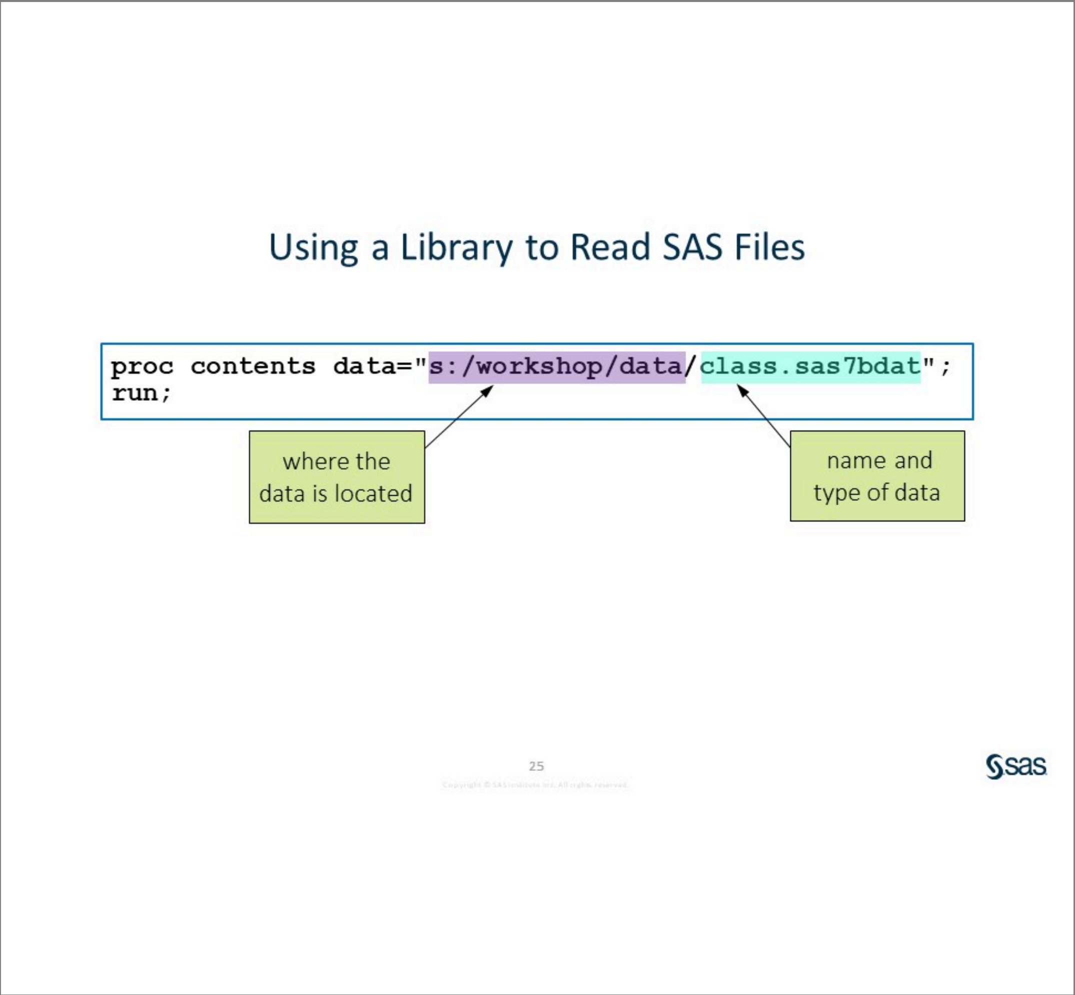
B2.2 - Accessing Data through Libraries

To-Do Date: Oct 28 at 11:59pm



Using a Library to Read SAS Files

```
proc contents data="s:/workshop/data/class.sas7bdat";
run;
```

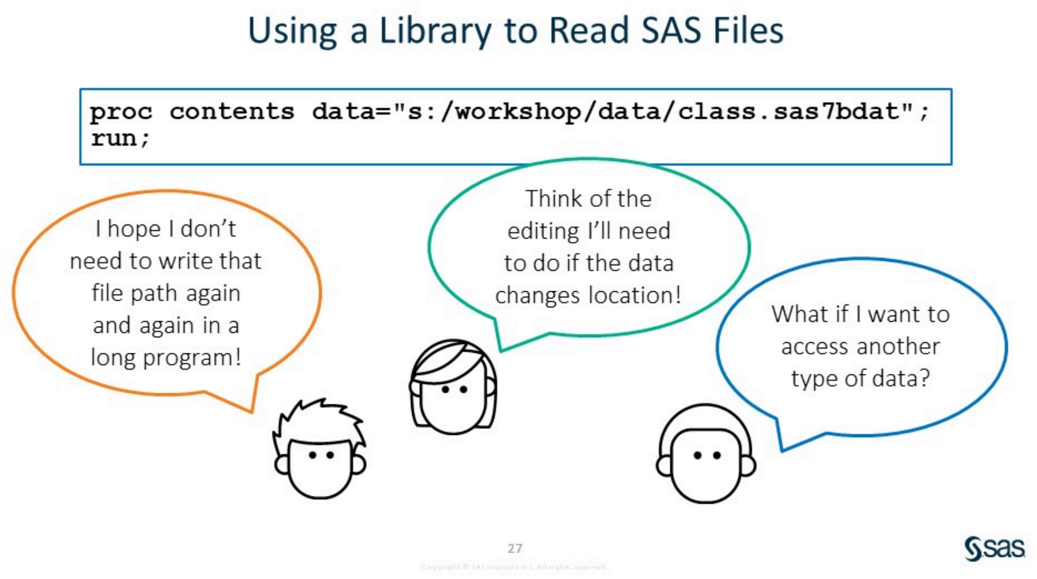
where the data is located
name and type of data
25

So far we've used a hardcoded filepath to the SAS table we want to access, and that file path has the two pieces of information that are required for SAS to read the file: where the data is located and what type of data it is. Because we've been reading a SAS table, providing a path to the data and file name in quotation marks works perfectly.

Let's discuss the issues that might arise from using a hardcoded path in your code.

Think about it:

What challenges might arise if you use a fixed path in your program?



Using a Library to Read SAS Files

```
proc contents data="s:/workshop/data/class.sas7bdat";
run;
```

I hope I don't need to write that file path again and again in a long program!
Think of the editing I'll need to do if the data changes location!
What if I want to access another type of data?
27

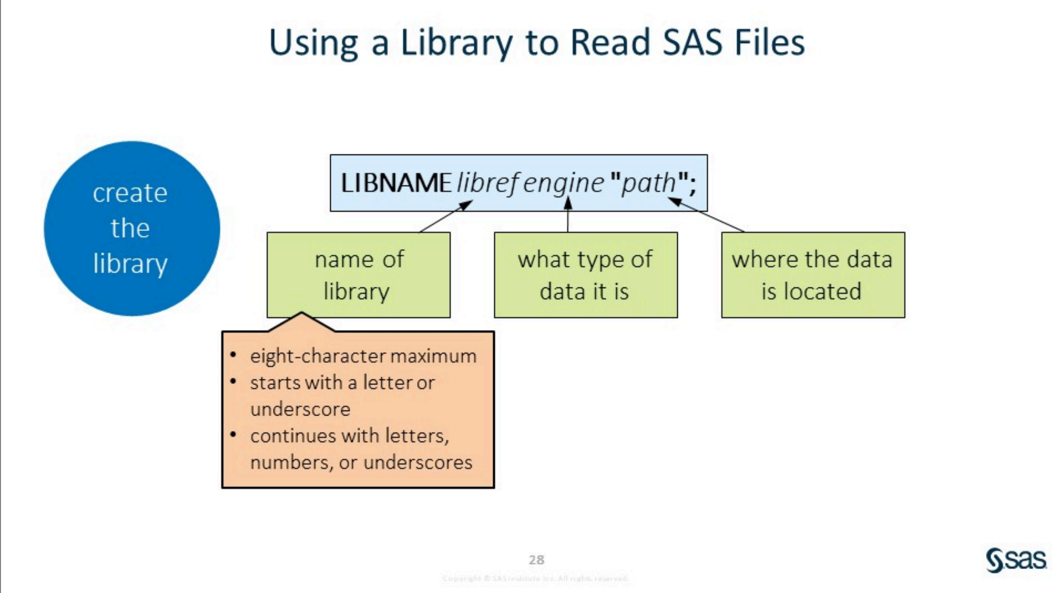
SAS Libraries

What if I write a longer, more complex program? It would be tedious to have to repeatedly provide the full path and file name each time!

What if the data changes locations? You would have to edit my program in multiple places!

What if the type of data I want to read is Excel or Teradata? Providing a simple file path to those files won't work. SAS needs more specific instruction for how to read other types of structured data.

All these issues can be solved by using libraries.



Using a Library to Read SAS Files

create the library

```
LIBNAME libref engine "path";
```

name of library
what type of data it is
where the data is located
eight-character maximum
starts with a letter or underscore
continues with letters, numbers, or underscores
28
sas

SAS Libraries

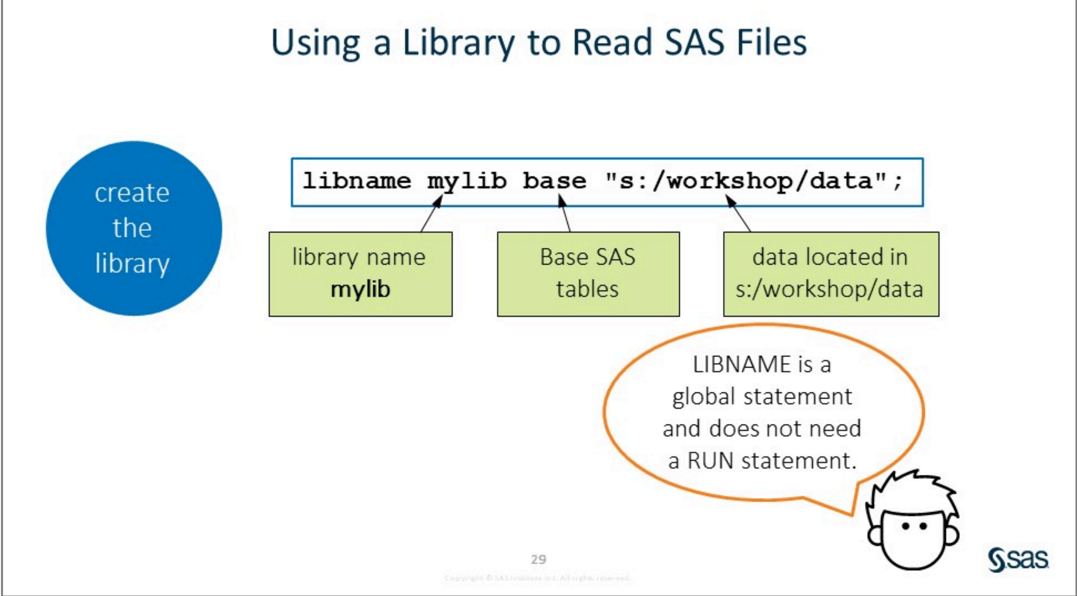
SAS libraries provide a way to specify the two required pieces of information – the location and file type – in a very simple and efficient way. You can think of a library as a collection of data files that are the same type and in the same location.

