USDA Natural Resources Conservation Service

NASIS SQL Guide

Structured Query Language for Query and Report Writing

5/4/2012

Table of Contents

NASIS SQL Guide

This guide is designed to provide the NASIS user an understanding of SQL and its various uses in NASIS. SQL is used to write queries, reports, properties, calculations and validations. An understanding of the NASIS data structure (tables and columns) is required before using SQL.

This document is to be used with the NASIS web site **References.** These documents are necessary for understanding the Query and Report writing process:

- The "Tables and Columns" document identifies the NASIS tables and its columns.
- The "NASIS CVIR Language Manual" Scripting language for NASIS Calculations, Validations, Interpretations and Reports.
- NASIS Data Structure Diagrams.
- Data types and Comparison Operators chart in this document.

SQL

To become adept at writing queries, the user must have knowledge of the Structured Query Language database language. SQL, as it is commonly referred to, was created by IBM in the early 1970s as a unified language for defining, querying, modifying and controlling the data in a relational database. There are now over 75 different flavors of SQL in commercial use. NASIS originally used the Informix database and is now using the Microsoft SQL Server database. The basic SQL structure is standardized between commercial databases however there are dialect differences. This document will focus on the SQL Server dialect and how it is used with the various soils databases. SQL is used in NASIS and the Soil Data Mart, the Soil Data Access site and Web Soil Survey. Understanding SQL will allow the user the ability to query data or write reports from these various databases and sites.

SQL Syntax

A SQL statement contains several elements. The SQL has certain "**Keywords**" that have special meaning. They are typically entered in UPPERCASE, however SQL is not case sensitive. This is done for organization purposes only. The statement also contains **identifiers** which are the names of the databases, tables and columns. Typically, identifiers are entirely in lower case. And, the statement contains **operators or functions** are used for comparisons or mathematical equations. The operator can be used for arithmetic (+ or -) or as comparisons (> or =) or as logical (AND, OR, NOT) or aggregate functions (MAX, MIN, SUM, COUNT, AVG).

Keywords

The basic SQL statement consists of 3 key words:

- **SELECT** (column)
- **FROM** (table)
- **WHERE** (condition)

The **SELECT** clause:

- specifies the columns (e.g. musym, muname, mukind) to be retrieved,
- each column must have a unique name,
- allows for expressions that must follow normal SQL syntax (e.g. sandtotal_r + silttotal_r + claytotal_r AS particle_size),
- if expressions are used in the select statement, an alias (e.g. “particle_size”) must be used with the expression to provide a unique name.

The **FROM** clause:

- specifies all the tables used in the query,
- and may specify aliases and outer joins.

The **WHERE** clause:

- filters which rows to use in the FROM clause
- uses normal SQL conditions and
- uses the NASIS "JOIN table TO table" syntax to simplify writing join conditions,
- and the two tables in a JOIN condition must have a relationship

Example:

SELECT nationalmusym, muname
FROM mapunit
WHERE muname = “Harney silt loam, 0 to 1 percent slopes”;.

Identifiers

The NASIS and SSURGO metadata reports, found on the appropriate web sites, contain the identifiers needed for SQL statements. The “Tables_and_Columns.pdf” document provides the list of tables and the columns within each. These documents are designed to provide the user with information necessary to write SQL statements.

Operators or Functions

The arithmetic operators, comparison operators, logical operators and the aggregate functions are used to filter the search functions of the WHERE clause. In this document on page 6 a chart is provided that identifies the data types and the various comparison operators that can be used. Further information on the various operators and functions will be discussed as they are introduced in this document.

NASIS 6 contains a national database (server), local database (client), and a selected set (screen). Queries and Reports are designed to run off either the national database or the local database. Queries run against the National database require an Object table (e.g. Legend, Mapunit, Datamapunit), whereas queries written for the Local database can be written to retrieve child table data (e.g. Correlation, Component, Horizon, etc).

Workshop Examples and Exercises

The examples in this section are a sample of some approaches to writing NASIS reports. Over time, you will develop your own techniques and style. Exercises in this section build on concepts demonstrated in the examples. These exercises provide an opportunity for you to develop your own approaches to creating NASIS reports.

When writing reports from scratch, it helps to have a report writing methodology similar to that described for writing queries in the *NASIS User Guide*. You may ask yourself several questions.

- What data do I want in the report?
- Are the data in the database or do they need to be calculated or decoded?
- In which tables are the data?
- What tables are needed to complete the joins between tables?
- How do I want the data organized?
- How do I want the page layout to look?

After you have defined what you want in your report, write your report using the statements and guidelines in this technical reference guide. Trial and error is almost always needed to write a report that runs cleanly. Seldom will a report be written perfect on the first try. With practice however, moderately complex reports can be written to meet a wide variety of uses.

In some cases, existing reports can be found that nearly meets the users' needs. A short-cut to writing a new report is to simply copy the existing report and modify it to meet the needs. Scan the existing report to identify parts that need to be modified and make the changes. Even with this short-cut, trial and error is needed to create a report that runs cleanly and provides the desired result.

Data types and comparison operators

The **data type** (integer, character, etc.) and **comparison operators** (matches, "=", ">", "is null", etc) are used to establish query conditions. There is a relationship between the comparison operator and data types that must be understood. When a query is written to specify a condition in the WHERE clause, there must be a comparison operator (such as = or MATCHES) that is compatible with the data element in the query conditions. For example, the data element "area name" is a "Variable Character" data type and the MATCHES operator is valid for this data type. (MATCHES is case insensitive except for Area symbols and Map unit symbols. Whereas equals "=" indicates an exact match [IMATCHES was previously used for case insensitive in the SQL, but now only works in DEFINE statements.]).

Data Type	Comparison Operator											
	=	!=	>	<	>=	<=	IS NULL	IS NOT NULL	LIKE " "	MATCHES " "	BETWEEN AND	IN ( )
Character												
Variable Character (String)												
Text (narrative text)	III	III	III	III	III	III			IV	IV	III	III
Float									IV	IV		
Smallfloat									IV	IV		
Integer									IV	IV		
Smallint									IV	IV		
Datetime									IV	IV		
Bit (Boolean)									IV	IV		
Ordered Code (choice)									IV	IV		
Unordered Code (choice)			II	II	II	II			IV	IV	II	
Property	III	III	III	III	III	III			IV	IV	III	III
Evaluation	III	III	III	III	III	III			IV	IV	III	III
Rule	III	III	III	III	III	III			IV	IV	III	III
Query	III	III	III	III	III	III			IV	IV	III	III

Notes: Date and date time values must be entered in the correct format or an SQL error will result. NOT, AND, and OR operators are used to combine two conditions; they are not related to data type.

Blank	Allowed
II	Allowed by query program, but results may not be meaningful
III	Allowed by query program, but will result in SQL error when query is executed.
IV	Not allowed

"MATCHES" allows a string of characters to be entered without regard to the case. Equals, "=", (case sensitive) allows a string of characters to be entered, but must be in the exact case as stored in the database. The MATCHES comparison operator is unique to NASIS SQL.

In Microsoft® SQL Server™, each column, local variable, expression, and parameter has a related data type, which is an attribute that specifies the type of data (integer, character, **money**, and so on) that the object can hold. SQL Server supplies a set of system data types that define all of the types of data that can be used with SQL Server. The set of system-supplied data types is shown below.

Character Strings
char Fixed-length non-Unicode character data with a maximum length of 8,000 characters.
varchar Variable-length non-Unicode data with a maximum of 8,000 characters.
text Variable-length non-Unicode data with a maximum length of 2^31 - 1 (2,147,483,647) characters.

Integers
int Integer (whole number) data from -2^31 (-2,147,483,648) through 2^31 - 1 (2,147,483,647).
smallint Integer data from -2^15 (-32,768) through 2^15 - 1 (32,767).
tinyint Integer data from 0 through 255
bit (boolean) Integer data with either a 1 or 0 value.

decimal and numeric
decimal Fixed precision and scale numeric data from -10^38 +1 through 10^38 –1.
numeric Functionally equivalent to **decimal.**

Approximate Numerics
float Floating double precision number data with the following valid values: -1.79E + 308 through -2.23E - 308, 0 and 2.23E + 308 through 1.79E + 308.
real (smallfloat) Floating single precision number data with the following valid values: -3.40E + 38 through -1.18E - 38, 0 and 1.18E - 38 through 3.40E + 38.

datetime and smalldatetime
datetime Date and time data from January 1, 1753, through December 31, 9999, with an accuracy of three-hundredths of a second, or 3.33 milliseconds.
smalldatetime Date and time data from January 1, 1900, through June 6, 2079, with an accuracy of one minute.

Examples of how comparison operators are used in an SQL query:

matches (or LIKE)
WHERE legend.legendsuituse matches "3"

not equal
mustatus != "additional"

equal to code value
repdmu = 1

between two values
muacres between ? and ?
muacres between 10 and 50

greater and less than
muacres >2000
muacress <5 acres

equal to text
legendsuituse = "current wherever mapped"

Wildcard characters

There are Wildcard characters that can be used with the operator 'LIKE' in the SQL to search for data. They are used to substitute one or more characters when searching for data. The standard NASIS wildcards are the '?' question mark for single character and the '*' asterisk for multiple characters. Other characters allowed include:

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for exactly one character
[charlist]	Any single character in charlist
[^charlist] or [!charlist]	Any single character not in charlist

The bracket '[ ]' wildcard uses a specific set of characters. It can be a continuous list e.g. [a-d] (will select any letter between a and d) or [a,c,d] (will select only the three letters identified in the bracket.

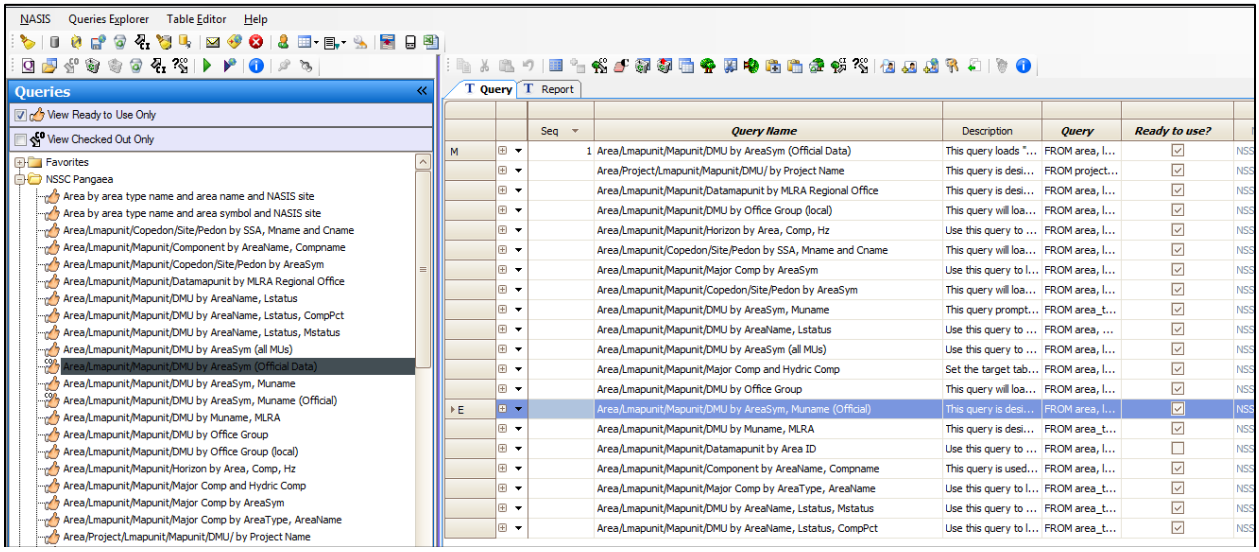
The '^' and '!' can be used to negate a set. e.g. [!MO123] or [^TN101] would not be selected in an area query.

Example	Mapunit name
All map units with Menfro in the name	%menfro%
All map units with 3 to 8 percent slope in the map unit name	%3 to 8%
All map units with flooded in the map unit name	%flooded*%
All map units with Menfro and silt loam texture in the map unit name	%menfro % silt loam%
All map units with a slope range of teens on the high end	%1_ slope%
All map units with "men" in the name and one letter before and after the three letters	_men_ %
All map units that start with CA in the name	[CA]%
All map units except the ones that that start with BC in the map unit name	![BC]%

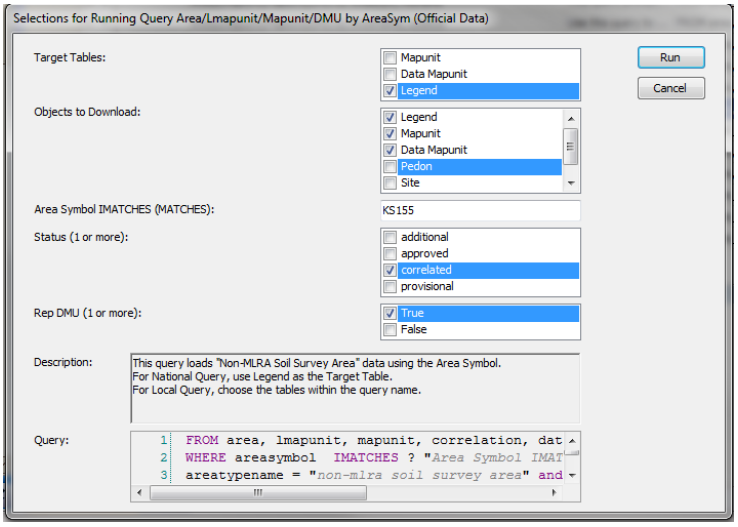
Avoid using wildcards at the beginning of the search pattern. Search patterns that begin with wildcards are the slowest to process. Pay careful attention to the placement of the wildcard symbols because the data returned may not be what was expected.

Queries

NASIS 6.2 has added enhancements to the Query function. The Query and Report "tables" allows the user to query for and manage Queries and Reports in a table format.



Another enhancement is how the user decides which objects to download when populating the Local Database.



The new National Query parameter box appears. The box includes the Description and Query panels and in addition, the box includes a new panel for 'Objects to Download' allowing the user to selected the various objects to be downloaded from the national server.

Queries

The purpose for a NASIS query is to load the local database and to populate the selected set with data that is filtered to meet the needs of the user. The NASIS "query" requires knowledge of SQL and the database structure. The NASIS query uses two Keywords: the **FROM** clause and the **WHERE** clause. The **SELECT** clause is not used since the NASIS query is designed to return the data for all the columns within the table(s) identified in the FROM clause. Since queries are understood to pull all columns, the SELECT * (select all columns) has been coded into the Query editor.

Queries are created to retrieve data from the National Database and used again to populate the Selected Set.
A query used to 'Run Against the National Database' requires an Object table. The Object Table is the Parent table within an object. (e.g. the legend table is the object table for the Legend Object).
Queries designed to 'Run Against the Local Database' do not require an Object Table.

A simple query would be to load all instances of a specific map unit name. Opening the 'Tables and Columns' report, identify the columns, table and data types necessary to write the query.

Table Physical Name: mapunit
Table Label: Mapunit

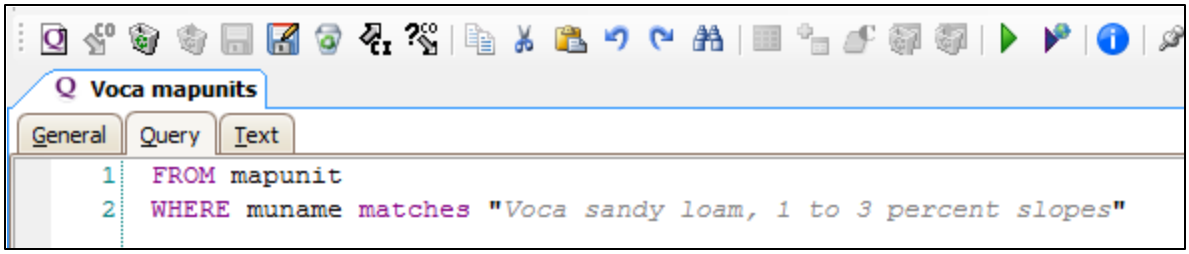
Seq	Column Physical Name	Column Label	Logical Data Type	Physical Data Type	Not Null?	Size
1	muname	Mapunit Name	String	Varchar	No	240
2	muname_s	S	Integer	Smallint	No	
3	mukind	Kind	Choice	Smallint	No	
4	nationalmusym	National Mapunit Symbol	String	Varchar	Yes	6
5	mapunitlfw_l	Low	Integer	Smallint	No	
6	mapunitlfw_r	RV	Integer	Smallint	No	
7	mapunitlfw_h	High	Integer	Smallint	No	
8	mapunitpfa_l	Low	Float	Real	No	

The map unit name field physical name is 'muname' and appears in the Mapunit table. The Mapunit table is the Object table. The muname field is a string data type and is compatible with all comparison operators. The simple query for use to extract map units from the National database would be:

FROM mapunit
WHERE muname matches "Voca sandy loam, 1 to 3 percent slopes"

The process would be to first select the Queries Explorer panel,

- choose to Open a new Query,
- enter a Query name, then
- select the Query tab and
- enter the SQL statement.



Load all instances of a named component into the Selected Set

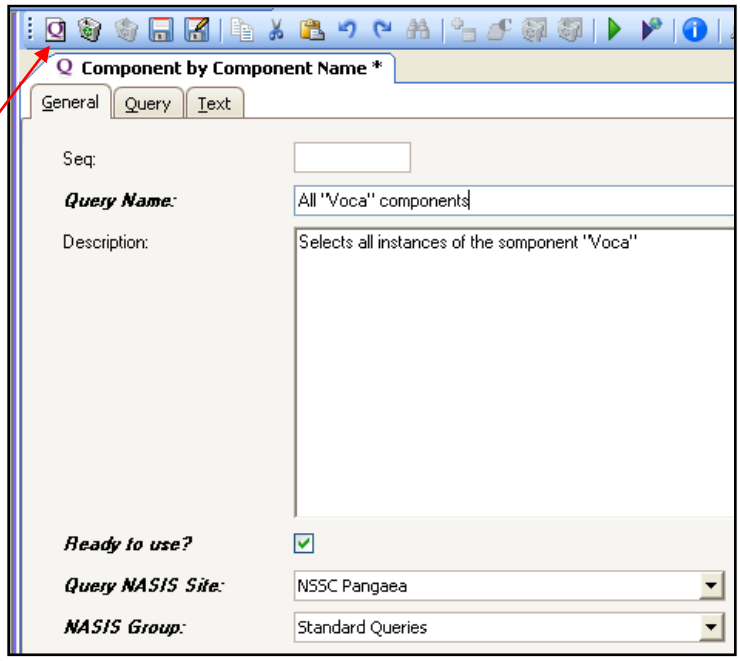
Assuming a simple local database query, to load all instances of a particular component name, then the first step in writing the query is to review the "Tables and Columns" report:

Table Physical Name: component
Table Label: Component

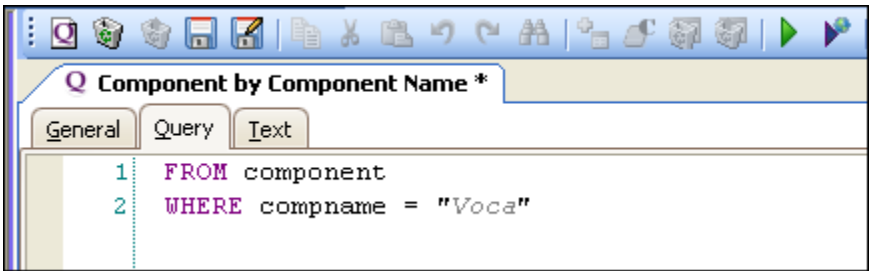
Seq	Column Physical Name	Column Label	Logical Data Type	Physical Data Type	Not Null?
1	dmuiidref	Lineage	Integer	Int	Yes
2	seqnum	Seq	Integer	Smallint	No
3	comppct_l	Low	Integer	Smallint	No
4	comppct_r	RV	Integer	Smallint	No
5	comppct_h	High	Integer	Smallint	No
6	compname	Component Name	String	Varchar	No
7	compname_s	S	Integer	Smallint	No
8	localphase	Local Phase	String	Varchar	No

The component name column (Physical Name is "compname") is found in the Component table and the field is a variable character (Varchar).

In NASIS, click on the "Add New Query" icon  . The General tab appears. Populate the query name and the description. Both fields are required.



The Query tab is used to write the SQL. The "**SELECT ***" is understood for all queries, therefore the query begins with the keyword **FROM**.



The above query is an example of a basic local database query. It includes one table in the FROM clause and one condition in the WHERE clause. This example will return and populate all component data for all instances where the component name is equal to the exact letters “Voca”.

The equal sign “=” is a “comparison operator”. The equal sign is used for an exact match in the field being compared. Any component in which the name is “VOCA” or “voca” or “VoCa” will not be loaded.

This query can be run against the local database in order to populate the selected set. This query will not work against the National Database because no Object table is identified. The Object table, ‘Datamapunit’, would need to be included in the SQL. Adding multiple tables will be discussed in a later section.

Use of the Question mark "?"

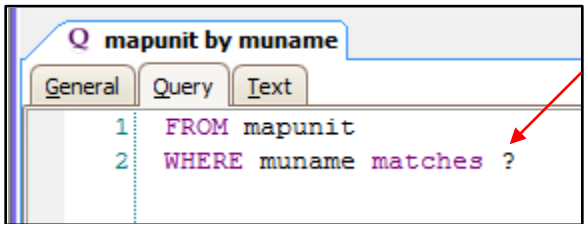
In the first query, a specific map unit name was written into the query. To use this query to load a different map unit name the user must modify the query. The user is interested in writing one query that can be used multiple times allowing different map unit names to be allowed. This introduces the concept of the question mark in queries.

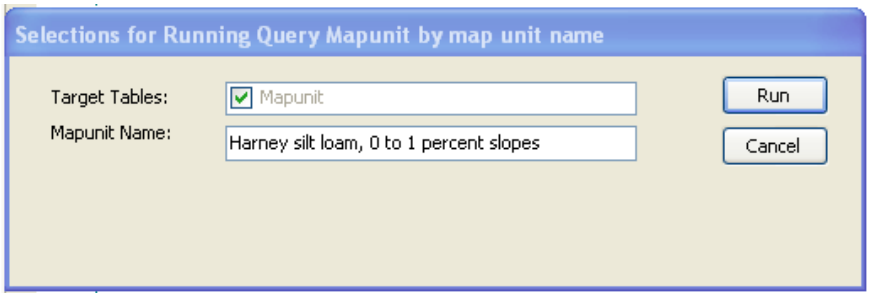
The first step is to review the Table and Columns report and, as previously shown, the map unit name column, 'muname', is in the map unit table with a 'String' data type.

Table Physical Name: mapunit
Table Label: Mapunit

Seq	Column Physical Name	Column Label	Logical Data Type	Physical Data Type	Not Null?	Size
1	muname	Mapunit Name	String	Varchar	No	240
2	muname_s	S	Integer	Smallint	No	
3	mukind	Kind	Choice	Smallint	No	

This query is written with a variable, the question mark '?'. The "?" question mark creates a parameter box. The parameter box is used to identify the map unit name to be queried. When run, the query will prompt the user to identify the map unit name to be queried:





Use of ">, <, =" comparison operator

In this example, the user is writing a query to load a component name with a component percentage greater than or equal to 80 percent. The first step is to review the Tables and Columns report to identify the columns for use in the query:

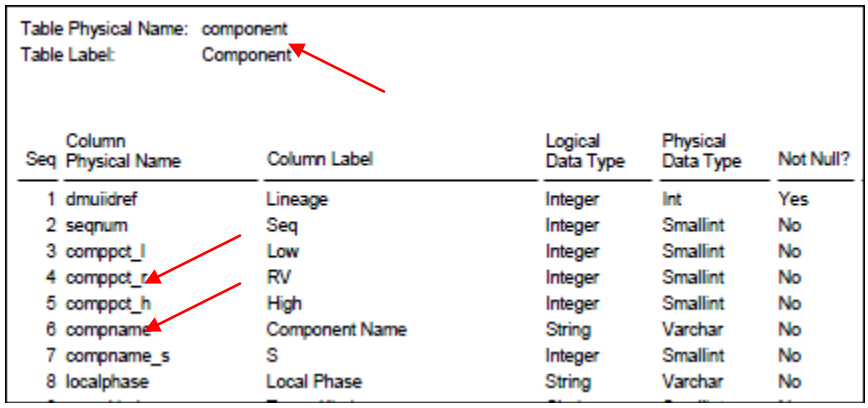
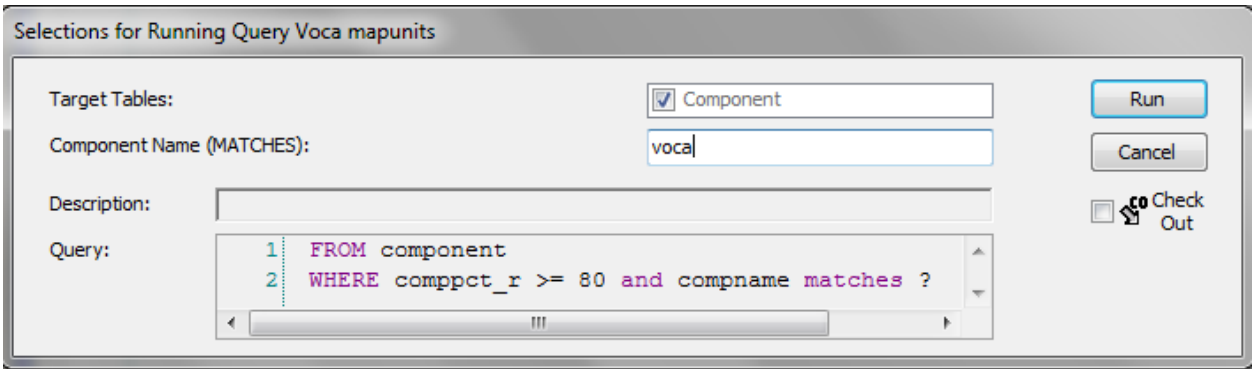


Table Physical Name: component
Table Label: Component

Seq	Column Physical Name	Column Label	Logical Data Type	Physical Data Type	Not Null?
1	dmuiidref	Lineage	Integer	Int	Yes
2	seqnum	Seq	Integer	Smallint	No
3	comppct_l	Low	Integer	Smallint	No
4	comppct_r	RV	Integer	Smallint	No
5	comppct_h	High	Integer	Smallint	No
6	compname	Component Name	String	Varchar	No
7	compname_s	S	Integer	Smallint	No
8	localphase	Local Phase	String	Varchar	No

Notice the component percentage has three columns: the Low, the RV and the High. The user chooses the RV value since it is most commonly populated. This field is an Integer data type and according to the Data type and comparison operator chart, only LIKE and MATCHES cannot be used.

FROM component
WHERE comppct_r >= 80 and compname matches ?



Notice the parameter box appears and prompts the user for a component name. The comparison operator "greater than or equal to" (>=) can only be viewed in the Query window of the parameter box. This query is used to search for those component names that the user enters in which the component percentage is greater than or equal to 80 percent.

Use of "?"parameter

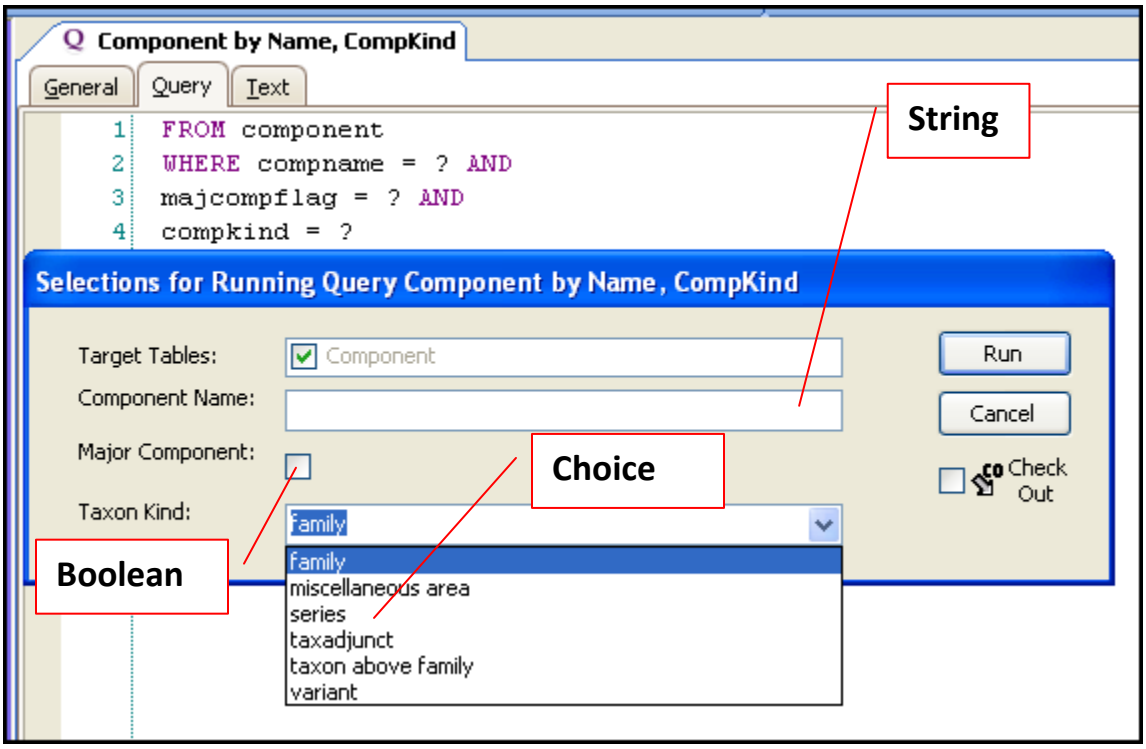
Continuing, the user requests the component name, component kind and the major components. The first step is to identify the Tables and Columns to be used. The columns are in the component table and include a variety of data types:

- Component Name is a "Variable Character".
- Component Kind is a "Choice" field.
- Major Component Flag is a "Boolean" (1 or 0, Yes or No, True or False, field).

Table Physical Name: component
Table Label: Component

Seq	Column Physical Name	Column Label	Logical Data Type	Physical Data Type	Not Null?
1	dmuiidref	Lineage	Integer	Int	Yes
2	seqnum	Seq	Integer	Smallint	No
3	comppct_l	Low	Integer	Smallint	No
4	comppct_r	RV	Integer	Smallint	No
5	comppct_h	High	Integer	Smallint	No
6	compname	Component Name	String	Varchar	No
7	compname_s	S	Integer	Smallint	No
8	localphase	Local Phase	String	Varchar	No
9	compkind	Taxon Kind	Choice	Smallint	No
10	majcompflag	Major Component	Boolean	Bit	No

This query will identify the various methods the question mark is used based on the data types. The use of the question mark for a 'String' has been discussed. The use of the '?' on the "Boolean" data type provides a 'radio box' to be either clicked on for 'Yes' or cleared for 'No'. The 'Choice' data type field for component kind provides the drop down choice list for component kind within the parameter box.



Use of "IN ( )" and "IN (?)" parameter

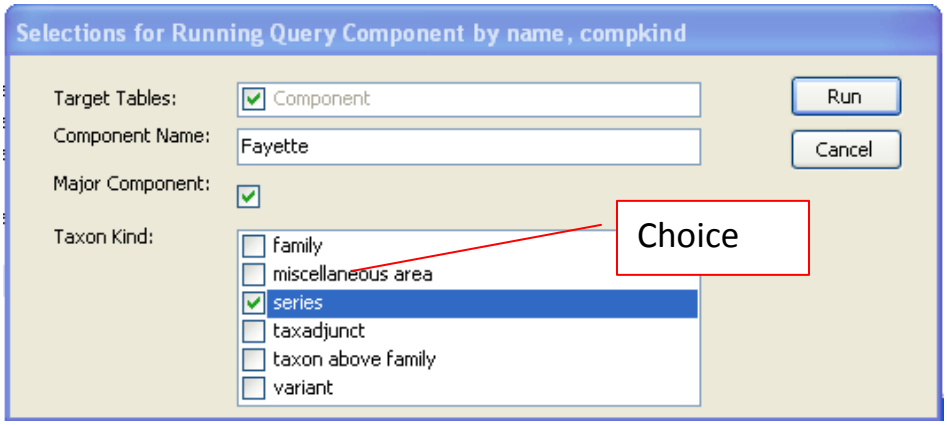
Using the same query but with a slight change now provides a different method of presenting the Component Kind choice list. The question mark for the Component Kind is now enclosed in parentheses behind the word "**IN**". The IN clause has several methods of use.

- **Parameter selection based on choice list fields**

The first is with a choice list field such as "component kind".

FROM component
WHERE compname = ? and majcompflag = ? and compkind IN (?)

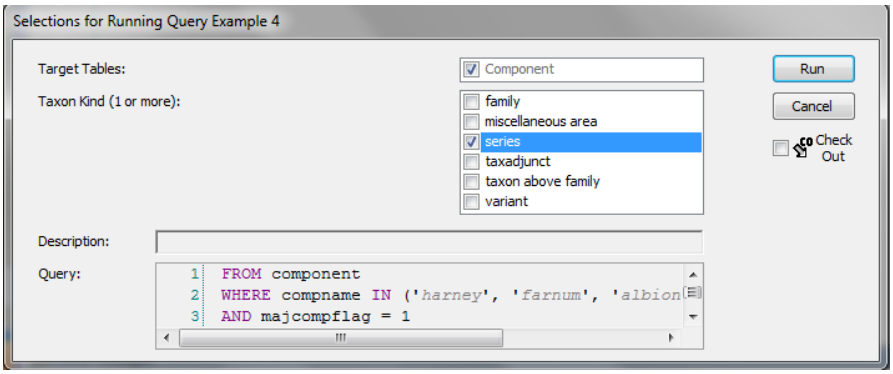
When used with a choice list field the parameter box will include the choice list for Component Kind with multiple selection ability:



- **Multiple entries in one string**

The "IN" command can be used to provide a list of variables to be searched. In this query, the user is loading several component names using a single query. The syntax used is the IN followed by a space, then open parentheses, then open quotation followed by the variable followed by the closed quote. Variables are strung using a comma to separate each variable. The last variable is followed with a closed parenthesis.

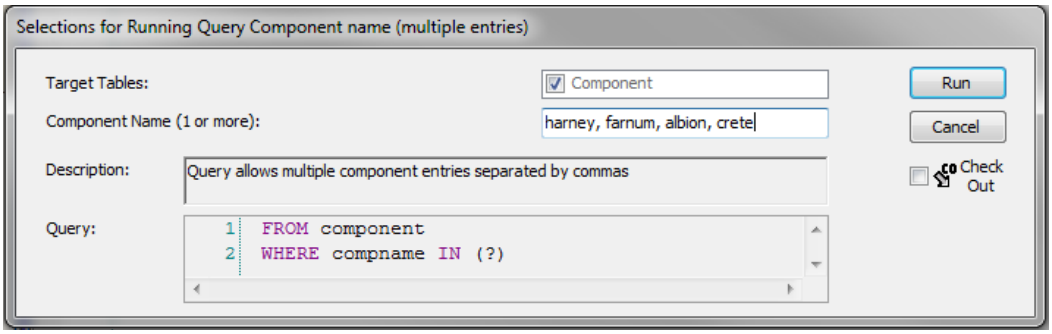
FROM component
WHERE compname IN ('harney', 'farnum', 'albion', 'crete') AND mapcompflag = 1 and compkind IN (?)



- **User defined multiple entries using the parameter screen**

A variation is to include the "?" inside the IN clause parentheses. When used with a 'String' data type, this allows multiple entries, in this case multiple component names, within the parameter box. Each entry is separated by a comma delimiter.

FROM component
WHERE compname IN (?)
AND mapcompflag = 1



Use of BETWEEN command

The 'BETWEEN' operator allows the user to select a range. What if the user would like a query that identifies a certain component name and the user wishes to provide a slope range.

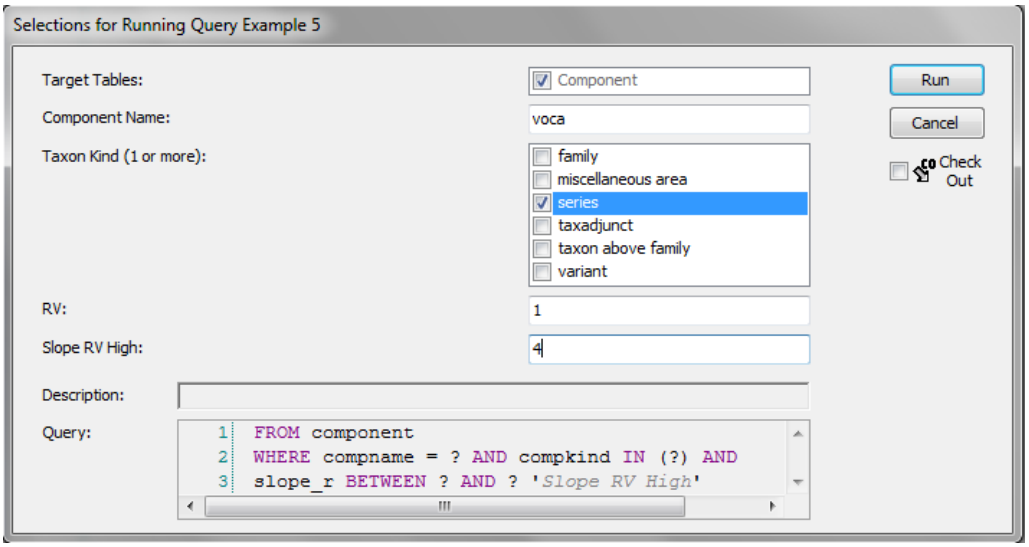
Reviewing the Tables and Columns report, the component table and its columns are identified.

Table Physical Name: component
Table Label: Component

Seq	Column Physical Name	Column Label	Logical Data Type	Physical Data Type	Not Null?
1	dmuiidref	Lineage	Integer	Int	Yes
2	seqnum	Seq	Integer	Smallint	No
3	comppct_l	Low	Integer	Smallint	No
4	comppct_r	RV	Integer	Smallint	No
5	comppct_h	High	Integer	Smallint	No
6	compname	Component Name	String	Varchar	No
12	slope_l	Low	Float	Real	No
13	slope_r	RV	Float	Real	No
14	slope_h	High	Float	Real	No

The slope_r is an integer data type. The parameters will be specific numbers. The use of the BETWEEN command allows the user to prompt for two fields to be entered in the parameter box.

FROM component
WHERE compname = ? AND compkind IN (?) AND slope_r BETWEEN ? AND ? "Slope RV High"



The query prompts the user for the Component Name, the various Taxon Kinds and two Slope Range fields. In addition, notice the use of the words in quotes after each question mark in the SQL. Placing the words "Slope RV High" after the question mark allows the user to override the default label and assign a more meaningful label for the specific field. Compare the default label of 'RV' to the assigned label of 'Slope RV High'.

Exercises

Using the **NASIS CVIR Language Manual**, write the following queries.

1. Load all map units with a given name with a status of “correlated”.
2. Identify a map unit name with the specific Prime Farmland class.
3. Load a component based on its drainage class.
4. Load all components that are mollic albaqualfs.

Adding additional tables

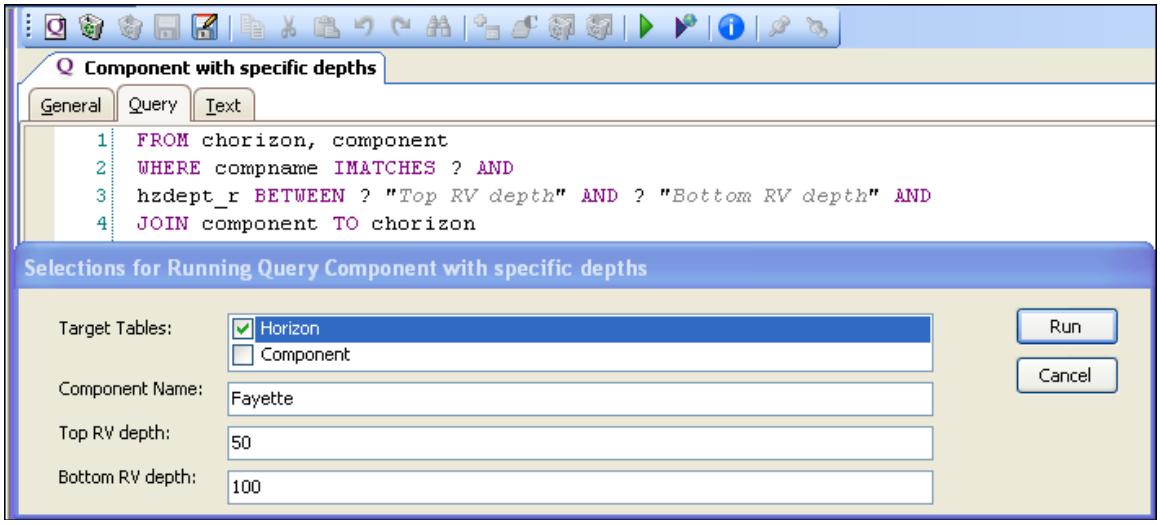
Loading specific components with specific horizon depths

In most instances the user will wish to load several tables of data into the local database or selected set. Additional filters (search conditions) may also be necessary to target certain data sets for editing purposes. Review the "Tables and Columns" report to identify the horizon table and its columns.

Table Physical Name: chorizon
Table Label: Horizon

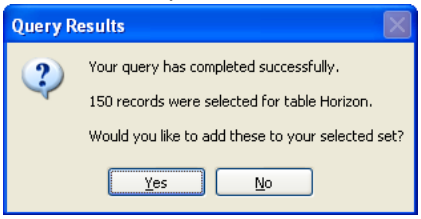
Seq	Column Physical Name	Column Label	Logical Data Type	Physical Data Type	Not Null?
1	coiidref	Lineage	Integer	Int	Yes
2	seqnum	Seq	Integer	Smallint	No
3	hzname	Designation	String	Varchar	No
9	hzdept_l	Low	Integer	Smallint	No
10	hzdept_r	RV	Integer	Smallint	No
11	hzdept_h	High	Integer	Smallint	No
12	hzdepb_l	Low	Integer	Smallint	No
13	hzdepb_r	RV	Integer	Smallint	No
14	hzdepb_h	High	Integer	Smallint	No

Item 10, "hzdept_r" is the top depth Representative value. This will be used to identify those horizons that fall within the users determined limits.



The query includes two tables in the FROM clause separated by a comma. The clause "WHERE" includes the search conditions for the component name and uses the previously defined BETWEEN to establish a range of top depths for the search. The query is completed with the JOIN statement joining the two tables, component and chorizon. The use of the JOIN statement in the WHERE clause is a legacy of the NASIS Informix database version.

The Target Table is set to Horizon because horizon depth is the focus of the query. The user then populates the component name to search and the top and bottom limits of the horizon top depth.

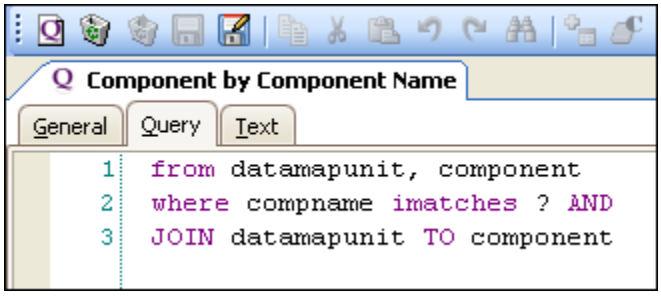


Target Tables

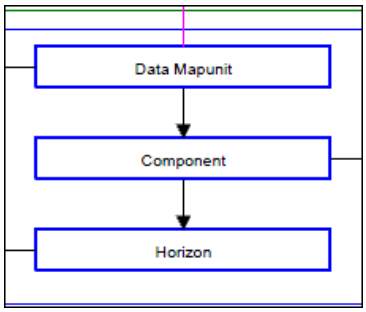
Simply put, the target table focuses the outcome of a particular query. In this way, the user can control the query so that it loads only the specific data to be worked on during that edit session. The target table can greatly restrict or expand the number of records returned by a particular query. To understand target tables, the user must understand the relationship between objects in the NASIS database. The data model diagrams help to visualize this relationship.

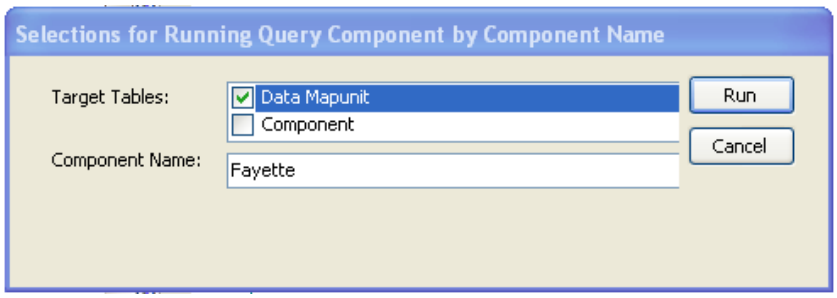
So, how does a Target Table restrict the records returned by a query? In an edit session, the user only wishes to work with components that are named "Fayette". The user would choose a query that loads components by ***compname*** and specify **Fayette**. Because component name is in the component table, either datamapunit or component could be selected as the target table. *Whether or not only the* ***Fayette series*** *is loaded depends on the target table choice*.

- If *datamapunit* is selected as the target table, *all* Data Mapunits that have at least one ***Fayette*** component, in addition to *all* the DMU other components, are loaded into the component table.
- If *component* is selected as the target table, only components named Fayette are loaded.

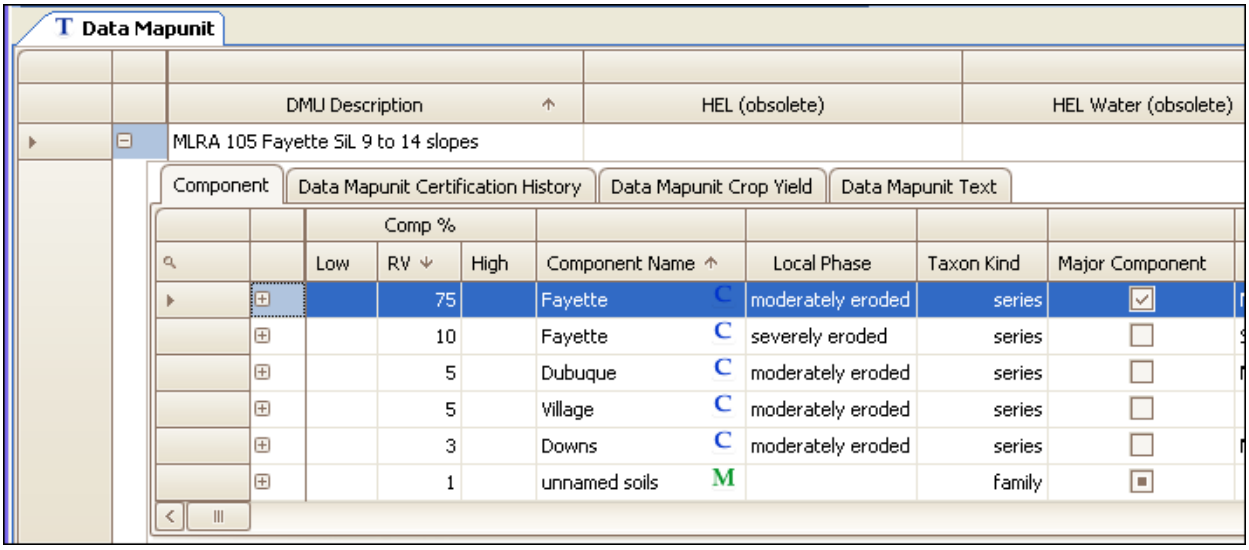


Using this simple query as an example:
The query has two tables in the FROM clause that become Target Tables in the parameter box. Setting the Target Table to "Data Mapunit"

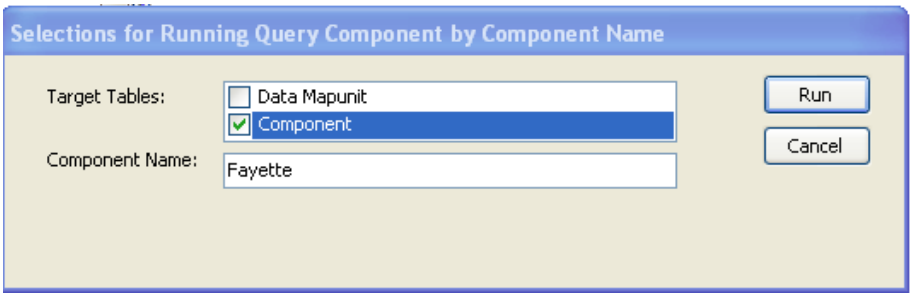




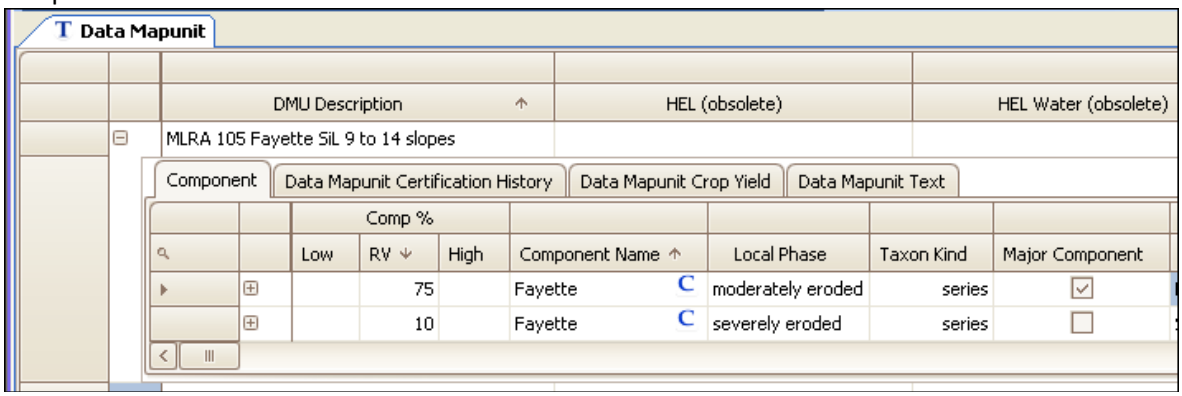
Provides for a selected set in which all Data Mapunits in which the Fayette component is a member.



Contrast the previous results with setting the Target Table to Component.



By using Component as the Target Table, the component table is populated with only the "Fayette" component.

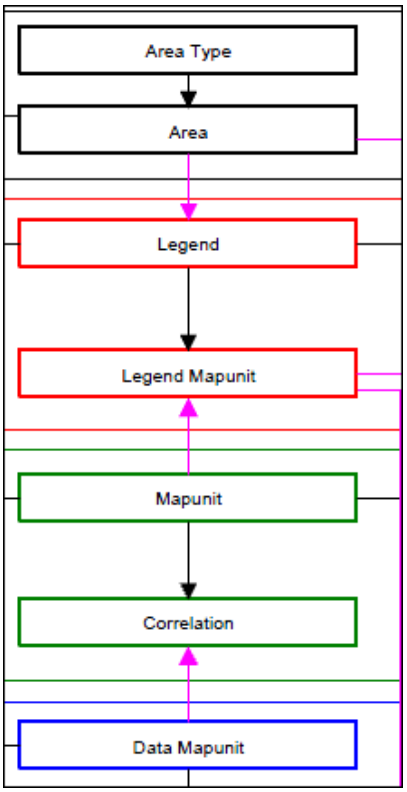


Contrast the results of the same query with different Target Tables.
Setting the Target Table to Data Mapunit requested all Data Mapunits in which Fayette is a component. What is a Data Mapunit? It is all the components and their data for a given map unit.
Setting the Target Table to Component requested to populate the Component table with only the Fayette components. The Component table contains only a specific component, therefore if used as the Target Table then only the specific named component is populated within the Data Mapunit for the selected set.

Join Conditions

In this example the user will load all map units and the associated data mapunit for a given Survey Area. The manuscript reports require the Area, Legend, Mapunit, and Datamapunit objects to be loaded. This query will require multiple tables and additional search conditions.

Using the database schema, the user identifies the tables needed for this query:



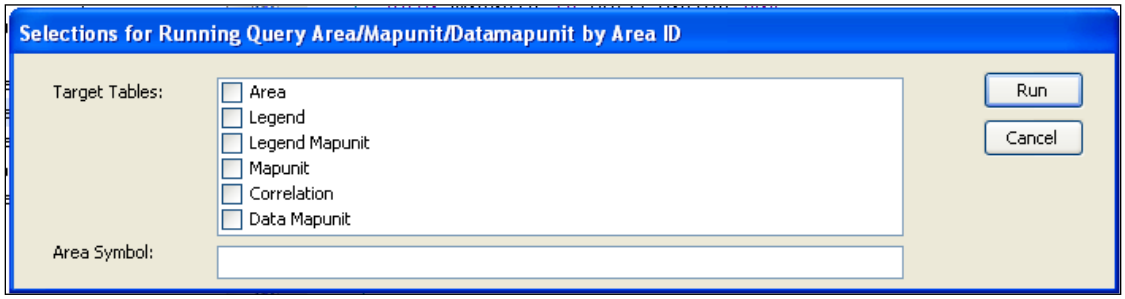
After the tables are identified the user can review the Tables and Columns report and enter the appropriate information into the NASIS query:

FROM area
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
WHERE areasymbol MATCHES ? and mustatus != 'additional' and repdmu = 1

The required tables are entered in the FROM clause. The clause "WHERE" has three search conditions. Notice the JOIN conditions exist in the FROM clause.

The search conditions include the area symbol (using an MATCHES), the map unit status (using "!=" does not equal) and the representative datamapunit (using "=" equal) as assigned in the correlation table.

This query prompts for the Target Tables and Area Symbol.



The use of SQL Express 2005 allows for a new method of joining tables and makes the queries and report more efficient. When the joins are performed in the WHERE clause the query has to make a large concatenated table of all the tables in the FROM clause then reduce the data with the joins. (You can put up to 256 tables in one FROM clause. The new syntax will make the first join and then pass the matching values on to the next join reducing the size of the file.

It's like going to a restaurant and ordering everything off the menu then restricting only eating one or two dishes after everything arrives. Using JOINs in the FROM clause selects only the item you want off the menu right away.

Write:

```
FROM mapunit
INNER JOIN correlation by default
INNER JOIN datamapunit by default;.
```

Instead of:

```
FROM mapunit, correlation, datamapunit
WHERE join mapunit to correlation
and join correlation to datamapunit;.
```

1. Use the highest level table needed for the query and work down by joining the tables because they have the smallest number of row unless one of the child tables has a restriction that reduces the number of records.
2. When using the BY condition you specify a relationship name defined in the NASIS data dictionary. In most cases, the relationship name is "by default". If more than one relationship exists between a pair of tables you must use the correct name. The Info page (Blue circle icon) for a table in NASIS will list the relationship names.
3. There is an OUTER JOIN explanation in the CVIR Language Manual page 35.

LEFT OUTER JOIN "Name" by default.

Data restrictions can be added in the "FROM" clause using the term ON. When the join conditions begin with ON, standard SQL syntax applies. You must specify the exact columns to be matched in each of the tables. Example below:

```
FROM datamapunit
INNER JOIN component ON datamapunit.dmuiid= component.dmuiidref
```

You can also add more conditions beyond just the key columns. Example below:

```
FROM component
INNER JOIN chorizon ON component.coiid=chorizon.coiidref and
hzname LIKE "%R%" and hzdept_r !=0
```

Joining data to a restrictive sub query runs quicker because the data set is reduced (sub queries run first thus reducing the data set size). Join the most restrictive tables first. By joining in the 'FROM' clause it restricts the data from the start and thus decreasing the size of what's being queried.

Exercises

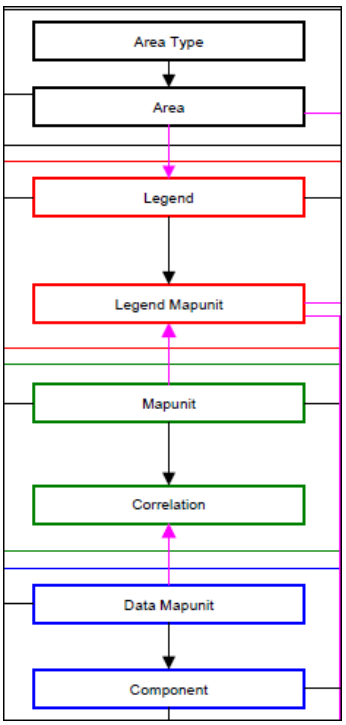
Create a query that

1. Load a map unit by name that ranges from 10,000 to 20,000 acres.
2. Includes the component name column and queries for a specific component.
3. Includes the horizons and queries for a specific depth
4. Identifies those components for a given survey area that area marked as “major component” yet are less than 15 percent in composition.

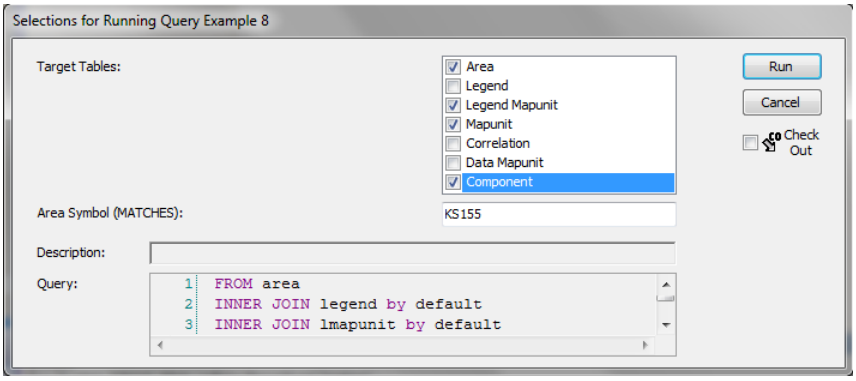
Use of the OR command

Load all major components and all hydric components

The 'OR' command is used to search for multiple conditions that may exist. For instance, to search for all major components and only those minor components that are 'hydric'. In this example, not all minor components, but only those hydric minors are searched.



FROM area
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
INNER JOIN component by default
WHERE areasymbol MATCHES ? AND repdmu = 1 AND mustatus != 'additional' AND ((majcompflag='yes') OR (hydricrating = 'yes'))



Explanation

This is a multi-table query that prompts the user to enter a Survey Symbol. This query includes all the tables from Area to Component. Notice the FROM clause lists the tables and their JOIN condition.

The 'WHERE' clause includes a question mark "?". This question mark creates a parameter box for entering the area symbol. The area symbol is using MATCHES to allow the user to enter a survey area symbol insensitive to case. If there are multiple records in the correlation table, then the 'repdmu' condition selects the one record identified as the representative DMU is loaded. This mustatus condition will load all map units except those with the status labeled as "additional".

Use of the OR command:

Notice there are two search conditions within a set of parentheses. The two search conditions are compared using the "OR" command. This search condition states "load all the major components OR load all the components in which the hydric rating matches "Yes". By using the OR command in this search condition, the computer will load all the major components and any hydric components whether the hydric component is a major or minor component.

Exercises

1. Create a query that selects components in which the flooding months contain a frequency of occasional or frequent.
2. Include in that query those components in which the frequency is also Long or very long

Arithmetic Operators

Load horizons in which Sand, Silt and Clay totals do not equal 100

After checking the Tables and Columns report, the sand, silt and clay columns are entered into the query.

```
FROM chorizon
WHERE sandtotal_r is not null and
      silttotal_r is not null and
      claytotal_r is not null and
      om_r < 36 and
(sandtotal_r+silttotal_r+claytotal_r) NOT BETWEEN 99.995 and 100.005
```

Line 1: selects the horizon table. The physical name is “chorizon”.
Line 2: begins the WHERE clause
Lines 2-5: verify that the sand, silt and clay fields for this query cannot be NULL fields and OM less than 36 percent.
Line 6: adds the sand, silt, clay RVs to verify they are between the two values.

If these conditions are met, then the query will load those horizons that meet these search criteria. Notice, there are no prompts for a parameter box. This query will load all horizons in the local database that meet these conditions. (No ‘object table’ is used in this query, therefore it can not be run against the national database)

Exercises

1. Create a query that will sum the sand fractions and compare the result to the sand total for a given survey area.
2. Create a query that compares the no. 200 sieve to the properties populated for particle size
3. Create a query that identifies those components with a clay texture that are less than 35 percent clay.

Outer Joins

Load all Components for a survey area based on component percent

When using multiple tables, the query assumes a one to one relationship between the two tables. There is a row in the parent table (Data Mapunit) that is linked directly to the child table (Component). If a data mapunit has an empty component table then the 1 to 1 match fails and that data mapunit does not meet the criteria.

FROM areatype
INNER JOIN area by default
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
INNER JOIN component by default
WHERE areaname MATCHES ? AND comppct_r > ? 'Component RV%' AND mustatus = 'correlated'

So, what if a Data Mapunit has an empty Component table? Should that information be loaded as part of the selected set? Most of the time the answer is Yes. This is over come by the use of the OUTER join.

FROM areatype
INNER JOIN area by default
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
LEFT OUTER JOIN component by default
WHERE areaname MATCHES ? AND comppct_r > ? 'Component RV%' AND mustatus = 'correlated'

With the query modified, it will now load all Data Mapunits associated with the map unit even if there is nothing populated in the Component table. The LEFT OUTER JOIN stipulates that all data mapunits are to be loaded regardless if there is data in the component table.

Once an outer join is used in a series of JOIN statements, the OUTER JOIN must be used on all joins below the first OUTER JOIN.

Exercises

Create a query to:

1. Load component surface fragments for a user defined official legend.
2. Load all sites, pedons and transects for a given survey area using the User Site ID.

Types of joins

Tables can be joined:

- by the relationship, many times this uses the term “default”
- by a defined relationship e.g. “mlra_sso” or “nonmlra_ssa” for area type joins
- on a specific relationship e.g. primary key:foreign key
- on any two fields with corresponding values

The relationship name is identified in the related parent table in the relationship Name column found with the blue i icon (view information) in NASIS.

The types of joins are:

- INNER JOIN
  - only matching values from both tables
  - the most common type of join
  - Allows you to join multiple tables in one query, but it requires specific condition for it to work.
  - You must ensure that the join statement has two tables with at least one common overlapping field.
  - An INNER JOIN is the default join type. If an inner join is omitted from the join clause of a query, the NASIS SQL server will assume it to be an inner join
- OUTER JOINS

If you understand inner joins, understanding OUTER JOINS is an easy progression. They both look for and display every match they find between two tables. Both joins require that you specify the matching field(s) in the ON clause. Outer joins show the records that inner joins omit.

  - left outer join
    - all values from the left (table left of the word join) table and only the matching values in the right (table right of the word join) table
  - right outer join
    - all values from the right table and only the matching values from the left table
  - full outer join
    - All values from both tables regardless of matching values
    - Fields will have null values that lack a matching row

Join Examples

1. The query below retrieves values that match in both tables. If a legend is not linked to an area it will not be retrieved.

   FROM area
   INNER JOIN legend by default

2. The query below retrieves all values in the left table (correlation table) and only matching in right table (data map unit table). This query will get all map units and only data map units that are linked to the correlation table. Any data map units that are not linked to a map unit will not be retrieved.

   FROM mapunit
   INNER JOIN correlation
   LEFT OUTER JOIN datamapunit by default

3. This query retrieves all values in right table (data map unit table) and only matching in left table (map unit table). This will get you all datamapunit and only mapunit where the mapunit are linked.

   FROM mapunit
   INNER JOIN correlation
   RIGHT OUTER JOIN datamapunit by default

4. This query retrieves all values in both tables. This will get all map units and data map units even if they are not linked

   FROM mapunit
   INNER JOIN correlation
   FULL OUTER JOIN datamapunit by default

5. This query joins on a specific primary key (legend.areiidref) and foreign key field (area.areaiid)

   FROM area
   INNER JOIN legend ON legend.areiidref=area.areaiid

6. This query joins a codename comparison (project.stateresponsible) and alias the table name (st)

   FROM legend
   INNER JOIN area st on CODENAME(project.stateresponsible)=Arkansas

7. This query performs a double joins on two values (atdbiidref and areatypename) with alias for areatype

   FROM AREA
   INNER JOIN areatype stt by default and stt.atdbiidref=1
   and stt.areatypename = "State or Territory"

8. This query creates a ‘parameter’ for state code and matches it to part of the area symbol. Parameter statement is discussed later:

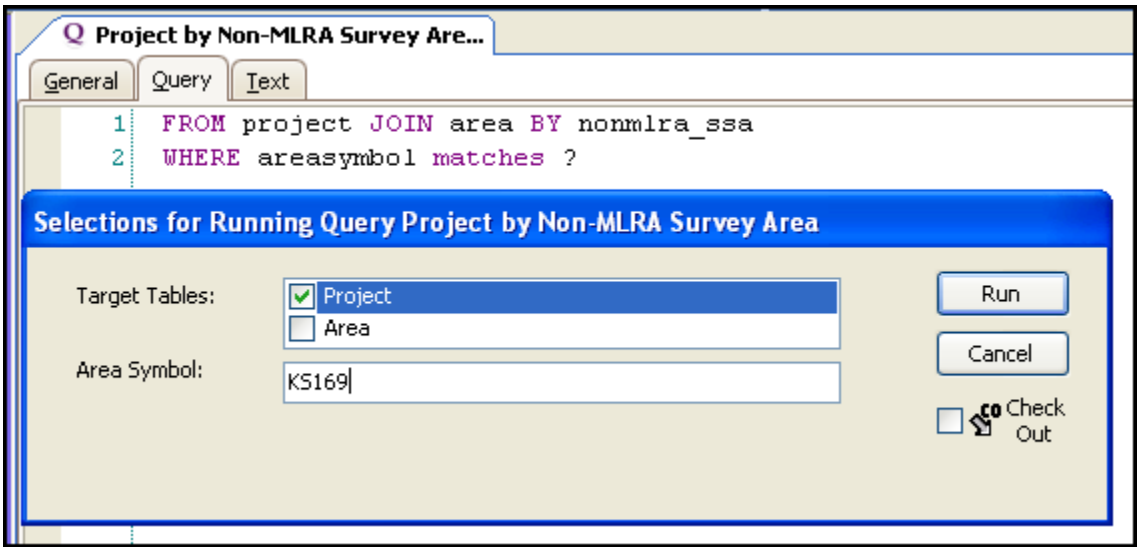
   PARAMETER areasym ELEMENT area.areasymbol PROMPT "State Symbol".
   SELECT LEFT((areasymbol), 2) imatches areasym
   FROM legend
   INNER JOIN area on areasymbol on areasymbol=areasym

9. This query creates a double self join (dmuiidref=c2.dmuiidref and c1.coiid!=c2.coiid). This will find duplicate components with the same name Captina in the same data map unit

   FROM component c1
   INNER JOIN component c2 on c1.dmuiidref=c2.dmuiidref and c1.coiid!=c2.coiid
   WHERE compname in (“Captina”)

Identifying Specific Joins

There will be instances where the JOIN between two tables must have a declared join column. The need for the "**BY**" is to determine the relationship (foreign keys) between the two tables.



The View Information is a tool used to identify the relationships between the related parent tables. The join between the Project and the Area could be on either the **nonmlra_sso** or the **mlra_sso**, therefore a BY is needed to identify the specific relationship.

Related Parent Tables

Table Label	NASIS Business Object Name	Relationship Name	Find/Load Related Enabled?
Area	Area Type	nonmlra_ssa	No
Area	Area Type	mlra_sso	Yes
NASIS Group	NASIS Site	default	Yes
NASIS Site	NASIS Site	default	Yes
NASIS User	NASIS User	object_last_udpated_by	No
NASIS User	NASIS User	record_last_updated_by	Yes

Subqueries using the EXISTS operator

Subquery is a query in a query. A subquery is usually added in the WHERE Clause of the sql statement. Most of the time, a subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value. Subqueries are an alternate way of returning data from multiple tables. A subquery is used to further restrict the results of the main query. The most common use for subqueries is filtering data in the WHERE clause of a SQL command. A subquery is set within the query using parentheses. Four special operators (EXISTS, IN, ALL, ANY), as well as the conventional operators like =, <, >, >=, <= are used to connect the containing command and the subquery.

The **EXISTS** operator is testing for existence of data, it either does exist (TRUE) or does not exist (FALSE). Therefore only one column or the asterisk (*) is necessary in the SELECT statement. What if a query is written to load all components that have more than one texture in the surface horizon? In this example, the query is written to prompt for a survey area and it selects the surface horizon. The subquery begins with EXISTS and tests the existence of more than one record ID (chiidref) in the horizon texture group table for the surface layer. Notice the use of the chorizon table which links the subquery to the query.

```
FROM areatype
INNER JOIN area by default
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
INNER JOIN component by default
INNER JOIN chorizon by default
WHERE areasymbol MATCHES ?
AND hzdept_r = 0
AND EXISTS (SELECT chorizon_iid_ref FROM chtexturegrp
                WHERE JOIN chorizon TO chtexturegrp
                GROUP BY chorizon_iid_ref
                HAVING COUNT(*) > 1)
```

Exercises

Create a query to load components by survey area that have multiple surface textures

Subqueries using the NOT EXISTS operator

Contrary to EXISTS, the NOT EXISTS is identifying the non existence of data. What if it was necessary to identify those components in which no horizon information is entered? The outer join is helpful, however another method is available. Using a subquery can be helpful to identify a child table with no open rows. In this example the NOT EXISTS is used. The subquery is in parentheses and it states to select everything for the horizon table and joins the component and the horizon table. The NOT EXISTS is a negative or reversal. If nothing exists, a table with no data, then it returns a TRUE statement and that data is loaded into the selected set.

