Access to PC Files

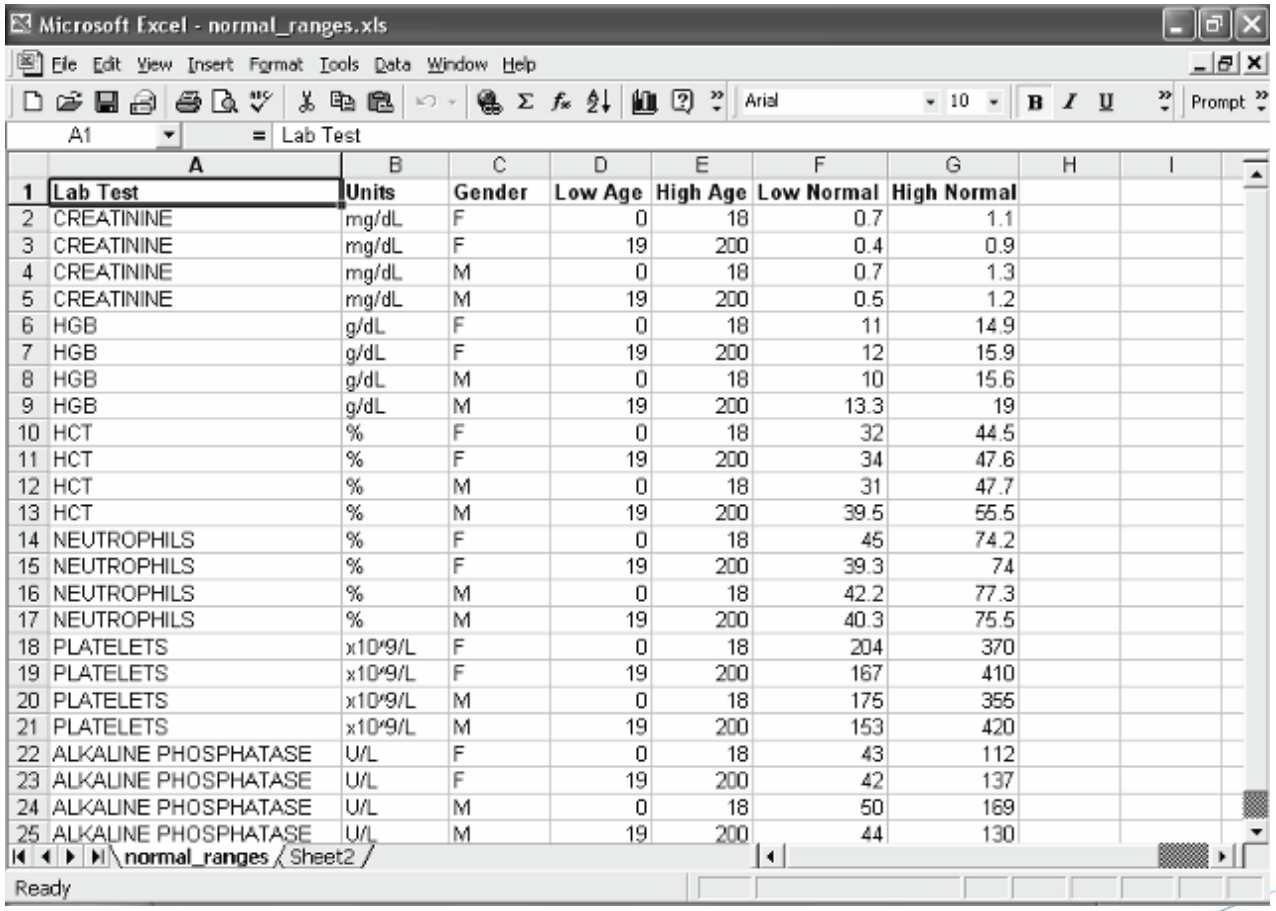
Contents

Introduction (1)

- SAS/ACCESS for PC files enables you to read data from PC files, to use that data in SAS reports or applications, and to use SAS data sets to create PC files in various formats.
- Because Microsoft Office is so widely used, it is sometimes necessary for you to import data directly from Microsoft Excel or Microsoft Access. Here in this section, we will take PC files of Microsoft Excel or Microsoft Access as example.
- SAS provides several ways to read Microsoft Excel and Access files. Some commonly used SAS tools include:
  - the LIBNAME statement
  - the Import Wizard/PROC IMPORT
  - the SQL Pass-Through Facility
  - SAS Enterprise Guide.

Introduction (2)

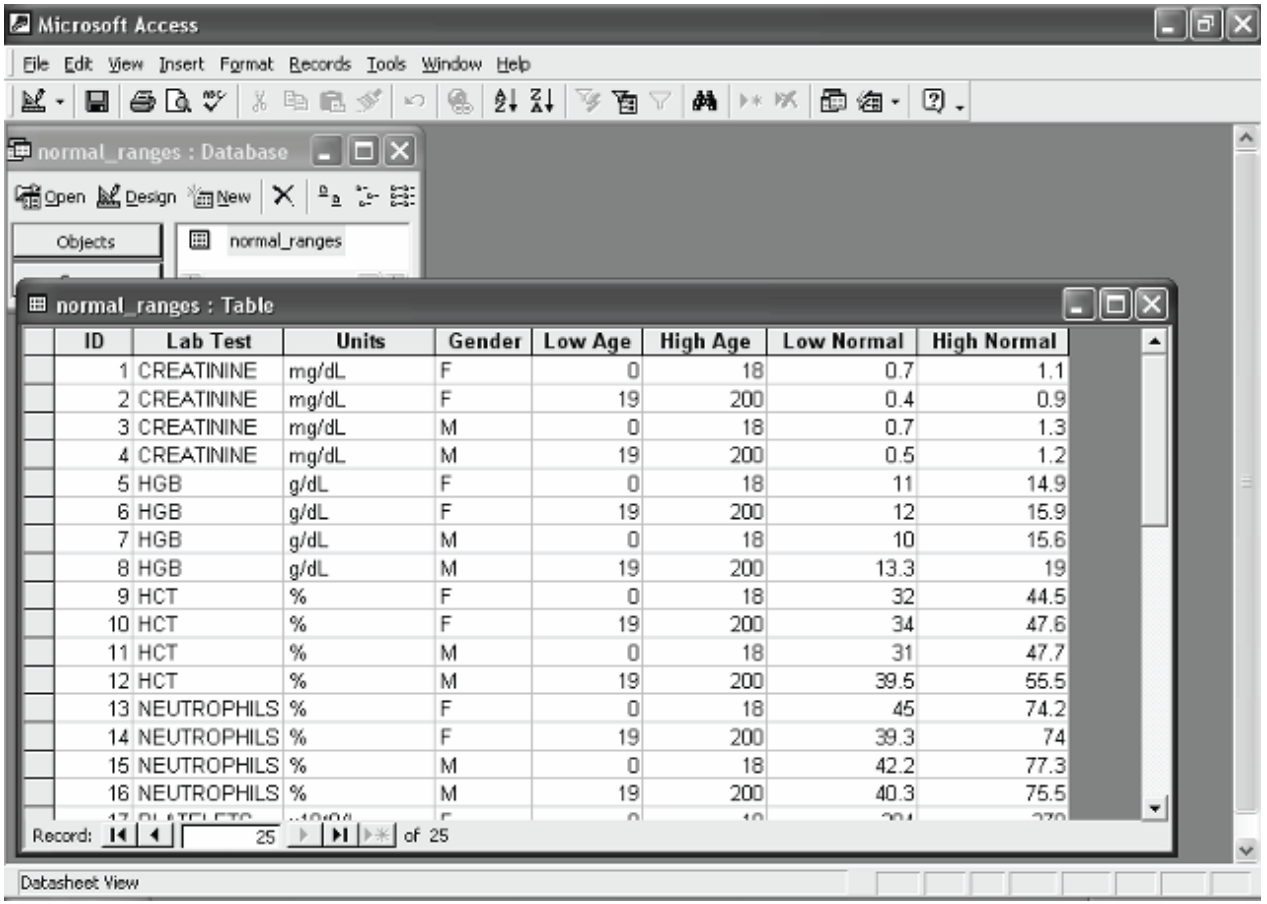
In Microsoft Excel, the lab normal data file might look like the following:



	A	B	C	D	E	F	G	H	I
1	Lab Test	Units	Gender	Low Age	High Age	Low Normal	High Normal		
2	CREATININE	mg/dL	F	0	18	0.7	1.1		
3	CREATININE	mg/dL	F	19	200	0.4	0.9		
4	CREATININE	mg/dL	M	0	18	0.7	1.3		
5	CREATININE	mg/dL	M	19	200	0.5	1.2		
6	HGB	g/dL	F	0	18	11	14.9		
7	HGB	g/dL	F	19	200	12	15.9		
8	HGB	g/dL	M	0	18	10	15.6		
9	HGB	g/dL	M	19	200	13.3	19		
10	HCT	%	F	0	18	32	44.5		
11	HCT	%	F	19	200	34	47.6		
12	HCT	%	M	0	18	31	47.7		
13	HCT	%	M	19	200	39.5	55.5		
14	NEUTROPHILS	%	F	0	18	45	74.2		
15	NEUTROPHILS	%	F	19	200	39.3	74		
16	NEUTROPHILS	%	M	0	18	42.2	77.3		
17	NEUTROPHILS	%	M	19	200	40.3	75.5		
18	PLATELETS	x10^9/L	F	0	18	204	370		
19	PLATELETS	x10^9/L	F	19	200	167	410		
20	PLATELETS	x10^9/L	M	0	18	175	355		
21	PLATELETS	x10^9/L	M	19	200	153	420		
22	ALKALINE PHOSPHATASE	U/L	F	0	18	43	112		
23	ALKALINE PHOSPHATASE	U/L	F	19	200	42	137		
24	ALKALINE PHOSPHATASE	U/L	M	0	18	50	169		
25	ALKALINE PHOSPHATASE	U/L	M	19	200	44	130		

