Import Excel Files into SAS (With Example)

Authored by

Mohammed looti

November 1, 2025

The Need for Seamless Data Integration

In the realm of contemporary data analysis, the capability to seamlessly integrate information originating from diverse sources is fundamentally important. While powerful statistical environments, such as SAS, are optimized for complex processing, modeling, and reporting, the initial raw data often resides in external formats. Among the most frequently encountered external formats utilized by business analysts and researchers is the spreadsheet created using Microsoft Excel.

Successfully transferring data stored within an Excel workbook into a functioning SAS Data Set mandates a reliable, efficient, and accurate procedure. Relying on manual data entry is not only highly susceptible to human error but is also entirely impractical when dealing with the substantial file sizes common in modern data projects. Fortunately, the SAS system provides a specialized utility specifically engineered for this process, ensuring rapid and precise data ingestion into the analytical environment.

This comprehensive guide is dedicated to exploring the standard and most streamlined methodology for importing data from Excel files: the application of the **PROC IMPORT** procedure. We will systematically detail the required syntax, explain the function of critical operational parameters, and provide a complete, practical example to guarantee successful and reproducible data migration from the external file into your SAS session.

The Core Tool: Introducing PROC IMPORT

The principal utility designed to convert external data files--which include Excel spreadsheets, delimited text files, and database tables--into native SAS data sets is the PROC IMPORT procedure. This procedure provides immense value because it substantially automates the challenging process of reading external data structures, intelligently inferring variable types (such as character or numeric), and dynamically generating the required internal SAS structure necessary for subsequent analysis.

To initiate the data transfer process using **PROC IMPORT**, the user must meticulously define three mandatory pieces of information. First, the name that the resulting SAS data set will be assigned (the **OUT=** parameter). Second, the exact location and full file path of the source Excel file (the **DATAFILE=** parameter). Third, the Database Management System identifier (**DBMS=**), which explicitly informs SAS how to interpret the external file structure. For recent versions of Excel files (those created since 2007), this crucial DBMS identifier is invariably specified as **XLSX**.

The syntax block below illustrates the fundamental structure required to utilize **PROC IMPORT** to transfer data seamlessly from an external Excel file into the SAS environment. This simple structure forms the foundation for nearly all Excel data imports.

```
/*import data from Excel file called my_data.xlsx*/
proc import out=my_data
datafile="/home/u13181/my_data.xlsx"
dbms=xlsx
replace;
getnames=YES;
run;
```

Deconstructing the Syntax: Essential Parameters

A thorough understanding of the specific parameters employed within the PROC IMPORT statement is absolutely essential for achieving successful, repeatable, and robust data integration processes. Each statement serves a well-defined and distinct purpose in accurately defining the source data, specifying the destination data set, and controlling how the file structure is handled during translation.

The **DBMS** option is particularly critical to the procedure's success, as it specifies the exact format of the external file being processed. For modern Excel files saved in the Office Open XML format, the necessary identifier is XLSX. Conversely, if an analyst were working with older Excel files created before the 2007 version, the correct identifier would be **XLS**. The **DATAFILE** parameter demands the complete and accurate operating system path to the source file, which may be located locally on the machine running SAS or accessed via a network share, depending on the specific SAS environment configuration.

Furthermore, several key options allow fine-tuning of the import process, ensuring the resultant data set accurately reflects the structure of the source spreadsheet. These options govern naming conventions and data handling during potential conflicts. Below is a detailed explanation of the most critical options utilized within the **PROC IMPORT** statement:

OUT=: This mandatory parameter establishes the name that the newly created SAS data set will be assigned once the import operation is fully complete. This resultant dataset then becomes the object of reference for all subsequent SAS analysis steps.
DATAFILE=: This is the mandatory parameter that defines the exact file path, including the full file name, of the Excel file intended for data transfer.
DBMS=: This option identifies the Database Management System (or file type) of the external source data, allowing SAS to correctly interpret and parse the internal file structure. For recent Excel workbooks, this must be **XLSX**.
REPLACE: This optional, yet frequently employed, statement instructs SAS to automatically overwrite the output data set if an existing data set with the identical name is already present in the target library. This feature is particularly useful when data sources are updated frequently.

GETNAMES=: When this parameter is set to **YES**, SAS intelligently utilizes the content of the first row of the external file as the variable names (column headers) in the newly created SAS data set. If the first row contains actual data values instead of descriptive headers, this parameter must be explicitly set to **NO**.

Step-by-Step Implementation: A Practical Case Study

To effectively demonstrate the practical application of the **PROC IMPORT** procedure, let us consider a highly common scenario. A researcher has gathered and stored critical sample data within a typical Excel spreadsheet, and this information must now be transferred into SAS for rigorous statistical analysis.