You create a library with the LIBNAME statement. This is one of the global statements in SAS that does not need a RUN statement at the end.

We start by creating a library that enables us to access the SAS tables in a particular folder. We begin with the keyword LIBNAME, followed by what is referred to as a *library reference*, or *libref*. The libref is the name of the library. The libref must be eight characters or less, must start with either a letter or underscore, and can include only letters, numbers, and underscores.

After the libref, we specify the engine, which is related to the type of data you are accessing. The engine is a behind-the-scenes set of instructions that enables SAS to read structured data files directly, without having to do a separate, manual import into SAS. There are dozens of engines available, including Base for SAS tables, Excel, Teradata, Hadoop, and many others.

Finally, we need to provide the location or connection information for the data we want to read. That can be a physical path or directory, or other options to connect to a database.



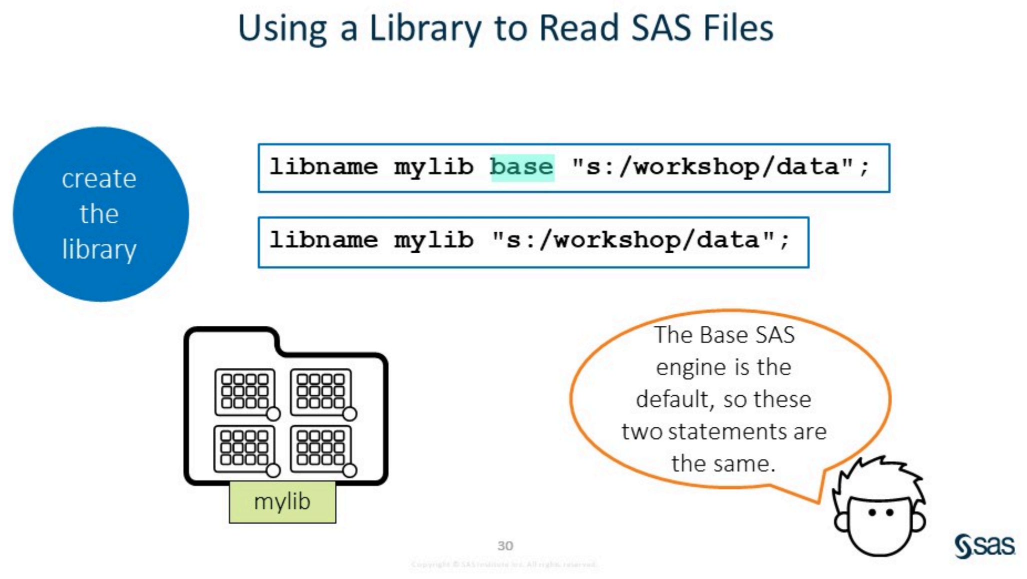
Using a Library to Read SAS Files

```
libname mylib base "s:/workshop/data";
```

create the library
library name **mylib**
Base SAS tables
data located in s:/workshop/data
LIBNAME is a global statement and does not need a RUN statement.
29

Libname Statement

This example LIBNAME statement creates a libref, or library, named **mylib** that uses the Base SAS engine to read SAS tables located in s:/workshop/data.



Using a Library to Read SAS Files

```
libname mylib base "s:/workshop/data";
```