FROM component
INNER JOIN chorizon by default
WHERE NOT EXISTS (SELECT chiid FROM chorizon WHERE JOIN component to chorizon)

Exercises

Create a query that loads all the components in a survey area in which the Parent Material table is not populated.

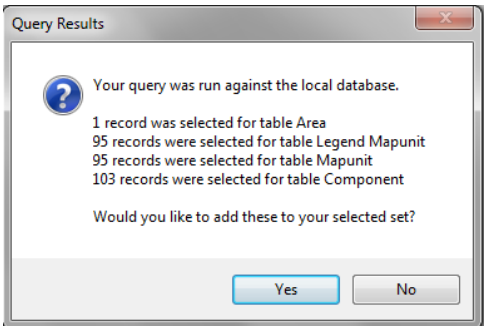
Subqueries using the = operator

What if it was necessary to identify all the component(s) with the maximum percentage in a survey area, or maximum in a datamapunit? The subquery using the = operator will return one result. The subquery, which extracts the maximum component percentage, is set apart using parentheses, and uses that value for comppct_r. What is returned for that value for use in the main query depends on whether the subquery is 'correlated' to the main query. This introduces the concept of a correlated versus uncorrelated sub query.

Correlated subquery

A correlated subquery has a more complex method of execution than single- and multiple-row subqueries and is potentially much more powerful. If a subquery references columns in the parent query, then its result will be dependent on the parent query (correlated). The SQL differences are subtle however notice the subquery FROM clause and compare the two versions. In the correlated subquery, the subquery and main query are linked using the WHERE clause that links the datamapunit in the subquery to the datamapunit in the main query.

FROM area
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
INNER JOIN component by default
WHERE areasymbol matches 'KS155'
AND repdmu = 1 and mustatus = 'correlated'
AND comppct_r = (SELECT max(comppct_r)
FROM component WHERE JOIN component to datamapunit)

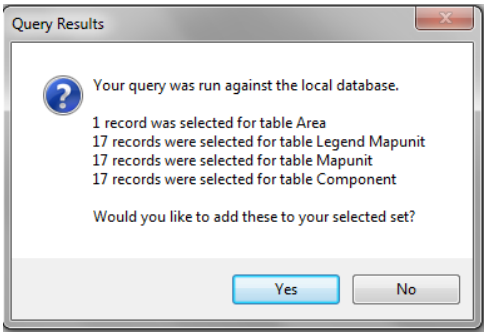


In the correlated subquery, the component with the highest component percentage in each datamapunit is presented as the value to be used in the main query.

Uncorrelated subquery

This subquery will load the component that contains that maximum percentage, but from where? Only 17 components contain the 100 percent found to be the highest comppct_r value in the survey area.

FROM area
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
INNER JOIN component by default
WHERE areasymbol matches 'KS155'
AND repdmu = 1 and mustatus = 'correlated'
AND comppct_r = (SELECT max(comppct_r) FROM component INNER JOIN datamapunit by default)



Since the WHERE clause does not exist in the subquery, the extracted comppct_r results come from the query of the entire component table. A maximum comppct_r value of 100 percent is returned and presented as the value to be used in the main query.

Subqueries using the IN operator

The IN operator requires that the subquery returns one value. In this example the IN operator is identifying the maximum bottom depth of the soil – the use of the term MAX on the horizon bottom depth column. In addition, this query contains a second subquery to identify those soils in which the parent material is “till”. This is an example of using multiple subqueries to load data.

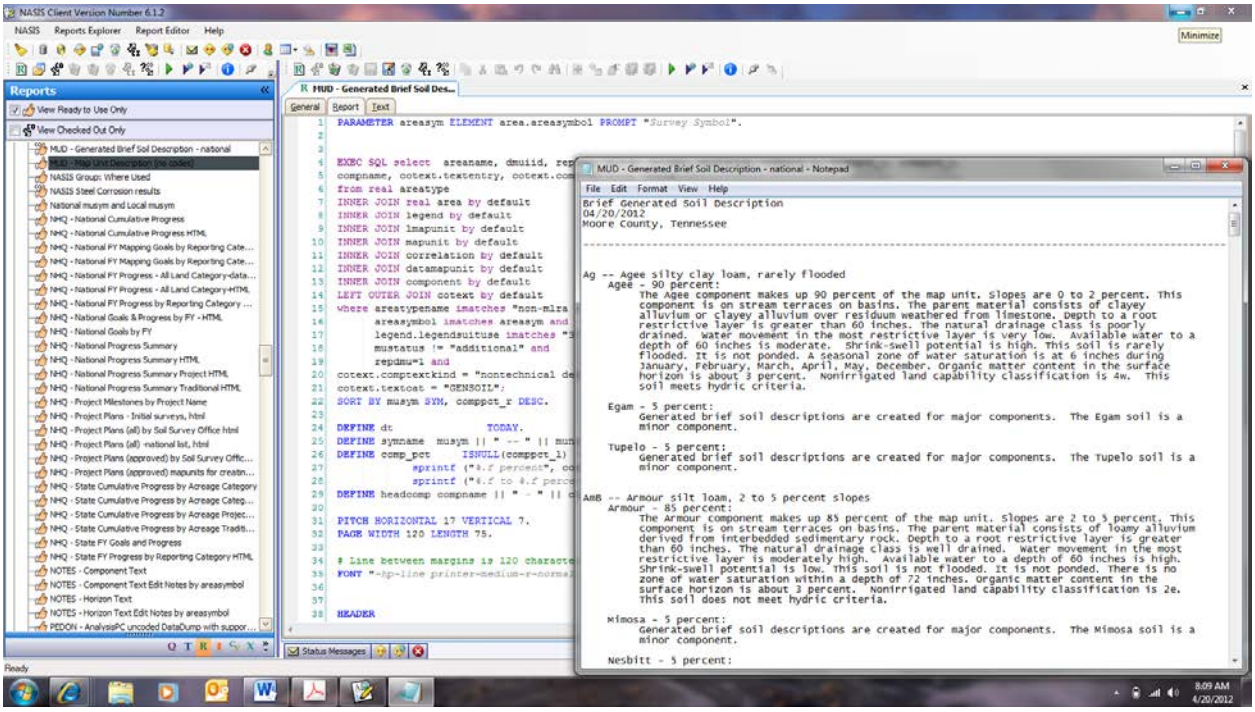
```
FROM area
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
INNER JOIN component by default
INNER JOIN chorizon by default
WHERE areasymbol matches ?
AND repdmu = 1 and mustatus = 'correlated'
AND ANY(SELECT * FROM copmgrp WHERE JOIN component to copmgrp and pmgroupname
matches '*till*' and rvindicator = 1)
AND hzdepb_r IN (SELECT MAX(hzdepb_r) FROM chorizon WHERE JOIN component to
chorizon)
```

The ANY keyword denotes that the search condition is TRUE if the comparison is TRUE for at least one of the values that is returned. If the subquery returns no value, the search condition is FALSE. This keyword is similar to the IN keyword.

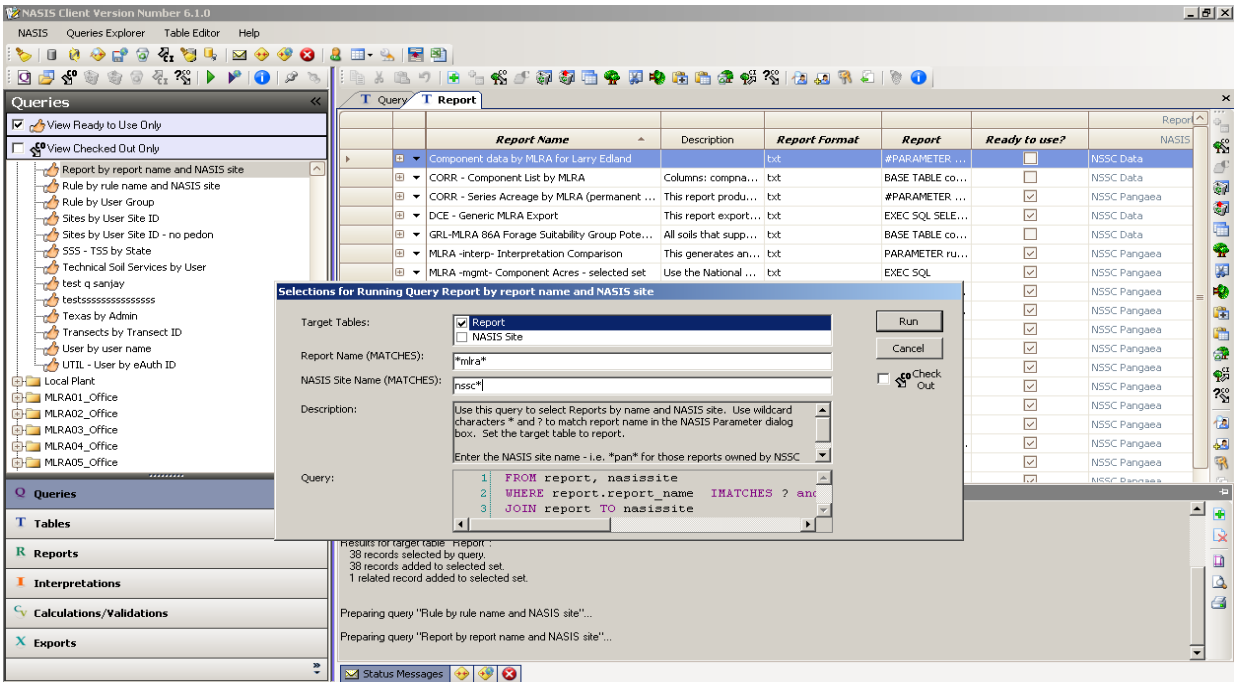
Exercises

1. Load components for a survey where the R horizon depths do not match the Restriction table depths

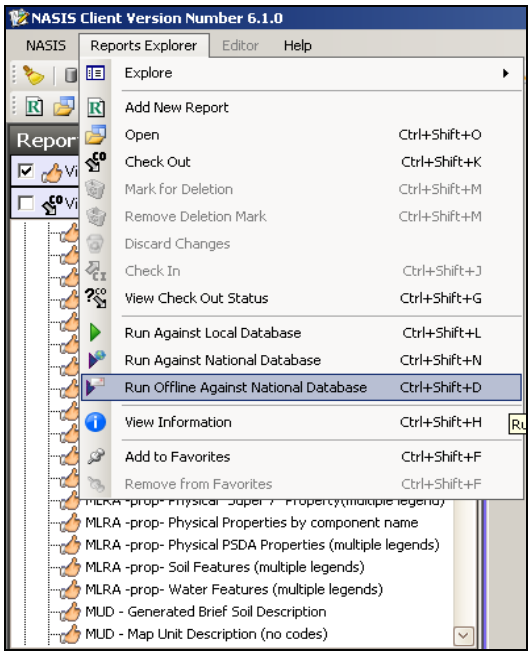
NASIS Reports



Reports can be loaded into the Report “table” allowing the user to query for and manage reports in a table format.



Reports can be ‘Run Against the Local Database’, ‘Run Against the National Database’ or “*Run Offline Against the National Database*”. Using the ‘offline’ option allows the user to run lengthy reports off the server instead of the local machine. The advantage is the process will release the NASIS screen allowing the user to continue working in NASIS while the report is running on the server. When completed, the server will send an email to the user with a link to retrieve the completed report. The metadata reports and CVIR guide are necessary references for report writing. Find these materials on the NASIS web site and keep them available.



NASIS Reports contain 3 major parts:

- QUERY
- DATA MANIPULATION
- OUTPUT

QUERY

```
EXEC SQL
select areaname, legenddesc, liid, lmapunit.seqnum, musym, muname
from legend, lmapunit, mapunit, outer area
where join area to legend and join legend to lmapunit and join lmapunit to mapunit;
SORT BY liid, lmapunit.seqnum, musym SYM.
```

DATA MANIPULATION

```
DEFINE dt               TODAY.
DEFINE legend_name      areaname || ": " || legenddesc.
DEFINE mu_name          muname.
```

OUTPUT

```
PITCH HORIZONTAL 15 VERTICAL 8.  PAGE LENGTH 80.
TEMPLATE basic SEPARATOR ""  AT LEFT FIELD WIDTH 10, FIELD WIDTH 90, "".
TEMPLATE head SEPARATOR ""  AT LEFT FIELD WIDTH 10, FIELD WIDTH 25,"".

HEADER INITIAL
   AT LEFT areaname WIDTH 75;
   AT 80 "Print date:  ", dt WIDTH 10.
   AT LEFT "Soil Map Legend".
END HEADER.

HEADER
   AT LEFT "Soil Map Legend".
END HEADER.

SECTION main
   HEADING
      SKIP 1 LINE.
      USING head
         "Map\nsymbol" ALIGN CENTER, "Soil name" ALIGN RIGHT.
      USING basic.
   DATA
      USING basic
         musym INDENT 1,
         mu_name INDENT -1.
END SECTION.

SECTION WHEN LAST OF liid
   DATA
      USING basic.
      NEW PAGE.
END SECTION.
```

The ‘Query’ section uses the SQL KEYWORDS

SELECT columns
FROM tables
WHERE conditions are met

The query can also contain optional sections, such as the ‘sort’, ‘aggregation’ and some ‘data manipulation’.

The ‘Data Manipulation’ section includes several tools used to transform the data ‘output’. These include the following statements:

CODELABEL/CODENAME,
DEFINE,
ASSIGN,
DERIVE,
INTERPRET, and
PARAMETER.

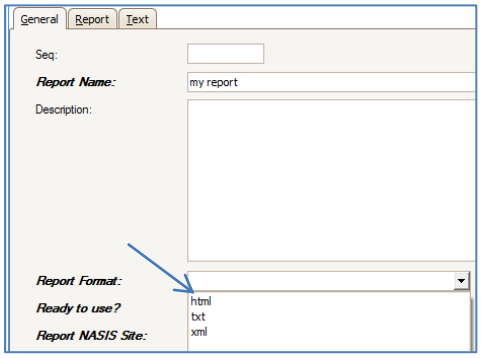
The ‘Output’ section includes the use of ‘text’ or ‘html’ formatting commands. These include the following commands:

INCLUDE/ACCEPT,
TEMPLATE,
PAGE,
MARGIN,
PITCH,
HEADER/FOOTER,
SECTION,
ELEMENT.

Explanations of these statements will be discussed in this document, but detailed information is found in the CVIR guide.

Query

A new NASIS report is created by choosing 'Add New Report' from the Reports Explorer menu or toolbar. A report name and report format is required field (bold and italic). The report format choice list includes: HTML, Text, or XML.



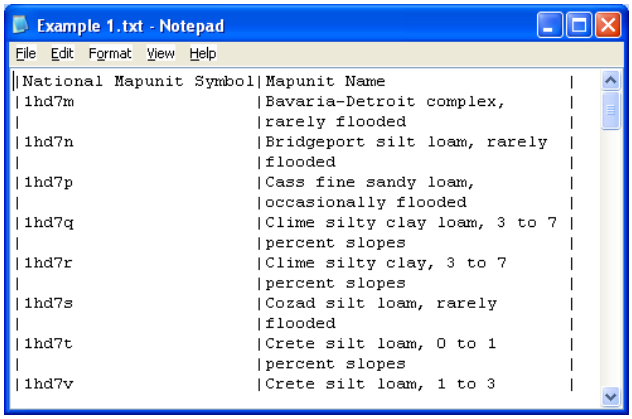
SINGLE TABLE QUERY

Report Script

A NASIS report can be created using the basic assigned report format. Reports use all SQL KEYWORDS: SELECT, FROM, and WHERE. The report script begins with the statement: **EXEC SQL**. This statement informs the NASIS software of an upcoming SQL statement. NASIS has created unique SQL terminology.

```
EXEC SQL
SELECT nationalmusym, muname
FROM mapunit;.
```

The NASIS SQL begins with **EXEC SQL** and ends with both the semicolon **";"** and **"."** period. The SELECT statement identifies the columns to be extracted FROM the identified tables and the WHERE sets the conditions. The report to the right is an example of the 'Text' report format.



This simple SQL, above, will pull two columns from the mapunit table: **nationalmusym** and the **map unit name**. The column labels and column widths are from the data dictionary. This information is found in the 'Table and Column' report found on the NASIS metadata web site or the View Information inside NASIS.

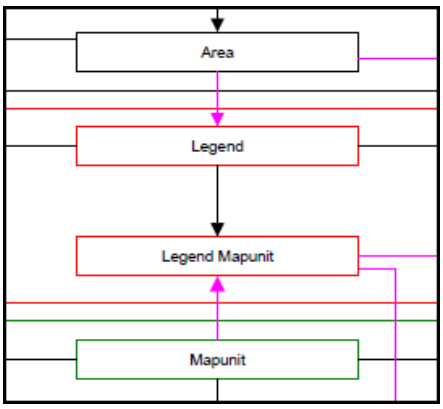
CVIR References

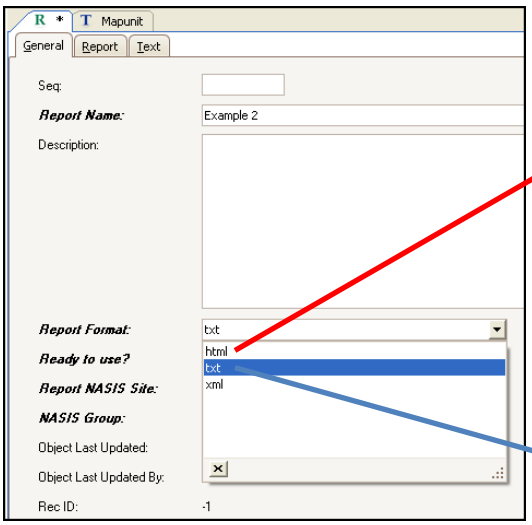
EXEC SQL statement
Table Structure Report

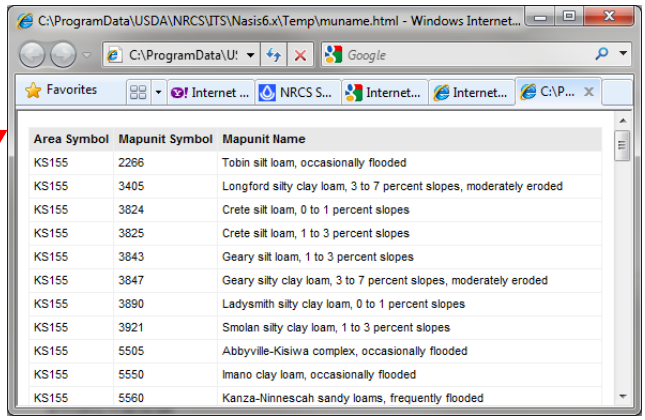
MULTIPLE TABLE QUERY

The SQL can be expanded to include more columns from additional tables. This example contrasts the use of the 'html' and 'text' report format.

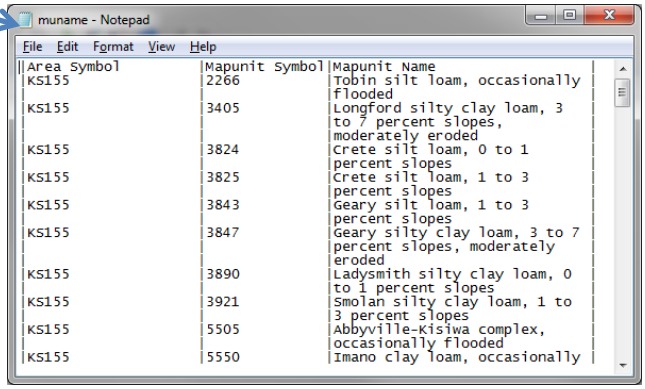
EXEC SQL
SELECT areasymbol, musym, muname
FROM area
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default;
SORT BY musym SYM.







Area Symbol	Mapunit Symbol	Mapunit Name
KS155	2266	Tobin silt loam, occasionally flooded
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded
KS155	3824	Crete silt loam, 0 to 1 percent slopes
KS155	3825	Crete silt loam, 1 to 3 percent slopes
KS155	3843	Geary silt loam, 1 to 3 percent slopes
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded
KS155	5550	Imano clay loam, occasionally flooded
KS155	5560	Kanza-Ninnescah sandy loams, frequently flooded



Area Symbol	Mapunit Symbol	Mapunit Name
KS155	2266	Tobin silt loam, occasionally flooded
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded
KS155	3824	Crete silt loam, 0 to 1 percent slopes
KS155	3825	Crete silt loam, 1 to 3 percent slopes
KS155	3843	Geary silt loam, 1 to 3 percent slopes
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded
KS155	5550	Imano clay loam, occasionally

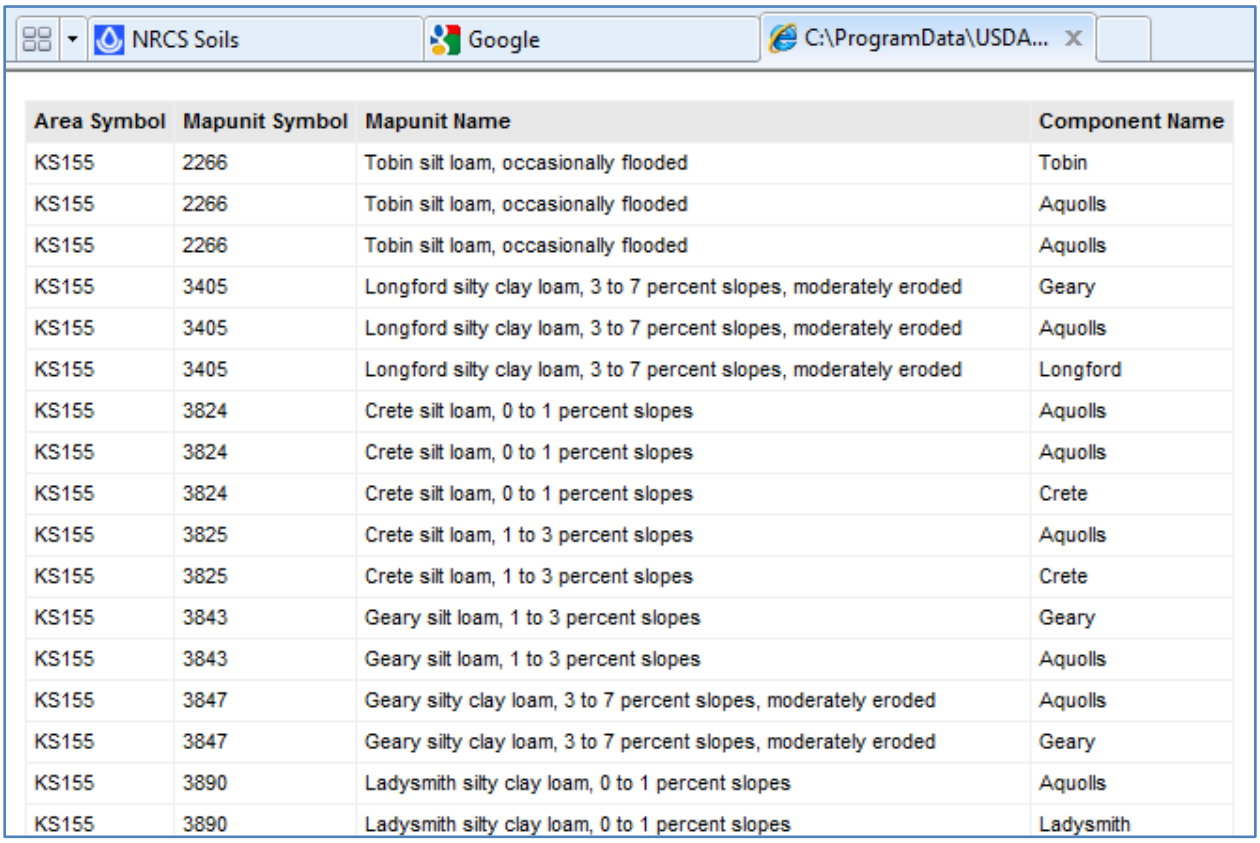
The report script, above, demonstrates a simple join between tables. Note that all tables are required in the FROM clause to connect all tables between the first to last tables. The areasymbol comes from the area table. The legend table is required because it is in the join path to the lmapunit table, where the musym is stored. The mapunit table stores the muname. The JOIN syntax is the same as in NASIS queries. The column physical name and the report column 'label' are found in the 'Tables and Columns' report.

CVIR References

EXEC SQL statement p. 35
Sort Specifications **p. 39**
Data Structure Diagrams

Exercise 1. Creating a Default Format Report

Create a default html format report that identifies the national mapunit symbol, mapunit name, and corresponding component names for mapunits for the legend in your selected set. Sort the report by mapunit symbol. Your report should look similar to the sample given here. Take a close look at your report; how can it be improved?

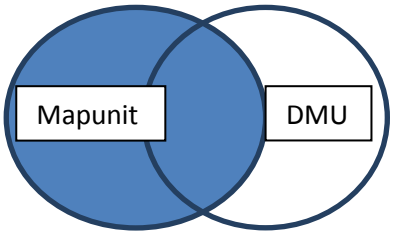


Area Symbol	Mapunit Symbol	Mapunit Name	Component Name
KS155	2266	Tobin silt loam, occasionally flooded	Tobin
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Crete
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Aquolls
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Crete
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Geary
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Aquolls
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Aquolls
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Ladysmith

JOINS

Exercise 1 was designed to create a 'legend' report identifying the national mapunit symbol, mapunit name, and component names. Some mapunits could be omitted because they are not linked to a data mapunit. To ensure that all mapunits appear the OUTER JOIN is needed to specify the return of all map units. The representative datamapunit column is added to verify that only the representative DMU is returned. The SORT on the component percentage column is included to sort the components by descending component percentage. In this script, the OUTER specification will return all mapunits even if no rows are returned from the join between the correlation table and the datamapunit table.

```
EXEC SQL
SELECT areasymbol, musym, muname, compname, comppct_r
FROM area
INNER JOIN  legend by default
INNER JOIN lmapunit by default
INNER JOIN  mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default
WHERE repdmu = 1;
SORT BY musym SYM, comppct_r desc.
```



NRCS Soils | Google | C:\ProgramData\USDA\N... | C:\ProgramData\U

Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	RV
KS155	2266	Tobin silt loam, occasionally flooded	Tobin	100
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	0
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	0
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford	90
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	10
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	0
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Crete	100
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	0
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	0
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Crete	100
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Aquolls	0
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Geary	100
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Aquolls	0
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	100
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	0
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Ladysmith	100
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Aquolls	0

CVIR References

EXEC SQL statement p. 35
Sort Specifications **p. 39**
Table Structure Report

Discussion:
Condition,
OUTER join,
Boolean

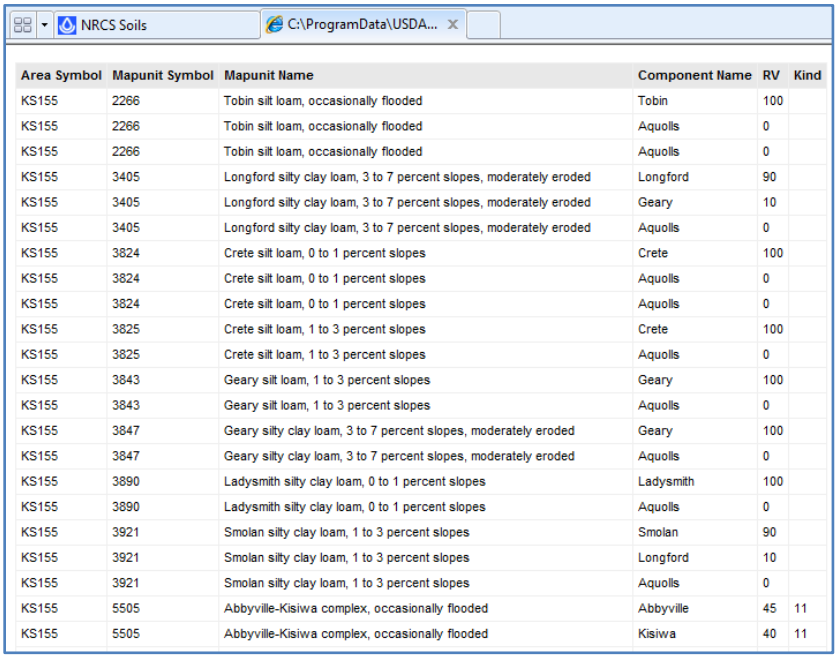
Exercise 2. OUTER Join for Component Restrictions

Using what has been learned, create a component restriction report that includes mapunit symbol, mapunit name, all components, and component restriction, sorted by mapunit symbol and component percent. Refer to the diagrams, 'tables and columns' report and 'data type and comparison' chart.

What tables are needed?

What columns are necessary?

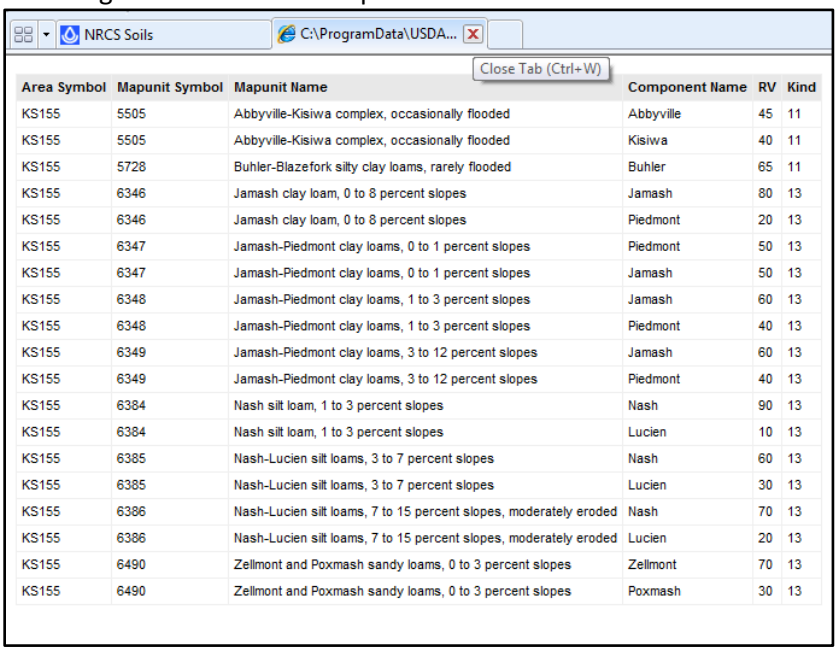
What comparison operators and conditions are needed?



Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	RV	Kind
KS155	2266	Tobin silt loam, occasionally flooded	Tobin	100	
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	0	
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	0	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford	90	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	10	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	0	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Crete	100	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Crete	100	
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Aquolls	0	
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Geary	100	
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Aquolls	0	
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	100	
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	0	
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Ladysmith	100	
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Smolan	90	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Longford	10	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Aquolls	0	
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Abbyville	45	11
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Kisiwa	40	11

In this report the restriction kind is 'uncoded' returning the result as populated in the database however all components appear whether there is a restriction or not.

Run this same report using INNER JOINS to compare the result:

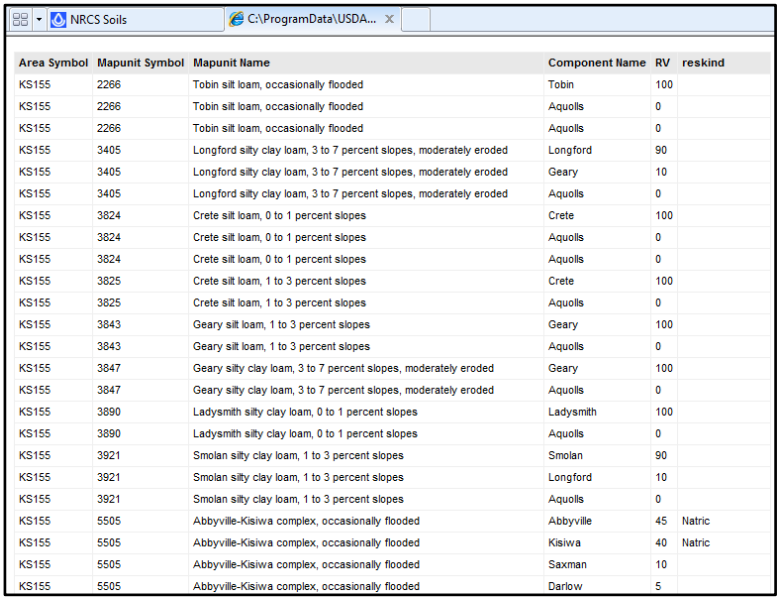


Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	RV	Kind
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Abbyville	45	11
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Kisiwa	40	11
KS155	5728	Buhler-Blazefork silty clay loams, rarely flooded	Buhler	65	11
KS155	6346	Jamash clay loam, 0 to 8 percent slopes	Jamash	80	13
KS155	6346	Jamash clay loam, 0 to 8 percent slopes	Piedmont	20	13
KS155	6347	Jamash-Piedmont clay loams, 0 to 1 percent slopes	Piedmont	50	13
KS155	6347	Jamash-Piedmont clay loams, 0 to 1 percent slopes	Jamash	50	13
KS155	6348	Jamash-Piedmont clay loams, 1 to 3 percent slopes	Jamash	60	13
KS155	6348	Jamash-Piedmont clay loams, 1 to 3 percent slopes	Piedmont	40	13
KS155	6349	Jamash-Piedmont clay loams, 3 to 12 percent slopes	Jamash	60	13
KS155	6349	Jamash-Piedmont clay loams, 3 to 12 percent slopes	Piedmont	40	13
KS155	6384	Nash silt loam, 1 to 3 percent slopes	Nash	90	13
KS155	6384	Nash silt loam, 1 to 3 percent slopes	Lucien	10	13
KS155	6385	Nash-Lucien silt loams, 3 to 7 percent slopes	Nash	60	13
KS155	6385	Nash-Lucien silt loams, 3 to 7 percent slopes	Lucien	30	13
KS155	6386	Nash-Lucien silt loams, 7 to 15 percent slopes, moderately eroded	Nash	70	13
KS155	6386	Nash-Lucien silt loams, 7 to 15 percent slopes, moderately eroded	Lucien	20	13
KS155	6490	Zellmont and Poxmash sandy loams, 0 to 3 percent slopes	Zellmont	70	13
KS155	6490	Zellmont and Poxmash sandy loams, 0 to 3 percent slopes	Poxmash	30	13

CODELABEL and CODENAME

In Exercise 2 the restriction kind is returned 'uncoded'. The data can be converted to the characters identified in the data dictionary using the CODENAME and CODELABEL. Explanation can be found on pages 22-23 in the CVIR guide but on page 36, it explains its use in the SELECT command. The SELECT command can be modified in exercise 2 to code the restriction kind, as follows:

```
EXEC SQL
SELECT areasymbol, musym, muname, compname, comppct_r, CODELABEL(reskind) reskind
FROM area
INNER JOIN  legend by default
INNER JOIN lmapunit by default
INNER JOIN  mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default
LEFT OUTER JOIN corestrictions by default
WHERE repdmu = 1;
SORT BY musym SYM, comppct_r desc.
```



NRCS Soils | C:\ProgramData\USDA...

Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	RV	reskind
KS155	2266	Tobin silt loam, occasionally flooded	Tobin	100	
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	0	
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	0	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford	90	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	10	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	0	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Crete	100	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Crete	100	
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Aquolls	0	
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Geary	100	
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Aquolls	0	
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	100	
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	0	
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Ladysmith	100	
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Smolan	90	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Longford	10	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Aquolls	0	
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Abbyville	45	Natric
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Kisiwa	40	Natric
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Saxman	10	
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Darlow	5	

The use of CODELABEL converts the number to the choice list description with the first letter UPPER CASE. The use of CODENAME is similar, but the name is all lower case. Notice the use requires the CODELABEL to be followed by the field in parentheses and renaming the field. The renaming of the field is referred to as an 'alias' and in this case the same name was used (reskind). The 'alias' can be any word; it could have been set to

'**CODELABEL(reskind) restrictions**'

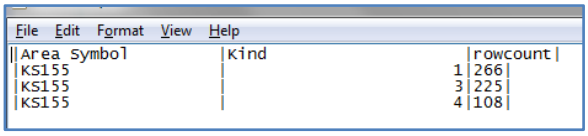
AGGREGATION

The aggregation of the Query can use either the 'Group by' function or the Aggregation clause.

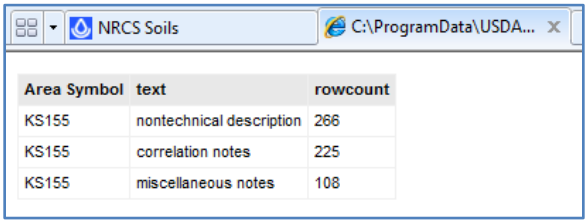
GROUP BY

This SQL uses the GROUP BY statement to aggregate data based on a column. In this Query, the aggregation finds all distinct instances of area symbols and mapunit text kinds then counts each occurrence. The Group By is used by sql to "aggregate" the data before the "aggregation" command 'count'. The count(*) is an SQL command that tells the query to tally up as it progresses using the alias named 'rowcount' (it could have been named anything). The Group By is necessary in the SQL (a new group) and count(*) tells how many are found for each group. The "sum" is not needed in a COLUMN field because sql has already completed the sum. This query uses the REAL command to return to the local database to retrieve data. In the example below, the map unit text kind records are grouped and counted:

```
EXEC SQL
SELECT areasymbol, CODENAME(mapunittextkind) AS text, count(*) as rowcount
FROM REAL area
INNER JOIN REAL legend by default
INNER JOIN REAL lmapunit by default
INNER JOIN REAL mapunit by default
INNER JOIN real mutext by default
WHERE areasymbol matches "KS155"
GROUP BY areasymbol, mapunittextkind;.
```

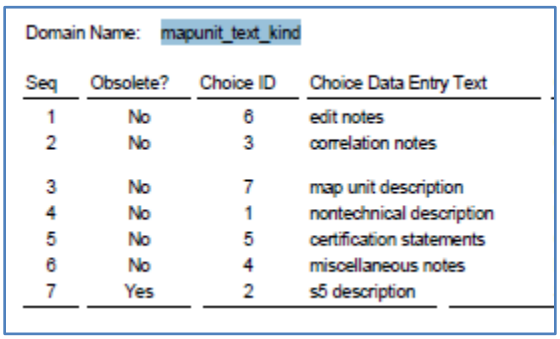


Without the use of CODENAME (text report)



With the use of CODENAME (html report)

There are 3 different choices for text 'Kinds' used in this survey area. There are 266 entries using 'Kind 1' (nontechnical descriptions). The value is deciphered using Choice ID in the Domains report:

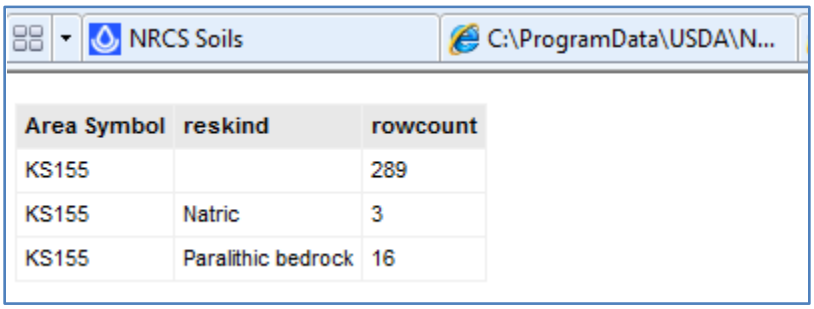


There are 266 nontechnical descriptions, 225 correlation notes, and 108 miscellaneous notes.

The GROUP BY and ORDER BY clauses do not allow column numbers or aliases as they did in Informix. You have to use a column name or expression. For example, GROUP BY areasymbol instead of GROUP BY 1 where '1' is the first column in the SELECT clause.

Exercise 3. Use of Group By

Using the SQL from Exercise 2, modify the query to identify the number and kinds of restrictions found in the selected set. The resulting report should look similar to this report:



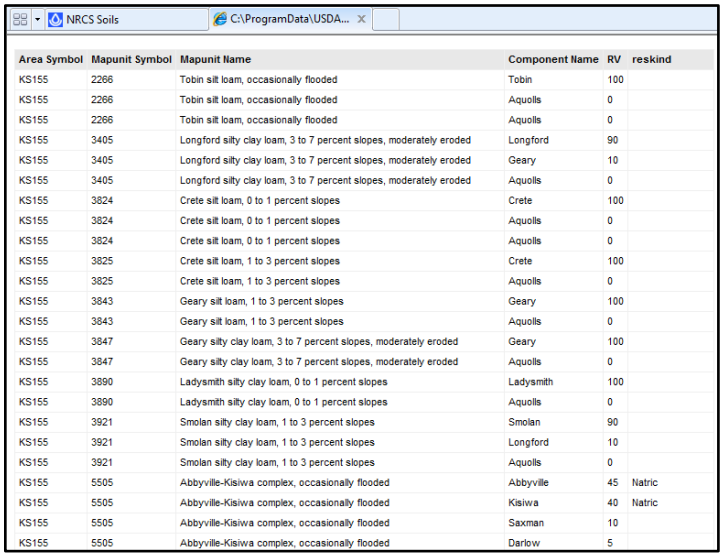
Area Symbol	reskind	rowcount
KS155		289
KS155	Natric	3
KS155	Paralithic bedrock	16

```
EXEC SQL
SELECT areasymbol, CODELABEL(reskind) reskind, count(*) as rowcount
FROM area
INNER JOIN  legend by default
INNER JOIN lmapunit by default
INNER JOIN  mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default
LEFT OUTER JOIN corestrictions by default
WHERE repdmu = 1
GROUP BY areasymbol, reskind;.
```

AGGREGATE

The use of the AGGREGATE function can eliminate duplication of many of the columns of data found in Exercise 2. Aggregation can be used to clean the report by removing the duplication of data in specified columns. To explain the AGGREGATE function, first review a set of data without aggregation. The following query produces this report:

```
EXEC SQL
SELECT areasymbol, musym, muname,
compname, comppct_r,
CODELABEL(reskind) reskind
FROM area
INNER JOIN  legend by default
INNER JOIN lmapunit by default
INNER JOIN  mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default
LEFT OUTER JOIN corestrictions by
default
WHERE repdmu = 1;
SORT BY musym SYM, comppct_r desc.
```



NRCS Soils | C:\ProgramData\USDA...

Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	RV	reskind
KS155	2266	Tobin silt loam, occasionally flooded	Tobin	100	
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	0	
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	0	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford	90	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	10	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	0	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Crete	100	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Crete	100	
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Aquolls	0	
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Geary	100	
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Aquolls	0	
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	100	
KS155	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	0	
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Ladysmith	100	
KS155	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	Aquolls	0	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Smolan	90	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Longford	10	
KS155	3921	Smolan silty clay loam, 1 to 3 percent slopes	Aquolls	0	
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Abbyville	45	Natric
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Kisiwa	40	Natric
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Saxman	10	
KS155	5505	Abbyville-Kisiwa complex, occasionally flooded	Darlow	5	

Notice the repetition of the area symbol, mapunit symbols, map unit names, etc. Each row has repeated information in this report.

Adding AGGREGATE to the SQL will remove the duplication of data in specific rows and make the data easier to read:

```
EXEC SQL
SELECT areasymbol, musym, muname, compname, comppct_r, CODELABEL(reskind) reskind
FROM area
INNER JOIN  legend by default
INNER JOIN lmapunit by default
INNER JOIN  mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default
LEFT OUTER JOIN corestrictions by
default
WHERE repdmu = 1;
SORT BY musym SYM, comppct_r desc
AGGREGATE ROWS BY areasymbol, musym, muname COLUMN compname NONE, comppct_r NONE.
```

Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	RV	reskind
KS155	2266	Tobin silt loam, occasionally flooded	Tobin	100	
			Aquolls	0	
			Aquolls	0	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford	90	
			Geary	10	
			Aquolls	0	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Crete	100	
			Aquolls	0	
			Aquolls	0	
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Crete	100	
			Aquolls	0	
KS155	3843	Geary silt loam, 1 to 3 percent slopes	Geary	100	
			Aquolls	0	

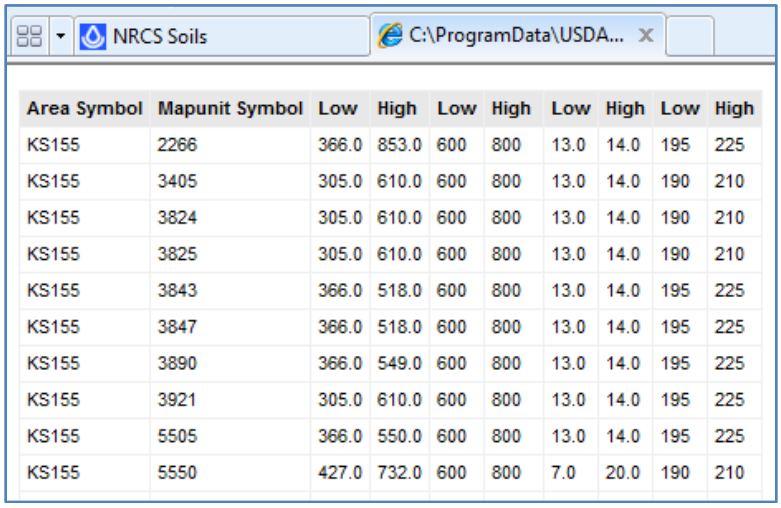
The aggregate specification is shown in **bold**. In this script, rows that have the same areasymbol, musym, and muname will be grouped together to form a single iteration for the report. Typically, aggregation should be applied to internal 'key' fields (liid, lmapunitiid, muiid, coiid, etc). The column aggregation "NONE" is specified allowing duplicates of the component name (see 2266 Aquolls). A default aggregation of UNIQUE could be applied to remove duplicate data from specified columns.

CVIR References

EXEC SQL statement p. 35
Sort Specifications **p. 39**
Aggregation Specifications p. 40

Exercise AGGREGATION of columns

After reading the information in CVIR on page 40, use the Aggregate clause to identify the climatic values (elev_l MIN, elev_h MAX, map_l MIN, map_h MAX, airtempa_l MIN, airtempa_h MAX, ffd_l MIN, ffd_h MAX.) by map unit symbols.



Area Symbol	Mapunit Symbol	Low	High	Low	High	Low	High	Low	High
KS155	2266	366.0	853.0	600	800	13.0	14.0	195	225
KS155	3405	305.0	610.0	600	800	13.0	14.0	190	210
KS155	3824	305.0	610.0	600	800	13.0	14.0	190	210
KS155	3825	305.0	610.0	600	800	13.0	14.0	190	210
KS155	3843	366.0	518.0	600	800	13.0	14.0	195	225
KS155	3847	366.0	518.0	600	800	13.0	14.0	195	225
KS155	3890	366.0	549.0	600	800	13.0	14.0	195	225
KS155	3921	305.0	610.0	600	800	13.0	14.0	195	225
KS155	5505	366.0	550.0	600	800	13.0	14.0	195	225
KS155	5550	427.0	732.0	600	800	7.0	20.0	190	210

EXEC SQL
SELECT areasymbol, musym, elev_l, elev_h, map_l, map_h, airtempa_l, airtempa_h, ffd_l, ffd_h
FROM area
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default
SORT BY areasymbol, musym
AGGREGATE ROWS BY areasymbol, musym COLUMN elev_l MIN, elev_h MAX, map_l MIN, map_h MAX, airtempa_l MIN, airtempa_h MAX, ffd_l MIN, ffd_h MAX.

Notice the column headers use the default column label as found in the Tables and Columns report.

Aggregation Exercise

Use what you learned in aggregating data from previous examples to create a report sums the map unit acres for farmland classifications within a given survey area. The area symbol, “class”, and summed map unit acres are the columns.

Sample Report Output

Your report parameter should look similar to the sample given here.

Area Symbol	farmlndcl	Total Acres
KS155	All areas are prime farmland	397,326
KS155	Farmland of statewide importance	287,819
KS155	Not prime farmland	129,031

```
EXEC SQL
SELECT areasymbol, CODELABEL(farmlndcl) farmlndcl, muacres
FROM area
INNER JOIN  legend by default
INNER JOIN lmapunit by default
INNER JOIN  mapunit by default;
SORT BY farmlndcl
AGGREGATE ROWS BY areasymbol, farmlndcl COLUMN muacres SUM.
```

ARITHMETRIC FUNCTIONS

The SELECT clause has the ability to use arithmetic functions (+, -, *, /) to validate or modify the data. Typical algebraic rules must be followed.

An example is the calculation of component acres using the map unit acres and the component percentage. Notice that the compacres field is created.

```
EXEC SQL
SELECT areasymbol, musym, muname, compname,
muacres, comppct_r, ((muacres*comppct_r)/100) as compacres
FROM area
INNER JOIN  legend by default
INNER JOIN lmapunit by default
INNER JOIN  mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default
WHERE repdmu = 1;
SORT BY musym SYM, comppct_r desc.
```

Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	Total Acres	RV	compacres
KS155	2266	Tobin silt loam, occasionally flooded	Tobin	513	100	513
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	513	0	0
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	513	0	0
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford	1,276	90	1148
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	1,276	10	127
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	1,276	0	0
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Crete	4,225	100	4225
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	4,225	0	0
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Aquolls	4,225	0	0
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Crete	5,237	100	5237
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Aquolls	5,237	0	0

Notice the 'RV' column is the default column label. Using arithmetic functions the column label can be converted with an alias:

Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	Total Acres	comppct_r	compacres
KS155	2266	Tobin silt loam, occasionally flooded	Tobin	513	100	513
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	513	0	0
KS155	2266	Tobin silt loam, occasionally flooded	Aquolls	513	0	0
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford	1,276	90	1148
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Geary	1,276	10	127
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Aquolls	1,276	0	0

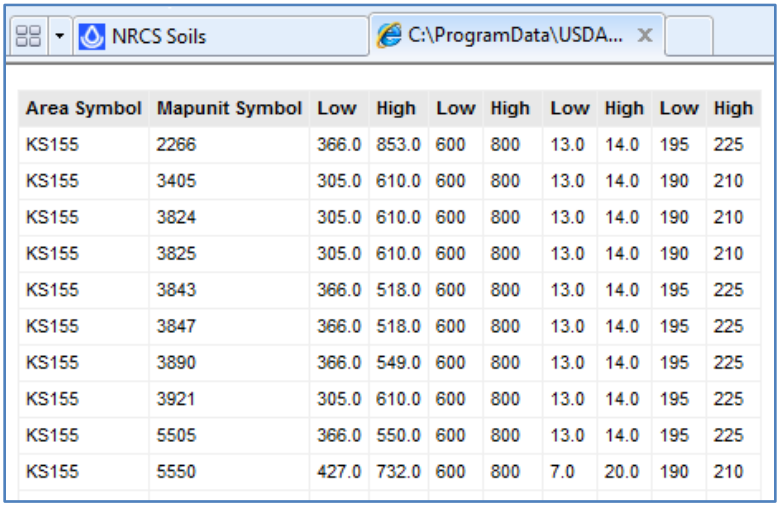
```
EXEC SQL
SELECT areasymbol, musym, muname, compname, muacres, comppct_r/1 as comppct_r,
((muacres*comppct_r)/100) as compacres
FROM area
```

By dividing the comppct_r by 1, it does not change the value, however it does allow for the alias column label to be assigned.

Another example is using the aggregation from the previous exercise.

EXEC SQL
SELECT areasymbol, musym, elev_l*1 as Elev_min_Low, elev_h*1 as Elev_max_High, map_l *1 as MAP_min_L, map_l * 1 as MAP_max_h, airtempa_l*1 as MAAT_min_l, airtempa_h * 1 as MAAT_max_h, ffd_l*1 as FFD_min_L, ffd_h*1 as FFD_max_h
FROM area
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default;
SORT BY areasymbol, musym
AGGREGATE ROWS BY areasymbol, musym COLUMN Elev_min_Low MIN, Elev_max_High MAX, MAP_min_L MIN, MAP_max_h MAX,
MAAT_min_l MIN, MAAT_max_h MAX, FFD_min_L MIN, FFD_max_h MAX.

From this:



Area Symbol	Mapunit Symbol	Low	High	Low	High	Low	High	Low	High
KS155	2266	366.0	853.0	600	800	13.0	14.0	195	225
KS155	3405	305.0	610.0	600	800	13.0	14.0	190	210
KS155	3824	305.0	610.0	600	800	13.0	14.0	190	210
KS155	3825	305.0	610.0	600	800	13.0	14.0	190	210
KS155	3843	366.0	518.0	600	800	13.0	14.0	195	225
KS155	3847	366.0	518.0	600	800	13.0	14.0	195	225
KS155	3890	366.0	549.0	600	800	13.0	14.0	195	225
KS155	3921	305.0	610.0	600	800	13.0	14.0	195	225
KS155	5505	366.0	550.0	600	800	13.0	14.0	195	225
KS155	5550	427.0	732.0	600	800	7.0	20.0	190	210

To this:

Area Symbol	Mapunit Symbol	Elev_min_Low	Elev_max_High	MAP_min_L	MAP_max_h	MAAT_min_l	MAAT_max_h	FFD_min_L	FFD_max_h
KS155	2266	366	853	600	600	13	14	195	225
KS155	3405	305	610	600	600	13	14	190	210
KS155	3824	305	610	600	600	13	14	190	210
KS155	3825	305	610	600	600	13	14	190	210
KS155	3843	366	518	600	600	13	14	195	225
KS155	3847	366	518	600	600	13	14	195	225
KS155	3890	366	549	600	600	13	14	195	225
KS155	3921	305	610	600	600	13	14	195	225

Notice that using the arithmetic functions, the column labels can be over ridden to create more meaningful column headers.

The arithmetic function "+" can also be used in a SQL to concatenate string values. For example, the component name and local phase can be concatenated into a single field using the plus. In addition, the term 'case' is used to validate the NULL possibility of the local phase field. Using the simple IF, THEN:ELSE statement the test for NULL local phase can be used first to identify the presence of a local phase term and if it is null then present the component name else put the two together with a comma and space in between:

```
EXEC SQL
SELECT areasymbol, musym, muname,
case when localphase is null then compname else compname + ', ' + localphase end as
compphase, muacres, comppct_r, ((muacres*comppct_r)/100) as compacres
FROM area
INNER JOIN  legend by default
INNER JOIN lmapunit by default
INNER JOIN  mapunit by default
LEFT OUTER JOIN correlation by default
LEFT OUTER JOIN datamapunit by default
LEFT OUTER JOIN component by default
WHERE repdmu = 1;
SORT BY musym SYM, comppct_r desc.
```

Notice the 'compphase' field contains a concatenation of the component and the local phase for the Orthents components.

Area Symbol	Mapunit Symbol	Mapunit Name	compphase	Total Acres	RV	compacres
IL075	91A	Swygert silty clay loam, 0 to 2 percent slopes	Swygert	19,290	92	17746
IL075	91A	Swygert silty clay loam, 0 to 2 percent slopes	Bryce	19,290	2	385
IL075	91A	Swygert silty clay loam, 0 to 2 percent slopes	Orthents, clayey, nearly level	19,290	2	385
IL075	91A	Swygert silty clay loam, 0 to 2 percent slopes	Urban land	19,290	2	385
IL075	91A	Swygert silty clay loam, 0 to 2 percent slopes	Streator	19,290	2	385
IL075	145B2	Saybrook silt loam, 2 to 5 percent slopes, eroded	Saybrook	3,633	90	3269
IL075	145B2	Saybrook silt loam, 2 to 5 percent slopes, eroded	Drummer	3,633	3	108
IL075	145B2	Saybrook silt loam, 2 to 5 percent slopes, eroded	Elpaso	3,633	3	108
IL075	145B2	Saybrook silt loam, 2 to 5 percent slopes, eroded	Harpster	3,633	3	108
IL075	192A	Del Rey silt loam, 0 to 2 percent slopes	Del Rey	8,625	92	7935
IL075	192A	Del Rey silt loam, 0 to 2 percent slopes	Urban land	8,625	2	172
IL075	192A	Del Rey silt loam, 0 to 2 percent slopes	Milford	8,625	2	172
IL075	192A	Del Rey silt loam, 0 to 2 percent slopes	Orthents, clayey	8,625	2	172
IL075	192A	Del Rey silt loam, 0 to 2 percent slopes	Montgomery	8,625	2	172

REAL TABLES

The use of the term REAL in front of the table name will link the table in the local database instead of the selected set. When the query is run against the national database the term REAL is ignored. "REAL" needs to be on all the tables that are in your local database by default (e.g. area, areatype, geomorphic tables, etc.).

Report Script

```
EXEC SQL
SELECT areasymbol, mapunittextkind, 1 as rowcount
FROM REAL area
INNER JOIN REAL legend by default
INNER JOIN REAL lmapunit by default
INNER JOIN REAL mapunit by default
INNER JOIN REAL mutext by default
WHERE  areasymbol matches "KS155";
SORT BY areasymbol sym, mapunittextkind
AGGREGATE ROWS mapunittextkind COLUMN rowcount sum.
```

SUBQuery

This report introduces the use of a subquery and how it can be used to specify results based on a second set of criteria. The values returned for the unified classification is uncoded, and CODELABEL/CODENAME could be used to code the result. In this example the report is used to verify the list of unified classifications for each horizon that is specifically a USDA texture of 'FSL or VFSL'.

```
EXEC SQL
SELECT compname, hzname, hzdept_r, hzdepb_r, unifiedcl
FROM component
INNER JOIN chorizon by default
INNER JOIN chunified by default
WHERE chiid IN (select chiidref from chtexturegrp where texture in ("FSL", "VFSL"));
sort by compname, hzdept_r
AGGREGATE rows compname, hzdept_r column unifiedcl list.
```

Report Script – alternative EXISTS

```
EXEC SQL
SELECT compname, hzname, hzdept_r, hzdepb_r, unifiedcl
FROM component
INNER JOIN chorizon by default
INNER JOIN chunified by default
WHERE EXISTS (select * from chtexturegrp where join chorizon to chtexturegrp and texture in
("FSL", "VFSL"));
SORT by compname, hzdept_r
AGGREGATE rows by compname, hzdept_r.
```

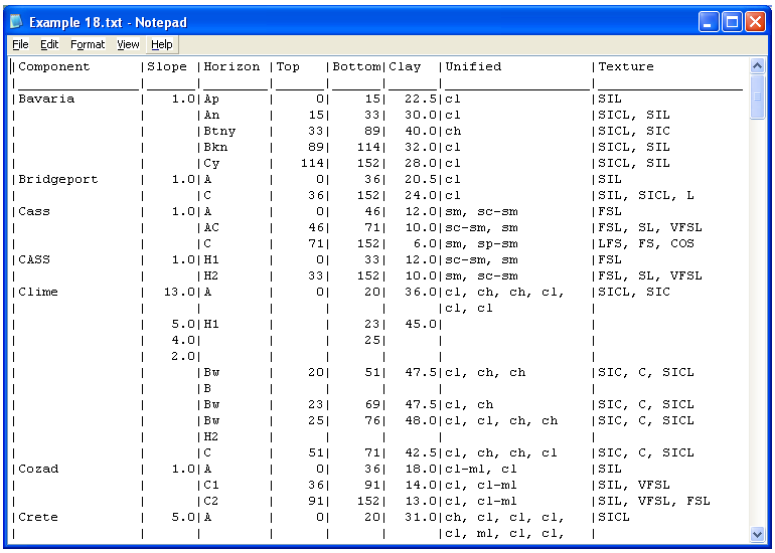
Component Name	Designation	RV	RV	Unified
Abbyville	A	0	20	23, 5
Aquents	2Bg	7	20	23
Attica	Ap	0	20	24
Attica	BA	20	38	24, 24
Attica	Bt1	38	76	19, 5, 1, 19, 5, 24, 1, 24
Attica	Bt2	76	127	19, 24, 5, 1, 1, 19, 5, 24
Attica	Bt3	127	165	1, 19, 1, 24, 5, 24, 19, 5
Attica	BC	165	203	24, 24, 5, 5
Buhler	2C1	148	192	23, 5, 13, 1

JOINING MULTIPLE SQL STATEMENTS

Multiple SQL statements can be used in a report. Each SQL can be used to extract data for specific uses in the report. A few rules for multiple SQL reports:

1. The report must use the 'BASE TABLE' command. This command identifies the one table that links all SQL statements.
2. Parameterized queries can be used in place of the BASE TABLE.

This report introduces the use of multiple queries used to compare UNIFIED and USDA textures data by horizon. The first query combines unified classes into a list using the aggregate command. Note the use of LIST to create a comma delimited list of the unified and the USDA textures. Note the use of codename in the SELECT clause to decode UNIFIED instead of using a DEFINE statement which would come after the list command, therefore, not working as part of the aggregation to create a list. The second query combines the USDA texture classes into a list, using the AGGREGATE and list, for the COLUMN. Notice that the use of AGGREGATION ROWS BY is limited to the first query, subsequent SQL statements can only use AGGREGATE COLUMNS.



Component	Slope	Horizon	Top	Bottom	Clay	Unified	Texture
Bavaria	1.0	Ap	0	15	22.5	cl	SIL
		An	15	33	30.0	cl	SICL, SIL
		Btny	33	89	40.0	ch	SICL, SIC
		Bkn	89	114	32.0	cl	SICL, SIL
		Cy	114	152	28.0	cl	SICL, SIL
Bridgeport	1.0	A	0	36	20.5	cl	SIL
		C	36	152	24.0	cl	SIL, SICL, L
Cass	1.0	A	0	46	12.0	sm, sc-sm	FSL
		AC	46	71	10.0	sc-sm, sm	FSL, SL, VFSL
		C	71	152	6.0	sm, sp-sm	LFS, FS, COS
CASS	1.0	H1	0	33	12.0	sc-sm, sm	FSL
		H2	33	152	10.0	sm, sc-sm	FSL, SL, VFSL
Clime	13.0	A	0	20	36.0	cl, ch, ch, cl, cl, cl	SICL, SIC
	5.0	H1		23	45.0		
	4.0			25			
	2.0						
		Bw	20	51	47.5	cl, ch, ch	SIC, C, SICL
		B					
		Bw	23	69	47.5	cl, ch	SIC, C, SICL
		Bw	25	76	48.0	cl, cl, ch, ch	SIC, C, SICL
		H2					
		C	51	71	42.5	cl, ch, ch, cl	SIC, C, SICL
Cozad	1.0	A	0	36	18.0	cl-ml, cl	SIL
		C1	36	91	14.0	cl, cl-ml	SIL, VFSL
		C2	91	152	13.0	cl, cl-ml	SIL, VFSL, FSL
Crete	5.0	A	0	20	31.0	ch, cl, cl, cl, cl, ml, cl, cl,	SICL

BASE TABLE chorizon.

EXEC SQL
SELECT compname, slope_r, hzname, hzdept_r, hzdepb_r, claytotal_r, codename(unifiedcl) unifiedcl
FROM component
INNER JOIN chorizon by default
INNER JOIN chunified by default;
SORT by compname, hzdept_r
AGGREGATE rows compname, hzdept_r column unifiedcl list.

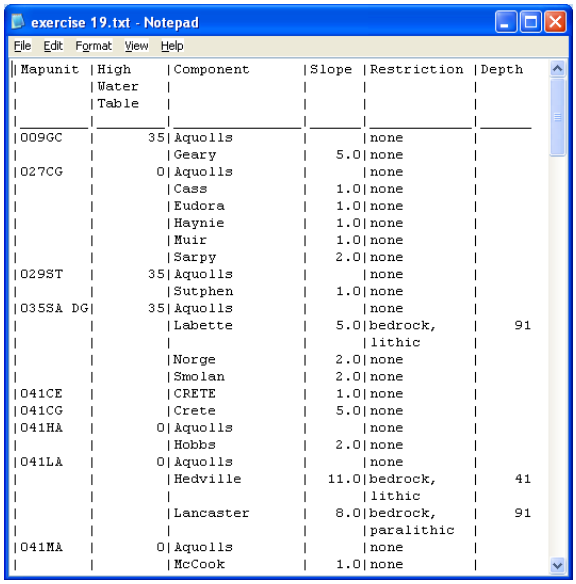
Second query combines texture classes into a list.

EXEC SQL
SELECT texture
FROM chorizon
INNER JOIN chtexturegrp by default;
AGGREGATE column texture list.

PARAMETERIZED QUERY

This report introduces the use of multiple queries used to compare results. This example uses parameterized queries to link to differing sets of data for comparison. A Parametric query is a special instance of joining two SQL statements using the "IN" or "=" operator. Using = is always faster than IN. Use IN if there is a possibility of multiple values for muiid. Assuming muiid is the output of your initial query, it would have to be aggregated in such a way that multiple values of muiid could occur. This will join one field in a secondary query to a corresponding value in the main query.

This report presents the High Water Table for a component along with the Restriction and its depth. Two different database paths require special treatment. A BASE TABLE is not set in a parameterized query.



Mapunit	High Water Table	Component	Slope	Restriction	Depth
O09GC	35	Aquolls		none	
		Geary	5.0	none	
O27CG	0	Aquolls		none	
		Cass	1.0	none	
		Eudora	1.0	none	
		Haynie	1.0	none	
		Muir	1.0	none	
		Sarpy	2.0	none	
O29ST	35	Aquolls		none	
		Sutphen	1.0	none	
O35SA DG	35	Aquolls		none	
		Labette	5.0	bedrock, lithic	91
		Norge	2.0	none	
		Smolan	2.0	none	
O41CE		CRETE	1.0	none	
O41CG		Crete	5.0	none	
O41HA	0	Aquolls		none	
		Hobbs	2.0	none	
O41LA	0	Aquolls		none	
		Hedville	11.0	bedrock, lithic	41
		Lancaster	8.0	bedrock, paralithic	91
O41MA	0	Aquolls		none	
		McCook	1.0	none	

Report Script using "=" (single result)

First query finds the depth to first restriction for each component.
Outer join is used in case a component has no restrictive features.

EXEC SQL
SELECT dmudesc, dmuiid, compname, slope_r, reskind, resdept_r
FROM datamapunit
INNER JOIN component by default
LEFT OUTER JOIN corestrictions by default;
sort by dmudesc, compname, resdept_r
ARREGATE ROWS dmudesc, compname COLUMN reskind first, resdept_r first.

Second query finds the highest water table for the component.

EXEC SQL
SELECT soimoistdept_r
FROM component
INNER JOIN comonth by default

```
INNER JOIN cosoilmoist by default
WHERE component.dmuiidref = $dmuiid and soimoiststat="wet";
AGGREGATE column soimoistdept_r min.
```

Report Script using "IN" (possible multiple results)

```
BASE TABLE mapunit.
EXEC SQL
SELECT liid, lmapunitiid, musym, muiid, muname, muacres, areasymbol, group_name
FROM areatype
INNER JOIN area by default
INNER JOIN legend by default
INNER JOIN lmapunit by default
INNER JOIN mapunit by default
INNER JOIN nasis_group by default
WHERE
 areatypename imatches "non-mlra soil survey area" and
 mustatus = "correlated" AND  legendsuituse = "current wherever mapped";
SORT BY muname, group_name, lmapunitiid, muacres.

EXEC SQL
SELECT  muname mname, nationalmusym, dmuiid
FROM  mapunit
INNER JOIN correlation by default
INNER JOIN datamapunit by default
WHERE repdmu = 1 and
muiid in ($muiid);
SORT BY mname, dmuiid.
```

Joining sub-query tables to main query

FROM area
WHERE areaiid=(select areaiidref FROM legend WHERE JOIN legend to area)

Join two sets of data from different sub-queries

Use the "UNION ALL" Between queries. This will select the mapunit symbols from the two different legends and make them into one list.

EXEC SQL
SELECT musym
From area
Where areasymbol IN (MO207);.

UNION ALL

SELECT musym
From area
Where areasymbol IN (MO103);.

Complex join to subquery in the "From" clause

```
SELECT DISTINCT   musym, compname
FROM area
INNER JOIN legend l ON l.areasymbol = area.areasymbol
INNER JOIN lmapunit lmu ON lmu.liidref = l.liid
INNER JOIN mapunit mu ON mu.muiid=lmu.muiidref
INNER JOIN correlation corr by default
INNER JOIN datamapunit by default
INNER JOIN  component co ON co.dmuiidref = corr.dmuiidref
INNER JOIN ( SELECT  sandtotal_r, silttotal_r, claytotal_r , texture FROM chorizon ch
LEFT OUTER JOIN chtexturegrp tex ON ch.chiid = tex.chiidref)
```

Query Exercises

Create a basic html report that provides the list of all components within the selected set presenting the area symbol, mapunit symbol, map unit name, component name and parent material group. The reports should appear similar to this example:

Area Symbol	Mapunit Symbol	Mapunit Name	Component Name	RV	Parent Material Group Name
KS155	2266	Tobin silt loam, occasionally flooded	Tobin	100	silty alluvium
			Aquolls	0	
			Aquolls	0	
KS155	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	Longford	90	silty alluvium or loess
			Geary	10	loess
			Aquolls	0	
KS155	3824	Crete silt loam, 0 to 1 percent slopes	Crete	100	silty and clayey loess
			Aquolls	0	
			Aquolls	0	
KS155	3825	Crete silt loam, 1 to 3 percent slopes	Crete	100	silty and clayey loess
			Aquolls	0	

Data Manipulation

The Data Manipulation phase of the report allows for data pulled from the SQL to be transformed to another value or class. The Data Manipulation section requires the use of the OUTPUT section in a report.

DEFINE

CVIR page 22 contains a full explanation. The DEFINE statement is the basic method of data manipulation. The DEFINE statement can be used to uncode coded data elements, concatenate two variables, transform NULL values, convert variables using arithmetic values, provide classes and manipulate decimals. Examples include:

CODENAME and CODELABEL

These terms can also be used in the DEFINE statements to return the code name for the coded value from the data dictionary. CODENAME codes using lower case, whereas CODELABEL returns mixed case codes.

```
DEFINE textkind    CODENAME(mapunittextkind).
DEFINE drncl       CODELABEL(drainagecl).
```

CONCATENATING OF FIELDS

The use of the double pipe "||" allows for concatenating of fields such as the Irrigated capability class and subclass.

```
DEFINE ilc                CODELABEL(irrcapcl) || CODELABEL(irrcapscl).
DEFINE nilc               CODELABEL(nirrcapcl) || CODELABEL(nirrcapscl).
```

Concatenating two fields with a space in between:

```
DEFINE symname        musym || "--" || muname.
```

Or concatenate a text string with a field:

```
DEFINE compsim        "Description of " || compname.
```

ARITHMETRIC FUNCTIONS

Arithmetic functions can be used to transform data fields, such as converting metric to English:

```
DEFINE mat_l    (airtempa_l*9/5)+32.
```

Or identifying the thickness of of a horizon by a property:

```
DEFINE rv       (hzdepb_r - hzdept_r) * om_r.
```

NULL TRANSFORMATION

Some data fields can be NULL within the database and a decision must be made to transform the data for use. The DEFINE uses the IF, THEN, ELSE to test for NULL values. (If the unified field is null, then assign 'NULL VALUE' else code label the unified class.

```
DEFINE un1      ISNULL(unifiedcl) ? "NULL VALUE": CODELABEL(unifiedcl).
```

SPRINTF commands

The C programming 'sprintf' commands (%s for character, %f for numeric) can be used to transform data (CVIR page 26). For instance, using the sprintf to assign the clay range and assign the number of floating fields each element will be assigned – 3 places for claytotal_l and 2 places for claytotal_h.

```
DEFINE clay          ISNULL(claytotal_l) OR
                     ISNULL(claytotal_h) ? "  ---" :
                     sprintf("%3.f-%-2.f",claytotal_l,claytotal_h).
```

Or assigning a fixed number of decimals as in awc will be 4 places with 2 decimal places:

```
DEFINE awc           ISNULL(awc_l) OR ISNULL(awc_h) ? "   ---" :sprintf("%4.2f-%-4.2f",awc_l,awc_h).
```

Or assigning a value to a character field to test for a null texture field and if not null then define the character field:

```
DEFINE tex2      ISNULL(tex) ? "-" : sprintf("%s",tex).
```

ASSIGN

The ASSIGN statement recalculates the value of a variable that was defined in a previous DEFINE, DERIVE, or EXEC SQL statement. No alias is used with an ASSIGN, the transformation literally replaces the previous data for the specifical named field.

For example, the compkind is listed in the SELECT command. The CODENAME for this field could be been created in the SELECT command, or as a DEFINE with an alias name, or as an ASSIGN command in which no alias is necessary.

ASSIGN compkind CODENAME(compkind).

To reassign the map value from metric to English:

ASSIGN map_l map_l/25.4.

To use with the ROUND versus the sprint command:

ASSIGN elev_l ROUND(elev_l * 3.28, -1).

To test for NULL:

ASSIGN fraggt10_l if isnull(fraggt10_l) then 0 else fraggt10_l.

In all instances, the original variable is recalculated but maintains the original variable name.

DERIVE

The DERIVE requires the use of BASE TABLE and calls a NASIS interpretation *property* to compute available water capacity for each component. The BASE TABLE in the report must match that one identified in the Property. Note that table.column structure for the component.seqnum is included in the SORT clause to specify the specific seqnum column to be used. If sequence number is populated, it will override sorting by component percent.

BASE TABLE component.

EXEC SQL
SELECT nationalmusym, compname, component.seqnum, comppct_r
FROM mapunit
INNER JOIN correlation by default
INNER JOIN datamapunit by default
INNER JOIN comonent by default
WHERE repdmu = 1;
SORT BY nationalmusym SYMBOL, component.seqnum, comppct_r DESC, compname.

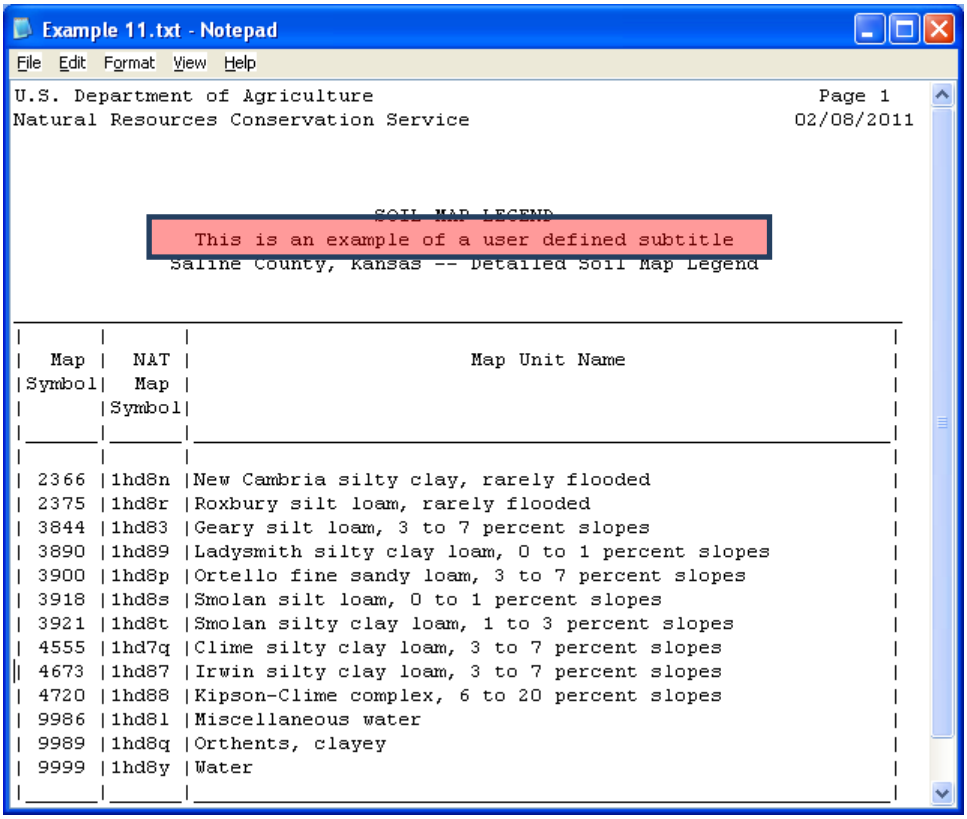
DERIVE awc FROM rv USING "NSSC Pangaea":"AVAILABLE WATER CAPACITY".

CVIR References

BASE TABLE statement p.11
EXEC SQL statement p. 35
SORT BY Specifications **p. 39**
DERIVE statement p. 34

PARAMETER

Parameter statements are explained on page 54 of the CVIR guide. This report uses the Parameter statement creating a user defined title. The format used in this report is similar to standard NASIS reports. There are many examples of the PARAMETER statement. The example text formatted report:



Example 11.txt - Notepad

File Edit Format View Help

```
U.S. Department of Agriculture                                  Page 1
Natural Resources Conservation Service                          02/08/2011


                          SOIL MAP LEGEND
          This is an example of a user defined subtitle
          Saline County, Kansas -- Detailed Soil Map Legend

_____________________________________________________________________
|       |       |                                                   |
|   Map |   NAT |                        Map Unit Name              |
|Symbol|   Map |                                                   |
|       |Symbol|                                                   |
|_______|_______|___________________________________________________|
|       |       |                                                   |
| 2366  |1hd8n  |New Cambria silty clay, rarely flooded             |
| 2375  |1hd8r  |Roxbury silt loam, rarely flooded                  |
| 3844  |1hd83  |Geary silt loam, 3 to 7 percent slopes             |
| 3890  |1hd89  |Ladysmith silty clay loam, 0 to 1 percent slopes   |
| 3900  |1hd8p  |Ortello fine sandy loam, 3 to 7 percent slopes     |
| 3918  |1hd8s  |Smolan silt loam, 0 to 1 percent slopes            |
| 3921  |1hd8t  |Smolan silty clay loam, 1 to 3 percent slopes      |
| 4555  |1hd7q  |Clime silty clay loam, 3 to 7 percent slopes       |
| 4673  |1hd87  |Irwin silty clay loam, 3 to 7 percent slopes       |
| 4720  |1hd88  |Kipson-Clime complex, 6 to 20 percent slopes       |
| 9986  |1hd8l  |Miscellaneous water                                |
| 9989  |1hd8q  |Orthents, clayey                                   |
| 9999  |1hd8y  |Water                                              |
|_______|_______|___________________________________________________|
```

```
PARAMETER subtitle CHARACTER PROMPT "Report Subtitle".
EXEC SQL
SELECT areaname, liid, legenddesc, nationalmusym, musym, muname
FROM area
INNER JOIN  legend by default
INNER JOIN  lmapunit by default
INNER JOIN  mapunit by default
WHERE mustatus = "correlated";
SORT BY areaname, musym SYM.

DEFINE dt TODAY.
DEFINE legend_name areaname || " -- " || legenddesc.

   TEMPLATE muline SEPARATOR "|" AT LEFT FIELD WIDTH 6, FIELD WIDTH 6, FIELD WIDTH 58, "".

   HEADER
    AT 1 "U.S. Department of Agriculture";
    AT RIGHT "Page ", PAGE WIDTH 3.
    AT 1 "Natural Resources Conservation Service";
    AT RIGHT dt WIDTH 10.
    SKIP 3 LINES.
    AT CENTER "SOIL MAP LEGEND".
   END HEADER.

   SECTION main
    HEADING
         AT CENTER subtitle WIDTH 75 ALIGN CENTER.
         AT CENTER legend_name WIDTH 75 ALIGN CENTER.
     SKIP 1 LINE.
     AT LEFT "_" WIDTH 74 REPEAT.
     USING muline.
     USING muline "Map Symbol" ALIGN CENTER," NAT Map Symbol" ALIGN CENTER,
      "Map Unit Name" ALIGN CENTER.
     USING muline "_" REPEAT, "_" REPEAT, "_" REPEAT.
     USING muline.

    DATA
     USING muline musym INDENT 1, nationalmusym INDENT -1, muname INDENT -1.
   END SECTION.

   SECTION WHEN LAST OF liid
    DATA
    USING muline "_" REPEAT, "_" REPEAT, "_" REPEAT.
    NEW PAGE.
   END SECTION.
```

CVIR References

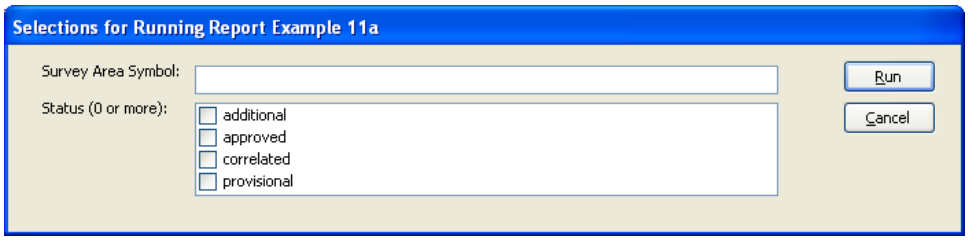
PARAMETER statement p. 54

PARAMETER Example

Use what you know from previous examples to create a report with a conditional statement allowing the user to choose by survey area number and their choice of map unit status and present a formatted legend report that includes the parameter selections as part of a title. The mapunit symbol, national map unit symbol, map unit name and map unit acres are the columns.

Sample Report Output

Your report parameter should look similar to the sample given here.

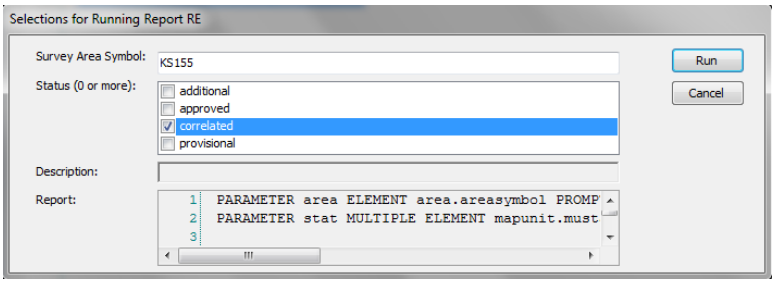


U.S. Department of Agriculture — Page 1
Natural Resources Conservation Service — 02/18/2011

SOIL MAP LEGEND
Reno County, Kansas -- Detailed Soil Map Legend
KS155--4
KS155--3

Map Symbol	NAT Map Symbol	Map Unit Name	Map Unit Acres	MU Status
1	37vf	Udic Argiustolls, 0 to 1 percent slope	0	additional
2	37vd	Udic Argiustolls, 1 to 3 percent slopes	0	additional
6	37vj	Sewage lagoons	0	additional
8	37vk	Redbed drains silt loam (fine, mixed, thermic typic ustifluvents)	0	additional
14	37vc	Carway-Kisiwa Complex	0	additional
21	37vl	Urban land-Aquic Argiustolls-Aquic Natrustolls complex	0	additional
48	37vg	Aquic Argiustolls, fine-loamy	0	additional

```
PARAMETER area ELEMENT area.areasymbol PROMPT "Survey Area Symbol".
PARAMETER stat MULTIPLE ELEMENT mapunit.mustatus.

EXEC SQL
SELECT areasymbol, areaname, liid,
legenddesc, nationalmusym,
musym, muname, muacres,
codename(mustatus) mustatus
FROM area
INNER JOIN  legend by default
INNER JOIN  lmapunit by default
INNER JOIN  mapunit by default
WHERE areasymbol = area AND
        mustatus IN( stat);
   SORT BY areaname, musym SYM.
```



Area Symbol	Area Name	Rec ID	Legend Description	National Mapunit Symbol	Mapunit Symbol	Mapunit Name	Total Acres	mustatus
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgcj	2266	Tobin silt loam, occasionally flooded	513	correlated
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgck	3405	Longford silty clay loam, 3 to 7 percent slopes, moderately eroded	1,276	correlated
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgcl	3824	Crete silt loam, 0 to 1 percent slopes	4,225	correlated
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgcm	3825	Crete silt loam, 1 to 3 percent slopes	5,237	correlated
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgcn	3843	Geary silt loam, 1 to 3 percent slopes	1,317	correlated
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgcp	3847	Geary silty clay loam, 3 to 7 percent slopes, moderately eroded	252	correlated
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgcq	3890	Ladysmith silty clay loam, 0 to 1 percent slopes	758	correlated
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgcr	3921	Smolan silty clay loam, 1 to 3 percent slopes	1,625	correlated
KS155	Reno County, Kansas	10,605	Detailed Soil Map Legend	1kgcs	5505	Abbyville-Kisiwa complex, occasionally flooded	6,896	correlated

PARAMETER Examples

PARAMETER area ELEMENT area.areasymbol PROMPT "Survey Symbol".

PARAMETER geographic_applicability ELEMENT legend.legendsuituse PROMPT "Geographic Applicability? (Normally want current wherever mapped)".

PARAMETER comp_nam ELEMENT component.compname PROMPT "Use * wildcard for all comps, or type in a component name".

PARAMETER majcomp ELEMENT component.majcompflag PROMPT "Check if major comps only. Don't check for majors and minors".

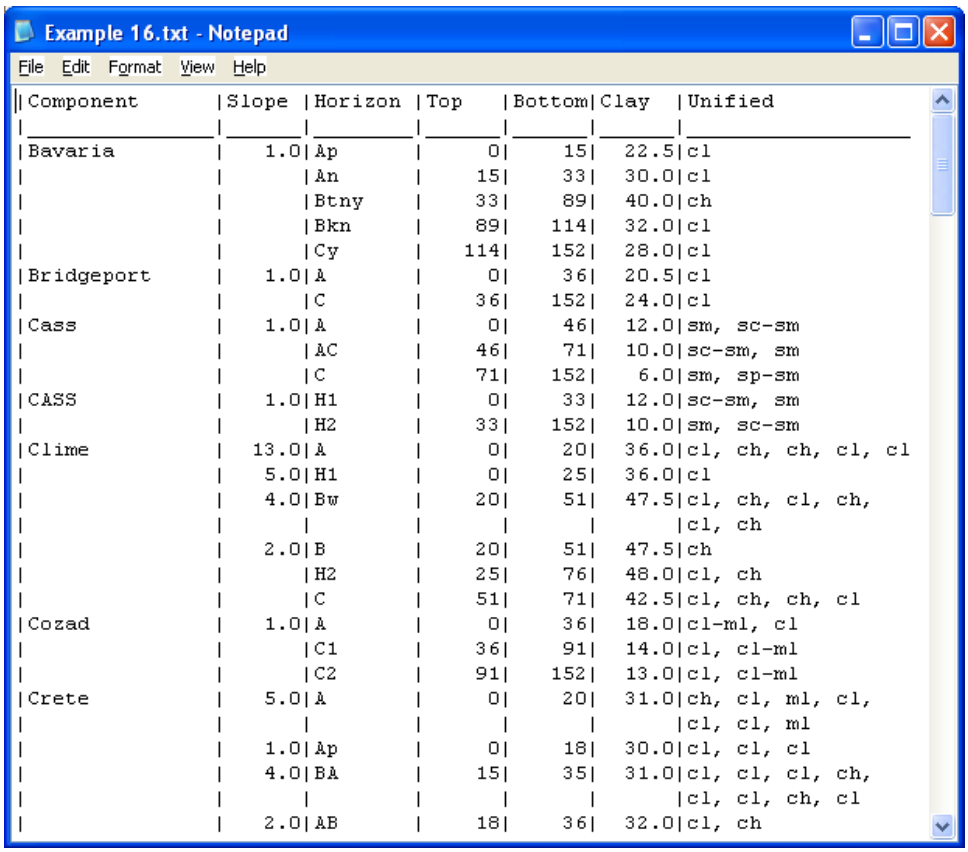
PARAMETER naname ELEMENT nasissite.nasissitename PROMPT "MLRA Office imatches".

PARAMETER soil CHARACTER PROMPT "Sampled as name to be queried".

PARAMETER fy NUMERIC PROMPT "Fiscal Year (4 digits)" required.

REGROUP

This report introduces the use of the REGROUP statement that is a secondary aggregation. REGROUP reduces each variable to one value. Note in this report that hzname must be done last so numbers of values match. REGROUP is part of the aggregation clause and allows for re-aggregation of your data after the initial aggregation. The unified (all horizon data) is aggregated using NONE so that the horizon level data is not aggregated to the component name. It is regrouped below to have it aggregate at the horizon level instead of the component level.


Example 16.txt - Notepad

File Edit Format View Help

Component	Slope	Horizon	Top	Bottom	Clay	Unified
Bavaria	1.0	Ap	0	15	22.5	cl
		An	15	33	30.0	cl
		Btny	33	89	40.0	ch
		Bkn	89	114	32.0	cl
		Cy	114	152	28.0	cl
Bridgeport	1.0	A	0	36	20.5	cl
		C	36	152	24.0	cl
Cass	1.0	A	0	46	12.0	sm, sc-sm
		AC	46	71	10.0	sc-sm, sm
		C	71	152	6.0	sm, sp-sm
CASS	1.0	H1	0	33	12.0	sc-sm, sm
		H2	33	152	10.0	sm, sc-sm
Clime	13.0	A	0	20	36.0	cl, ch, ch, cl, cl
	5.0	H1	0	25	36.0	cl
	4.0	Bw	20	51	47.5	cl, ch, cl, ch,
						cl, ch
	2.0	B	20	51	47.5	ch
		H2	25	76	48.0	cl, ch
		C	51	71	42.5	cl, ch, ch, cl
Cozad	1.0	A	0	36	18.0	cl-ml, cl
		C1	36	91	14.0	cl, cl-ml
		C2	91	152	13.0	cl, cl-ml
Crete	5.0	A	0	20	31.0	ch, cl, ml, cl,
						cl, cl, ml
	1.0	Ap	0	18	30.0	cl, cl, cl
	4.0	BA	15	35	31.0	cl, cl, cl, ch,
						cl, cl, ch, cl
	2.0	AB	18	36	32.0	cl, ch

BASE TABLE component.

EXEC SQL
SELECT compname, slope_r, hzname, hzdept_r, hzdepb_r, claytotal_r, unifiedcl, chunified.rvindicator
FROM component
INNER JOIN chorizon by default
INNER JOIN chunified by default
WHERE chunified.rvindicator = 1;
SORT by compname, hzdept_r
AGGREGATE ROWS compname COLUMN hzname none, hzdept_r none, hzdepb_r none, claytotal_r none, unifiedcl none.

This query creates the first aggregation on the component name so each component has all of its various horizons and clays and unified texture. Many components and all of their data.

ASSIGN unifiedcl REGROUP codename(unifiedcl) by hzname aggregate list ", ".
ASSIGN hzdept_r REGROUP hzdept_r by hzname aggregate first.
ASSIGN hzdepb_r REGROUP hzdepb_r by hzname aggregate first.
ASSIGN claytotal_r REGROUP claytotal_r by hzname aggregate first.
ASSIGN hzname   REGROUP hzname by hzname aggregate first. #always regroup hzname last.

The second aggregation is the REGROUP and each regroup is on the horizon name for each component - first put the multiple unified texturres on one cell all concatenated using a comma. Next aggregate all top depths by component name and take the first top depth in the group. Then the same aggregation with bottom depth and with clay total. The last regroup is the regroup all horizons by hzname.

REGROUP uses an ASSIGN statement so that they each can be aggregated via hzname. The list is defaulted to the comma ", " but is listed is limited to the first value for assign.

CVIR References
REGROUP statement p. 31
ASSIGN statement p.12

LOOKUP

This report introduces the use of the LOOKUP (CVIR page 25) which selects values from an array based on an index or condition. Two or three parameters can be used. The first expression is the key, which must be a single value, and the second expression is the index array. The key and the index must have the same type of data. In this example the maximum bulk density by horizon, by component name is being displayed. (OUTPUT is necessary to show results).

EXEC SQL
SELECT compname, slope_r, hzname, hzdept_r, hzdepb_r, dbthirdbar_r, dbthirdbar_r maxdb
 # third bar is used twice, once to pull the value for the horizon, the second renamed with an alias of maxdb, for use in the aggregation below.
FROM component
inner join chorizon by default
WHERE majcompflag = 1;
SORT by compname, hzdept_r
AGGREGATE rows compname column hzname none, hzdept_r none, hzdepb_r none, dbthirdbar_r none, maxdb max.

using the aggregation COLUMN, the maximum value from maxdb value is selected.

PAGE WIDTH 10 in.

Component	Slope	Horizon	Top	Bottom	Db
Carbika	0.0	Ap	0	28	1.50
		Bt1	28	38	
		Bt2	38	56	
		Bt3	56	86	
		Bt4	86	104	
		Bt5	104	152	
		Btk	152	203	
Farnum	0.0	Btk	185	203	1.48
Funmar	0.0	2Btkb2	168	203	1.55
Nalim	0.0	Ap	0	15	1.61
Naron	0.0	BC	140	168	1.58
		C	168	203	
Taver	0.0	2Btk2	135	163	1.55
		3Bt	163	203	
Carbika	0.0	Ap	0	28	1.50
		Bt1	28	38	
		Bt2	38	56	
		Bt3	56	86	
		Bt4	86	104	
		Bt5	104	152	
		Btk	152	203	
Farnum	0.0	Btk	185	203	1.48
Nalim	0.0	Ap	0	22	1.61
Naron	5.0	BC	140	168	1.58
		C	168	203	
Saltcreek	5.0	Ap	0	15	1.50
		Bt1	15	25	
		Bt2	25	66	
		Bt3	66	99	
		2Bt4	99	142	
		2Btk1	142	168	
		2Btk2	168	203	
Avans	10.0	Btk1	134	165	1.54
		Btk2	165	203	
Naron	10.0	BC	140	168	1.58

Using ASSIGN and LOOKUP, the maxdb is the key and the third bar and horizon depth are
the parameters. Lookup finds the data for the horizon with the greatest bulk density.

ASSIGN hzdept_r LOOKUP (maxdb, dbthirdbar_r, hzdept_r).
ASSIGN hzdepb_r LOOKUP (maxdb, dbthirdbar_r, hzdepb_r).
ASSIGN hzname LOOKUP (maxdb, dbthirdbar_r, hzname).

maxdb is the "key" and is used in the LOOKUP to find the horizon top depth and bottom depths in
which the max BD occurs. Once again, the final LOOKUP is on the hzname.

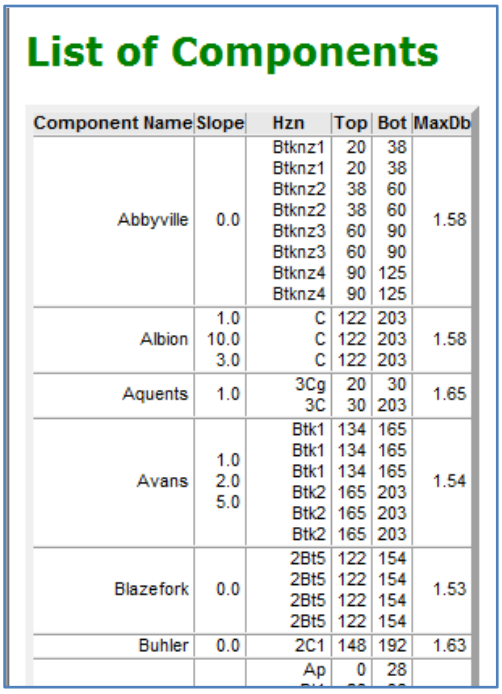
TEMPLATE basic separator "|"
AT LEFT field width 15, field width 6, field width 8, field width 6, field width 6, field width 6.

SECTION
HEADING
USING basic "Component", "Slope", "Horizon", "Top", "Bottom", "Db".
USING basic "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_" repeat.
DATA
USING basic compname, slope_r, hzname, hzdept_r, hzdepb_r, maxdb.
END SECTION.

Same Lookup Report using HTML coding

EXEC SQL
SELECT compname, slope_r, hzname, hzdept_r, hzdepb_r, dbthirdbar_r, dbthirdbar_r maxdb
third bar is used twice, once for the with an alias of maxdb
FROM component
inner join chorizon by default
WHERE majcompflag = 1;
SORT by compname, hzdept_r
AGGREGATE rows compname column hzname none, hzdept_r none, hzdepb_r none, dbthirdbar_r none, maxdb max.

ASSIGN hzdept_r LOOKUP (maxdb, dbthirdbar_r, hzdept_r).
ASSIGN hzdepb_r LOOKUP (maxdb, dbthirdbar_r, hzdepb_r).
ASSIGN hzname   LOOKUP (maxdb, dbthirdbar_r, hzname).



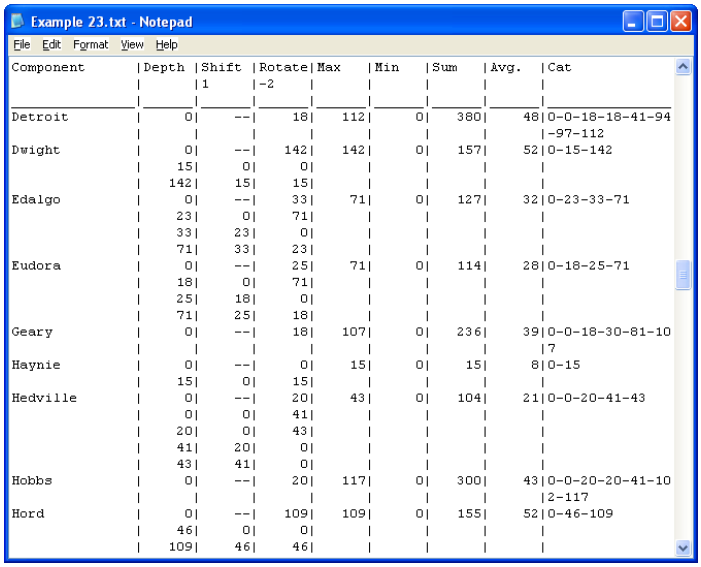
List of Components

Component Name	Slope	Hzn	Top	Bot	MaxDb
Abbyville	0.0	Btknz1	20	38	1.58
		Btknz1	20	38	
		Btknz2	38	60	
		Btknz2	38	60	
		Btknz3	60	90	
		Btknz3	60	90	
		Btknz4	90	125	
		Btknz4	90	125	
Albion	1.0	C	122	203	1.58
	10.0	C	122	203	
	3.0	C	122	203	
Aquents	1.0	3Cg	20	30	1.65
		3C	30	203	
Avans	1.0	Btk1	134	165	1.54
	2.0	Btk1	134	165	
	5.0	Btk1	134	165	
		Btk2	165	203	
		Btk2	165	203	
		Btk2	165	203	
Blazefork	0.0	2Bt5	122	154	1.53
		2Bt5	122	154	
		2Bt5	122	154	
		2Bt5	122	154	
Buhler	0.0	2C1	148	192	1.63
		Ap	0	28	

```
TEMPLATE basic TAG "td" ATTRIB ("role", "center")
   ELEMENT "tr" FIELD, FIELD, FIELD, FIELD, FIELD, FIELD.

SECTION WHEN AT START
DATA
          ELEMENT OPEN "HTML".
          ELEMENT OPEN "body" .
          ELEMENT "h2"  ATTRIB("style", "color:Green") "List of Components" .
          ELEMENT OPEN "table" ATTRIB("border", "5") ATTRIB("style", "background-color:white").
          ELEMENT OPEN "thead" ATTRIB("align", "center") .
          using basic "Component Name", "Slope", "Hzn", "Top", "Bot", "MaxDb".
          ELEMENT CLOSE "thead".
END SECTION.
SECTION
DATA
          ELEMENT OPEN "tbody".
          ELEMENT OPEN "tr".
                    USING basic
                    compname TAG "td" VALUETAG "para" ATTRIB ("role", "number"),
                    slope_r TAG "td" VALUETAG "para" ATTRIB ("role", "number"),
                    hzname TAG "td" VALUETAG "para" ATTRIB ("role", "number"),
                    hzdept_r TAG "td" VALUETAG "para" ATTRIB ("role", "number"),
                    hzdepb_r TAG "td" VALUETAG "para" ATTRIB ("role", "number"),
                    maxdb TAG "td" VALUETAG "para" ATTRIB ("role", "number").
          ELEMENT CLOSE "tr".
          ELEMENT CLOSE "tbody".
END SECTION.
SECTION WHEN AT END
   DATA
          ELEMENT CLOSE "table".
          ELEMENT CLOSE "body".
          ELEMENT CLOSE "HTML".
END SECTION.
```

ARRAY

This report introduces the use of the various ARRAY features (begin on p 23 CVIR) that computes a value in a multiple valued expression.



Component	Depth	Shift 1	Rotate -2	Max	Min	Sum	Avg.	Cat
Detroit	0	--	18	112	0	380	48	0-0-18-18-41-94-97-112
Dwight	0	--	142	142	0	157	52	0-15-142
	15	0	0					
	142	15	15					
Edalgo	0	--	33	71	0	127	32	0-23-33-71
	23	0	71					
	33	23	0					
	71	33	23					
Eudora	0	--	25	71	0	114	28	0-18-25-71
	18	0	71					
	25	18	0					
	71	25	18					
Geary	0	--	18	107	0	236	39	0-0-18-30-81-107
Haynie	0	--	0	15	0	15	8	0-15
	15	0	15					
Hedville	0	--	20	43	0	104	21	0-0-20-41-43
	0	0	41					
	20	0	43					
	41	20	0					
	43	41	0					
Hobbs	0	--	20	117	0	300	43	0-0-20-20-41-102-117
Hord	0	--	109	109	0	155	52	0-46-109
	46	0	0					
	109	46	46					

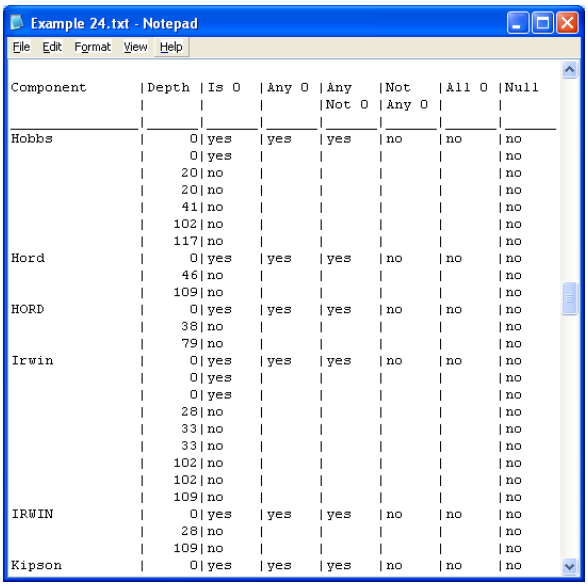
```
EXEC SQL select compname, hzdept_r
from component, outer chorizon
where join component to chorizon;
sort by compname, hzdept_r
aggregate rows compname
column hzdept_r none.

PAGE WIDTH 20 in.

# Create new variables to hold results of
each array manipulation function.

DEFINE ashift ARRAYSHIFT (hzdept_r, 1).
DEFINE arot   ARRAYROT (hzdept_r, -2).
DEFINE amax   ARRAYMAX (hzdept_r).
DEFINE amin   ARRAYMIN (hzdept_r).
DEFINE asum   ARRAYSUM (hzdept_r).
DEFINE aavg   ARRAYAVG (hzdept_r).
DEFINE acat   ARRAYCAT (sprintf("%.f", hzdept_r), "-").
#arraycat works on a set of numbers for one given variable it is not used on example L-R-H

TEMPLATE basic separator "|" width 6 replace null with "--"
at left field width 15 separator "", field, field, field, field, field, field, field, field width 15.

SECTION
 HEADING
  USING basic "Component", "Depth", "Shift 1", "Rotate -2", "Max", "Min", "Sum", "Avg.", "Cat".
  USING basic "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_"
repeat, "_" repeat.
 DATA
  USING basic compname, hzdept_r, ashift, arot, amax, amin, asum, aavg, acat.
END SECTION.
```

CVIR References pp. 20-25

ARRAYSHIFT, ARRAYROT, ARRAYMAX, ARRAYMIN, ARRAYSUM, ARRAYAVG, ARRAYCAT

ARRAY - another method

This report uses the DEFINE statement to create an array.

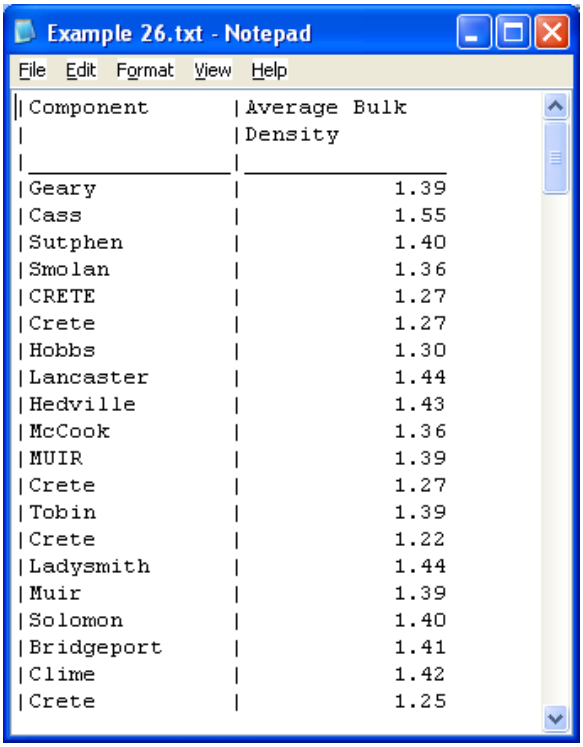


Report Script

```
EXEC SQL
select compname, hzdept_r
from component
left outer join chorizonby default;
sort by compname, hzdept_r
aggregate rows compname
column hzdept_r none.

# Create new variables to hold results of each array
manipulation function.

DEFINE is0          if hzdept_r == 0            then "yes" else "no".
DEFINE anyis0       if any(hzdept_r == 0)       then "yes" else "no".
DEFINE anynot0      if any(hzdept_r != 0)       then "yes" else "no".
DEFINE notany0      if not any(hzdept_r == 0) then "yes" else "no".
DEFINE allis0       if all(hzdept_r == 0)       then "yes" else "no".
DEFINE isanull      if isnull(hzdept_r)         then "yes" else "no".

PAGE WIDTH 20 in.
TEMPLATE basic separator "|" width 6 replace null with "--"
at left field width 15 separator "", field, field, field, field, field, field, field.

SECTION
 HEADING
  USING basic "Component", "Depth", "Is 0", "Any 0", "Any Not 0", "Not Any 0", "All 0", "Null".
  USING basic "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_" repeat, "_"
repeat.
 DATA
  USING basic compname, hzdept_r, is0, anyis0, anynot0, notany0, allis0, isanull.
END SECTION.
```

WTAVG

(CVIR p. 28) The WTAVG computes the sum of the first expression's values after multiplying each by a weighting factor, taken from the corresponding value of the second expression, and then divides the result by the sum of the weights. In this example the weighted average of the Bulk Density is component based on the coiid.



Component	Average Bulk Density
Geary	1.39
Cass	1.55
Sutphen	1.40
Smolan	1.36
CRETE	1.27
Crete	1.27
Hobbs	1.30
Lancaster	1.44
Hedville	1.43
McCook	1.36
MUIR	1.39
Crete	1.27
Tobin	1.39
Crete	1.22
Ladysmith	1.44
Muir	1.39
Solomon	1.40
Bridgeport	1.41
Clime	1.42
Crete	1.25

Report Script

```
BASE TABLE component.
EXEC SQL
select compname, hzdept_r, dbthirdbar_r, coiid,
chiid
from component
inner join chorizon by default;
sort by coiid, compname, chiid, hzdept_r
aggregate rows coiid column hzdept_r none,
dbthirdbar_r none, coiid none, compname, chiid
none.

PAGE WIDTH 20 in.

Property returns a multi-valued variable containing
the RV thickness of  that part of each horizon which
is in the specified depth range. If a horizon is entirely
outside the range its thickness is set to 0.

DERIVE layer_thickness USING "NSSC Pangaea":"LAYER THICKNESS IN RANGE (Generic)" (0, 2000).

# Compute average bulk density weighting each horizon by its thickness.

DEFINE average_db  WTAVG(dbthirdbar_r, layer_thickness).

TEMPLATE basic separator "|"
AT LEFT field width 15, field width 15.

SECTION when first of coiid
HEADING
   USING basic "Component", "Average Bulk Density".
   USING basic "_" repeat, "_" repeat.
DATA
   USING basic compname, average_db decimal 2.
END SECTION.
```

INCLUDE and ACCEPT

The INCLUDE command runs another report (subreport) and inserts its output as a logical line in the first report. Parameters may be passed to the subreport, and they must correspond with variables in the subreport's ACCEPT statement. Typically a record key would be passed as a parameter, which would be used by the subreport to query for information related to that record.

```
AT LEFT musym WIDTH 8, muname WIDTH 50.
INCLUDE "MLRA10_Office":"Component drainage" (dmuiid, coiid).
```

An output specification is used either to control spacing on the page or to produce actual report output. Output specifications can be either conditional or unconditional. When the IF clause is used, the IF expression is evaluated each time the section is processed. The expression follows the same rules as expressions for the DEFINE statement (see page 17). If it results in a True (non-zero) value, the output content is produced. If the value of the expression is False (a null, a zero or an empty character string) nothing is output. Without the IF clause, the output is always produced when its section is printed.

The ACCEPT statement defines variables that are passed into the script. These variables can be used in expressions to calculate values for other variables. They can also be used in the WHERE clause of a query by writing $name, where name is the name of the variable. This creates a parameterized query, as discussed under EXEC SQL.

The ACCEPT statement could be used in a subreport. The value of a key column such as dmuiid might be passed by a higher level report, and the subreport would use it in a query to find data related the to data mapunit being processed in the higher level report.

```
ACCEPT dmuiid, coiid.

EXEC SQL
SELECT  dmudesc, compname, comppct_r
FROM  datamapunit
INNER JOIN component by default
WHERE dmuiid = $dmuiid AND
SORT BY dmudesc, comppct_r DESC, compname LEX.
```

Using Subreports

This topic is discussed in the INCLUDE and ACCEPT section of this document.

The purpose of a subreport is to produce some output that is loosely coupled to the primary report, meaning that a subreport has its own set of queries and output specifications that might not be related to those of the primary report. It allows for greater flexibility in cases where complex formatting is required.

A subreport is requested with an INCLUDE statement in the data block of an output section. The entire output of a subreport is inserted in the data block as a single logical line. If keep processing is in effect, it will attempt to keep the subreport output together on a single page. Therefore it is advisable to design subreports so their output is less than a page. Longer output will spill over onto additional pages of the

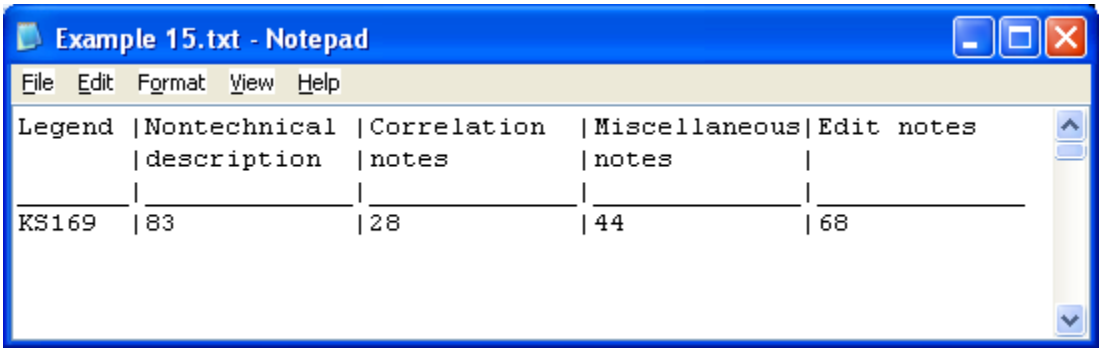
main report and possibly produce unwanted results. However, it is also possible to have a main report that produces no output of its own and only calls a series of subreports, in which case the main report will be a page by page copy of the subreports.

Subreports may not specify any page layout, such as the page size, font, headers or footers. The page layout of the main report controls all output from subreports.

A report and its subreports do not need to use the same base table, and no automatic synchronization is done as with properties in a DERIVE statement. Subreports may call themselves in a recursive fashion to produce a report on recursively organized data. An example is a report to list rules and all their subrules at any depth. It is important to pass the right parameters to a subreport so that it will find the right records to report on and not get into an endless recursion.

CROSSTAB

Crosstab allows data to be transformed from rows of data to columns of data. In a SQL, the data would be returned as rows of data. Using the CROSSTAB, the aggregated tab is assigned as columns. The column labels are those assigned in the data dictionary. The *crosstab* is a special type of aggregation that assigns values to positions in an array based on the value of a controlling column. It requires a CROSSTAB column, and one or more CELLS columns. The CROSSTAB function requires an OUTPUT formatted report.



Report Script

```
EXEC SQL select areasymbol, mapunittextkind, count(*) as rowcount
from real area
INNER JOIN real legend by default
INNER JOIN real lmapunit by default
INNER JOIN real mapunit by default
INNER JOIN real mutext by default
where areasymbol matches "KS169"
group by areasymbol, mapunittextkind;

SORT BY areasymbol sym, mapunittextkind
AGGREGATE ROWS areasymbol
CROSSTAB mapunittextkind
CELLS rowcount.
```

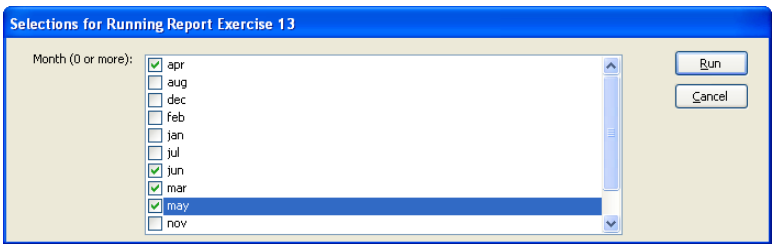
The Crosstab in the above report identifies the number of text notes, by category, for a given survey area.

Another example is this report that uses the Parameter statement for a user to define the months to be used in the report. The Parameter allows for multiple selections from the component month table. The format used in this report is similar to standard NASIS reports.

Notice the columns are those user defined months selected from the Parameter box.

Report Script

PARAMETER months ELEMENT comonth.month SELECTED CODEVAL MULTIPLE.



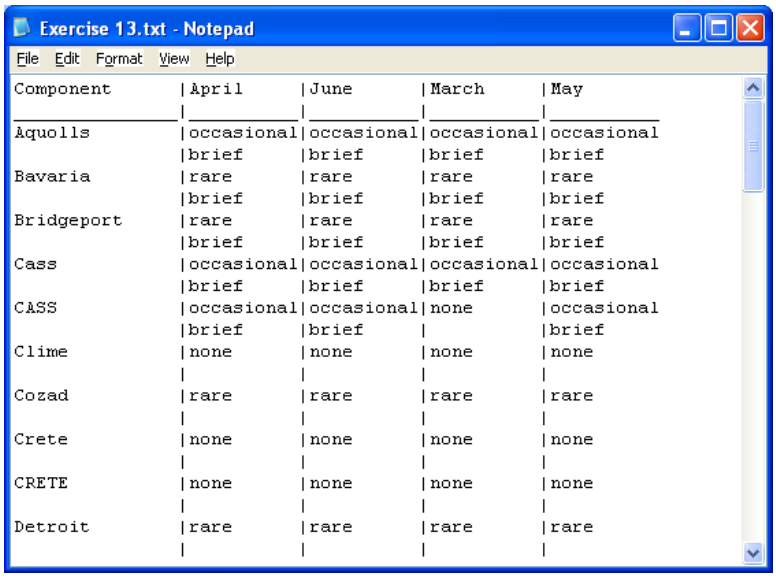
EXEC SQL
select compname, month, flodfreqcl, floddurcl
from component
INNER JOIN comonth by default;

Query aggregation selects flooding frequency and duration by month.

SORT BY compname, month
AGGREGATE ROWS compname
CROSSTAB month VALUES (months)
CELLS flodfreqcl, floddurcl.

PAGE LENGTH unlimited WIDTH unlimited.

Translate internal code representations into names. For the month, use
CODELABEL to get the correct capitalization.



DEFINE monthname CODELABEL(month).
DEFINE floodfreq CODENAME(flodfreqcl).
DEFINE floodduration CODENAME(floddurcl).

The ARRAY format places array values across the page rather than in a column.

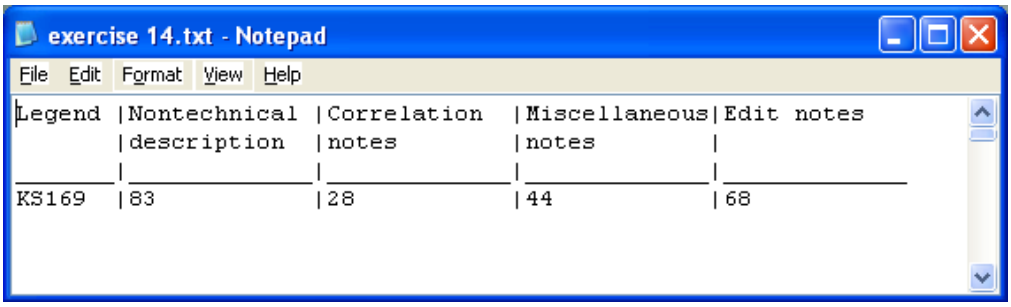
TEMPLATE compline
AT LEFT field width 15 separator "", ARRAY (field width 10 separator "|").

SECTION

```
HEADING
   USING compline "Component", monthname.
   USING compline "_" REPEAT, "_" REPEAT.
DATA
   USING compline compname, floodfreq.
   USING compline "", floodduration.
END SECTION.
```

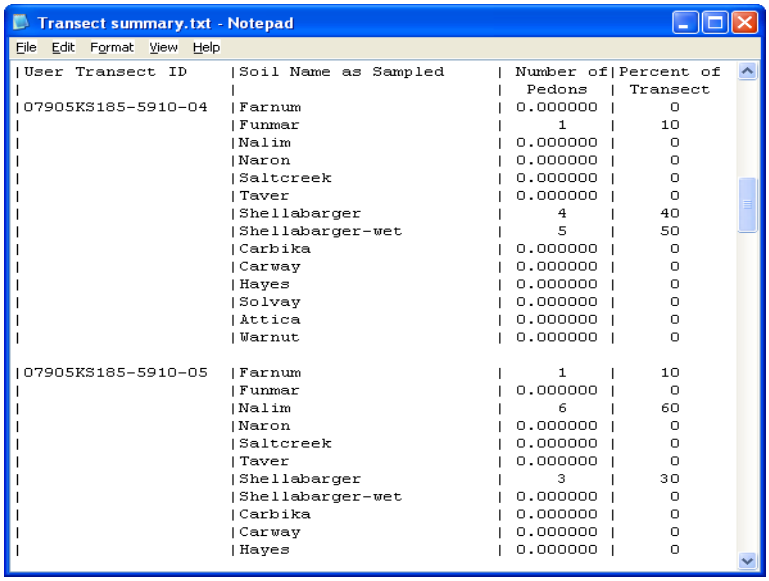
COUNT

This report includes the Aggregation statement and a Crosstab that identifies the number of text notes, by category, for a given survey area. The previous crosstab was used to report the flooding frequency by the selected month(s). This example query uses the crosstab to identify the various choices of map unit text Kinds and the number of times each kind is used for a given survey. The Query aggregation finds all distinct mapunit text kinds, then counts how many times each occurs for each area symbol. The crosstab must be in a sort by for this to work. The use of 1 is a mathematical expression used in the select or define clauses this instance every time it comes up, there is a 1 and rowcount is the name. The word "as" is optional. rowcount is another column and it lists a 1 for each of the areasymbols and mutextkinds. The sum in COLUMN is used to tally the rowcount's. There are only 3 columns used (areasymbol, mutextkind, and rowcount). This query uses the REAL command to return to the local database to retrieve data. Column labels are from the data dictionary.



Exercise Transect Summary report

Develop a transect summary sheet. Using what you know from previous examples, create a new report with formatted columns and headings for Transect IDs "07905KS185-5910*". The report is to show User Transect ID, sampled names, Number of Pedons, and Percent of Transect. Sort the report by User Transect ID. Adjust your column format so that the report fits one page wide. Use aggregation to eliminate duplicate User Transect IDs in your report.



User Transect ID	Soil Name as Sampled	Number of Pedons	Percent of Transect
07905KS185-5910-04	Farnum	0.000000	0
	Funmar	1	10
	Nalim	0.000000	0
	Naron	0.000000	0
	Saltcreek	0.000000	0
	Taver	0.000000	0
	Shellabarger	4	40
	Shellabarger-wet	5	50
	Carbika	0.000000	0
	Carway	0.000000	0
	Hayes	0.000000	0
	Solvay	0.000000	0
	Attica	0.000000	0
	Warnut	0.000000	0
07905KS185-5910-05	Farnum	1	10
	Funmar	0.000000	0
	Nalim	6	60
	Naron	0.000000	0
	Saltcreek	0.000000	0
	Taver	0.000000	0
	Shellabarger	3	30
	Shellabarger-wet	0.000000	0
	Carbika	0.000000	0
	Carway	0.000000	0
	Hayes	0.000000	0

```
PARAMETER trans ELEMENT utransectid MULTIPLE.


EXEC SQL
   select utransectid, soinmassamp, count(*) soilcount
   from REAL transect, REAL pedon
   WHERE utransectid matches trans and join transect to pedon
   group by utransectid, soinmassamp;
   sort by utransectid, soinmassamp
   aggregate rows by utransectid crosstab soinmassamp cells soilcount.

   PAGE WIDTH 15 in.

   assign soilcount if isnull(soilcount) then 0 else soilcount.
   define tot ARRAYSUM(soilcount).
   define pct (soilcount/tot) * 100.


   template basic separator "|"
   at left field width 20, field width 25, field width 10 align center,
   field width 10 align center.

SECTION
  HEADING
   USING basic "User Transect ID", "Soil Name as Sampled", "Number of",
   "Percent of".
   USING basic "","","Pedons", "Transect".
  DATA
   SKIP 1 LINE.
   USING basic utransectid, soinmassamp, soilcount, pct.
END SECTION.
```

Output

The report output format is controlled with a few commands. Detailed instructions and syntax can be found in the CVIR manual for reports.

TEMPLATE, SECTION, COLUMN FORMAT

Report Output

This next example is a text report has been formatted using column formats and a heading. Note that some mapunits may not have component names but are included in your report because of an OUTER join is specified in the query. Note also that map symbols and map unit name repeat when a mapunit has several components. This is not a default format report. The page layout is specified by use of templates, sections, and column formats.

Report Script

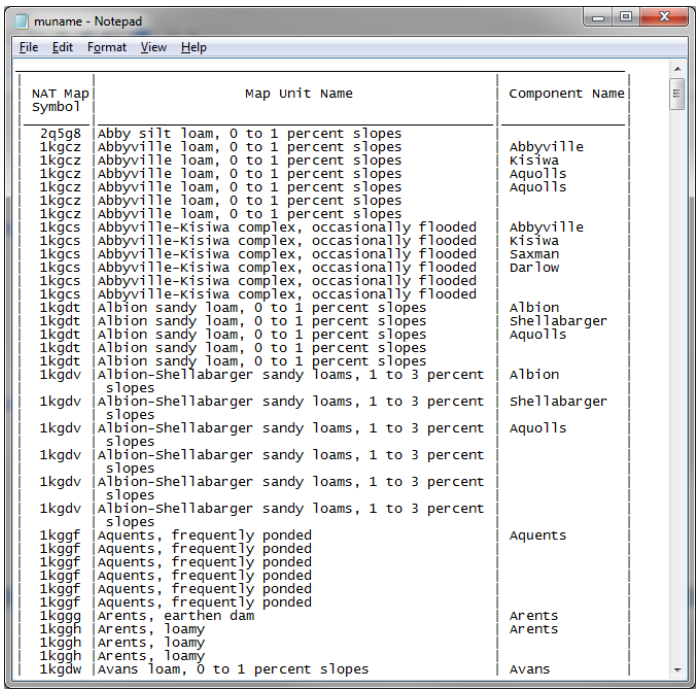
The EXEC SQL simply collects the data; the SECTION specifies which columns will be printed. A TEMPLATE is used to simplify coordination between headings and data. Report formatting is contained in a SECTION which includes both the HEADING and DATA. Column layout specifications are shown in **bold**. Note that the SELECT clause returns map units even though no component data (symbol 2q5g8) is available to print in the report.

EXEC SQL
SELECT nationalmusym, muname, repdmu, compname, comppct_r
FROM mapunit
LEFT OUTER JOIN correlation by default
LEFT OUTER jOIN datamapunit by default
LEFT OUTER JOIN component by default;
SORT BY muname, comppct_r DESC.

TEMPLATE dline SEPARATOR "|" AT LEFT FIELD WIDTH 8, FIELD WIDTH 48, FIELD WIDTH 15, "".

SECTION
HEADING
AT LEFT "_" REPEAT WIDTH 74.
USING dline.
USING dline
"NAT Map Symbol" ALIGN CENTER,
"Map Unit Name" ALIGN CENTER,
"Component Name" ALIGN CENTER.
USING dline "_" REPEAT, "_" REPEAT, "_" REPEAT.

DATA
USING dline
nationalmusym ALIGN RIGHT INDENT 1,
muname INDENT -1,
compname INDENT 1.


muname - Notepad

File Edit Format View Help

NAT Map Symbol	Map Unit Name	Component Name
2q5g8	Abby silt loam, 0 to 1 percent slopes	
1kgcz	Abbyville loam, 0 to 1 percent slopes	Abbyville
1kgcz	Abbyville loam, 0 to 1 percent slopes	Kisiwa
1kgcz	Abbyville loam, 0 to 1 percent slopes	Aquolls
1kgcz	Abbyville loam, 0 to 1 percent slopes	Aquolls
1kgcz	Abbyville loam, 0 to 1 percent slopes	
1kgcz	Abbyville loam, 0 to 1 percent slopes	
1kgcs	Abbyville-Kisiwa complex, occasionally flooded	Abbyville
1kgcs	Abbyville-Kisiwa complex, occasionally flooded	Kisiwa
1kgcs	Abbyville-Kisiwa complex, occasionally flooded	Saxman
1kgcs	Abbyville-Kisiwa complex, occasionally flooded	Darlow
1kgcs	Abbyville-Kisiwa complex, occasionally flooded	
1kgcs	Abbyville-Kisiwa complex, occasionally flooded	
1kgdt	Albion sandy loam, 0 to 1 percent slopes	Albion
1kgdt	Albion sandy loam, 0 to 1 percent slopes	Shellabarger
1kgdt	Albion sandy loam, 0 to 1 percent slopes	Aquolls
1kgdt	Albion sandy loam, 0 to 1 percent slopes	
1kgdt	Albion sandy loam, 0 to 1 percent slopes	
1kgdv	Albion-Shellabarger sandy loams, 1 to 3 percent slopes	Albion
1kgdv	Albion-Shellabarger sandy loams, 1 to 3 percent slopes	Shellabarger
1kgdv	Albion-Shellabarger sandy loams, 1 to 3 percent slopes	Aquolls
1kgdv	Albion-Shellabarger sandy loams, 1 to 3 percent slopes	
1kgdv	Albion-Shellabarger sandy loams, 1 to 3 percent slopes	
1kgdv	Albion-Shellabarger sandy loams, 1 to 3 percent slopes	
1kggf	Aquents, frequently ponded	Aquents
1kggf	Aquents, frequently ponded	
1kggf	Aquents, frequently ponded	
1kggf	Aquents, frequently ponded	
1kggf	Aquents, frequently ponded	
1kggf	Aquents, frequently ponded	
1kggg	Arents, earthen dam	Arents
1kggh	Arents, loamy	Arents
1kggh	Arents, loamy	
1kggh	Arents, loamy	
1kgdw	Avans loam, 0 to 1 percent slopes	Avans

END SECTION.

CVIR References

EXEC SQL statement p. 35

This SQL is discussed in previous sections in the query section and report section.

Sort Specifications **p. 39**

TEMPLATE statement p.80

The TEMPLATE is used in report formatting. The function allows the user to define a column format and use it repeatedly to format data. This function can be used to identify the column separator, the width of each column and column format.

SECTION statement p.59

The SECTION query is used to format the various formats on the page. The Heading and the Data 'sections' identify those fields used as the column headers and those data fields filling each column.

Column Layout Specifications p.72

Column layout specifications allow the format of the data within the column. Notice the use of the Align – left, center, right' along with other available options.

Discussion:

Defining template – first and last columns
Separator and pipe
AT LEFT
FIELD WIDTH
SECTION
Align center
Align right

FONT

The font is defaulted no user control over the font in a text output.

MARGIN

The default margin is half inch on all margins; otherwise, you can specify the margins.
If the page length is UNLIMITED the top and bottom margins are ignored, and if the page width is UNLIMITED the left and right margins are ignored.
EXAMPLE:

MARGIN TOP 1 inch BOTTOM 1 inch LEFT 1 inch RIGHT 1 inch.

PAGE

The default page width is 8.5 by 11 inches otherwise, you can specify the size. If inches is not specified then the width and length are controlled by the pitch.
LENGHT UNLIMITED WIDTH UNLIMITED is used for data exports and files to be used in other programs to eliminate extra lines and page breaks.
EXAMPLE:

PAGE WIDTH 144 LENGTH 88.

OR

PAGE WIDTH 8 INCHES LENGTH 11 INCHES.

OR

LENGTH UNLIMITED WIDTH UNLIMITED.

PITCH

The pitch defines the character spacing, in characters or lines per inch. The default is horizontal 10 characters per inch and vertical 6 lines per inch, which corresponds to a 12-point fixed-width font such as Courier

EXAMPLE:

PITCH HORIZONTAL 17 VERTICAL 8.

TEMPLATE

A template describes the format of a report line without the data. Templates are not required, but are useful to avoid repetitive specification of layout options. Putting the statement "USING template-name" into an output specification copies all the column layout information from the template into the output specification. The template is invoked with a USING statement, other layout options can be given, which take precedence over the template.

EXAMPLE:

TEMPLATE basic SEPARATOR "|" AT LEFT FIELD WIDTH 8, FIELD WIDTH 50.

HEADER and FOOTER

INITIAL HEADER is printed only once on the first page.
HEADER is the headers printed on every page.
FINAL FOOTER is print only on the last page.
FOOTER is printed at the bottom of every page.
EXAMPLE:

```
HEADER
AT CENTER "Sample Report".
SKIP 2 LINES.
END HEADER.
```

HEADERS

Report Output

This report uses headers to improve the appearance. Note that a horizontal line has been added to the bottom of the report.

Report Script

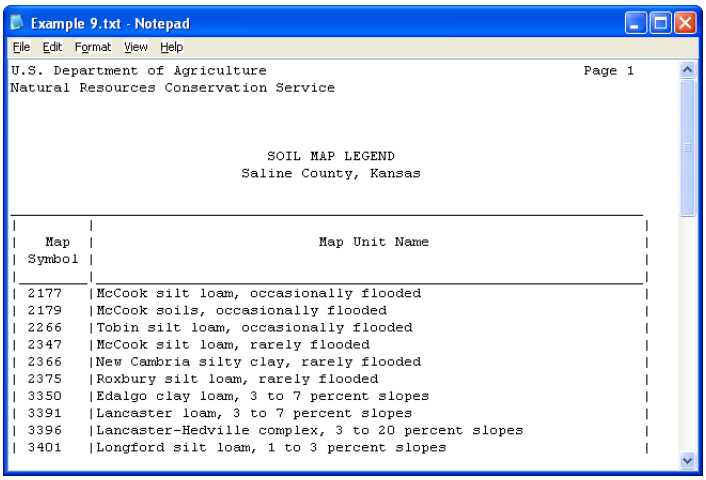
Specifications for the report header are in the HEADER part of the report. Note the difference in usage between HEADER and a SECTION HEADING.

```
EXEC SQL
SELECT areaname, musym, muname
FROM area
INNER JOIN  legend by default
INNER JOIN  lmapunit by default
```

INNER JOIN mapunit by default
WHERE mustatus = "correlated";
SORT BY areaname, musym SYM.

TEMPLATE muline SEPARATOR "|" AT LEFT FIELD WIDTH 8, FIELD WIDTH 64, "".

HEADER
AT 1 "U.S. Department of Agriculture";
AT RIGHT "Page ", PAGE WIDTH 3.
AT 1 "Natural Resources Conservation Service".
SKIP 3 LINES.
AT CENTER "SOIL MAP LEGEND".
END HEADER.



SECTION
HEADING
AT CENTER areaname WIDTH 75 ALIGN CENTER.
SKIP 1 LINE.
AT LEFT "_" WIDTH 74 REPEAT.
USING muline.
USING muline "Map Symbol" ALIGN CENTER,
"Map Unit Name" ALIGN CENTER.
USING muline "_" REPEAT, "_" REPEAT.

DATA
USING muline musym INDENT 1, muname INDENT -1.
END SECTION.

SECTION WHEN LAST OF areaname
DATA
USING muline "_" REPEAT, "_" REPEAT.
NEW PAGE.
END SECTION.

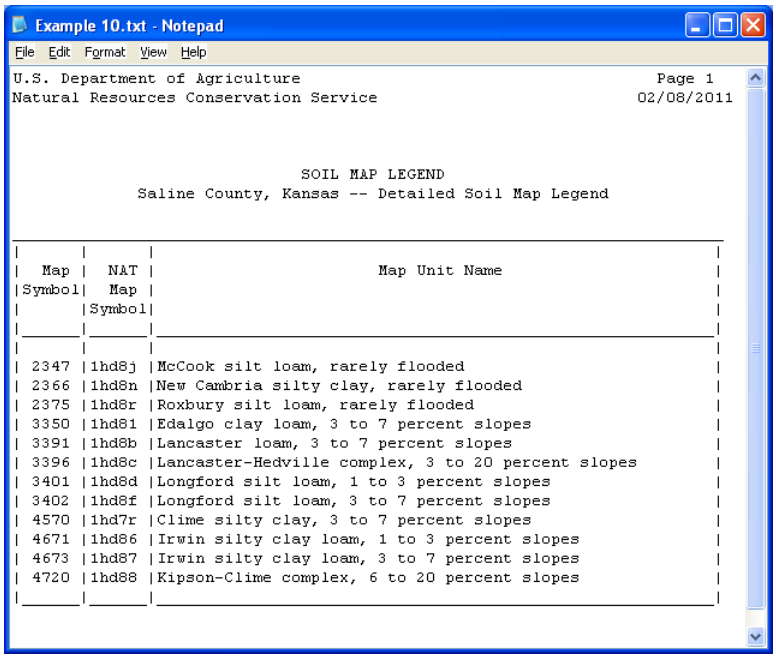
CVIR References

HEADER statement p. 47

DATE, SUBTITLE, SKIP LINES

Report Output

This report includes a date, dynamic subtitle, and additional line in the column titles. The format used in this report is similar to standard NASIS reports.



Report Script

The DEFINE statement is used to include today's date and concatenate two fields that will be used in the HEADING.

```
EXEC SQL
    SELECT areaname, liid,
legenddesc, nationalmusym, musym,
muname
FROM area
INNER JOIN  legend by default
INNER JOIN  lmapunit by default
INNER JOIN  mapunit by default
 WHERE mustatus = "correlated";
    SORT BY areaname, musym SYM.

    DEFINE dt TODAY.
    DEFINE legend_name areaname || " -- " || legenddesc.

    TEMPLATE muline SEPARATOR "|"
AT LEFT FIELD WIDTH 6, FIELD WIDTH 6, FIELD WIDTH 58, "".

    HEADER
     AT 1 "U.S. Department of Agriculture";
     AT RIGHT "Page ", PAGE WIDTH 3.
     AT 1 "Natural Resources Conservation Service";
     AT RIGHT dt WIDTH 10.
     SKIP 3 LINES.
     AT CENTER "SOIL MAP LEGEND".
    END HEADER.

    SECTION main
     HEADING
      AT CENTER legend_name WIDTH 75 ALIGN CENTER.
      SKIP 1 LINE.
      AT LEFT "_" WIDTH 74 REPEAT.
      USING muline.
      USING muline "Map Symbol" ALIGN CENTER," NAT Map Symbol" ALIGN CENTER,
       "Map Unit Name" ALIGN CENTER.
      USING muline "_" REPEAT, "_" REPEAT, "_" REPEAT.
```

```
 USING muline.

DATA
 USING muline musym INDENT 1, nationalmusym INDENT -1, muname INDENT -1.
END SECTION.

SECTION WHEN LAST OF liid
 DATA
 USING muline "_" REPEAT, "_" REPEAT, "_" REPEAT.
 NEW PAGE.
END SECTION.
```

CVIR References

DEFINE statement p. 13
Line Specifications p. 64

SECTION

A report section defines a block of report output that is produced as a unit. A section can be unconditional, meaning that the section's data block is printed on each cycle of the report's main query, or it can be printed only when certain conditions occur. A report can have any number of sections. The sections are printed in the order determined by their conditions, as discussed below under Section Conditions. Names are used in the KEEP option, and can be useful as documentation.
Sections are divided into two parts HEADING and DATA. HEADINGs create the column headings or a title for the section. The DATA section prints the data based on section and query conditions in the report section.

SECTION: Conditions
A condition specifies when the section is used. If no condition is provided, the section appears for each report cycle. Sections are set with conditions. All conditions begin with WHEN and the condition. The order of processing for section conditions is:

AT START (only once per report)
FIRST OF (per report cycle)
Other sections, in the order they appear in the script
LAST OF (per report cycle)
AT END (only once per report)

The other sections conditions could use any kind of comparison or boolean condition. A NO DATA section prints only if there are no input records, and could be used print a message such as "No data found". If the NO DATA section is not used and there is no input, no report output is produced. Instead, a warning dialog is displayed to the user.

EXAMPLE:

```
SECTION WHEN type == 2

SECTION WHEN FIRST OF muiid

SECTION WHEN LAST OF liid
```

KEEP option
This option will keep the data together when it gets to the bottom of the page. If KEEP is not used the data could be split on to different pages.
EXAMPLE:

```
SECTION a
END SECTION.
SECTION b KEEP WITH a
END SECTION.
```

KEEP options are ignored when XML style output is produced.
All page layout is controlled by the style sheet applied to the output of the report generator.

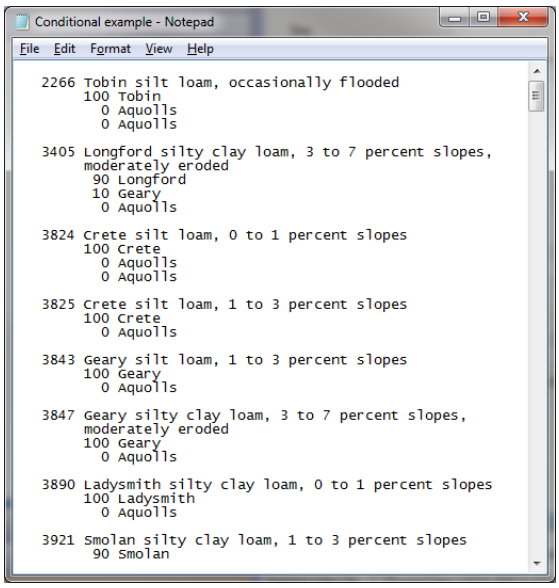
SECTION CONDITIONAL STATEMENTS

This report prints the publication symbol, map unit name, component names and percentages. But conditional sections are used to organize the page layout differently.

Report Script

This script uses a conditional statement on the SECTION to control the grouping and reporting of data from the database view. The section condition is shown in **bold**. Note that aggregation is not used in this example and each input row from the database view is an iteration of the report. The report is processed one input row (one iteration) at a time and both sections are evaluated for possible printing before the next input row is processed.

```
EXEC SQL
SELECT musym, muname, compname, comppct_r,
repdmu
FROM lmapunit
INNER JOIN mapunit by default
INNER JOIN correlation by default
INNER JOIN datamapunit by default
LEFT OUTER JOIN component by default;
SORT BY musym SYM, comppct_r DESC.

TEMPLATE muline
AT LEFT FIELD WIDTH 8 ALIGN RIGHT INDENT 1, FIELD
WIDTH 50.

TEMPLATE compline
AT 7 FIELD WIDTH 6 ALIGN RIGHT INDENT 1, FIELD
WIDTH 25.

SECTION WHEN FIRST OF musym
 DATA
 SKIP 1 LINE.
 USING muline musym, muname.
END SECTION.

SECTION
 DATA
 USING compline comppct_r, compname.
END SECTION.
```



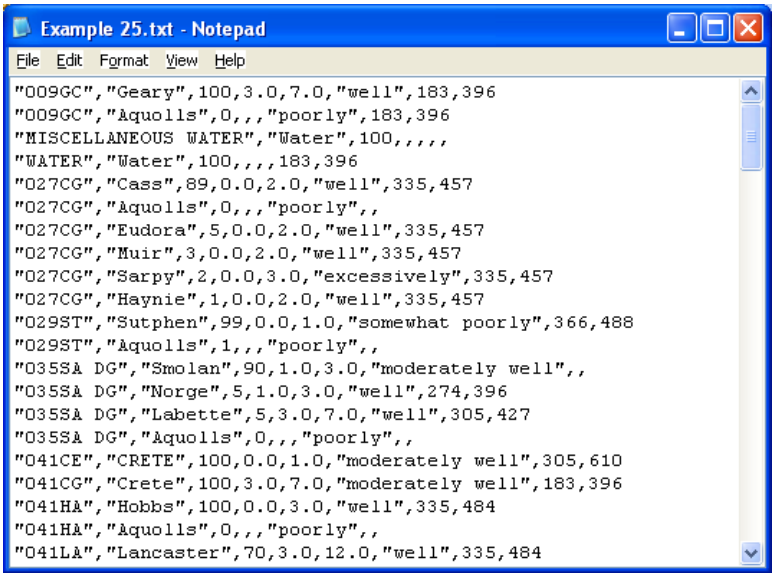
CVIR References

EXEC SQL statement p. 35
Sort Specifications **p. 39**
TEMPLATE statement p.80
SECTION statement p.59
Column Layout Specifications p.72

QUOTED function

Report Output

This report introduces other methods of formatting data for importing into other softwares. (p. 77)



Report Script

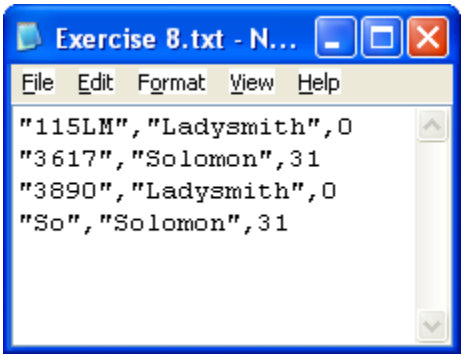
```
EXEC SQL select dmudesc, compname, comppct_r, slope_l, slope_h, drainagecl, elev_l, elev_h
from datamapunit
inner join  component by default;.

PAGE LENGTH UNLIMITED WIDTH UNLIMITED.
DEFINE drainage codename(drainagecl).

TEMPLATE export SEPARATOR "," WIDTH UNLIMITED
  AT LEFT field SEPARATOR "", field, field, field, field, field, field, field.

SECTION
DATA
  USING export
     dmudesc QUOTED,
     compname QUOTED,
     comppct_r DECIMAL 0,
     slope_l DECIMAL 1,
     slope_h DECIMAL 1,
     drainage QUOTED,
     elev_l DECIMAL 0 NO COMMA,
     elev_h DECIMAL 0 NO COMMA.
END SECTION.
```

Exercise 6. Creating a Data Export Format

Use an existing NASIS Property and what you know from previous examples to create a report with comma delimited columns, quoted text, and no headings that shows minimum representative depth to seasonal high water table for the dominant component of each map unit. You should use the PAGE statement to eliminate page breaks. You should also use column layout specifications to format rows of data. Your report output will be suitable for saving as an ASCII file that can be imported into other databases.

Sample Report Output

Your report should look similar to the sample given here. Depth is reported in centimeters.



LINE SPECIFICATIONS

This option controls each line of data with key terms and IF THEN conditions.
The most common specifications are

USING (template name),
SKIP line-controls spacing;
NEW PAGE-creates a new page;
INCLUDE (report name)-Adds data from a subreport;
AT Statement -controls the position on the page.

EXAMPLE:

```
SKIP 2 LINES.

AT LEFT musym WIDTH 8, muname WIDTH 50.

IF comp_pct > 10 USING comp_tmpl compname, slope_l, slope_h.

INCLUDE “MLRA10_Office”:”Flood Subreport” (dmudbsidref, coiid).
```

If the condition is True the output content is produced. If the value of the expression is False nothing is output. Without the IF clause, the output is always produced when its section is printed.

EXAMPLE:

```
AT LEFT musym WIDTH 8, muname WIDTH 50.

AT CENTER title WIDTH 20 CENTERED; AT RIGHT date WIDTH 12.
```

COLUMN SPECIFICATIONS

This option identifies exactly what will be printed at a particular spot in a report. These specifications are added after each column of data. Some of the most common are WIDTH, DECIMAL, ALIGN, SEPARATOR, NO COMMA and SUPRESS DUPLICATES.

EXAMPLE:

AT LEFT musym width unlimited SEPARATOR "|", muname width unlimited.

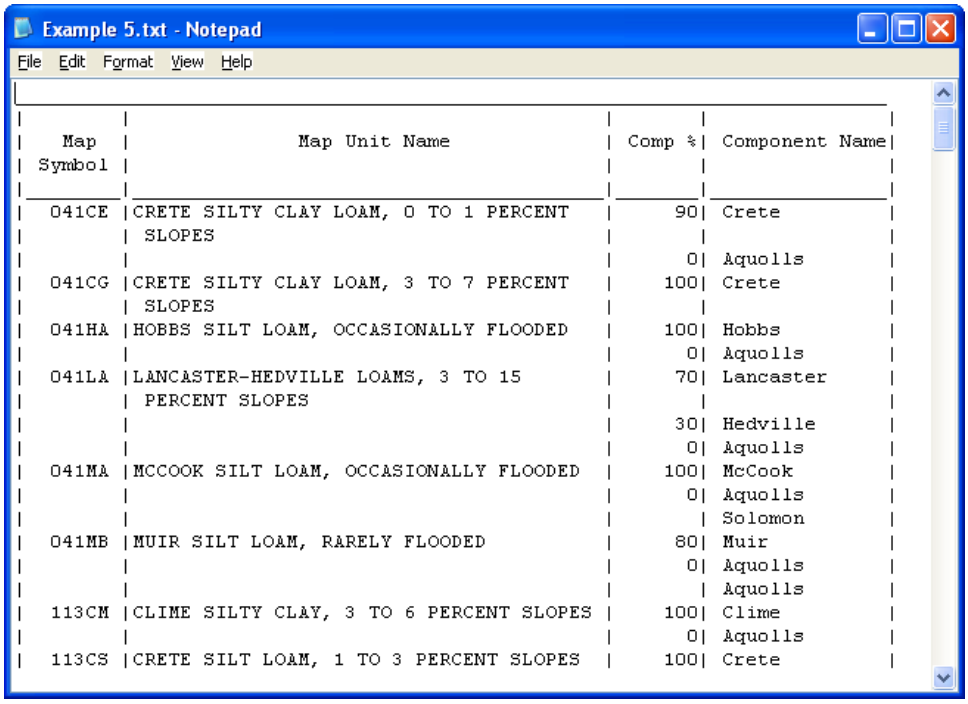
The SEPARATOR is used to separate columns. The SEPARTOR precedes the column of stat so if you do not want a SEPARTOR at the beginning of the table you have to specify "NO SEPARATOR" as a column specification and if you want a vertical line at the end of the table you have to create a null field "" with SEPARTOR.

Exercise Creating a Formatted Report

Using what you know, create a new report with formatted columns and headings that shows mapunit symbols, mapunit names, components, and component percentages. Sort the report by mapunit symbol and show the dominant components first. Adjust your column format so that the report fits one page wide. Use aggregation to eliminate duplicate map symbols and map unit names in your report. Be sure to keep component percent correctly coordinated with their corresponding component names.

Sample Report Output

Your report should look similar to the sample given here.



Map Symbol	Map Unit Name	Comp %	Component Name
041CE	CRETE SILTY CLAY LOAM, 0 TO 1 PERCENT SLOPES	90	Crete
		0	Aquolls
041CG	CRETE SILTY CLAY LOAM, 3 TO 7 PERCENT SLOPES	100	Crete
041HA	HOBBS SILT LOAM, OCCASIONALLY FLOODED	100	Hobbs
		0	Aquolls
041LA	LANCASTER-HEDVILLE LOAMS, 3 TO 15 PERCENT SLOPES	70	Lancaster
		30	Hedville
		0	Aquolls
041MA	MCCOOK SILT LOAM, OCCASIONALLY FLOODED	100	McCook
		0	Aquolls
			Solomon
041MB	MUIR SILT LOAM, RARELY FLOODED	80	Muir
		0	Aquolls
			Aquolls
113CM	CLIME SILTY CLAY, 3 TO 6 PERCENT SLOPES	100	Clime
		0	Aquolls
113CS	CRETE SILT LOAM, 1 TO 3 PERCENT SLOPES	100	Crete

Interpretation Reports

The complexity of reporting interpretations increased in NASIS 6. Interpretation reports require specific formating to handle running the interpretation, retrieving the results and reporting the final product.

MANU - Sewage Disposal text

```
BASE TABLE component.

INTERPRET "NSSC Pangaea": "ENG - Septic Tank Absorption Fields",
                "NSSC Pangaea": "ENG - Sewage Lagoons"
                  MAX RULEDEPTH 1
                  MAX REASONS 5
AGGREGATE CROSSTAB BY PrimaryRuleInterpRuleName
VALUES ("ENG - Septic Tank Absorption Fields",   "ENG - Sewage Lagoons")
LABELS "Septic Tank \nAbsorption Fields",  "Sewage Lagoons"
CELLS RatingValueHighRV,  InterpRuleDepth, RatingClassNameHighRV.

EXEC SQL
SELECT areaname, legenddesc, liid, musym, lmapunitiid, lmapunit.seqnum,
   compname, component.seqnum, comppct_r, coiid, localphase
FROM real area, legend, lmapunit, mapunit, correlation, datamapunit, component
WHERE join area to legend and
    join legend to lmapunit and
    join lmapunit to mapunit and
    join mapunit to correlation and
    join correlation to datamapunit and
    repdmu=1 and
    join datamapunit to component ;
SORT BY liid, lmapunit.seqnum, musym SYM, component.seqnum, comppct_r DESC,
   compname, coiid.

DEFINE mu_lead                  CLIP(musym) || ":".
DEFINE dt                TODAY.
#--------------------edited-----------------------------

DEFINE soilnm                   ISNULL(localphase) ? compname : compname||", "||localphase.
DEFINE flag                     NEW(coiid)  ? RatingClassNameHighRV : flag.
DEFINE fuzznum                  ((flag matches "Not rated*" or flag matches "Not Rated*") or
                                 (InterpRuleDepth == 0)) ? "" :
                                 RatingValueHighRV > 0.00 and RatingValueHighRV <= 0.005 ? "0.01" :
                                 RatingValueHighRV >= 0.99 and RatingValueHighRV < 1.00 ? "0.99" :
                                 sprintf("%.2f", RatingValueHighRV).

DEFINE rate_reason      InterpRuleDepth > 0 and (flag matches "Not rated*" or
                                flag matches "Not Rated*") ? "" : RatingClassNameHighRV.

#-------------------------------------------------------
```

```
PITCH HORIZONTAL 15 VERTICAL 8.
PAGE LENGTH 80 WIDTH 8.5 INCH.

# Line between margins 105 columns
TEMPLATE heading1
   AT LEFT FIELD WIDTH 21 SEPARATOR "", FIELD WIDTH 4 SEPARATOR "|",
   ARRAY ( FIELD WIDTH 25) SEPARATOR "|", "" SEPARATOR "".

TEMPLATE heading2
   AT LEFT FIELD WIDTH 27 SEPARATOR "",
   ARRAY ( FIELD WIDTH 19 SEPARATOR "|", FIELD WIDTH 5 SEPARATOR "|"),
   "" SEPARATOR "".

TEMPLATE mapunit
   AT LEFT FIELD WIDTH 21 SEPARATOR "",
   FIELD WIDTH 4 SEPARATOR "|",
   ARRAY ( FIELD WIDTH 19 SEPARATOR "|", FIELD WIDTH 5 SEPARATOR "|"),
   "" SEPARATOR "".

TEMPLATE component
   AT LEFT " " SEPARATOR "", FIELD WIDTH 20 SEPARATOR "";
   AT 22 BOTTOM FIELD WIDTH 4 SEPARATOR "|", ARRAY ( FIELD WIDTH 19 SEPARATOR "|", FIELD
WIDTH 5 SEPARATOR "|" ), "" SEPARATOR "".

TEMPLATE rating
   AT LEFT FIELD WIDTH 21 SEPARATOR "", FIELD WIDTH 4 SEPARATOR "|", ARRAY ( FIELD WIDTH 20
SEPARATOR "|", FIELD WIDTH 5 SEPARATOR "|" ), "" SEPARATOR "".

HEADER INITIAL
   AT LEFT areaname WIDTH 75; AT 82 "Print date:  ", dt WIDTH 10.
   AT LEFT "Sewage Disposal".
   SKIP 2 LINES.
   AT LEFT  "(The information in this table indicates the dominant soil condition but does not eliminate
the need for onsite investigation.  The numbers in the value columns range from 0.01 to 1.00.  The
larger the value, the greater the limitation.  See text for further explanation of ratings in this table.)"
WIDTH 78 INDENT -5.
   SKIP 1 LINE.
END HEADER.
#74
HEADER
  AT LEFT areaname WIDTH 75; AT 82 "Print date:  ", dt WIDTH 10.
   AT LEFT "Sewage Disposal".
   SKIP 1 LINE.
END HEADER.

SECTION
   HEADING
         AT LEFT "_" REPEAT WIDTH 104.
```

```
        USING heading1.
        USING heading1 "Map symbol\nand soil name" ALIGN CENTER,"Pct.\nof\nmap\nunit" ALIGN
CENTER,
           PrimaryRuleInterpRuleName LABEL ALIGN CENTER.
        USING heading1 "", "", "_" REPEAT.
        USING mapunit "", "", "Rating class and\nlimiting features" ALIGN CENTER, "Value".
        USING mapunit "_" REPEAT, "_" REPEAT, "_" REPEAT, "_" REPEAT.
        USING mapunit "","","","".
END SECTION.

SECTION mapunit WHEN FIRST OF musym
  DATA
        USING mapunit mu_lead.
END SECTION.

SECTION ratings
  KEEP WITH mapunit, ratings
  DATA
#---------------edited----------------------
        if not all (rate_reason == "" or isnull (rate_reason))
                USING component soilnm INDENT -1 PAD "-" SUPPRESS DUPLICATES by coiid,

#---------------------------------------
     comppct_r ALIGN CENTER SUPPRESS DUPLICATES by coiid,
     rate_reason,
     fuzznum.
END SECTION.

SECTION WHEN LAST OF coiid
  KEEP WITH ratings
    DATA
        USING mapunit.
END SECTION.

SECTION WHEN LAST OF liid
  DATA
        USING mapunit "_" REPEAT, "_" REPEAT, "_" REPEAT, "_" REPEAT.
        NEW PAGE.
END SECTION.
```

MANU - Sewage Disposal html

```
BASE TABLE component.

INTERPRET "NSSC Pangaea": "ENG - Septic Tank Absorption Fields",                    #page 49
                "NSSC Pangaea": "ENG - Sewage Lagoons"
        MAX RULEDEPTH 1
        MAX REASONS 5
 AGGREGATE CROSSTAB BY PrimaryRuleInterpRuleName
 VALUES ("ENG - Septic Tank Absorption Fields",  "ENG - Sewage Lagoons")
 LABELS "Septic Tanks", "Sewage Lagoons"
 CELLS RatingValueHighRV,  InterpRuleDepth,  RatingClassNameHighRV.

EXEC SQL
SELECT areaname, legenddesc, liid, musym, lmapunitiid, lmapunit.seqnum,
   compname, component.seqnum, comppct_r, coiid, localphase
FROM real area, legend, lmapunit, mapunit, correlation, datamapunit, component
WHERE join area to legend and
    join legend to lmapunit and
    join lmapunit to mapunit and
    join mapunit to correlation and
    join correlation to datamapunit and
    repdmu=1 and
    join datamapunit to component ;
SORT BY liid, lmapunit.seqnum, musym SYM, component.seqnum, comppct_r DESC,
   compname, coiid.


DEFINE mu_lead          CLIP(musym) || ":".
DEFINE space            INITIAL "".
DEFINE soilnm           SECASE (ISNULL(localphase) ? compname : compname||", "||localphase).

# Toggle a shading type for alternating components
DEFINE toggle           IF NEW (lmapunitiid) THEN 0
                        ELSE IF NEW(coiid) THEN 1-toggle
                        ELSE toggle.
DEFINE shading          IF toggle == 1 THEN "odd" ELSE "even".

# Format fuzzy rating value to avoid rounding off number close to 0 and 1.

DEFINE flag             NEW(coiid)  ? RatingClassNameHighRV : flag.

DEFINE fuzznum                  ((flag matches "Not rated*" or flag matches "Not Rated*") or
                                (InterpRuleDepth == 0)) ? "" :
                                RatingValueHighRV > 0.00 and RatingValueHighRV <= 0.005 ? "0.01" :
                                RatingValueHighRV >= 0.99 and RatingValueHighRV < 1.00 ? "0.99" :
                                sprintf("%.2f", RatingValueHighRV).
```

```
DEFINE rate_reason              InterpRuleDepth > 0 and (flag matches "Not rated*" or
                                flag matches "Not Rated*") ? "" : RatingClassNameHighRV.

# Indent value is used to select the proper indentation for reasons.
DEFINE class                    sprintf ("reason%.f", InterpRuleDepth).


TEMPLATE head1
   ELEMENT "tr"
                   FIELD TAG "td" ATTRIB ("rowspan", "2"),
                   FIELD TAG "td" ATTRIB ("rowspan", "2"),
                   ARRAY (FIELD TAG "td" ATTRIB ("colspan", "2")
                   ATTRIB ("class", "begindatagroup enddatagroup")).

TEMPLATE head2
   ELEMENT "tr"
                   ARRAY (FIELD TAG "td" ATTRIB("class", "begindatagroup"),
                    FIELD TAG "td" ATTRIB("class", "enddatagroup")).

TEMPLATE mapunit
   ELEMENT "tr" ATTRIB ("class", "mapunit")
                   FIELD TAG "td" VALUETAG "para" ATTRIB ("role", "mu-name"),
                   space TAG "td",
                   ARRAY (space TAG "td" ATTRIB("class", "begindatagroup"),
                   space TAG "td" ATTRIB("class", "enddatagroup")).

TEMPLATE component
   ELEMENT "tr" ATTRIB ("class", shading)
         FIELD TAG "td" VALUETAG "para" ATTRIB ("role", "comp-name"),
         FIELD TAG "td" VALUETAG "para" ATTRIB ("role", "number"),
         ARRAY (FIELD TAG "td" ATTRIB("class", "begindatagroup") VALUETAG "para" ATTRIB ("role",
class),
         FIELD TAG "td" ATTRIB("class", "enddatagroup") VALUETAG "para" ATTRIB ("role", "number")).

SECTION WHEN AT START
# Produces report title.
DATA
   ELEMENT OPEN "section" ATTRIB ("label", "SoilReport").
   ELEMENT "title" "Sewage Disposal Interpretations".
END SECTION.

SECTION WHEN FIRST OF liid
# Produces the survey area name and table headings
DATA
   ELEMENT OPEN "table".
   ELEMENT "title" areaname.
   ELEMENT OPEN "thead".
```

```
   USING head1
                "Map symbol and soil name",
                "Pct. of map unit",
                PrimaryRuleInterpRuleName.
   USING head2
                "Rating class and limiting features", "Value".
   ELEMENT CLOSE "thead".
   ELEMENT OPEN "tbody".
END SECTION.

SECTION mapunit WHEN FIRST OF lmapunitiid
# Produces one line with the mapunit symbol
DATA
   USING mapunit mu_lead.
END SECTION.

SECTION ratings
# Produces output for a component. First line has component name and interp ratings.
# Additonal lines are produced as needed for reasons.
DATA
        IF NOT ALL (rate_reason == "" OR ISNULL (rate_reason))
        USING component
                soilnm SUPPRESS DUPLICATES by coiid,
                comppct_r SUPPRESS DUPLICATES by coiid,
                rate_reason,
                fuzznum.
END SECTION.

SECTION WHEN LAST OF liid
# Closes the table
DATA
   ELEMENT CLOSE "tbody".
   ELEMENT CLOSE "table".
END SECTION.

SECTION WHEN AT END
# Closes the document.
DATA
   ELEMENT CLOSE "section".
END SECTION.
```

HTML Reports

HTML output style has no concept of a "page", so page layout features, such as KEEP and HEADING, are ignored. Be sure to identify the width and length as unlimited.

PAGE LENGTH UNLIMITED WIDTH UNLIMITED.

HTML output is produced with the ELEMENT command. A report cannot use both ELEMENT and AT commands. An ELEMENT always has a name, and may also have attributes and content:
Three options exist

TAG (Tags surround the whole data set),
VALUE TAG (VALUE TAG surrounds a single data value),
ATTRIB (for use with additional attributes).

HTML elements

An element in HTML represents some kind of structure or semantics and generally consists of a start tag, content, and an end tag. The following is a paragraph element:

```
<p>
This is the content of the paragraph element.
</p>
```

A plain ELEMENT command produces both the opening and closing tags. Sometimes it is not possible to put all the content you want into a single ELEMENT command, the ELEMENT OPEN "name" can be used to produce just the opening tag. For each ELEMENT OPEN there must have a matching ELEMENT CLOSE. Later in the report script there must be ELEMENT CLOSE "name" to produce the closing tag.

HTML tags

Tags are used to mark up the start and end of an HTML element.
A start tag consists of an opening angle bracket (<) followed by the element name, zero or more space separated attribute/value pairs, and a closing angle bracket (>).

A start tag with no attributes:

```
<p>
```

A start tag with an attribute:

```
<p class="info">
```

End tags consist of an opening angle bracket followed by a forward slash, the element name, and a closing angle bracket:

```
</p>
```

There are also some elements that are empty, meaning that they only consist of a single tag and do not have any content. In HTML, such tags look just like opening tags:

```
<br>
```

HTML Value Tags

Value tags are used to mark up one field at a time.

HTML attributes

An attribute defines a property for an element, consists of an attribute/value pair, and appears within the element's start tag. An element's start tag may contain any number of space separated attribute/value pairs. The most popular misuse of the term "tag" is referring to alt attributes as "alt tags". There is no such thing in HTML. Alt is an attribute, not a tag.

<img src="foobar.gif" alt="A foo can be balanced on a bar by placing its fubar on the bar's foobar.">

In most cases the TAG is used to define the structure of the report and the ATTRIB is used to define the style.

An element name must correspond with a standard HTML tag (table in Appendix) for the type of output to produce. Many HTML tags have standard attributes that modify the output appearance. For example, a table can have borders drawn between cells with a tag like <table border="1">. In a NASIS report this is written as ELEMENT "table" ATTRIB("border", "1").

To produce reports that look like Web Soil Survey reports, use the elements and attributes listed in the appendix for the DocBook HTML standard. This is converted automatically to HTML for displaying in a web browser. If you are familiar with HTML you can use regular HTML tags as NASIS elements instead of DocBook HTML.

The standard format for ATTRIB is ATTRIB("style", "color:#FF0000").
If you want to use special character the SPECIAL term must be applied

EXAMPLE:

textdata TAG "p" SPECIAL.

Every HTML output report needs to be started and ended with two tags and are usually in a section with an AT START and AT END condition.

EXAMPLE:

ELEMENT "p" ATTRIB ("class", "subhead") musym, ": ", muname.
ELEMENT "tr" musym TAG "td", muname TAG "td" ATTRIB ("class" "namecol").

TEMPLATE row2 ELEMENT "tr" FIELD TAG "td", FIELD TAG "td" VALUETAG "p" ATTRIB ("class" "namecol").

USING row2 compname, hzname.

Document sections

Another related term is "section". An HTML document is divided into a "head" section (the contents of the head element) and a "body" section (the contents of the body element). Here are three simple guidelines for markup syntax:

- Use lowercase for all element and attribute names.
- Explicitly include all start and end tags, including the optional tags.
- Quote all attribute values, use double-quoted syntax, and do not use any whitespace around the equals sign: name="value".

The link below give the complete list of properties for attribute values
http://www.w3.org/TR/CSS21/propidx.html

HTML Examples

Each report will typically have three sections, an opening, main and closing section.

```
SECTION WHEN AT START
ELEMENT OPEN “HTML”.
ELEMENT OPEN “BODY”.
END SECTION.

SECTION
DATA
Main body of data or table…….
END SECTION.

SECTION WHEN AT END
ELEMENT CLOSE “BODY”.
ELEMENT CLOSE “HTML”.
END SECTION.
```

Open a HTML page align the header and color the background

```
SECTION WHEN AT START
DATA
ELEMENT OPEN "HTML".
ELEMENT OPEN "body" ATTRIB("style", "background-color:#DCDCDC").
ELEMENT OPEN "h2" ATTRIB ("align", "center") "This report make over 200 checks on the data.".
ELEMENT CLOSE "h2".
```

Add blank line

```
ELEMENT "br".
```

Create a header

```
SECTION WHEN AT START
DATA
ELEMENT OPEN "h4" "Summary of RV horizon data".
ELEMENT CLOSE "h4".
```

Multiple sytles in Attribute separated by “;”

```
ATTRIB("style", "color:#DCDCDC;font:12")
```

Color a heading

```
ELEMENT OPEN "h3" ATTRIB ("style", "color:FF0000") "Red text indicates critical errors.".
ELEMENT CLOSE "h3".
```

Create table with border

```
ELEMENT OPEN "table" ATTRIB("border", "3").  Open table with border
ELEMENT "tr"  ATTRIB ("class", "heading")
    "Hzname" TAG "td"  ATTRIB ("class", "begindatagroup") VALUETAG "h4" ,
    "Horizon" TAG "td"  VALUETAG "h4",
    "Texture" TAG "td"  VALUETAG "h4",
    "Bulk" TAG "td" VALUETAG "h4" ATTRIB ("class", "enddatagroup").
ELEMENT "tr"  ATTRIB ("class", "heading")
    "" TAG "td"  ATTRIB ("class", "begindatagroup") VALUETAG "h4" ,
    "Depth" TAG "td"  VALUETAG "h4",
    "" TAG "td"  VALUETAG "h4",
    "Density" TAG "td" VALUETAG "h4" ATTRIB ("class", "enddatagroup").

SECTION
data
 ELEMENT "tr"
    hzname TAG "td" ATTRIB ("class", "begindatagroup") VALUETAG "h4",
    hzdepth TAG "td"  VALUETAG "h4",
    texture TAG "td"   VALUETAG "h4",
    dbthirdbar_r TAG "td"ATTRIB ("class", "enddatagroup") VALUETAG "h4".
end SECTION.
```

Set the report name and initial header

```
SECTION WHEN AT START
DATA
ELEMENT OPEN "SECTION" ATTRIB ("label", "SoilReport").
ELEMENT OPEN "table".
ELEMENT OPEN "thead".
USING basic  "State", "Office", "Project Name", "mukey", "Nationalsym.

ELEMENT CLOSE "thead".
ELEMENT OPEN "tbody".
END SECTION.
```

Define a point location for plotting

```
DEFINE  y 490-(477*(claytotal_r)*0.01).
DEFINE  x 580-577*(sandtotal_r)*0.01-(0.5*(claytotal_r)*555*0.01).
DEFINE  y1 y||" px".
DEFINE  x1 x||" px".
DEFINE  p12 "position:absolute;left:"||x1||";top:"||y1||";color:3366FF".
```

Display point location identified above

```
SECTION when c1==1
data
ELEMENT OPEN "h3"  ATTRIB ("style", p12) "x".
ELEMENT CLOSE "h3".
end SECTION.
add a point as a bullet (special character)
```

```
ELEMENT OPEN "h1" ATTRIB("style",p1)"&#8226;" Special.
ELEMENT CLOSE "h1".
```

Create a table box with different attributes

```
IF counter==1 ELEMENT "tr" hzname TAG "td" ATTRIB("height", “10px”) ATTRIB("width", "75px")
ATTRIB ("bgcolor", "#3E1F0F").
```

Open osd page with concatenated soil name

```
DEFINE osd1 "http://www2.ftw.nrcs.usda.gov/osd/dat/"||fl1||"/"||name1||".html".

SECTION when at start
data
ELEMENT OPEN "a" Attrib("href",osd1) Attrib("target", "_blank") "VISIT OSD SITE".
ELEMENT CLOSE "a".
end SECTION.
```

Create a link in an rectangle area of the report

```
ELEMENT OPEN "img"
ATTRIB("src","https://nrcs.sc.egov.usda.gov/ssra/nssc/Projects/NASIS/triangle.jpg") ATTRIB
("traget", "_blank") ATTRIB ("alt","'NASIS site link broken'") ATTRIB ("width", "600") ATTRIB ("height",
"536") ATTRIB ("usemap","#texture").
ELEMENT CLOSE "img".
ELEMENT OPEN "map" ATTRIB ("name", "texture")ATTRIB ("style","color:blue").
ELEMENT OPEN "area" ATTRIB ("Shape", "rect") ATTRIB("coords","200,200,300,300") ATTRIB ("alt",
"plot ranges")
ATTRIB ("href",
"https://nasis.sc.egov.usda.gov/NasisReportsWebSite/limsreport.aspx?report_name=WEB-
plot%20texture%20with%20ranges%20one%20horizon&#38;chiid2=3609432") ATTRIB ("target",
"_blank") ATTRIB("style","background-color:#B2FF99").
ELEMENT CLOSE "area".
ELEMENT CLOSE "map".
```

Create a email link

```
ELEMENT OPEN "a"
ATTRIB("href","mailto:kevin.godsey@mo.usda.gov?Subject=Error%20in%20report") "PLEASE SEND
ERRORS BY CLICKING HERE".
ELEMENT CLOSE "a".
```

Add a picture to your report

```
ELEMENT OPEN "img"
ATTRIB("src","https://nrcs.sc.egov.usda.gov/ssra/nssc/Projects/NASIS/triangle.jpg") ATTRIB ("traget",
"_blank") ATTRIB ("alt","'NASIS site link broken'") ATTRIB ("width", "600px") ATTRIB ("height", "536px").
ELEMENT CLOSE "img".
```

Create a main report to open sub-reports

```
SECTION
DATA
ELEMENT OPEN "html".
ELEMENT OPEN "body".
INCLUDE "MLRA16_Office":"HTML-plot texture on triangle".
ELEMENT "br".  Line break
ELEMENT OPEN "p" "IF THE POINTS DO NOT PLOT CORRECTLY, RESET YOU DISPLAY.".
ELEMENT CLOSE "p".
ELEMENT OPEN "p" "THE ORIGINAL DISPAY WAS SET TO 1920 BY 1080 ON A DUAL SCREEN.".
ELEMENT CLOSE "p".
INCLUDE "MLRA16_Office":"HTML-texture summary".
INCLUDE "MLRA16_Office":"HTML-rock  summary".
INCLUDE "MLRA16_Office":"HTML-DMU description".
INCLUDE "MLRA16_Office":"HTML-Mini profile description".
INCLUDE "MLRA16_Office":"HTML-plot profile".
ELEMENT CLOSE "body".
ELEMENT CLOSE "html".
end SECTION.
```

Make a web link

```
ELEMENT OPEN "a" ATTRIB("href", "ftp://ftp-fc.sc.egov.usda.gov/NSSC/GDS/GDS_v4_11.pdf")
"GEOMORPHIC DESCRIPTION SYSTEM".
ELEMENT CLOSE "a".
```

Create a heading with message for no data

```
SECTION WHEN NO DATADATA
ELEMENT OPEN h2" “If this message is displayed, critical data is missing for the description".
ELEMENT CLOSE "h2".
END SECTION.
```

HTML Report Format Rules

KEEP option for SECTION is ignored in HTML output

“p” vs “pre” p by its self is a nonformatted paragraph; all spaces, indentations, page breaks etc are ignored. The paragraph will alter its shape based on the size of the html window. “PRE” retains the formatting of the paragraph. So white spaces can be inserted before a word to separate two sections on the same line.