Introduction (3)

- In Microsoft Access the lab normal data might look like this:



ID	Lab Test	Units	Gender	Low Age	High Age	Low Normal	High Normal
1	CREATININE	mg/dL	F	0	18	0.7	1.1
2	CREATININE	mg/dL	F	19	200	0.4	0.9
3	CREATININE	mg/dL	M	0	18	0.7	1.3
4	CREATININE	mg/dL	M	19	200	0.5	1.2
5	HGB	g/dL	F	0	18	11	14.9
6	HGB	g/dL	F	19	200	12	15.9
7	HGB	g/dL	M	0	18	10	15.6
8	HGB	g/dL	M	19	200	13.3	19
9	HCT	%	F	0	18	32	44.5
10	HCT	%	F	19	200	34	47.6
11	HCT	%	M	0	18	31	47.7
12	HCT	%	M	19	200	39.5	55.5
13	NEUTROPHILS	%	F	0	18	45	74.2
14	NEUTROPHILS	%	F	19	200	39.3	74
15	NEUTROPHILS	%	M	0	18	42.2	77.3
16	NEUTROPHILS	%	M	19	200	40.3	75.5

LIBNAME Statement (1)

Beginning with SAS 9.1, the LIBNAME statement can be used to simply map to an Excel or Access database. For example, the following SAS code reads in and then prints the lab normal file normal_ranges.xls.

```
libname xlsfile EXCEL "C:\normal_ranges.xls";
proc contents
   data = xlsfile._all_;
run;
proc print
   data = XLSFILE.'normal_ranges$'n;
run;
```

- Note that the “EXCEL” engine specification is optional, because SAS would read the “.xls” extension in the physical filename and assume it indicates a Microsoft Excel file.
- Also note that the “xlsfile” libref refers to the entire Excel workbook. In the subsequent PROC PRINT, the “normal_ranges” must be specified so SAS will know which Excel worksheet to read.
- The data set/worksheet name in the PROC PRINT looks odd because of the existence of a special “$” character, which is normally not allowed as part of a data set name.

LIBNAME Statement (2)

The normals_ranges.mdb Microsoft Access file could be read in with the following similar SAS code.

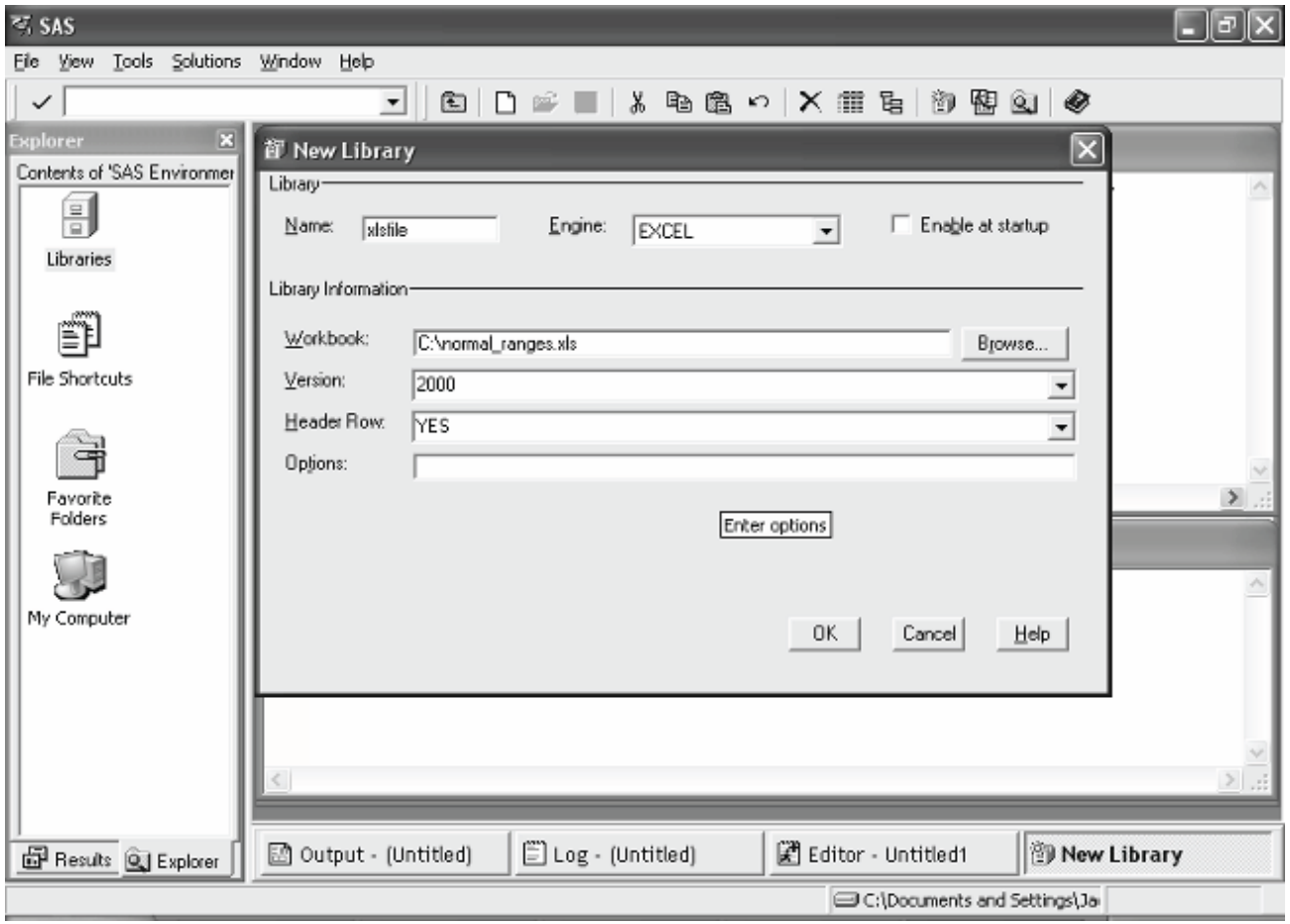
```
libname accfile ACCESS "C:\normal_ranges.mdb";
proc contents
   data = accfile._all_;
run;
proc print
   data = accfile.normal_ranges;
run;
```