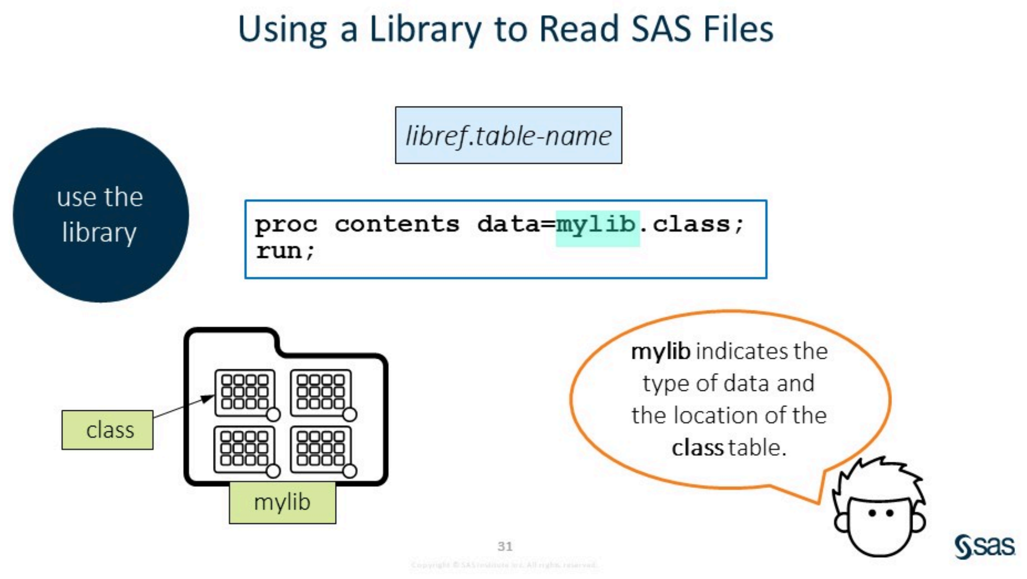
```
libname mylib "s:/workshop/data";
```

mylib
The Base SAS engine is the default, so these two statements are the same.
30

Base is the default engine, so you could write the LIBNAME statement without specifying the Base SAS engine and it would be fine.

One thing to keep in mind is that the path you specify must be relative to where SAS is running. If SAS is local, you can specify a path to a folder of files on your own machine. If SAS is on a remote server, the path or folder must be to a location known to the server.

When you submit the LIBNAME statement, all the information about the location and file type is associated with the library name, or libref.



Using a Library to Read SAS Files

libref.table-name

```
proc contents data=mylib.class;
run;
```

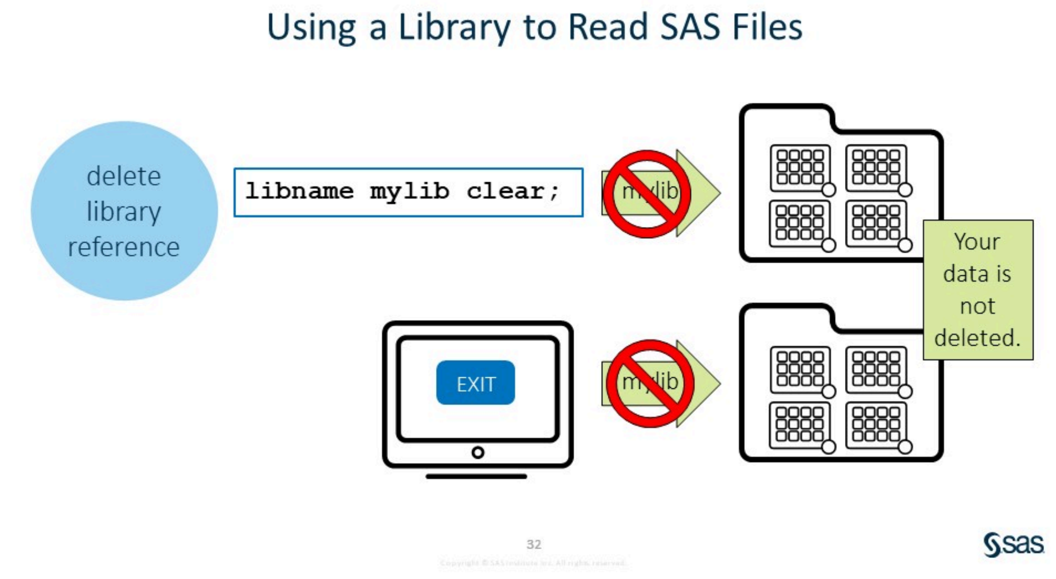
class
mylib
mylib indicates the type of data and the location of the class table.
31

After the library is defined, you can use it as a shortcut to access tables in your program. To do this, you simply specify the libref (or library name), a period, and the table name.

Remember that SAS needs to know where the data file is located and the type of data.

Because **mylib** is defined, SAS knows the location of the data (s:/workshop/data) and they type of data that is in the location (SAS tables). You only have to specify which of the tables you want to read, and you don't have to type the file extension because SAS already knows you're accessing SAS tables from that location!

Pretty easy, isn't it? Now if your data moves to another location, you can edit one statement in your program!



Using a Library to Read SAS Files

```
libname mylib clear;
```

By default, a libref that you define remains active until you delete it or end your SAS session. Remember that the libref is simply a pointer or shortcut to your existing data, so although the libref might be deleted when SAS shuts down, your data remains in the same place. When you restart SAS, you simply re-establish your library and libref by submitting the LIBNAME statement again before you access your data. This is why

SAS programs often begin with one or more LIBNAME statements to connect to the various data sources that are used in the code.