When using the default output the columns are in the order of the select list.

“Tr” stands for table row and
“Td” stands for table data or cell data.

Be careful in using an INPUT file for data needed to aggregate columns that are not unique with the column specification of “none” or duplicates will be eliminated.

Regroup errors cannot regroup because two columns are of different lengths. Usually due to not aggregating the data and duplicates are turned to null.

Column specifications coded values need “-“dashes while other values can be separated with “,” commas.

When creating parameters the term ‘IN” will create check boxes and is used with multiple choice lists. the “=” sign is used for comparing only one value.
Group by and Having cannot use aliases they have to be NASIS table names.

The term special is used to change a special character to its form. The example below changes the charater “&#8226” to a bullet in the output.
EXAMPLE

```
element OPEN "h1" ATTRIb("style",p1)"&#8226;" Special.
```

Example of some of the codes: complete list can be found on many internet sites; look for html special codes.

–	–
—	—
‘	‘
’	’
‚	‚
“	“
”	”
„	„
†	†
‡	‡
•	•
…	…
‰	‰
€	€
™	™

APPENDIX html formatting

The first table below identifies Element that are used with the WSS reports for standardization of output; the second table are standard HTML tags and the third table are color option values.

ELEMENTS

Element	Attribute Name	Attribute Value	Meaning
”section”	“label”	“SoilReport”	Required to identify the outermost section of a soil report
		”Survey_Area”	Data for one survey area. The title of the section is the survey area name. Used in reports that don’t have tables.
		”Map_Unit_Description”	Data for one mapunit in a map unit description report.
		others	The Map Unit Description report has labels to signify the type of data in each section, but they are currently ignored by the formatting program,
”title”	”role”	”suppressTitle”	Do not display a title for the current section. Although the <title> element is required in a <section>, this will suppress display of the title.
”table”	”orient”	”land”	Table is wide and should be displayed in landscape orientation.
“col”	“width”	“n*”	Relative column width expressed as a number followed by an asterisk. A column with width “3*” is 3 times as wide as a column with width “1*”. The exact width of a column depends on the overall width of the table and width of the other columns.
”tr”	”class”	”mapunit”	A table row that begins the data for a map unit.
		”even”	A table row that can be shaded in alternating colors to improve readability. Alternates with “odd”.
		”odd”	A table row that can be shaded in alternating colors to improve readability. Alternates with “even”.
		”interpdata”	A table row containing data about an interpretation in the Survey Area Data Summary report.
		”units”	A table row containing units of measure. It is a type of subheading for

			the table
”td”	”rowspan”	a number	Number of rows in the heading occupied by this cell. Part of standard HTML tables.
	“colspan”	a number	Number of columns in the heading occupied by this cell. Part of standard HTML tables.
	“class”	“begindatagroup”	Cell is the first of a group of cells that are set off visually by a heavier vertical border on the left edge.
		“enddatagroup”	Cell is the last of a group of cells that are set off visually by a heavier vertical border on the right edge.
		“datetime”	Cell contains a date/time field that should be formatted according to the date conventions for the report.
“para”	“role”	“mu-name”	Content is a mapunit symbol or name.
		“comp-name”	Content is a component name.
		“number”	Numeric data, normally displayed right justified in a cell.
		“class-name”	Non-numeric data, such as a class name, normally displayed centered in a cell.
		“hang-list”	Multiple values of character type data which are displayed in a vertical list with hanging indents
		“preservenewlines”	Content is a text field that may include newlines, tabs, and significant spaces. Normally this “white space” is removed. This attribute will preserve the layout of the original text.

TAGS

Tag	Description
<!--...-->	Defines a comment
<!DOCTYPE>	Defines the document type
<a>	Defines an anchor
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines contact information for the author/owner of a document
<area />	Defines an area inside an image-map
<b>	Defines bold text
<base />	Defines a default address or a default target for all links on a page
<bdo>	Defines the text direction
<big>	Defines big text
<blockquote>	Defines a long quotation
<body>	Defines the document's body
 	Defines a single line break
<button>	Defines a push button
<caption>	Defines a table caption
<cite>	Defines a citation
<code>	Defines computer code text
<col />	Defines attribute values for one or more columns in a table
<colgroup>	Defines a group of columns in a table for formatting
<dd>	Defines a description of a term in a definition list
<del>	Defines deleted text
<dfn>	Defines a definition term
	Defines a section in a document
<dl>	Defines a definition list
<dt>	Defines a term (an item) in a definition list
	Defines emphasized text
<fieldset>	Defines a border around elements in a form
<form>	Defines an HTML form for user input
<frame />	Defines a window (a frame) in a frameset
<frameset>	Defines a set of frames
<h1> to <h6>	Defines HTML headings
<head>	Defines information about the document
<hr />	Defines a horizontal line
<html>	Defines an HTML document
<i>	Defines italic text
<iframe>	Defines an inline frame
<img />	Defines an image
<input />	Defines an input control
<ins>	Defines inserted text

<kbd>	Defines keyboard text
<label>	Defines a label for an input element
<legend>	Defines a caption for a fieldset element
	Defines a list item
<link />	Defines the relationship between a document and an external resource
<map>	Defines an image-map
<meta />	Defines metadata about an HTML document
<noframes>	Defines an alternate content for users that do not support frames
<noscript>	Defines an alternate content for users that do not support client-side scripts
<object>	Defines an embedded object
	Defines an ordered list
<optgroup>	Defines a group of related options in a select list
<option>	Defines an option in a select list
<p>	Defines a paragraph
<param />	Defines a parameter for an object
<pre>	Defines preformatted text
<q>	Defines a short quotation
<samp>	Defines sample computer code
<script>	Defines a client-side script
<select>	Defines a select list (drop-down list)
<small>	Defines small text
	Defines a section in a document
	Defines strong text
<style>	Defines style information for a document
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<table>	Defines a table
<tbody>	Groups the body content in a table
<td>	Defines a cell in a table
<textarea>	Defines a multi-line text input control
<tfoot>	Groups the footer content in a table
<th>	Defines a header cell in a table
<thead>	Groups the header content in a table
<title>	Defines the title of a document
<tr>	Defines a row in a table
<tt>	Defines teletype text
	Defines an unordered list
<var>	Defines a variable part of a text

COLOR CODING

AliceBlue	#F0F8FF
AntiqueWhite	#FAEBD7
Aqua	#00FFFF
Aquamarine	#7FFFD4
Azure	#F0FFFF
Beige	#F5F5DC
Bisque	#FFE4C4
Black	#000000
BlanchedAlmond	#FFEBCD
Blue	#0000FF
BlueViolet	#8A2BE2
Brown	#A52A2A
BurlyWood	#DEB887
CadetBlue	#5F9EA0
Chartreuse	#7FFF00
Chocolate	#D2691E
Coral	#FF7F50
CornflowerBlue	#6495ED
Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B
DarkGray	#A9A9A9
DarkGrey	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
Darkorange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B
DarkSlateGray	#2F4F4F
DarkSlateGrey	#2F4F4F

DarkTurquoise	#00CED1	
DarkViolet	#9400D3	
DeepPink	#FF1493	
DeepSkyBlue	#00BFFF	
DimGray	#696969	
DodgerBlue	#1E90FF	
FireBrick	#B22222	
FloralWhite	#FFFAF0	
ForestGreen	#228B22	
Fuchsia	#FF00FF	
Gainsboro	#DCDCDC	
GhostWhite	#F8F8FF	
Gold	#FFD700	
GoldenRod	#DAA520	
Gray	#808080	
Grey	#808080	
Green	#008000	
GreenYellow	#ADFF2F	
HoneyDew	#F0FFF0	
HotPink	#FF69B4	
IndianRed	#CD5C5C	
Indigo	#4B0082	
Ivory	#FFFFF0	
Khaki	#F0E68C	
Lavender	#E6E6FA	
LavenderBlush	#FFF0F5	
LawnGreen	#7CFC00	
LemonChiffon	#FFFACD	
LightBlue	#ADD8E6	
LightCoral	#F08080	
LightCyan	#E0FFFF	
LightGoldenRodYellow	#FAFAD2	
LightGray	#D3D3D3	
LightGrey	#D3D3D3	
LightGreen	#90EE90	
LightPink	#FFB6C1	
LightSalmon	#FFA07A	
LightSeaGreen	#20B2AA	
LightSkyBlue	#87CEFA	

LightSlateGray	#778899	
LightSlateGrey	#778899	
LightSteelBlue	#B0C4DE	
LightYellow	#FFFFE0	
Lime	#00FF00	
LimeGreen	#32CD32	
Linen	#FAF0E6	
Magenta	#FF00FF	
Maroon	#800000	
MediumAquaMarine	#66CDAA	
MediumBlue	#0000CD	
MediumOrchid	#BA55D3	
MediumPurple	#9370D8	
MediumSeaGreen	#3CB371	
MediumSlateBlue	#7B68EE	
MediumSpringGreen	#00FA9A	
MediumTurquoise	#48D1CC	
MediumVioletRed	#C71585	
MidnightBlue	#191970	
MintCream	#F5FFFA	
MistyRose	#FFE4E1	
Moccasin	#FFE4B5	
NavajoWhite	#FFDEAD	
Navy	#000080	
OldLace	#FDF5E6	
Olive	#808000	
OliveDrab	#6B8E23	
Orange	#FFA500	
OrangeRed	#FF4500	
Orchid	#DA70D6	
PaleGoldenRod	#EEE8AA	
PaleGreen	#98FB98	
PaleTurquoise	#AFEEEE	
PaleVioletRed	#D87093	
PapayaWhip	#FFEFD5	
PeachPuff	#FFDAB9	
Peru	#CD853F	
Pink	#FFC0CB	
Plum	#DDA0DD	

PowderBlue	#B0E0E6	
Purple	#800080	
Red	#FF0000	
RosyBrown	#BC8F8F	
RoyalBlue	#4169E1	
SaddleBrown	#8B4513	
Salmon	#FA8072	
SandyBrown	#F4A460	
SeaGreen	#2E8B57	
SeaShell	#FFF5EE	
Sienna	#A0522D	
Silver	#C0C0C0	
SkyBlue	#87CEEB	
SlateBlue	#6A5ACD	
SlateGray	#708090	
Snow	#FFFAFA	
SpringGreen	#00FF7F	
SteelBlue	#4682B4	
Tan	#D2B48C	
Teal	#008080	
Thistle	#D8BFD8	
Tomato	#FF6347	
Turquoise	#40E0D0	
Violet	#EE82EE	
Wheat	#F5DEB3	
White	#FFFFFF	
WhiteSmoke	#F5F5F5	
Yellow	#FFFF00	
YellowGreen	#9ACD32	

Suggested reading.

http://www.w3schools.com/xml/xml_whatis.asp

http://www.htmlhelp.com/reference/html40/

http://www.mountaindragon.com/html/tables.htm

http://www.htmlgoodies.com/

http://www.htmlcompendium.org/index.htm

The report scripting will be similar to writing NASIS reports, however the formatting of the reports is using XML and HTML scripting. This will be new to many.

Web URL reports

Any report in NASIS can be tweaked to work as a URL output through windows explore.

The output can be text format and is displayed in explorer window.
URL reports can be called with python scripts in ARCMAP and the data be incorporated with other criteria.

The parameters are slightly different. The program cannot convert text to code so if a parameter is a coded value you have to enter the code not the code name.

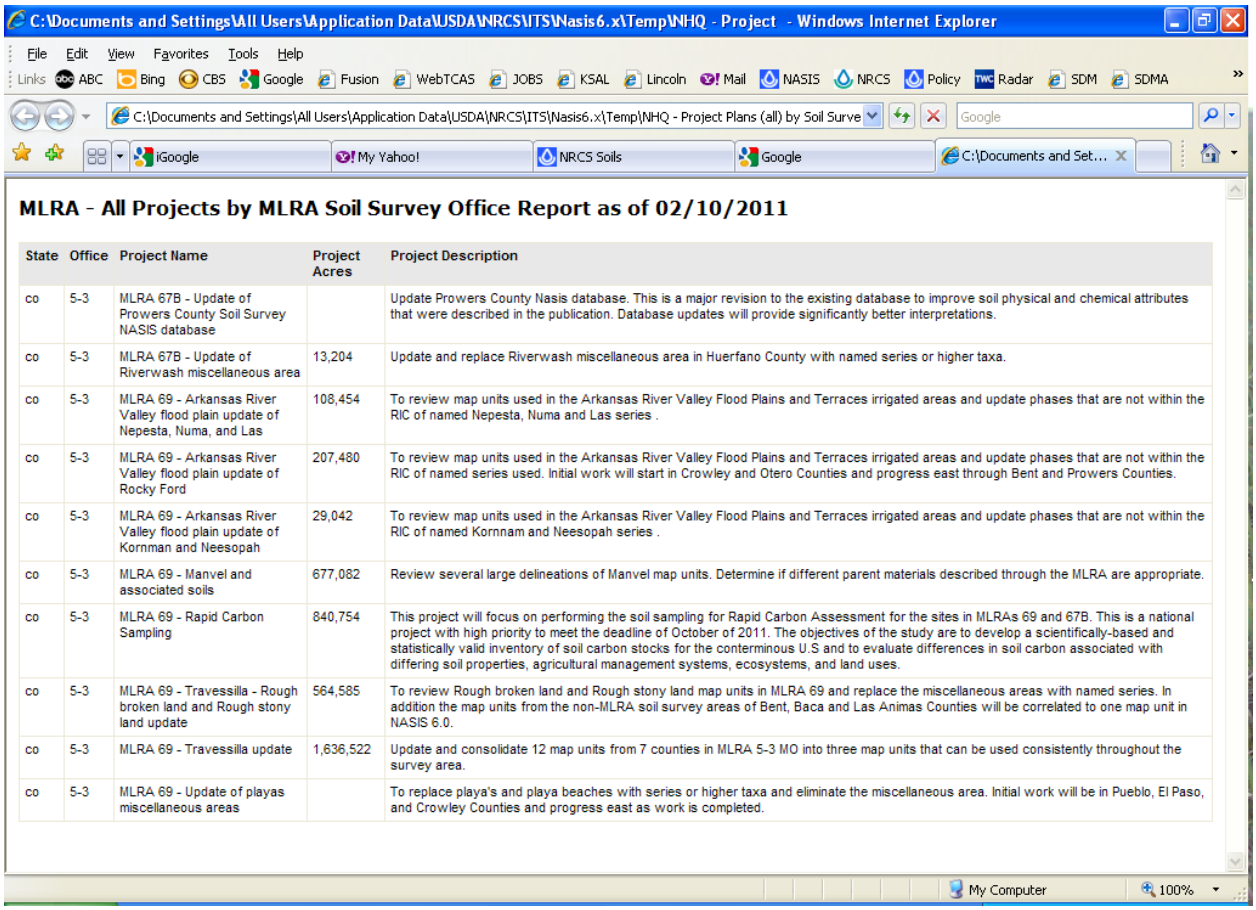
If the URL has the parameter identified with the ampersand (&) the report will run.
https://nasis.sc.egov.usda.gov/NasisReportsWebSite/limsreport.aspx?report_name=WEB-Mapunits%20by%20area%20symbol&area_sym=MO123

If the URL report does not have a parameter identified it will open a parameter page with fill in boxes.

https://nasis.sc.egov.usda.gov/NasisReportsWebSite/limsreport.aspx?report_name=Prime%20and%20statewide%20soils%20by%20area%20and%20musym

These reports have to be in one of two folders in NASIS: NSSL and the NSSC Data folders.

NASIS 6 provides the ability to write reports in text, html or xml programming languages. The CVIR book discusses the methods of writing using html and xml. This section will discuss the programming needs for html report coding.



MLRA - All Projects by MLRA Soil Survey Office Report as of 02/10/2011

State	Office	Project Name	Project Acres	Project Description
co	5-3	MLRA 67B - Update of Prowers County Soil Survey NASIS database		Update Prowers County Nasis database. This is a major revision to the existing database to improve soil physical and chemical attributes that were described in the publication. Database updates will provide significantly better interpretations.
co	5-3	MLRA 67B - Update of Riverwash miscellaneous area	13,204	Update and replace Riverwash miscellaneous area in Huerfano County with named series or higher taxa.
co	5-3	MLRA 69 - Arkansas River Valley flood plain update of Nepesta, Numa, and Las	108,454	To review map units used in the Arkansas River Valley Flood Plains and Terraces irrigated areas and update phases that are not within the RIC of named Nepesta, Numa and Las series .
co	5-3	MLRA 69 - Arkansas River Valley flood plain update of Rocky Ford	207,480	To review map units used in the Arkansas River Valley Flood Plains and Terraces irrigated areas and update phases that are not within the RIC of named series used. Initial work will start in Crowley and Otero Counties and progress east through Bent and Prowers Counties.
co	5-3	MLRA 69 - Arkansas River Valley flood plain update of Kornman and Neesopah	29,042	To review map units used in the Arkansas River Valley Flood Plains and Terraces irrigated areas and update phases that are not within the RIC of named Kornnam and Neesopah series .
co	5-3	MLRA 69 - Manvel and associated soils	677,082	Review several large delineations of Manvel map units. Determine if different parent materials described through the MLRA are appropriate.
co	5-3	MLRA 69 - Rapid Carbon Sampling	840,754	This project will focus on performing the soil sampling for Rapid Carbon Assessment for the sites in MLRAs 69 and 67B. This is a national project with high priority to meet the deadline of October of 2011. The objectives of the study are to develop a scientifically-based and statistically valid inventory of soil carbon stocks for the conterminous U.S and to evaluate differences in soil carbon associated with differing soil properties, agricultural management systems, ecosystems, and land uses.
co	5-3	MLRA 69 - Travessilla - Rough broken land and Rough stony land update	564,585	To review Rough broken land and Rough stony land map units in MLRA 69 and replace the miscellaneous areas with named series. In addition the map units from the non-MLRA soil survey areas of Bent, Baca and Las Animas Counties will be correlated to one map unit in NASIS 6.0.
co	5-3	MLRA 69 - Travessilla update	1,636,522	Update and consolidate 12 map units from 7 counties in MLRA 5-3 MO into three map units that can be used consistently throughout the survey area.
co	5-3	MLRA 69 - Update of playas miscellaneous areas		To replace playa's and playa beaches with series or higher taxa and eliminate the miscellaneous area. Initial work will be in Pueblo, El Paso, and Crowley Counties and progress east as work is completed.

NHQ - Project Plans (all) by Soil Survey Office html

```
PARAMETER sso ELEMENT areasymbol  PROMPT "Office Responsible e.g. 5-2".
BASE TABLE Project.

EXEC SQL
SELECT areasymbol, areaname, projectname, projectdesc, projectiid, stateresponsible
FROM REAL area
INNER JOIN REAL project by mlra_sso
LEFT OUTER JOIN REAL projectmilestone by default
LEFT OUTER JOIN REAL milestonetype by default
WHERE projectname matches "MLRA*"
and areasymbol matches sso ;
SORT BY stateresponsible, areasymbol, projectname
AGGREGATE ROWS by areaname, projectname.

EXEC SQL
SELECT liid, muiid, musym, muname, muacres
FROM REAL area muarea
INNER JOIN REAL project by mlra_sso
INNER JOIN REAL projectmapunit by default
```

```
INNER JOIN REAL mapunit by default
INNER JOIN REAL lmapunit by default
INNER JOIN REAL legend by default;
AGGREGATE COLUMN muiid NONE, muacres NONE.

DEFINE asum                ARRAYSUM(muacres).
DEFINE dt                  TODAY.
DEFINE location            areasymbol || ": " || areaname.
DEFINE title               "MLRA - All Projects by MLRA Soil Survey Office Report".
DEFINE state               CODENAME(stateresponsible).
TEMPLATE basic             TAG "td" ELEMENT "tr" FIELD, FIELD, FIELD, FIELD, FIELD.

SECTION WHEN AT START                                  #sets the report name and initial header
DATA
ELEMENT OPEN "section" ATTRIB ("label", "SoilReport").   #opens section
ELEMENT "title" title, " as of ", dt.
ELEMENT OPEN "table".                                  #opens table
ELEMENT OPEN "thead".                                  #opens table header
USING basic     "State",
                "Office",
                "Project Name",
                "Project Acres",
                "Project Description" TAG "td" ATTRIB ("role", "center").
ELEMENT CLOSE "thead".                                   #closes table header
ELEMENT OPEN "tbody".                                  #opens table body
END SECTION.
SECTION                                                  #adds the data to the report
DATA
          USING basic
         state,
         areasymbol,
         projectname,
         asum,
         projectdesc.
END SECTION.
SECTION when at end                                    #closes the report
DATA
ELEMENT CLOSE "tbody".                                 #closes table body
ELEMENT CLOSE "table".                                 #closes table
ELEMENT CLOSE "section".                                 #closes section
END SECTION.                                             #ends section
```

Exercises

Queries

Arithmetic

FROM area, legend, lmapunit, mapunit, correlation, data_mapunit, component, chorizon
WHERE area.area_symbol MATCHES ? AND
legend.soil_survey_area_status IN (?) AND
correlation.representative_dmu = "yes" AND
mapunit.mapunit_status != "additional" AND
component.major_component_flag IN (?) AND
((**ROUND** (sandvc_r + sandco_r + sandmed_r + sandfine_r + sandvf_r, 1)) != ROUND (sandtotal_r, 1)) AND
JOIN area TO legend AND
JOIN legend TO lmapunit AND
JOIN lmapunit to mapunit AND
JOIN mapunit TO correlation AND
JOIN correlation TO data_mapunit AND
JOIN data_mapunit TO component AND
JOIN component TO chorizon

FROM chorizon, area, legend, component, correlation, areatype, mapunit, data_mapunit, lmapunit
WHERE area.areasymbol MATCHES ? "AREA SYMBOL eg. mo*" AND
mapunit.mapunit_status IN (?) AND areatypename matches "non-mlra*" and
correlation.representative_dmu = "yes" and
ABS(sandtotal_r-(sandvc_r+sandco_r+sandmed_r+sandfine_r+sandvf_r)) >0.09 and
JOIN areatype TO area AND
JOIN area TO legend AND
JOIN legend to lmapunit AND
JOIN lmapunit to mapunit AND
JOIN mapunit TO correlation AND
JOIN correlation TO data_mapunit AND
JOIN data_mapunit TO component AND
JOIN component TO chorizon

Compare sieve to PSDA
FROM area, legend, lmapunit, mapunit, correlation, data_mapunit, component, chorizon
WHERE area.area_symbol MATCHES ? AND
mapunit.mapunit_status != "additional" AND
correlation.representative_dmu = 1 AND
component.component_name MATCHES ? AND
component.major_component_flag = 1 AND
sieveno200_r != ((sandvf_r/2) + silttotal_r + claytotal_r) and sandtotal_r is not null and
JOIN area TO legend AND
JOIN legend TO lmapunit AND
JOIN lmapunit to mapunit AND
JOIN mapunit TO correlation AND
JOIN correlation TO data_mapunit AND
JOIN data_mapunit TO component AND
JOIN component TO chorizon

NOT EXISTS
FROM areatype, area, legend, mapunit, correlation, data_mapunit, component
WHERE area.area_symbol matches ? AND

```
legend.legend_suitability_for_use = "2" AND
mapunit.mapunit_status != "additional" AND
correlation.representative_dmu = "yes" AND
area_type_name = "Non-MLRA Soil Survey Area" AND
component.major_component_flag = "yes" AND
component.component_kind != "miscellaneous area" and
NOT EXISTS (SELECT * FROM component_parent_material, component_parent_material_grp WHERE JOIN component TO
component_parent_material_grp and join component_parent_material_grp to component_parent_material) and
JOIN areatype to area AND
JOIN area TO legend AND
JOIN legend TO mapunit AND
JOIN mapunit TO correlation AND
JOIN correlation TO data_mapunit AND
JOIN data_mapunit to component
```

```
FROM chorizon, mapunit, corestrictions, area, component, areatype, legend, correlation, data_mapunit, lmapunit
WHERE  areasymbol MATCHES ? "AREA SYMBOL eg. mo*"  and
areatype.areatypename MATCHES "non-mlra*" and mapunit.mapunit_status IN (?) AND correlation.repdmu=1 and  reskind IN
("bedrock, lithic") and hzname  matches "*R*" and (hzdept_r!=resdept_r or hzdept_l!=resdept_l or hzdept_h!=resdept_h) and
Join area TO areatype and
JOIN area to legend and
JOIN legend to lmapunit AND
JOIN lmapunit to mapunit AND
join mapunit to correlation and
join correlation to data_mapunit and
join data_mapunit to component and
join component to chorizon and
join component to corestrictions
```

```
FROM chorizon, area, legend, mapunit, correlation, data_mapunit, component, areatype, lmapunit
WHERE area.area_symbol MATCHES ? "AREA SYMBOL eg. mo*" AND areatypename matches "non-mlra*" and
chorizon.horizon_depth_to_top_r = 0 AND
mapunit.mapunit_status IN (?) AND
JOIN area TO legend AND repdmu="yes" and
JOIN area to areatype and
JOIN legend to lmapunit AND
JOIN lmapunit to mapunit AND
JOIN mapunit TO correlation AND
JOIN correlation TO data_mapunit AND
JOIN data_mapunit TO component AND
JOIN component TO chorizon AND
EXISTS (SELECT chorizon_iid_ref FROM chorizon_texture_group
WHERE JOIN chorizon TO chorizon_texture_group
GROUP BY chorizon_iid_ref
HAVING COUNT(*) > 1)
```

```
FROM pedon, site, nasisgroup, OUTER site_association_site, OUTER transect, siteobs
WHERE nasisgroup.grpname matches ? "group name" and
        pedon.pedon_type  IN (?) AND
        JOIN pedon to siteobs and
        JOIN siteobs to site and
        JOIN pedon to nasisgroup AND
        JOIN pedon TO transect AND
        JOIN site TO site_association_site
and nasissite.nasissitename MATCHES ? and
```

JOIN pedon to nasissite

Reports

Exercise 2: multiple tables and SORT

```
EXEC SQL
SELECT nationalmusym, muname, dmudesc
FROM mapunit, correlation, datamapunit
WHERE JOIN mapunit TO correlation AND
JOIN correlation TO datamapunit;
SORT BY muname SYM.
```

Exercise 3

```
EXEC SQL
SELECT nationalmusym, muname, compname, comppct_r, repdmu
FROM mapunit, OUTER(correlation, datamapunit, component)
WHERE repdmu = 1 AND
JOIN mapunit TO correlation AND
JOIN correlation TO datamapunit AND
JOIN datamapunit to component;
SORT BY nationalmusym SYM, comppct_r DESC.
```

Exercise 5a

```
EXEC SQL
SELECT musym, muname, compname, comppct_r, repdmu
FROM legend, lmapunit, mapunit, OUTER(correlation, datamapunit, component)
WHERE repdmu = 1 AND
JOIN legend to lmapunit AND
JOIN lmapunit to mapunit AND
JOIN mapunit TO correlation AND
JOIN correlation TO datamapunit AND
JOIN datamapunit to component;
SORT BY musym SYM, comppct_r DESC
AGGREGATE ROWS BY musym COLUMN compname NONE.

TEMPLATE dline SEPARATOR "|"
AT LEFT FIELD WIDTH 8, FIELD WIDTH 40, FIELD WIDTH 7, FIELD WIDTH 15, "".

SECTION
 HEADING
 AT LEFT "_" REPEAT WIDTH 74.
 USING dline.
 USING dline
   "Map Symbol" ALIGN CENTER,
   "Map Unit Name" ALIGN CENTER,
   "Comp %" ALIGN CENTER,
   "Component Name" ALIGN CENTER.
 USING dline "_" REPEAT,  "_" REPEAT, "_" REPEAT, "_" REPEAT.

 DATA
 USING dline
   musym ALIGN RIGHT INDENT 1,
   muname INDENT -1,
   comppct_r INDENT -1,
   compname INDENT 1.
```

```
END SECTION.
```

Exercise 8a

```
BASE TABLE component.

   EXEC SQL
   SELECT musym, compname, comppct_r
   FROM legend, lmapunit, mapunit, correlation, datamapunit, component
   WHERE JOIN legend to lmapunit
            AND JOIN lmapunit to mapunit
            AND JOIN mapunit TO correlation
   AND JOIN correlation TO datamapunit
   AND JOIN datamapunit TO component
   AND repdmu = 1;
   SORT BY musym SYM, comppct_r DESC
   AGGREGATE ROWS BY musym COLUMN compname FIRST.

DERIVE shwt FROM rv USING "NSSC Pangaea" : "DEPTH TO HIGH WATER TABLE MINIMUM".

PAGE LENGTH UNLIMITED WIDTH UNLIMITED.

SECTION WHEN not isnull(shwt)
    DATA
    AT LEFT
            musym QUOTED WIDTH UNLIMITED,
            compname QUOTED WIDTH UNLIMITED SEPARATOR ",",
            shwt NO COMMA WIDTH UNLIMITED SEPARATOR ",".
END SECTION.

Exercise 14a
      EXEC SQL
         select utransectid, soinmassamp, count(*) soilcount
         from REAL transect, REAL pedon
         WHERE utransectid matches "07905KS185-5910*" and join transect to pedon
         group by utransectid, soinmassamp;
         sort by utransectid, soinmassamp
         aggregate rows by utransectid crosstab soinmassamp cells soilcount.

         PAGE WIDTH 15 in.

         assign soilcount if isnull(soilcount) then 0 else soilcount.
         define tot ARRAYSUM(soilcount).
         define pct (soilcount/tot) * 100.


         template basic separator "|"
         at left field width 20, field width 25, field width 10 align center,
         field width 10 align center.

         SECTION
         HEADING
         USING basic "User Transect ID", "Soil Name as Sampled", "Number of",
         "Percent of".
         USING basic "","","Pedons", "Transect".
         DATA
         SKIP 1 LINE.
         USING basic utransectid, soinmassamp, soilcount, pct.
```

```
   end section.

#07905KS185-5910
#07105NE125-2171*


Exercise 12a.

exec sql select areasymbol, CODELABEL(farmlndcl) class, muacres
from areatype, area, legend, lmapunit, mapunit
where join areatype to area and join area to legend and
join legend to lmapunit and join lmapunit to mapunit and
areatypename matches "Non-MLRA*" and mustatus != "additional";
sort by areasymbol, class
aggregate rows areasymbol, class column muacres sum.

TEMPLATE basic SEPARATOR "|"
     AT LEFT FIELD WIDTH 8, FIELD WIDTH 33, FIELD WIDTH 10, "".

    SECTION
     HEADING
      AT LEFT "Farmland Classification Acres" WIDTH 54 ALIGN CENTER.
      SKIP 1 LINE.
      AT LEFT "_" WIDTH 54 REPEAT.
      USING basic.
      USING basic
       "Area Symbol" ALIGN CENTER,
       "Classification" ALIGN CENTER,
       "Total Acres" ALIGN CENTER.
      USING basic "_" REPEAT, "_" REPEAT, "_" REPEAT.

     DATA
      USING basic
       areasymbol INDENT 1 SUPPRESS,
       class INDENT -1,
       muacres DECIMAL 2 REPLACE NULL WITH      "---".
    END SECTION.

    SECTION WHEN LAST OF areasymbol
     DATA
      USING basic.
    END SECTION.

Exercise 20a

EXEC SQL
SELECT areaname, musym, nationalmusym, muname, compname, comppct_r, repdmu, CODELABEL(reskind) kind
FROM area, legend, lmapunit, mapunit, correlation, datamapunit, OUTER (component, corestrictions)
WHERE repdmu = 1 AND
JOIN area to legend and
JOIN legend to lmapunit and
JOIN lmapunit to mapunit AND
JOIN mapunit TO correlation AND
JOIN correlation TO datamapunit AND
JOIN datamapunit to component AND
JOIN component to corestrictions;
SORT BY nationalmusym SYM, comppct_r DESC
AGGREGATE ROWS BY nationalmusym COLUMN compname NONE.
```

```
TEMPLATE dline SEPARATOR "|"
AT LEFT FIELD WIDTH 8, FIELD WIDTH 36, FIELD WIDTH 15, FIELD WIDTH 11, "".

SECTION
 HEADING
 AT LEFT "_" REPEAT WIDTH 74.
 USING dline.
 USING dline
   "NAT Map Symbol" ALIGN CENTER,
   "Map Unit Name" ALIGN CENTER,
   "Component Name" ALIGN CENTER,
   "Restriction" ALIGN CENTER.
 USING dline "_" REPEAT,  "_" REPEAT, "_" REPEAT,  "_" REPEAT.

 DATA
 USING dline
   nationalmusym ALIGN RIGHT INDENT 1,
   muname INDENT -1,
   compname INDENT 1,
   kind.
END SECTION.
```

ERROR MESSAGES

When text is identified on the general tab and html format on the report tab

```
Preparing to run report "HTML-soil extent map" on the local database...
ERROR
While running ReportScript "HTML-soil extent map"
ELEMENT cannot be used in a report whose output format is TXT
```

When html is checked on the general tab and the data is text

```
ERROR
While running ReportScript "UTIL - Topsoiling"
  with coiid=1195676
A USING statement has more columns than the TEMPLATE "tier1"
```

When page width and length are not unlimited and output is text

```
ERROR
While running ReportScript "test errrors"
Count must be positive and count must refer to a location within the string/array/collection.
Parameter name: count
```

Regroup error when the second column is not aggregated to none.

```
ERROR
While running ReportScript "test errrors"
Error in DEFINE grpgrp
Variables used in REGROUP have unequal numbers of values (2 vs. 1)
```

When a period is missing the error references the first column of the line below the line that is missing the period.

```
Preparing to run report "test errrors" on the local database...
While verifying ReportScript "test errrors"
Syntax error at line 16, column 1
'page' is out of context.
```

When a Right parenthesis is missing the error reads

```
Preparing to run report "test errrors" on the local database...
While verifying ReportScript "test errrors"
Error in query beginning at line 3
expecting "RPAREN", found 'FROM'
```

If the left parenthesis is missing, the error reads

```
Preparing to run report "test errrors" on the local database...
While verifying ReportScript "test errrors"
Syntax error at line 4, column 28
')' is out of context.
```

When you codelabel a column but forget to give it an alias

```
Preparing to run report "test errrors" on the local database...
While verifying ReportScript "test errrors"
Error in query beginning at line 3
An expression in the Select clause must have an alias
```

When you have an extra comma at the end of the select list

```
Preparing to run report "test errrors" on the local database...
While verifying ReportScript "test errrors"
Syntax error at line 5, column 1
'FROM' is out of context.
```

If you forget to open the HTML and body you get the following message

```
Preparing to run report "test report" on the local database...
ERROR
While running ReportScript "test report"
There are multiple root elements. Line 2, position 2.
```