- Again, the “ACCESS” specification as a LIBNAME engine is optional, as the libref would default to Microsoft Access because “.mdb” is in the physical filename.
- Note that the ACCESS LIBNAME engine seems by default to import all text fields as 255 characters in length.
- Also note that all dates that come from Microsoft Access via the ACCESS engine are represented in SAS as SAS datetime fields. This is because Access has only datetime fields compared with SAS, which has date, time, and datetime variables.

LIBNAME Statement (3)

SAS ACCESS and EXCEL librefs can be specified interactively by right-clicking on the Libraries icon in the SAS Explorer window and completing the parameters in the New Library window. You can define a libref called xlsfile that points to normal_ranges like following picture.

LIBNAME Statement (4)



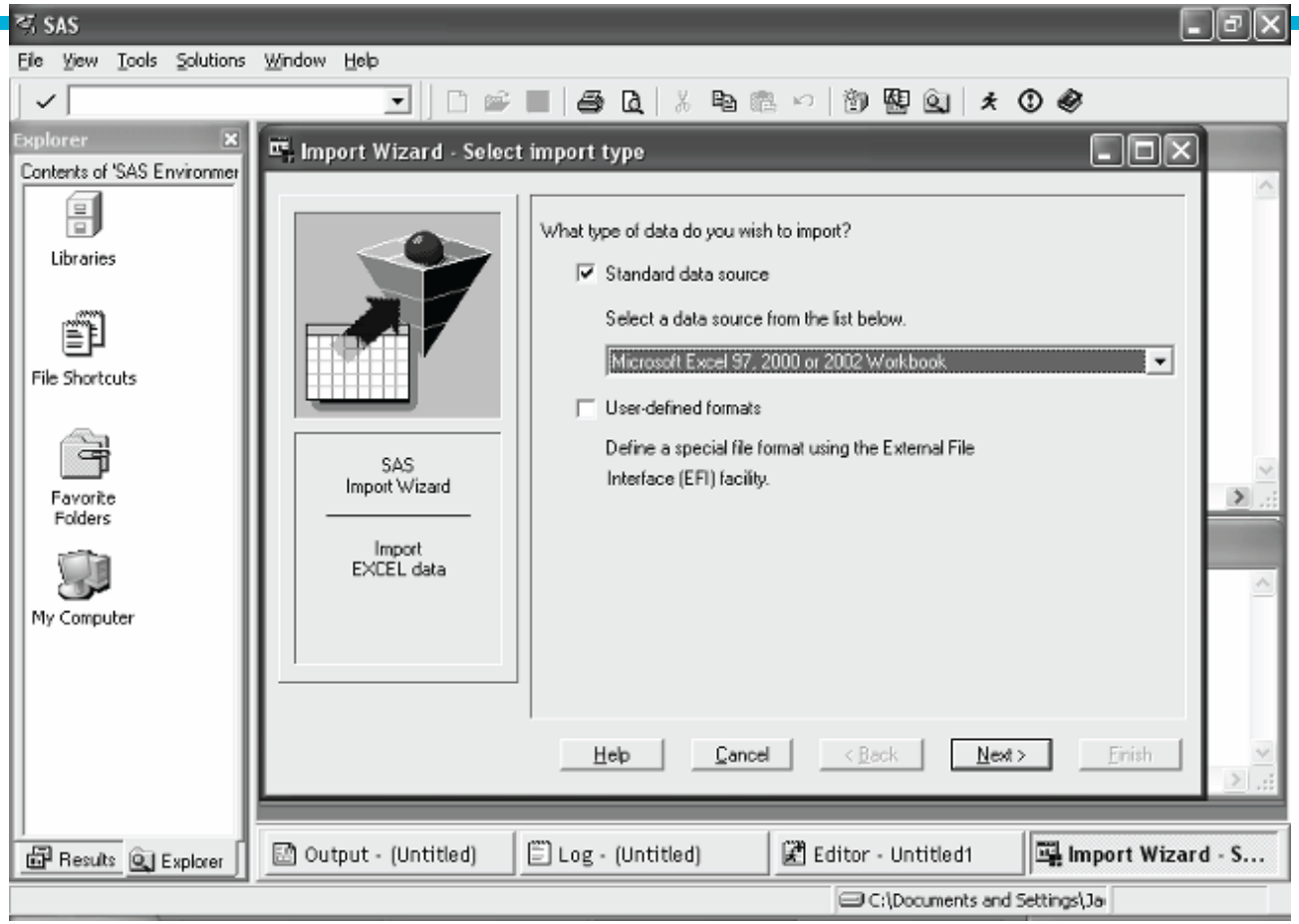
Be aware that the LIBNAME statement approach allows for both reading and writing to and from Microsoft Office files, which means the contents of the Microsoft Office files can be changed by SAS.

Import Wizard and PROC IMPORT (1)

- The interactive SAS Import Wizard provides an easy way to import the contents of Microsoft Excel and Access files into SAS. Here again, the Import Wizard is essentially a graphical user interface that builds the PROC IMPORT code for you.

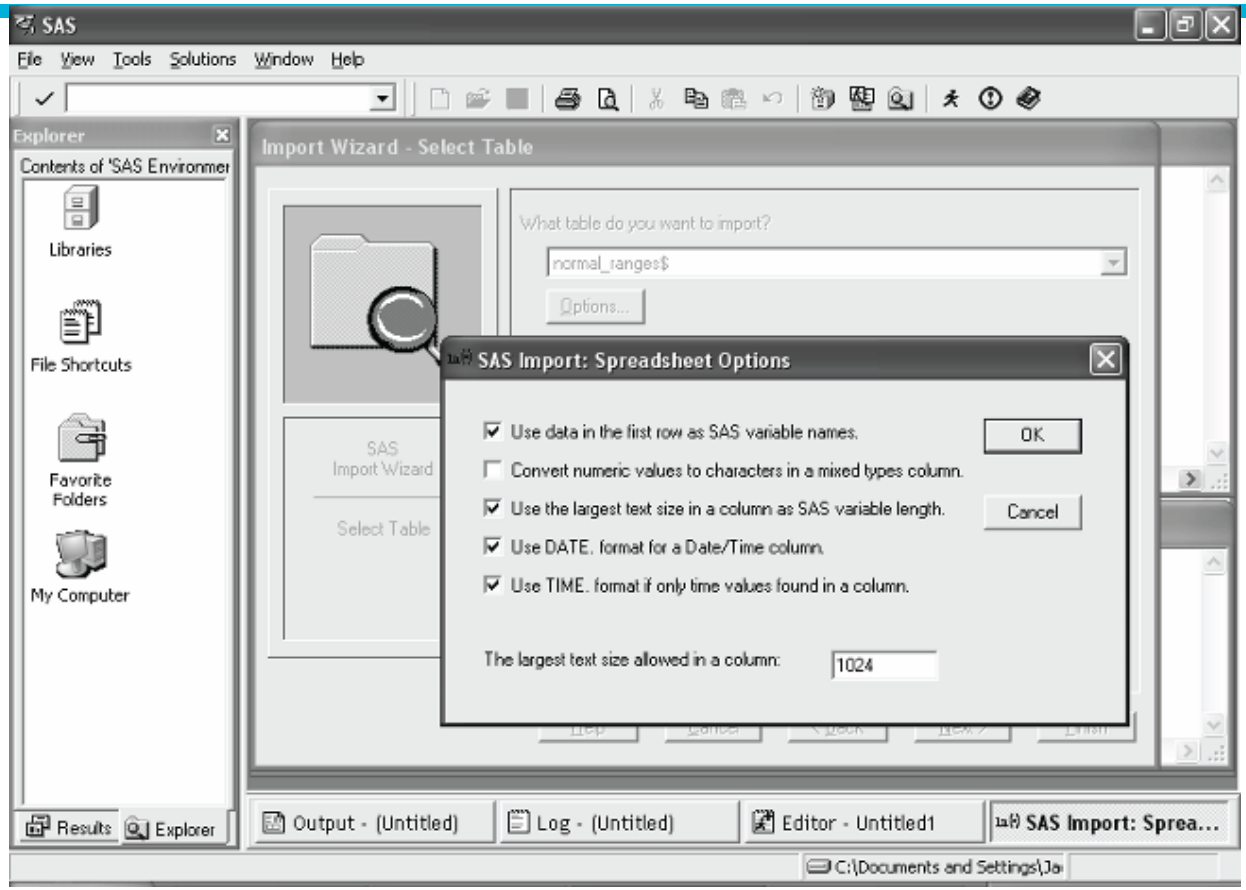
- Begin in the interactive SAS windowing environment by selecting “File” from the toolbar and then “Import Data…” from the drop-down menu. A window like the following will appear, where you can select Microsoft Excel as a standard data source.

Import Wizard and PROC IMPORT (2)



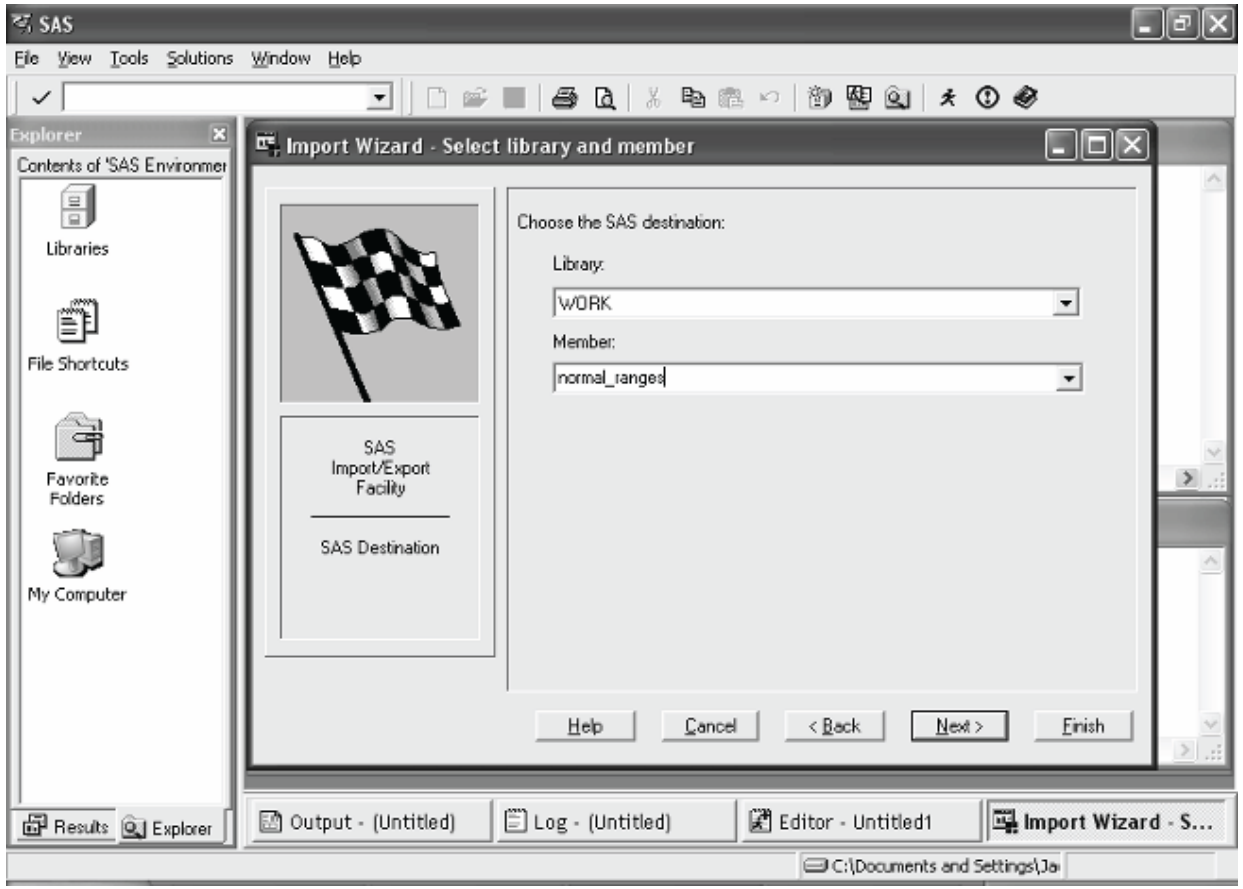
Click “Next,” and a file browser window will open that allows for the drill-down and selection of the Microsoft Excel file of interest. Once the file is selected, a Select Table window will open. This window allows you to pick which worksheet in the Excel file you want to turn into a SAS data set. Click the “Options” button to see the new options available with SAS 9.1 and PROC IMPORT.

Import Wizard and PROC IMPORT (3)



Note that the default options were chosen in the preceding window. We will look at those further when we explore the subsequent PROC IMPORT code. Now, click “OK” in the Spreadsheet Options window and then click “Next” in the Select Table window. The Select Library and Member window opens, which allows for the selection of a SAS library and data set name as follows.

Import Wizard and PROC IMPORT (4)



Click “Next” and SAS will prompt you to see if you want to save the PROC IMPORT code generated by the Import Wizard. Click “Finish” to complete the file import.

Import Wizard and PROC IMPORT (5)

Here is the PROC IMPORT code generated by SAS from this run.

```
PROC IMPORT OUT= WORK.normal_ranges
            DATAFILE= "C:\normal_ranges.xls"
            DBMS=EXCEL REPLACE;
   SHEET="normal_ranges$";
   GETNAMES=YES;
   MIXED=NO;
   SCANTEXT=YES;
   USEDATE=YES;
   SCANTIME=YES;
RUN;
```

- If the options on PROC IMPORT do not produce what is desired, they can be changed and resubmitted, or the code can be saved to edit and run in batch mode later.

Import Wizard and PROC IMPORT (6)

Here are the new options available with SAS 9.1 along with my recommended settings.

Option	Purpose
DBSASLABEL	By default, the SAS label for an imported variable is set to the column name. Setting DBSASLABEL=NONE places null values into the SAS labels.
MIXED	Converts numeric values to character values if a column displays numeric and character text cells. Note that the default here is NO. Keep in mind that SAS scans only the first eight rows of the Excel column to determine whether the column is numeric or character. If SAS picks character and there are numeric cells later, then those will be set to blank. For this reason consider setting MIXED=YES.
SCANTEXT	When set to YES, this option tells SAS to scan the entire column to determine the width of the column. Always leave this set to YES.
SCANTIME	When set to YES, this option applies a SAS time format to a field if it appears to contain only time entries.
TEXTSIZE	This option hardcodes the maximum width of a character variable. It overrides SCANTEXT=YES.
USEDATE	When set to YES, this option formats SAS datetime fields with a date format. If you prefer to use datetime formats with datetime fields, set USEDATE=NO.

► The Import Wizard process for Microsoft Access files works like the one for Excel files and produces similar PROC IMPORT code. Keep in mind that text fields get a default length of 255 characters when PROC IMPORT is used with Microsoft Access files.

SAS/ACCESS SQL Pass-Through Facility (1)

The SAS/ACCESS SQL Pass-Through Facility is another way for SAS to dynamically establish a connection to Microsoft Excel or Access files. You can connect to the Microsoft Excel file normal_ranges.xls by using the following SAS code.

```
**** OBTAIN AVAILABLE WORKSHEET NAMES FROM EXCEL FILE;
proc sql;
   connect to excel (path = "C:\normal_ranges.xls");
   select table_name from connection to excel(jet::tables);
quit;
**** GO GET NORMAL_RANGES WORKSHEET FROM EXCEL FILE;
proc sql;
   connect to EXCEL (path = "C:\normal_ranges.xls" header = yes
                     mixed = yes version = 2000 );
   create table normal_ranges as
      select * from connection to excel
      (select * from [normal_ranges$]);
   disconnect from excel;
quit;
```

► Study the preceding SAS code. Notice how the first SQL step uses a special Microsoft Jet Engine query to obtain the names of the worksheet in normal_ranges.xls. Also note that the SQL step that fetches the normal_ranges worksheet from normal_ranges.xls does so by placing the worksheet in braces in the inner SELECT statement.

SAS/ACCESS SQL Pass-Through Facility (2)

The following SAS code uses the SQL Pass-Through Facility to connect to the Microsoft Access file normal_ranges.mdb.

```
*** OBTAIN AVAILABLE TABLE NAMES FROM ACCESS FILE;
proc sql;
   connect to access (path = "C:\normal_ranges.mdb");
   select table_name from connection to access(jet::tables);
quit;
**** GO GET NORMAL_RANGES WORKSHEET FROM ACCESS FILE;
proc sql;
   connect to access (path="C:\normal_ranges.mdb");
   create table normal_ranges as
      select * from connection to access
      (select * from normal_ranges);
   disconnect from access;
quit;
```

- Note that the SQL Pass-Through Facility to Microsoft Excel and Access files does default to 255 characters in length for character fields.

Access to Oracle

Contents

Introduction

Importing relational databases and clinical data management systems

- Most **clinical data management systems** used for clinical trials today store their data in relational database software such as Oracle or Microsoft SQL Server.
- A **relational database** is composed of a set of rectangular data matrices called “tables” that relate or associate with one another by certain key fields.
- The language most often used to work with relational databases is *structured query language* (*SQL*). The SAS/ACCESS SQL Pass-Through Facility and the SAS/ACCESS LIBNAME engine are the two methods that SAS provides for extracting data from relational databases.

SAS/ACCESS SQL Pass-Through Facility (1)

The SAS/ACCESS SQL Pass-Through Facility has long been one of the only ways of getting data out of a relational database. It is still a flexible means of obtaining relational database data, as it allows for using SAS SQL as a means of filtering or modifying data on the way into SAS.

```
proc sql;
   connect to oracle as oracle_tables
          (user = USERID orapw = PASSWORD path = "INSTANCE");

   create table AE as
      select * from connection to oracle_tables
      (select * from AE_ORACLE_TABLE );

   disconnect from oracle_tables;
quit;
```

SAS/ACCESS SQL Pass-Through Facility (2)

- This code simply extracts the data from the table named “AE_ORACLE_TABLE” within Oracle and places it in its entirety in a SAS work library data set called AE.

- The USER, ORAPW, and PATH parameters are specific to the Oracle database settings at a particular site, so you would need to consult an Oracle database administrator to get the proper values.

- The good thing about the SQL Pass-Through Facility is that it lets you create a more desirable SAS data set with some slight modifications.

SAS/ACCESS SQL Pass-Through Facility (3)

```
proc sql;
   connect to oracle as oracle_tables
     (user = USERID orapw = PASSWORD path ="INSTANCE");

   create table library.AE as
      select * from connection to oracle_tables
      (select subject, verbatim, ae_date, pt_text
          from AE_ORACLE_TABLE
          where query_clean=”YES”);
   disconnect from oracle_tables;
quit;
```

- Notice how the highlighted changes allow for a permanent SAS data set to be created containing only the variables desired and only the records that have all data queries resolved by data management.

SAS/ACCESS LIBNAME statement (1)

- The Oracle specific syntax for the LIBNAME statement is:

  LIBNAME *libref* **oracle** <*connection-options*> <*LIBNAME - options*>

  - *libref* is any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.
  - oracle is the SAS/ACCESS engine name for the interface to Oracle.
  - *connection – options* provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS.
  - *LIBNAME – options* define how DBMS objects are processed by SAS. Some LIBNAME options can enhance performance; others determine locking or naming behavior.

SAS/ACCESS LIBNAME statement (2)

- Beginning with SAS 7, a new SAS/ACCESS LIBNAME statement interface was available for accessing data in relational databases.

- For example, the previous example of the SQL Pass-Through Facility can be distilled to the following LIBNAME statement and associated DATA step.

```
libname oratabs oracle user=USERNAME
        orapw = PASSWORD path = "@INSTANCE" schema = TRIALNAME;
data adverse;
   set oratabs.AE_ORACLE_TABLE;
       where query_clean = “YES”;
       keep subject verbatim ae_date pt_text;
run;
```

SAS/ACCESS LIBNAME statement (3)

- In this program the “oratabs” libref allows all of the tables found in that Oracle data instance to be treated like SAS data sets. This is a simple and fast way of accessing relational databases, and it requires no knowledge of SQL to implement.

- Although the preceding examples import Oracle data, SAS/ACCESS can be used to access quite a number of relational databases, including Oracle, Microsoft SQL Server.

Access to SQL

Contents

- SAS/ACCESS SQL Pass-Through Facility
- SAS/ACCESS LIBNAME statement

SAS/ACCESS SQL Pass-Through Facility

The following example sends Microsoft SQL Server 6.5 an SQL query for processing. The results from the query serve as a virtual table for the PROC SQL FROM clause. In this example, MYDB is the connection alias.

```
proc sql;
   connect to SQLSVR as mydb
      (datasrc="SQL Server" use=testuser password=testpass);
   select * from connection to mydb
      (select CUSTOMER, NAME, COUNTRY
          from CUSTOMERS
          where COUNTRY <> 'USA');
quit;
```

SAS/ACCESS LIBNAME statement (1)

The Microsoft SQL Server specific syntax for the LIBNAME statement is:

LIBNAME *libref* **sqlsvr** <*connection-options*> <*LIBNAME - options*>

- *Libref* is any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.
- Sqlsvr is the SAS/ACCESS engine name for the interface to Microsoft SQL Server.
- *connection – options* provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS.
- *LIBNAME – options* define how DBMS objects are processed by SAS. Some LIBNAME options can enhance performance; others determine locking or naming behavior.

SAS/ACCESS LIBNAME statement (2)

- Microsoft SQL server LIBNAME statement examples
  - In following example, USER= and PASSEORD= are connection options.

```
Libname mydblib sqlsvr user=testuser password=testpass;
```

  - In the following example, the libref MYDBLIB connects to a Microsoft SQL Server database using the NOPROMPT= option.

```
libname mydblib sqlsvr
   noprompt="uid=testuser;
   pwd=testpass;
   dsn=sqlservr;"
   stringdates=yes;

proc print data=mydblib.customers;
   where state='CA';
run;
```