2.05 Activity

1. Open a new program. Write a LIBNAME statement to create a library named **PG1** that reads SAS tables in the **data** folder. (Note: you will need to change the path to the path of YOUR data folder)

```
libname pg1 base "s:/workshop/data";
```

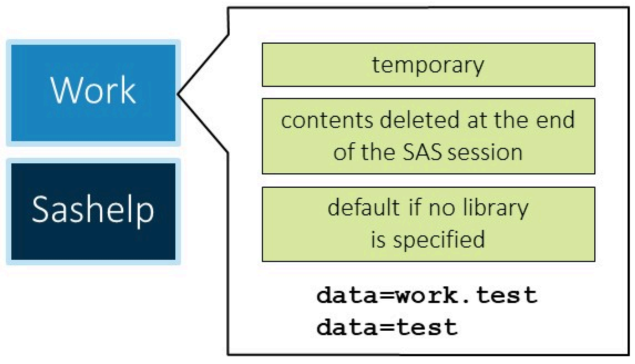
2. Run the code and verify that the library was successfully assigned in the log.
3. Go back to your program and save it as **libname.sas** in YOUR main course files folder. Replace the file if it exists.

2.06 Activity

1. Select **Libraries** in the navigation pane and expand **My Libraries**.
2. Expand the **PG1** library. Why are the Excel and text files in the **data** folder not included in the library?

Click here for Solution.

Automatic SAS Libraries


Work
Sashelp
temporary
contents deleted at the end of the SAS session
default if no library is specified

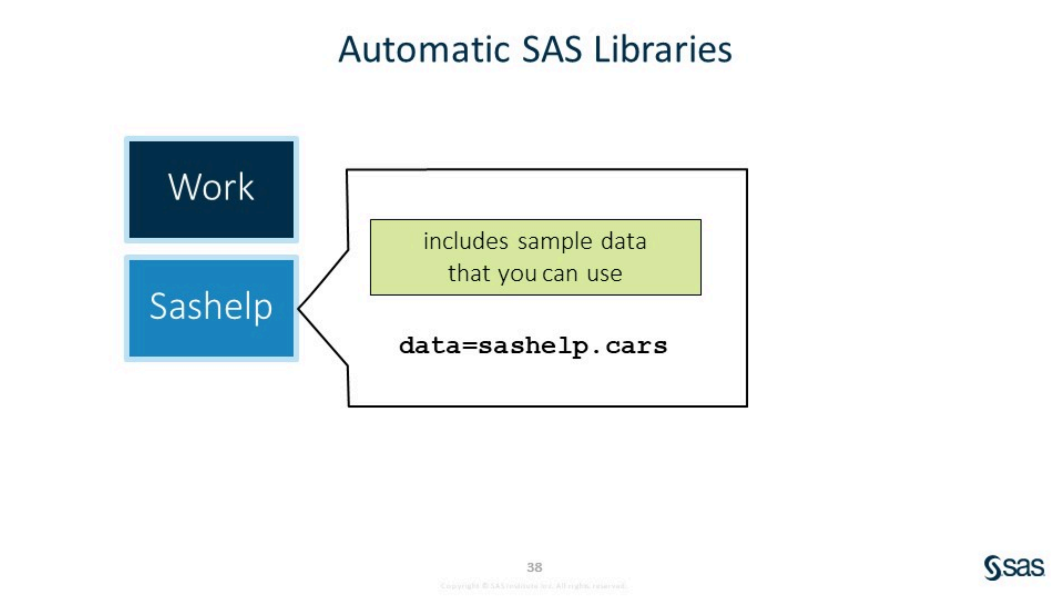
```
data=work.test
data=test
```

37

Work Library

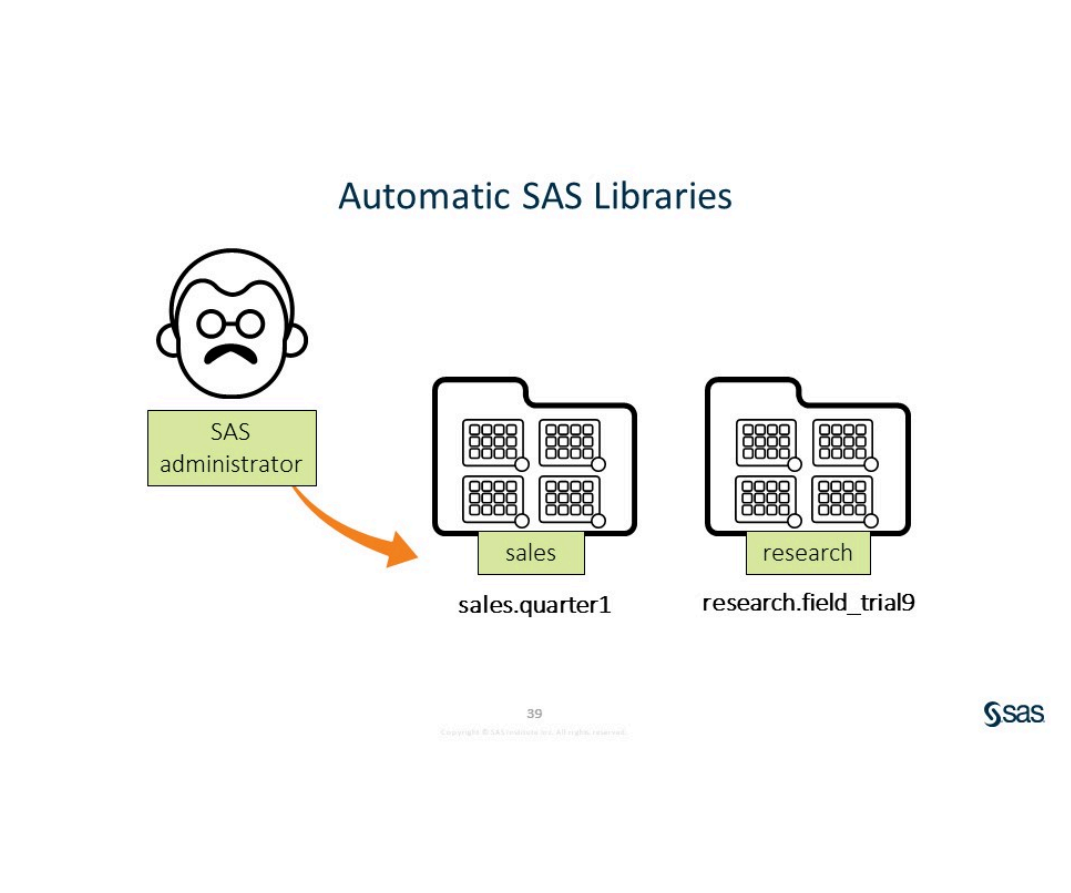
In addition to creating your own libraries, there are other ways libraries can be created. When you start your SAS session, there are several libraries that are defined automatically by SAS. I'd like to point out two that you'll be using in this class.

The **Work** library is a temporary library that is automatically defined by SAS at the beginning of each SAS session. We say the **Work** library is temporary because any tables written to the **Work** library are deleted at the end of each SAS session. This library is commonly used in SAS programs because it is a great way to create working files that you do not need to save permanently. The **Work** library is also considered to be the default library. If you do not type a libref in front of a table name, SAS assumes that the library is **Work**. For example, these code snippets both reference the temporary table named **test** in the **Work** library.



SAShelp Library

Another library that SAS automatically defines is the **Sashelp** library. **Sashelp** contains a collection of sample tables and other files. We use several of the sample tables in **Sashelp** in the examples in this course.



It's worth mentioning that there might be other libraries established for you in your environment by a SAS administrator. If you do have libraries defined for you, you don't need to submit a LIBNAME statement. You can just use the libref created by your administrator and the table name to use those data sources in your program.

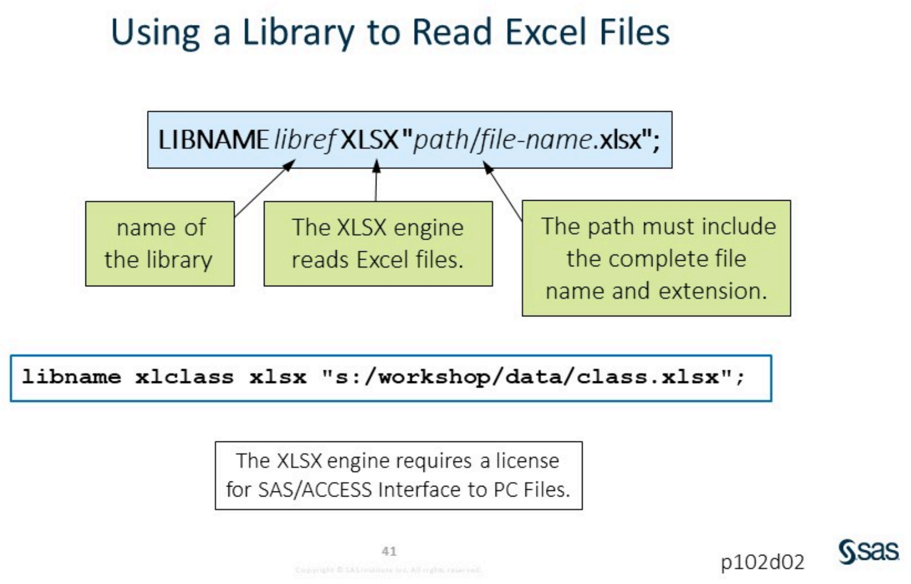
This is true with the Clemson Cloud Server.

Demo: Exploring Automatic SAS Libraries

Use **p102d01.sas** in the **demo** folder in SAS

[2_2 - Demo - Exploring Automatic SAS Libraries.pdf (https://clemson.box.com/s/m0kcj0c5ilig3zkczeysk3dy6ktvuemt)](https://clemson.box.com/s/m0kcj0c5ilig3zkczeysk3dy6ktvuemt)

Using a Library to Read Excel Files



Using a Library to Read Excel Files

LIBNAME *libref* **XLSX** "*path/file-name*.xlsx";

name of the library
The XLSX engine reads Excel files.
The path must include the complete file name and extension.

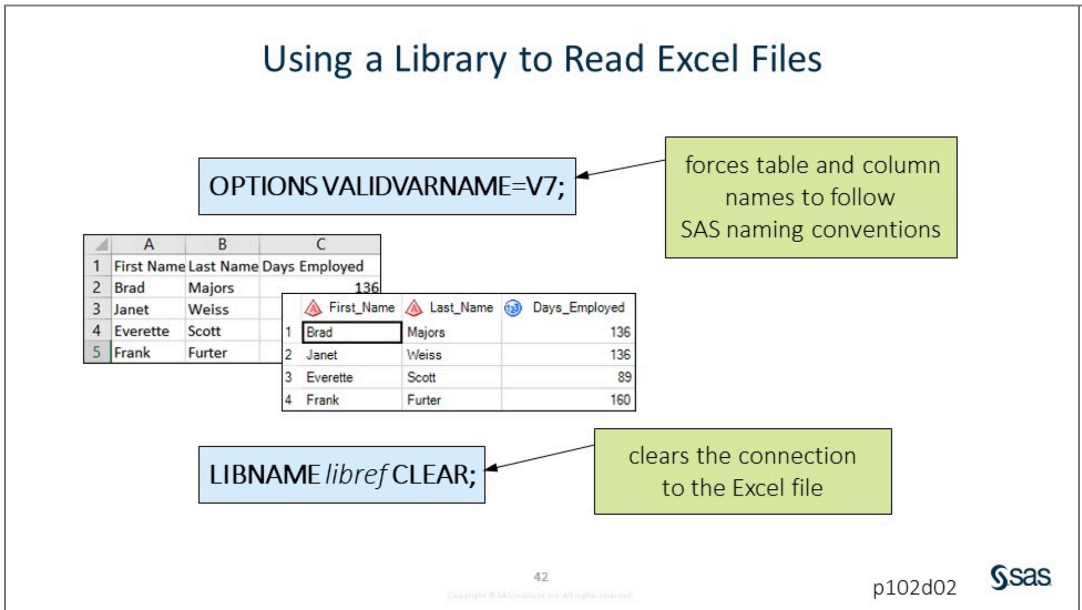
```
libname xlclass xlsx "s:/workshop/data/class.xlsx";
```

The XLSX engine requires a license for SAS/ACCESS Interface to PC Files.

41
p102d02
sas

In addition to SAS data, you can use libraries to access many other types data. For example, you can use a library with the XLSX engine to read data directly from Excel spreadsheets. I should point out that this engine requires a license for SAS/ACCESS to PC Files.

Remember that when SAS reads or writes data in a program, it must know where the data is located and what format is it in. The only change to the LIBNAME statement syntax is that we specify the XLSX engine, and a path that includes the complete Excel workbook file name and extension. You can think of the Excel workbook as a collection of tables. Each individual worksheet or named range is one table in the collection.



Using a Library to Read Excel Files

```
OPTIONS VALIDVARNAME=V7;
```

forces table and column names to follow SAS naming conventions

	A	B	C
1	First Name	Last Name	Days Employed
2	Brad	Majors	136
3	Janet	Weiss	
4	Everette	Scott	
5	Frank	Furter	

	First_Name	Last_Name	Days_Employed
1	Brad	Majors	136
2	Janet	Weiss	136
3	Everette	Scott	89
4	Frank	Furter	160

LIBNAME *libref* CLEAR;

clears the connection to the Excel file

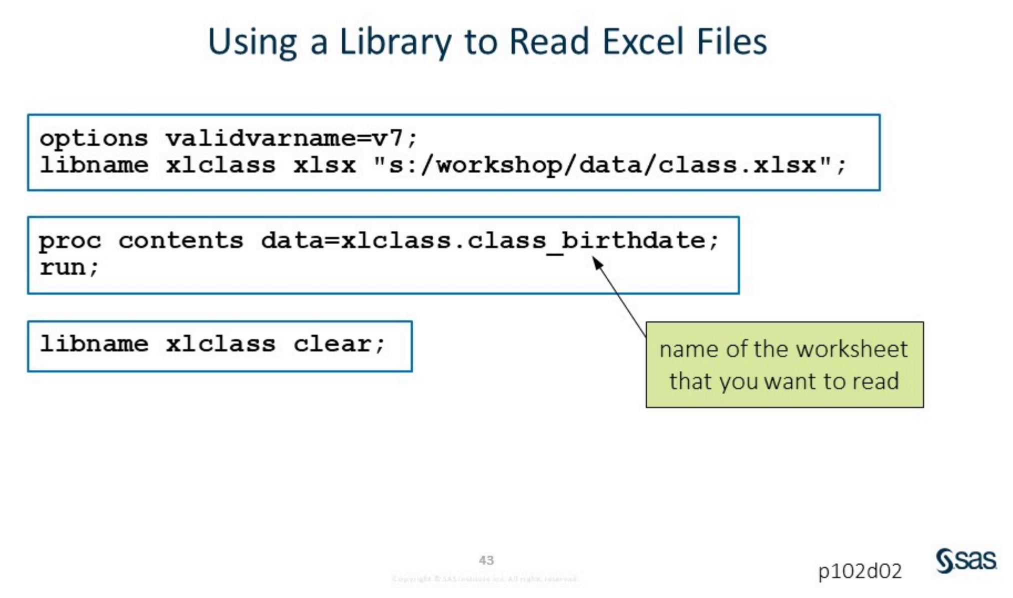
42

Copyright © SAS Institute Inc. All rights reserved.

p102d02

There are two extra statements that you often use when you read Excel data. The first is the OPTIONS statement, a global statement for specifying system options. Excel doesn't have any rules for column headings, so they can be longer than 32 characters and include spaces or other special symbols. When SAS reads the Excel data, we can force column names to adhere to strict SAS naming conventions by using the VALIDVARNAME=V7 system option. Technically, this enforces the column naming rules established with SAS 7. With this option set, SAS replaces any spaces or special symbols in column names with underscores, and names greater than 32 characters are truncated.

When you define a connection to data sources such as Excel or other databases, it's a good practice to clear, or delete, the libref at the end of your program. While your library is active, it might create a lock on the data preventing others from accessing the file, or it could maintain an active connection to the data sources that is unnecessary. To clear the library reference, use the LIBNAME statement again, name the libref, and use the keyword CLEAR.



Using a Library to Read Excel Files

```
options validvarname=v7;
libname xlclass xlsx "s:/workshop/data/class.xlsx";
```

```
proc contents data=xlclass.class_birthdate;
run;
```

```
libname xlclass clear;
```

name of the worksheet that you want to read
43
p102d02

In this example, we use the OPTIONS statement to enforce SAS naming rules for columns, and then we create the **xlclass** library with the XLSX engine to read data from the class.xlsx Excel workbook located in s:/workshop/data. The PROC CONTENTS step is reading the **class_birthdate** worksheet in the **class** workbook. At the end, we clear the **xlclass** libref.

As you can see, it's easy to establish a library to an Excel workbook and read or write to Excel directly without having to do an extra step to import or export the data.

Demo: Using a Library to Read Excel Files

Before we read the data in SAS, it's a good idea to examine the Excel file to understand what we have. Remember Excel isn't truly a database, and there are no rules or restrictions as to what can be entered in each spreadsheet. In the Hurricane.xlsx file, we have 5 worksheets. Each are organized as a data table, having defined columns with names in row 1. In the STORM_SUMMARY spreadsheet, 2 of the column names contain spaces. We want to make sure when SAS reads this data that our column naming rules, including no spaces or special symbols, is followed. And also note that Date is formatted as a 2 digit day, 3-letter month and 2-digit year. We want these values in SAS to be converted to a SAS date, or the number of days from January 1st, 1960.

Let's use what we have learned to connect directly to the **Hurricane** Excel workbook. First, I need to add an OPTIONS statement with VALIDVARNAME=V7 so that our column names will be modified to follow the SAS naming conventions. Spaces or other symbols will be replaced with underscores. Next I will create the LIBNAME statement to provide the two necessary pieces of information: where is the data located and what format is it in? I start with LIBNAME, then provide the LIBREF to represent the name of this new library – I'll call it myxl. Next I need to give the engine, so I will use XLSX. Recall the engine provides all the necessary behind-the-scenes information to allow SAS to read and write to this non-SAS file. And for the path in quotes, rather than pointing to a folder in the operating system, I am going to provide the path and file for my Excel workbook. Each worksheet in the workbook will be treated a unique table. I'll run the OPTIONS and LIBNAME statements.

If I use my navigation pane or explorer, I can see the MYXL library, and the 5 spreadsheets listed as if they were SAS tables. I can double-click to open STORM_SUMMARY to view it. Notice that Hem_EW and Hem_NS have underscores added where there were previously spaces. And Date values LOOK like formatted dates, but let's examine the column attributes to determine if Date is defined as a character or numeric column. I can do it interactively by looking at the column properties. Or I can write a PROC CONTENTS step to create a report on the table metadata. I will update my previous PROC CONTENTS to change the libref to xlhur and the table to STORM_SUMMARY. We see the Date column is numeric with a format applied. The XLSX engine did the work for us to read the data properly and create the required and some optional attributes. For example, notice there is the LABEL attribute. While each column must have a name that follows specific rules, formats can be anything you can type, up to 256 characters. And most reporting procedures display labels in reports. So for the Hem_EW and Hem_NS column names that were modified to include underscores, notice the label matches the original value in the Excel worksheet.

Finally, I'll complete the statement to clear the xlhur library. Remember nothing happens to the Excel file, but the xlhur libref is no longer active in the SAS session.

Reading other types of data goes beyond the scope of this class, but it is worth pointing out that SAS offers many different engines through the SAS/ACCESS products to allow you to read directly from non-SAS data sources. The elegance of the LIBNAME statement and the ACCESS engines is once you or your administrator has created a library connecting to your various data sources, the majority of your code is independent of the type of data you are working with.

Use p102d02.sas in the demo folder

2_2 - Demo - Using a Library to Read Excel Files.pdf (https://clemson.box.com/s/52tntynx2eknrkqwvc2hv4udyhlsdie4)

2.07 Activity

Use **p102a07.sas** in the **activities** folder and perform the following tasks:

1. If necessary, update the path of the course files in the LIBNAME statement.
2. Complete the PROC CONTENTS step to read the **parks** table in the **NP** library.
3. Run the program. Navigate to your list of libraries and expand the **NP** library. Confirm that three tables are included: **Parks**, **Species**, and **Visits**.
4. Examine the log. Which column names were modified to follow SAS naming conventions?

Click here for Solution.

5. Uncomment the final LIBNAME statement and run it to clear the **NP** library.