Imagine we are working with the following sample dataset in Excel, which includes variables such as ID, Score, and Group. It is important to observe that the very first row of the spreadsheet clearly contains the descriptive headers, which are intended to serve as the variable names within the SAS environment:

	A	B	C	D	E	F
1	A	B	C			
2	1	4	76			
3	2	3	49			
4	2	3	85			
5	4	5	88			
6	2	2	90			
7	4	6	78			
8	5	9	80			
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						

We can now apply the previously defined syntax structure to import this specific dataset. We have chosen to name the resulting SAS data set **new_data**. Since our external file adheres to the best practice of having variable names in the first row, we must ensure that the **getnames=YES** option

is included in the procedure call to guarantee accurate structure creation and variable mapping.

```
/*import data from Excel file called my_data.xlsx*/
proc import out=new_data
datafile="/home/u13181/my_data.xlsx"
dbms=xlsx
replace;
getnames=YES;
run;

/*view dataset*/
proc print data=new_data;
```

Upon successful execution of the code block, the PROC IMPORT statement efficiently reads the specified Excel file, generates the corresponding SAS data structure, and populates it with all the rows and columns. We then immediately follow up using the **PROC PRINT** procedure, which is the standard mechanism for verifying the integrity and structure of the newly created data set. The resulting output confirms that the data has been imported flawlessly, preserving the original values, variable names, and underlying data structures:

Obs	A	B	C
1	1	4	76
2	2	3	49
3	2	3	85
4	4	5	88
5	2	2	90
6	4	6	78
7	5	9	80

As clearly demonstrated by the verifiable output above, the data now displayed in the SAS data set accurately and completely matches the source data contained within the original Microsoft Excel file, confirming the successful completion of the data migration.

Ensuring Data Integrity: Best Practices and Advanced Options

While the **PROC IMPORT** procedure is designed to be generally straightforward, adhering to specific best practices is essential to ensure robust and consistent data handling, particularly when dealing with large, complex, or frequently updated Excel files. These practices mitigate common

issues related to data structure and file handling.

The correct specification of the **GETNAMES** option is paramount for accurate variable mapping. If **getnames=YES** is erroneously used when the first row actually contains observation data, SAS will incorrectly assign that row as variable names, leading to corrupted header information. Conversely, if **getnames=NO** is used when descriptive headers are present, SAS will generate generic variable names (e.g., VAR1, VAR2) and treat the header row as observation data, inevitably causing systemic data misalignment.

Another critical consideration involves specifying the correct sheet within the Excel workbook. If the workbook contains multiple sheets, **PROC IMPORT** will, by default, only attempt to read the first available sheet. To import data from a specific sheet, the analyst must incorporate the **SHEET=** option within the procedure call, specifying the sheet name exactly as it appears in the external Excel file. Furthermore, always make it a habit to check the SAS Log immediately after execution; any errors concerning file access permissions, path inaccuracies, or character encoding mismatches will be clearly and explicitly reported there. Finally, utilizing the **DBMS=XLSX** option mandates that the appropriate SAS/ACCESS Interface to PC Files component is correctly installed and properly licensed on the operating system running SAS.

Beyond PROC IMPORT: Alternative Integration Strategies

Although PROC IMPORT offers an excellent solution for simple, one-time, or small-scale data transfers, SAS provides sophisticated alternative methodologies that are often far more suitable for complex or recurring data integration requirements. Understanding these alternatives is crucial for professional and scalable data management practices.

For analytical projects that demand continuous, live linking to an Excel file without the inefficiency of repeatedly importing and duplicating the data, the **LIBNAME** statement employing the **PCFILES** engine offers a superior solution. This advanced method treats the Excel file as a live, external SAS library (or 'libref'), which permits analysts to directly read, query, and even update the data contained within the Excel sheets as if they were standard, native SAS data sets. This approach drastically minimizes file duplication and ensures that the analytical data is always current and synchronized with the source.

For users who require the maximum possible control over variable data types, specific formatting, and the meticulous handling of problematic data elements (such as ambiguous date formats or columns containing mixed text and numeric entries), importing data via the Data Step remains the most granular and powerful approach. While the Data Step is inherently more complex and requires explicit programming, it allows the developer to define every single variable explicitly, overriding SAS's automatic guessing mechanism and effectively preventing potential data truncation, type mismatches, or other structural errors that can sometimes occur with fully

automated procedures like **PROC IMPORT**.

Summary and Conclusion

The **PROC IMPORT** procedure provides a powerful, highly efficient, and standardized mechanism for successfully migrating external data, particularly from ubiquitous Excel workbooks, into a fully functional and ready-to-use SAS data set. By correctly specifying the required data file path, the desired output data set name, and the critical XLSX DBMS type, data analysts can rapidly prepare their raw information for sophisticated statistical processing within the SAS environment.

It is vital to remember the crucial distinction provided by **getnames=YES** when your source Excel sheet includes headers; this simple option ensures that your variables are correctly named, significantly enhancing the readability and long-term usability of the imported data. Always confirm successful execution by diligently reviewing the SAS log for warnings or errors and, ideally, using **PROC PRINT** to visually inspect the structure and content of the first few observations in the newly created file.

The following tutorials explain how to perform other common tasks in SAS: