

SAS For DATA SCIENCE

Fundamentals of Programming in SAS



vijay kasotia

eMAIL ID: vk73742@gmail.com

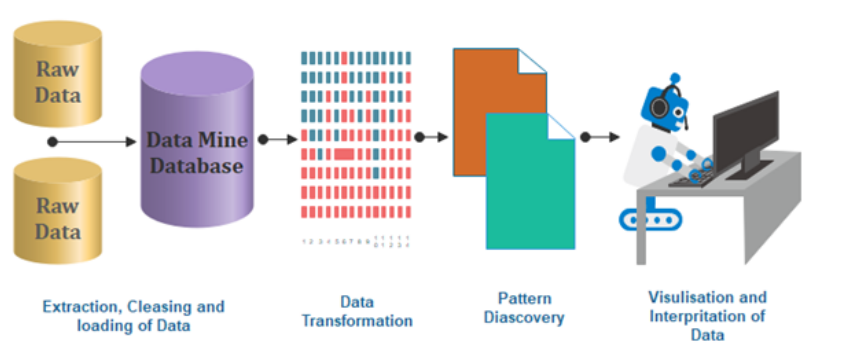
LinkedIn Profile: https://www.linkedin.com/in/vijay-kasotia-5602a6134/

**SAS For Data Science**

Introduction

SAS is a programming language that was developed on the 1st January 1960 by the Institute of SAS. SAS full form (or stands) is *Statistical Analysis Software*. It is used for data management, business modeling, data updation and modification, and statistical analysis (predictive, descriptive, and prescriptive Analysis), etc. With the help of this programming language, you analyze the data and make better decisions for forthcoming. Technically speaking, it is generally used for data analytics.

It can be read almost all the data files from any source such some common sources are raw data text files, MS Excel files (csv, xlsx, etc.), relational database files (Db2), and so on. Apart from that, SAS is an independent platform for any operating system such as Windows, Linux, Mac, Ubuntu, etc.



SAS offers broad help to automatically change and divide information in the examination of the intuitive (drag and drop facility) interface of other BI apparatuses.

This reading material covers the most basic part of SAS that is data management, programming tools provided in SAS.



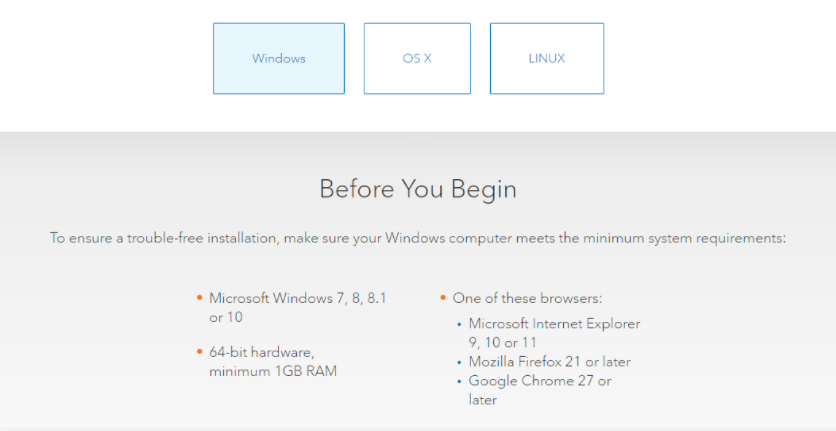
Download SAS University Edition

The SAS University Edition can be downloaded from the given link below <https://www.sas.com/en_in/software/university-edition/download-software.html>

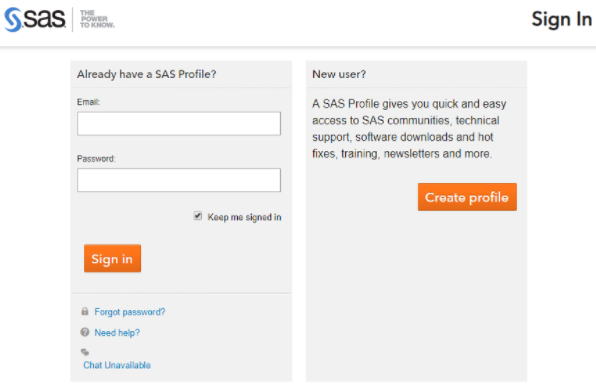
Click on this link and you will directly reach to the page shown below.



Choose according to your operating system. Here, we are going with the Windows operating system.



After that, you will be required to either create or sign-in on the SAS website. When you sign in or create a SAS profile and agree with the institute T&C’s, then you will have to download the SAS App file. The file size is around ~ 2 GB and it will take some time for downloading and then you can install it. Now, we could leave it for downloading and move it to Oracle VM Virtual Box.

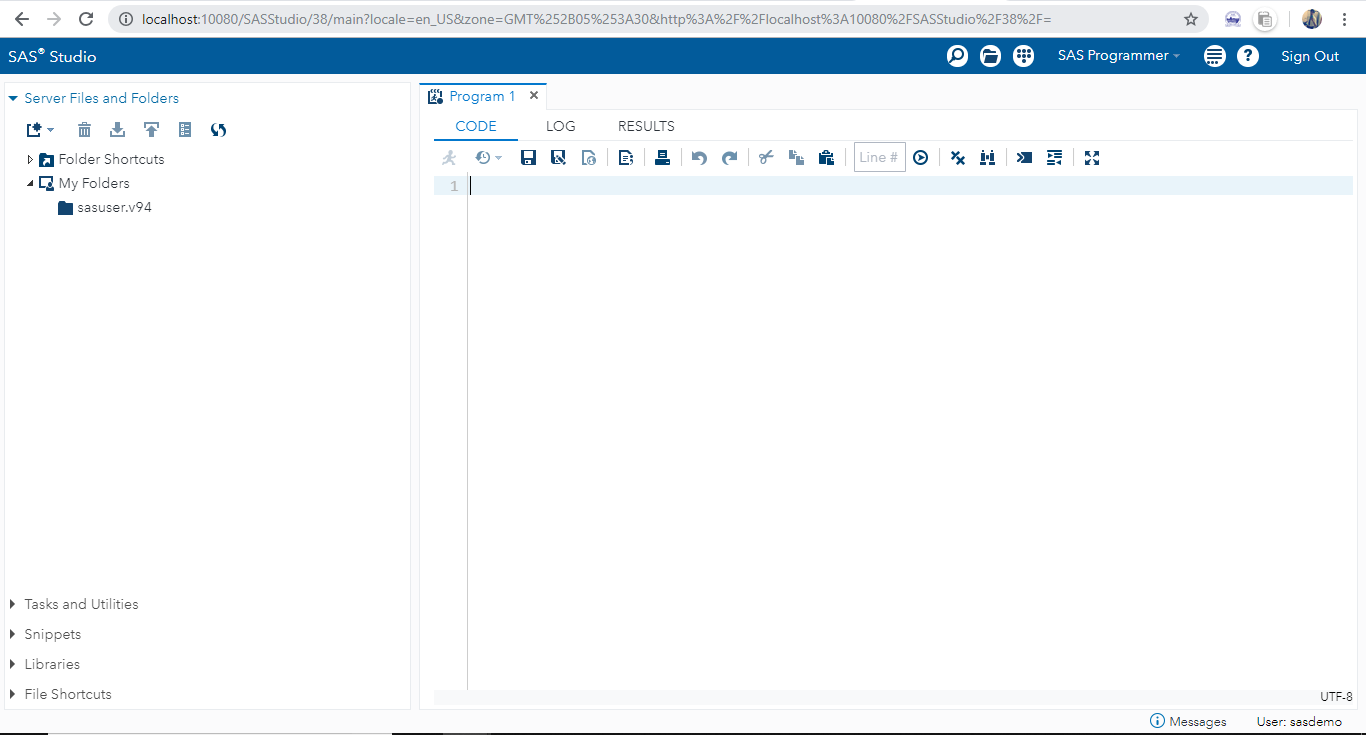




Now, download the *Oracle VM Virtual Box* from the given link <https://www.virtualbox.org/wiki/Downloads> because it will be used to run SAS studio.



You can select an appropriate option according to your operating system. Thus, download the virtual box software and install it in your system.

Finally, all the successful completion of the installation process, will show this window see below.

Alternatively, the simplest way to use SAS studio is to go online and the link is here <https://welcome.oda.sas.com/login>.You can sign in and click on the accept the terms of the license and the terms of use and conditions. After that choose the application of SAS studio and you direct reach the server of SAS programmer.

**SAS Names, Data Sets, and SAS Data Types**

There is a simple rule in SAS for variable names. The rule follows that SAS variable and data set names must begin with a letter or an underscore “\_” and be no more than 32 characters long. For example, FirstName, Last\_Name, Date\_290197. However, you need to know a basic thing that is cannot use the spaces or blanks and dashes because it is restricted and invalid characters for SAS names. For example, 21\_number, First Name, Last-Name. Note that, the variable name criteria are not case sensitive.

In SAS language programming, there is a data portion and descriptor portion. It is an important part to understand how it works and what is it, the SAS data portion stores or contains the data values. On the other hand, the SAS descriptor portion (also known as metadata) stores the information which is about the features of the dataset as well as information about each variable such as formats, length, labels, etc.



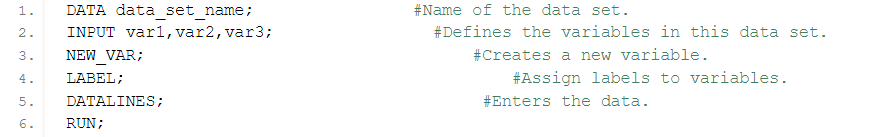
There are two types of variables i.e., Numeric and Character. It is easier to recognize as well as use rather than some other programs that include a variety of data types, such as integer, float, long-integer, and so on. Further, SAS fixes the storage of 8 bytes length for the numeric variable which implies the storage of 16 to 17 significant digits. On the other side, each character value (characters/letters, digits, special symbols, and spaces i.e., non-numeric data values) is stored with lengths ranging from 1 to 32 which implies that 767 bytes are possible. Note that, 1 byte is the same as 1 character.

**SAS Program Structure**

There are two main mechanisms of SAS i.e., Data Step and Proc Step. In this section, we deliver knowledge about the SAS program structure. Let’s start with the data step.

DATA STEP begins with the keyword “Data” statement. It helps to read the data, data manipulation, update, modify the data and perform calculations, according to the program. After the “Data” statement, you assign a dataset name and then start with the data set structure process as well as assign a variable name. Thus, it creates a SAS data set.

Syntax:



PROC STEP begins with the “Proc” statement (i.e., procedure). It helps to analyze the data, create a summary of the dataset, generate reports/outputs in the form of a table, create graphs for visualization purposes, and more.

Syntax:



In this process, you need to keep one thing in your mind, the data steps and proc steps end with the RUN statement. Note that, it is a hard and fast rule to use all SAS statements end with the semicolon “ **;** ”, otherwise you may deal with the error or syntax error.

Apart from that, there is a global statement that comes out any place in the SAS environment. TITLE and OPTIONS are the global statements. It affects the SAS session.

**The Data Statement**

The data statement is for assigning the name of the data set and a wide range of capabilities.

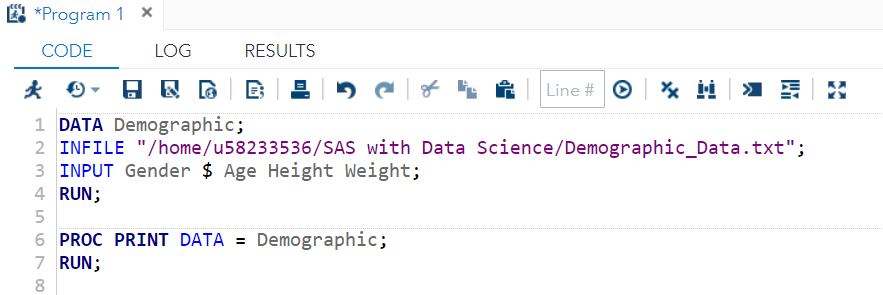
**The Infile Statement**

The infile statement coming out where the raw data is stored implies that the path of the file is given to the infile statement. It is valid in Data Step.

**The Input Statement**

The INPUT statement explains how values in an input record are organized. The INPUT statement reads records from a file given in the INFILE statement that was previously executed. SAS handlers who are familiar with a series of prompts can save time by entering data at their own pace rather than waiting for all of them to appear. Responses to an INPUT prompt sequence are accepted in the order they are entered.

Here is a simple and first program to read raw data and produce a report:

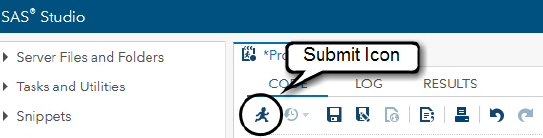


Output:



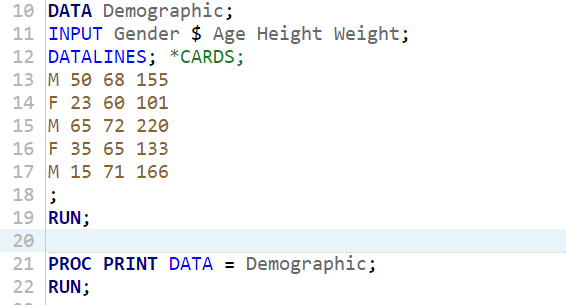
The above program is just an example of how Data Step, Infile, Input statement, and Proc Statement works in the program.

Note that, the below picture displays that how to run your program in SAS Studio. When you click on the submit icon (which is available on the windowing environment), then it will run the program. Alternatively, the shortcut key is F3 (Function plus F3).



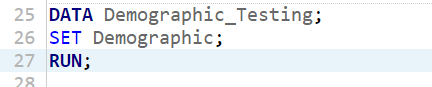
Alternatively, if you don’t have a txt file or any data file, then you may use DATALINES statement as the source of the data where you can put your data. It is the last statement of the data step. Thus, in this way, you could create your dataset by using the DATALINES statement.

Here is the sample program for DATALINES:



**The Set Statement**

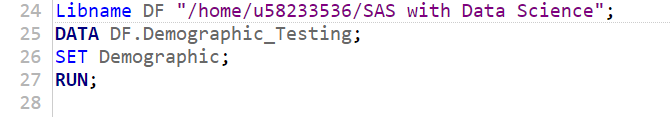
The set statement reads the data from an existing SAS data set. The function of the set statement is used in place of the infile and input statements. The statement is as follows:



The above statement creates a new data set as Demographic\_Testing which is similar to the Demographic data set.

Apart from that, you can permanently store the data set in the library by using the LIBNAME statement. You could clarify that the library DF refers to the directory “/home/u58233536/SAS with Data Science”. Subsequently, you are required to prefixture “DF**.**” with the data set name. In this way, the SAS dataset is loaded in that directory.

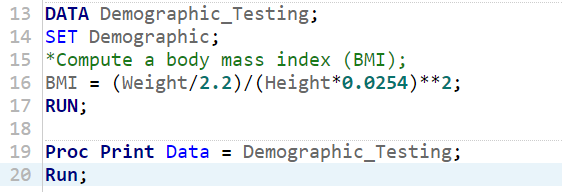
Here is the program:



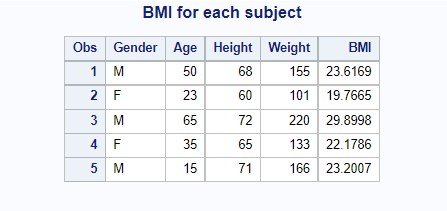
**Modifying SAS Data with SAS Operators (Arithmetic, Comparison, Logical, Minimum/Maximum, Concatenation)**

In this section, you will learn how to create new variables and modify the existing data. In this way, you can absorb more about SAS works. Let’s start with the how to create variables in SAS Studio.

Here is the program:



Program’s Output:



The statement which begins with *BMI equals to* is known as an assignment statement. It is a guide to play out the calculation on the right-hand side (RHS) of the equivalent sign and refer the subsequent value to the variable named on the left. In this above program, you are making another variable named BMI that is characterized as an individual's weight (in kilograms) divided by an individual's Height (in meters) squared. BMI is a valuable list of weight. Clinical specialists frequently use BMI when figuring the wellbeing dangers of different sicknesses, (for example, respiratory failures).

Apart from that, the comment statements are beginning with an asterisk mark (\*). They allow you to comment or make notes for yourself or others who will be reviewing your code later. Starting with an asterisk, writing as many comment lines as you want, and ending the statement (as you do with other SAS statements) with a semicolon (;) is one approach to make a SAS comment. Comments are useful not only for people trying to read and understand your code but also for you.

Furthermore, starting with a slash star (/\*) and ending with a star slash (\*/) is another technique to add a comment to a SAS programme. This type of comment can even be included in a SAS statement. You could, for example, write:

INPUT Gender $ Age /\* age is in years \*/ Ht Wt;

Moreover, the forward-slash (/) for division, the asterisk (\*) for multiplication, and the double asterisk (\*\*) for exponentiation are all utilised in this assignment statement. This is an excellent moment to go through the entire set of ***arithmetic operators***. The following are the details:

|  |  |
| --- | --- |
| Operator (Symbols) | Description/Function |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| \*\* | Raise to the power (Exponentiation) |
| - | Negation |

You can play with these athematic operators like a simple mathematics rule of algebra operations. You can follow the rule in this way the addition and subtraction happen after the multiplication and division.

For example,

a = 3 + 2 \* 5;

In this assignment statement, the value of a is 13. But if you would like to perform addition first then multiply, you require to use parentheses in the expression like this:

a = (3 + 2) \* 5;

In this assignment statement, the value of a is 25. Because parentheses around the 3 and 2, so that’s why 3 and 2 added composed first, then multiplied by 5 that is a = 25. Based on the above expression, we can say that, whenever the parentheses are included in the expression, you will need to execute/run it first then other operations.

One more example of exponentiation and negation are as follows:

a = 3\*\*2 + 2 \* - 4;

a = 9 + 2 \* - 4;

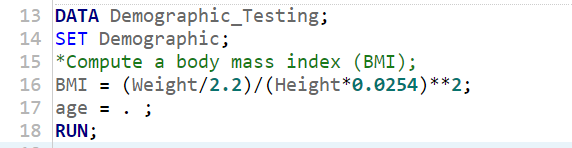
a = 9 + (-8);

\*Thus, this expression will give you;

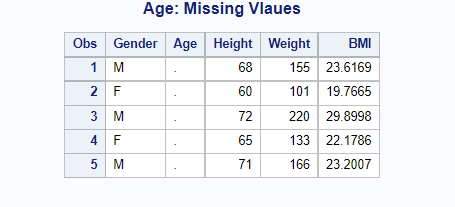
a = 1

From the above expression, you learned exponentiation and negation performed first.

Now, move to the next part of the modification in SAS. Here, we give you a basic knowledge about the missing value through a programme:



In the 17th line of code, the age variable is set to a missing value through an assignment statement (see the output of the program):



The missing value of the numeric variable is shown as period (.). Looking at the output where age variable values are represented by a period (**.**). On the other hand, if the variable type is the character, then you can assign a value to the variable (like this Gender = “Male”) in the enclosed quotes.

Quick Note: An expression is a part of a statement where an actual operation is performed and the complete line of code is known as a statement.

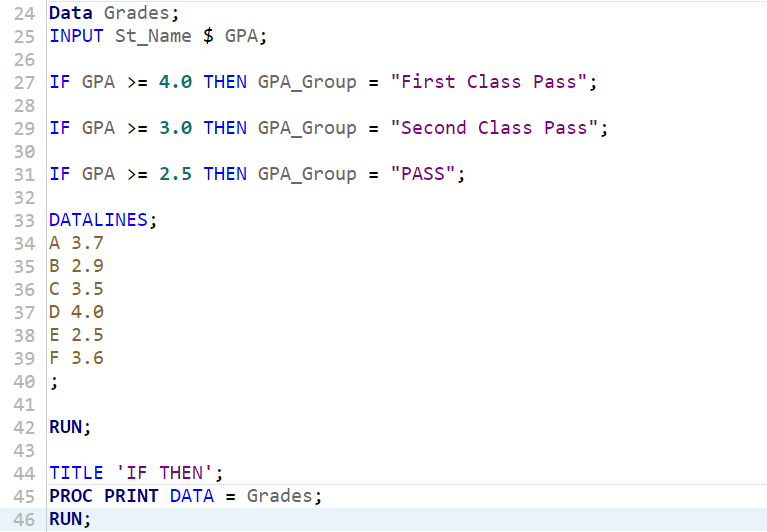
Let’s talk about the ***Comparison Operator****,* the assigned expression could be used inside an *if then* statement to change the value of a variable for certain data but not for others, or to create different changes for separate sets of values. The below table shows comparison operators as:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbolic | Mnemonic | Meaning | Example | |
| = | EQ | equals | | IF Sex = ‘M’; or IF Sex EQ ‘M’; |
| ^= or ~= | NE | not equal | | IF Income NE; |
| > | GT | greater than | | IF Income GT 2700; |
| < | LT | less than | | IF Income LT 2700; |
| >= | GE | greater than or equal | | IF Income GE 2700; |
| <= | LE | less than or equal | | IF Income LE 4500; |
| in | IN | selecting multiple values | | IF state IN (‘DL’ ’IN’); |

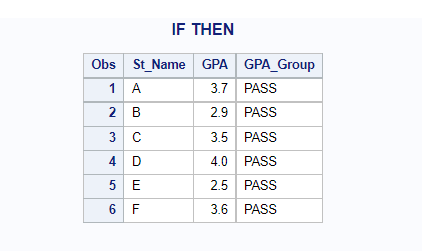
Now, generating a new variable by using IF THEN condition. Consider the case of a student's GPA. Let's say we want to create a variable named GPA\_Group that can take one of three values:

* "First Class Pass" for those with a GPA\_Group greater than or equal to 4.0,
* "Second Class Pass" for those with a GPA\_Group greater than or equal to 3.0 and
* "Pass" for those with a GPA\_Group greater than or equal to 2.5.

Here is the program:

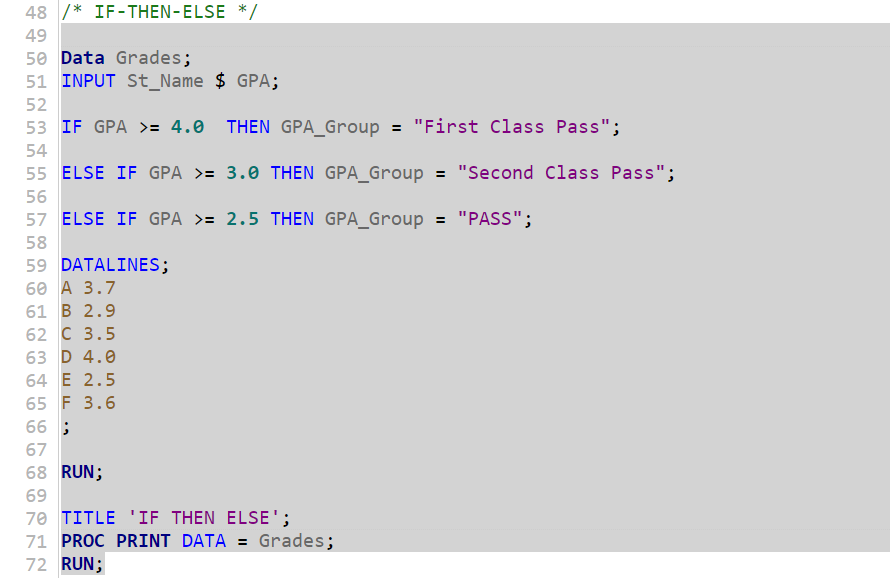


Now, we run the code and get the following output:

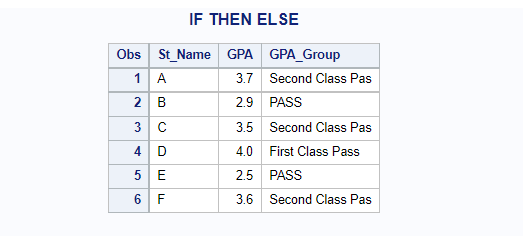


The output creates the new variable correctly but the result is not like what we want. So, we have to change the code and use the concept of IF-THEN-ELSE. Let’s try this concept in the above program:

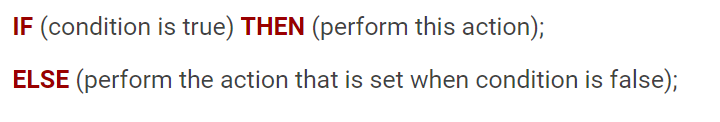
Here is the program:



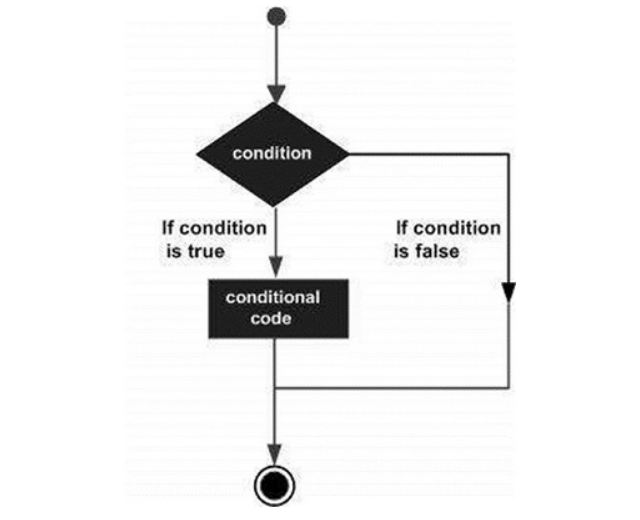
Program’s Output:



This program’s output is satisfactory or matched with the example conditions. Besides, from the IF-THEN statement, you learned that the SAS code follows logic rules: if-then statements are evaluated in the order they appear in the data step. On the other hand, the If Then Else statement is working, IF clause is met or True THEN statement run this action, and Else condition is an alternate action if the THEN statement is not performed or False (see blow picture as a syntax).



Apart from that, this type of condition is called **decision making structures** where the programmer evaluated/tested one or more conditions at the same time. The standard form of decision making structure works like this chart (see below)

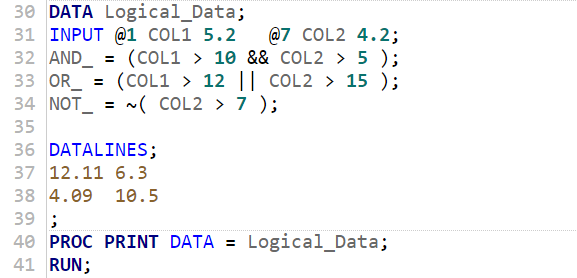


When the statements will pass or are executed at that time condition is determined either to be true or false.

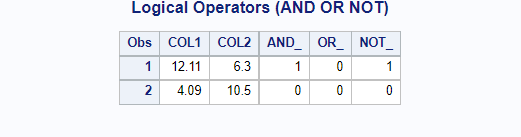
Now, Let’s talk about the ***Logical Operator***, a logical operator is a symbol or word that connects two or more related logically, so that the value of the composite statement created is solely determined by the value of the underlying statements and the operator's meaning. AND, OR, and NOT are examples of common logical operators.

|  |  |  |  |
| --- | --- | --- | --- |
| SAS Logical (Boolean) Operators | | | |
| Operator | **Description** | **Example** | **SAS Output** |
| && | **Logical AND** | If a = 3 and b = 5  a < 10 && b > 3 | TRUE |
| || | **Logical OR** | If a = 3 and b = 5  a == 2 OR b == 5 | TRUE |
| ! | **Logical NOT** | If a = 3 and b = 5  !(a == b) | TRUE |

The logical operators are labelled in detail in the above table. Let’s say that two data variables, **a and b** have values of 3 and 5, respectively.



After running the above code, we get the following output:



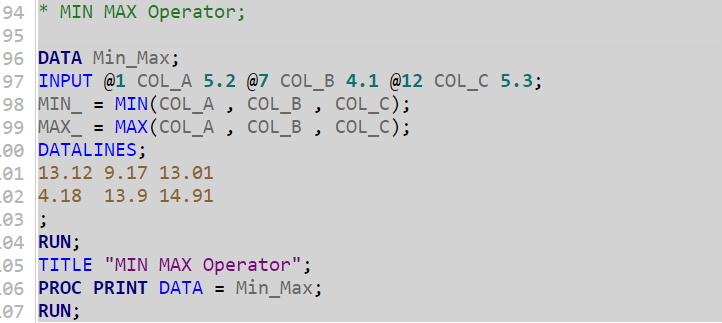
The explanation of Logical Operators are as follows:

* AND – When both the conditions are TRUE, then the result is 1, otherwise 0.
* OR – When any one of the conditions is TRUE, then the result is 1, otherwise 0.
* NOT – When the condition is satisfied or TRUE, then the result is 1, otherwise 0.

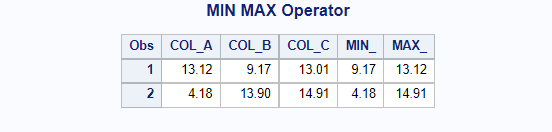
Now, Let’s talk about the ***Minimum/Maximum Operator,*** if you would like to find the minimum or maximum value of two quantities, use the MIN and MAX operators. Around the operators with the 2 variables, you wish to know their minimum and maximum values. The smaller of the two numbers is returned by the MIN (><) operator. The higher of the two values are returned by the MAX (<>) operator.

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| MIN | The MIN Operator. The result of MIN operator is minimum value from all the data observations in the row. | MIN (5.2, 9.7, 7.1)  >> 5.2 |
| MAX | The MAX Operator. The result of MAX operator is maximum value from all the data observations in the row. | MAX (5.2, 9.7, 7.1)  >> 9.7 |

Here is the code of min and max operator:



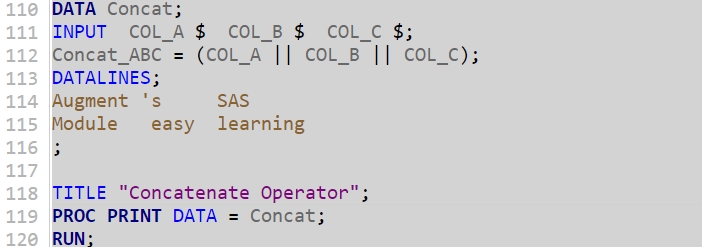
We run the above code and get the output (It shows in the RESULTS tab):



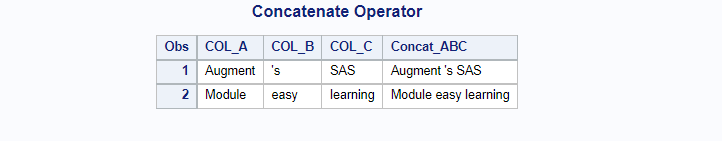
Now, Let’s talk about the last Operator i.e., ***Concatenation Operator,*** the meaning and work of concatenation are to add or combine two more string values and produce a single character value. The symbol of the concatenation operator is || (double pipe), as well as SAS, which support the concatenation function i.e., CAT function.

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| || | The Concatenate Operator. The result of Concatenate operator is to combine two or more values. | “Hello“||”SAS”  >> Hello SAS |

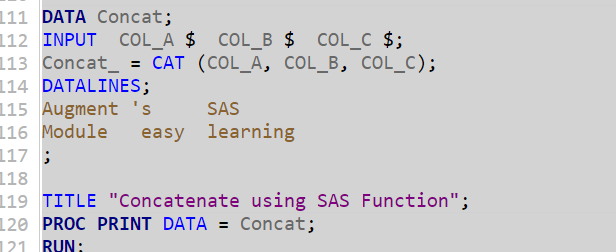
A simple example of Concatenate SAS Program using Operator (that is ||):



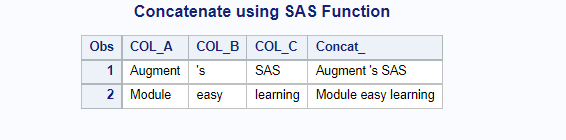
We run the above program (F3), and get the concatenate output as:



Alternative, concatenate Example by using SAS function (i.e., CAT Function):



SAS Output:

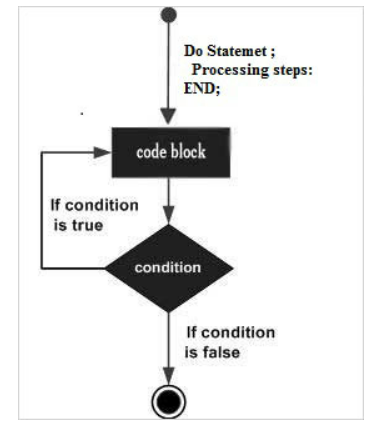


**SAS- LOOPS (DO Loop, DO UNTIL Loop, & DO WHILE Loop)**

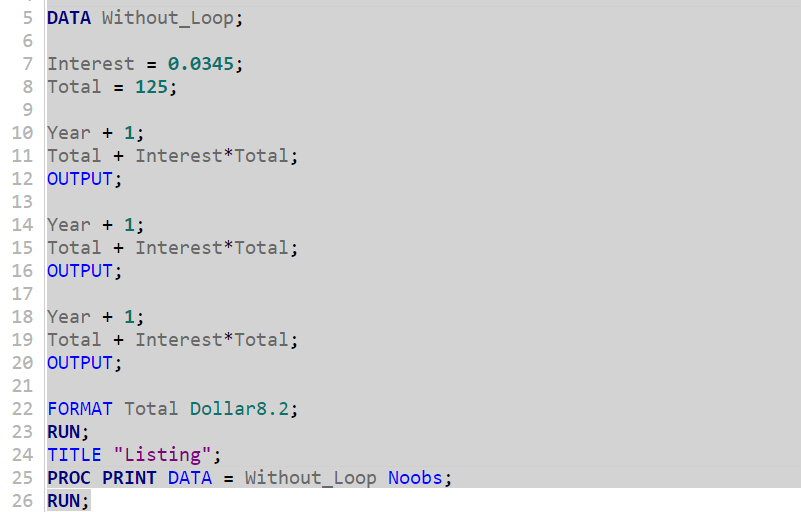
SAS Loops offers an extensive range of options performing repetitive movements on SAS data sets persistently without having to create identical/duplicate code or automatically performing the same statements several times.

* **DO Loop**

DO Loops are the most basic form of loops in the SAS which can be performed in the DATA Step. An iterative DO loop's operations are unrestricted, which means that if you tell it to execute the program 20 times, then it will execute only 20 times without ending (without mistake arising throughout program running). Do Loops use when the task is like calculate something over and over again or incremental counting. The below chart is a flow diagram of the DO Statement.

****

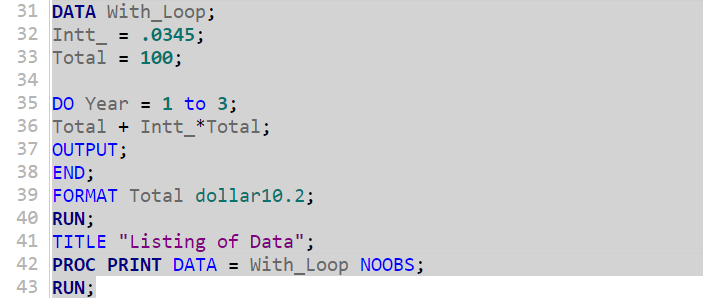
First, we will write the program without iterative loops, thereafter we write the program with iterative loops.



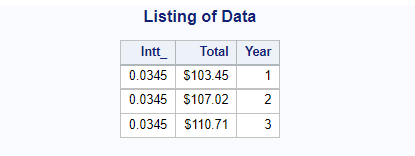
Output:



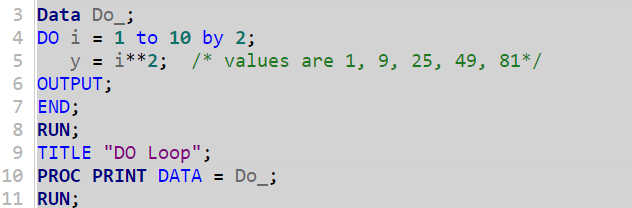
With Iterative Loop:



Output (similar as above):



Now, let's look at one more example of how to use a basic iterative DO loop.



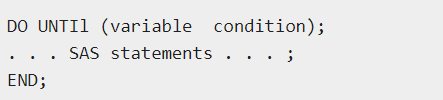
Output:



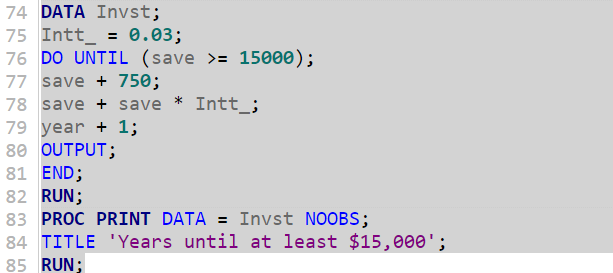
Note: The ***by*** option with the loops statement means that the increment in the value of the index variable in each iteration of the DO loop. The increment value is not belonging to zero. However, if the ***by*** the statement is silent in the case of loops, then it can be treated as increment the value of the counter by 1 which is the default. On the other hand, SAS always produces an automatic output, however, when an OUTPUT statement appears wherever in the DATA STEP program, then at that time SAS does not able to perform automatic output.

* **DO UNTIL Loop**

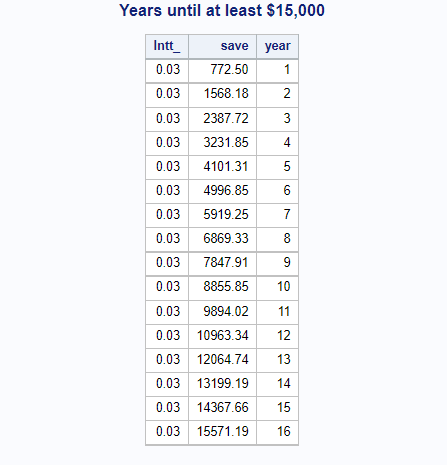
It is better to choose a condition like where you could stop a loop value or condition is true rather than terminate an iterative Do loop. The DO UNTIL statement performs until a logical statement is met or true. The condition of Do Until Loop is sited in the parentheses which use the UNTIL clause and the statement is tested at the end/bottom of the loop. The syntax is as follows:



Example: Let's say you want to know how long it would take to save $15,000 if you put $175 every year into a 3% interest account. Here is a demonstration program that performs the calculation by using SAS Do Until Loop as:



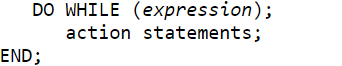
Output:



**Note**: The Do Until Loop always runs at least on one occasion. As you can check the last observation of the output’s value is greater than or equal to the 15,000 but not more than that which means that DO UNTIL statement is not performed until the bottom of the loop.

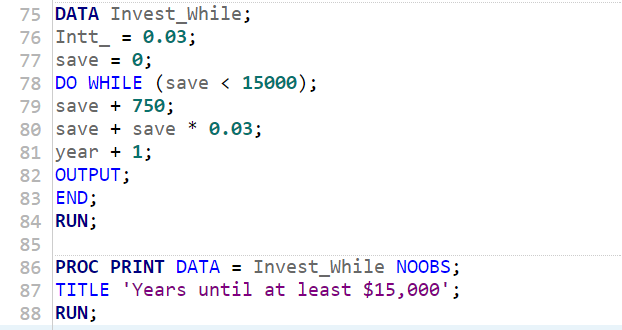
* **DO WHILE Loop**

The Do While Loop is a substitute for Do Until. The condition of DO WHILE Loop is sited in the parentheses which use the WHILE clause before the Do while loop statement and the statement is tested at the top of the loop rather than end/bottom of the loop. If DO while loop expression is True, then it will be iterating. However, the while clause (or expression) is not true, then the loop does not even execute at one time. This statement is only valid in the Data Step. The syntax is as follows



Example: The same example has been taken as you did previously.

Here is the correct program is as follows:



After launching or running the above program, you will get this result as:



Additionally, if you change the WHILE expression case like save > 15,000, then there will be no output in the Result tab or output tab of SAS Studio. In this case, the meaning of no output is that the SAS program is not effective or fails or DO WHILE is unable to execute this program.

**SAS** – **Arrays**

Many languages use arrays for programming but SAS arrays are different from others. They don't have any values, but they do enable you to quickly refer to a group of SAS variables. SAS Array processing is a way for doing the same operation on multiple variables at the same time. Array processing is useful for a range of tasks, such as repeating calculations on numerous variables or producing additional variables with the same features. Arrays make sense to practice when the values are related and continuously recurring, so in that case, the programmer wants to loop through the majority of them. In the data step, the combination of arrays and do loops gives tremendous power to programming.

The syntax of the SAS array is followed as:



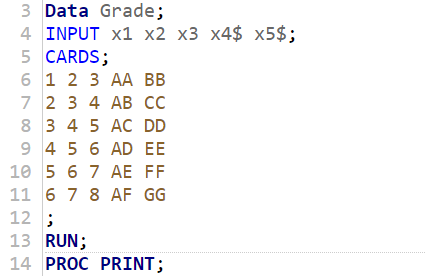
The description of the above syntax-

* Array- It is the SAS Keyword that uses an array
* ARRAY–NAME – It is the name of the array similar to variable names.
* SUBSCRIPT – the number of values that are going to store in parenthesis.
* ($) – it is an optional parameter and the dollar sign indicates that the array is going to store in character value.
* VARIABLE-LIST- SAS variables list which includes array value.
* ARRAY-VALUES- initial value stored for each of the array elements.

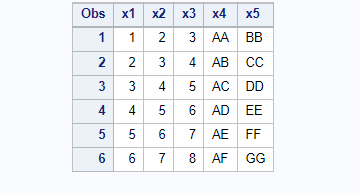
There are some examples which show that how to define or assign the character and numeric values to an array.

* array marks {\*} $ m1 m2 m3 ('a', 'b', 'c');
* array marks (1:4) $ m1 - m4;
* array marks {4} m1 m2 m3 m4 (95 90 85 85);
* array marks {4} m1 - m4 (95 90 2\*85);

Here is the SAS simple code or program:

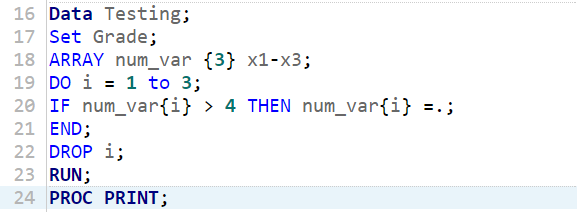


Output:

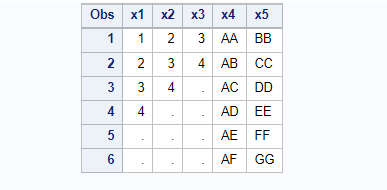


In the above program, we are going to modify the output with the help of an array. Let’s start with the simple task i.e., if the numerical value is greater than 3 then treat it as a missing value.

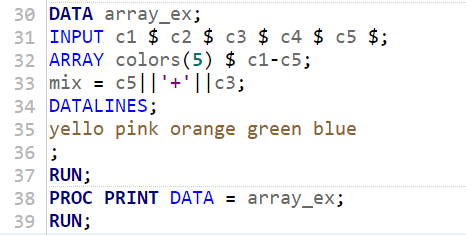
The program is as follows:



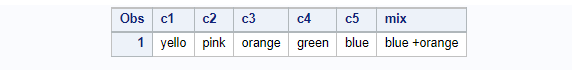
After running the above program, we get the result, shown below:



One more example for SAS Array:

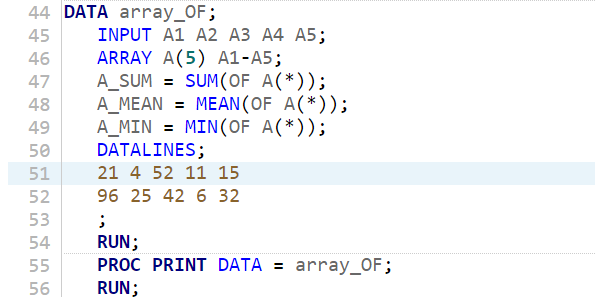


Output:

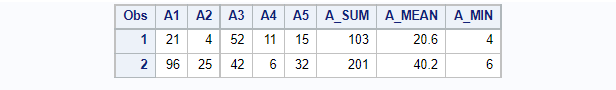


**SAS Array with OF operator**

The OF operator is used to execute calculations on the whole row of an array. The below program uses the OF operator with an array.



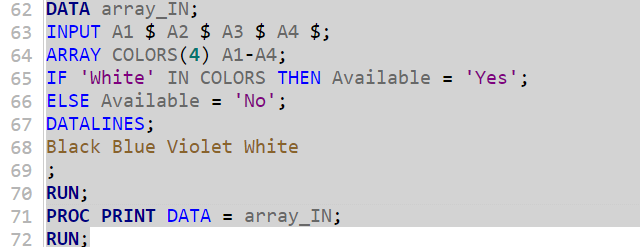
Output:



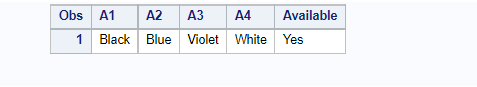
**SAS Array with IN operator**

The IN operator, which tests for the presence of a value in the array's row, can also be used to retrieve an array's value.

In the below program, we will verify the presence of the color “White” in the data. Let’s write the code in SAS Studio:

****

We run the above code and get the result as below:

****

**SAS – Functions**

SAS offers a large number of built-in functions that aid in data analysis and processing. The functions are used in the part of the Data step. They consider data elements as inputs and produce a response that is saved in a different variable. The number of parameters a program accepts varies depending on the type of function. Several functions will take no parameters, while others take a predetermined number of variables. The types of functions SAS offers are listed below.

The standard syntax of SAS Function as:



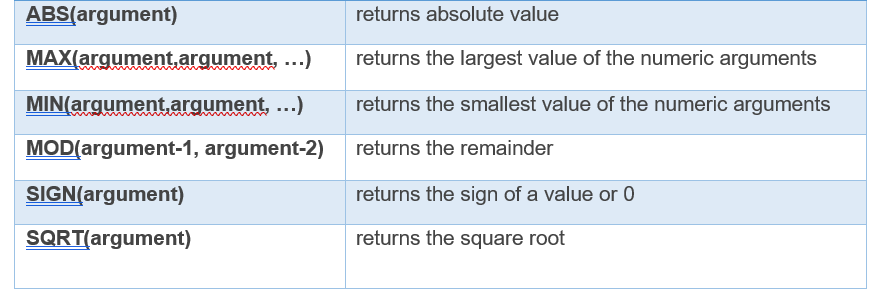
Note that the argument might be the constant, variable, expression, or so on.

SAS functions are classified into several types based on how they are used and what tasks they perform. The following are the many types of SAS functions:

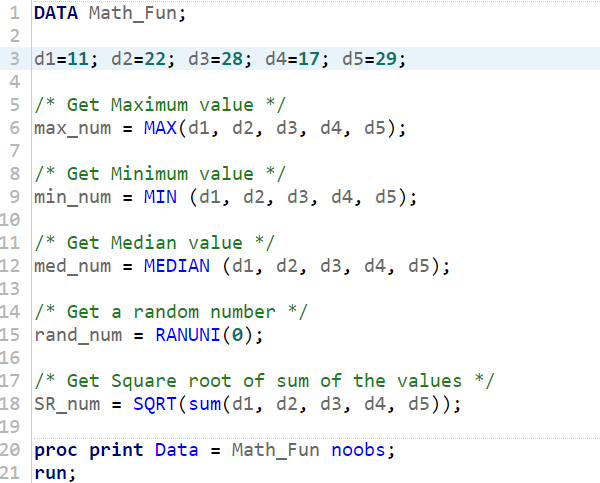
* *Mathematical or Arithmetic*
* *Character (String)*
* *Date and Time*
* *Truncation*
* *Miscellaneous*

**Mathematical or Arithmetic Functions**

SAS mathematical functions are another name is numeric functions. It carries out mathematical operations on a single input or a group of arguments. In the below box, these are some functions with its description.



We will give you some examples of the most common and important functions that are used mostly by programmers.



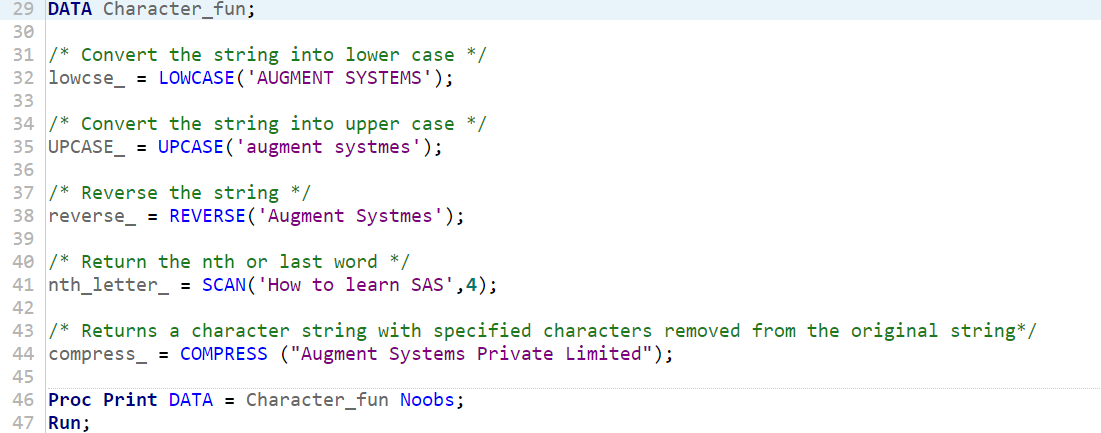
After running the above code, we get the result in the results log as:



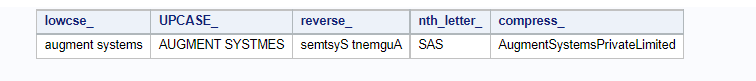
Note: If we run the program again and again, then all the output remains unchanged, however, the output of Random Number (i.e., comes from the RANUNI function) will always change or differ.

**Character (String) Functions**

This course includes an example of the most often used SAS character functions. Dealing with character strings is more difficult than dealing with numeric data. As a result, understanding how to use character functions in practice is necessary.

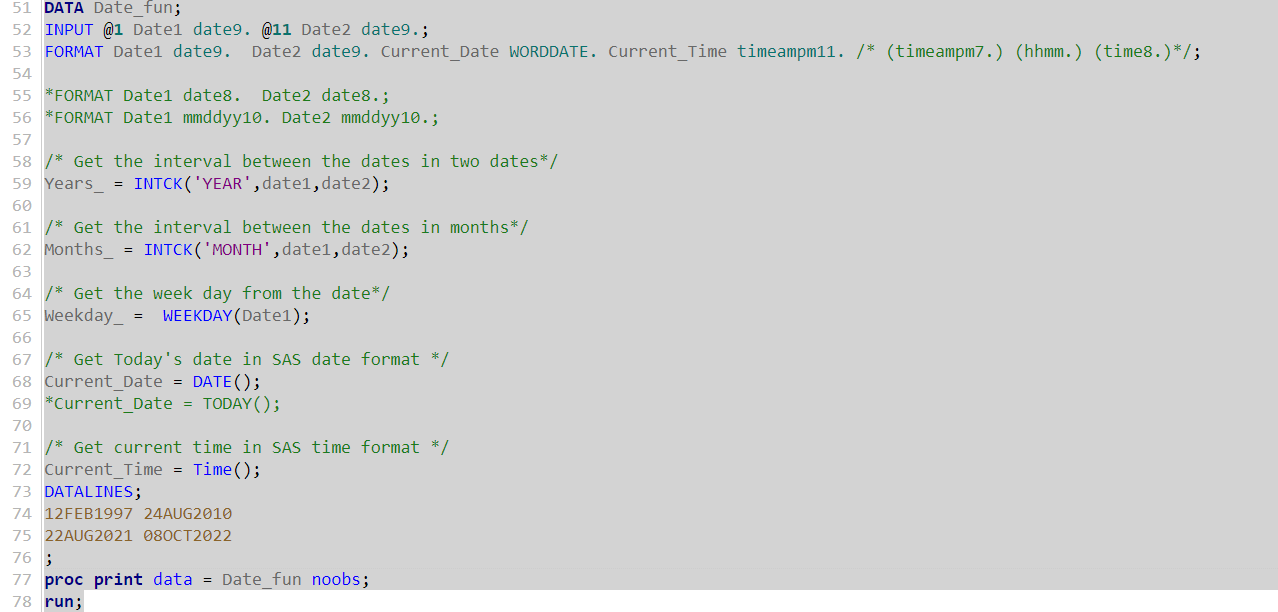


After running the above code, we get the result as:

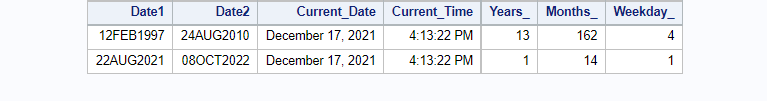


**Date and Time Functions**

The date and time functions help in the procedure of date and time values. We will give you examples of some important functions see below:

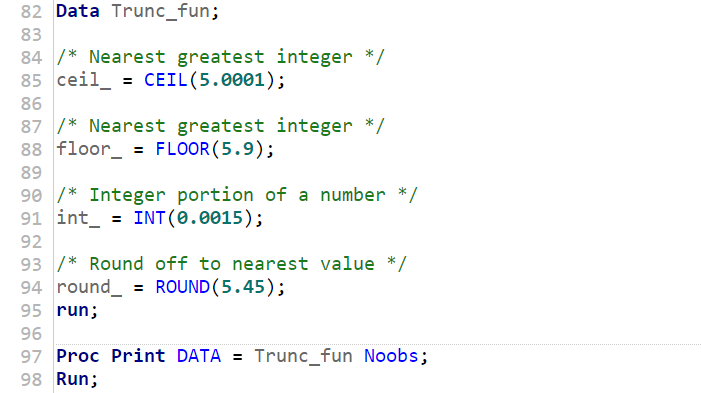


After launching or running the above program, you will get this result as:

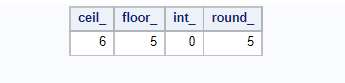


**Truncation Functions**

First, start with the meaning of Truncation which could help us to understand that how truncations functions work. The meaning of truncation is that the process of cutting anything shorter or faster, especially by omitting or removing the end of it. Let’s write a program of some truncation functions through this you can easy to get these functions are as follows:

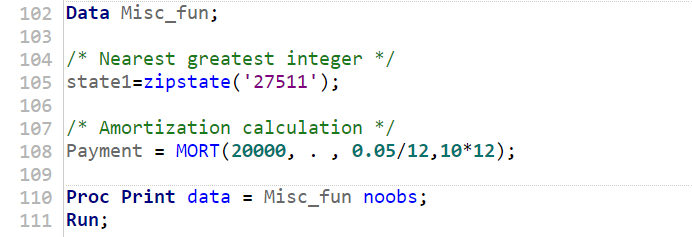


Output:



**Miscellaneous Functions**

Here are some examples of miscellaneous functions are as:



Output:



**SAS – Input Methods**

The input statement helps to read the raw data set which is valid under the Data Step. In the input statement, you must specify a variable with the name to each field which is followed by $ (which indicates a character value), pointer control, column-specifications, informat, and/or line hold specifiers (i.e., @, and @@). We are going to discuss several input methods are listed below.

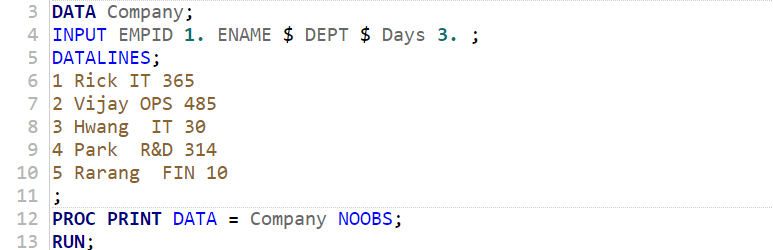
* List Input Method
* Named Input Method
* Column Input Method
* Formatted Input Method

The next sections go into the specifics of each input method.

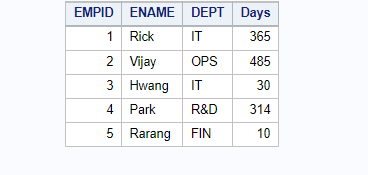
**List Input Method**

In this process, the variables are mentioned together with the data types in the input statement. The raw data is carefully examined to ensure that the defined variable order aligns with the data. All pairs of end-to-end columns should have the same delimiter (generally a blank/space).

Let’s take an example and understand the program (see below).



Output of the above program:



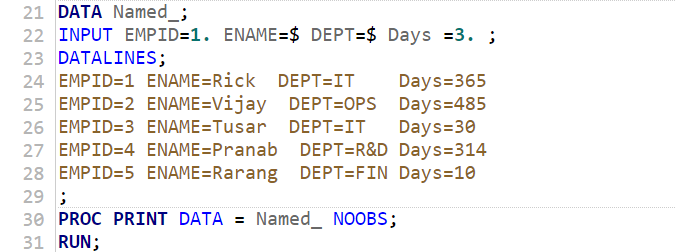
The shown above and all the data elements are well placed in the output table. A $ (dollar sign) should be used after a character variable in the input statement. In the datalines statement and input format, the space or blank does not indicate a missing value. If the value is missing, then it will be always treated as a period (.). In SAS, the maximum length of the string variable is 8 characters which means that if the string variable is greater than 8 characters, then it will be automatically trimmed. So, in that case, you must specify the length of the character. We will show you the next couple of examples.

*Note: At least a single space must specify between every value.*

**Named Input Method**

In this method, the named input uses the read data values followed by the variable names and an equal sign (=). The INPUT statement that follows reads the datalines that have equal signs. Let's have a look at an example.

Here is the program:



Output:

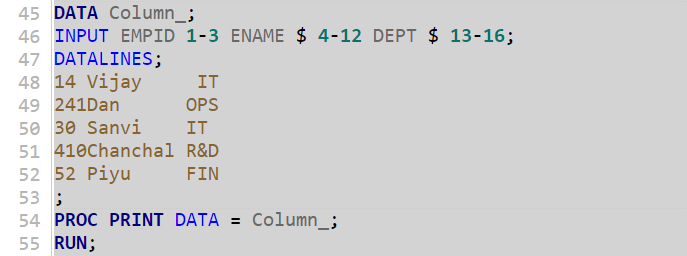


This method is the easy way to create the data set. If the data set is small then this method could be useful. But if we talk about practical life the datasets are huge or millions of observations. So, this method is not useful/appropriate and is too time-consuming.

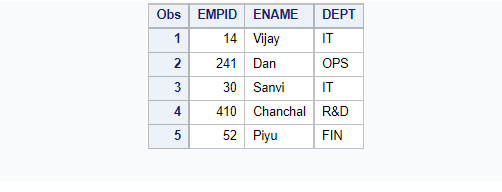
**Column Input Method**

It is the method or style which allows you to read the data values that are ordered and fixed in the columns. In simple words, the column lengths describe the value of a single column of data. Suppose student name has a maximum of more than 8 characters, so you have to begin in the 10th column in the datalines statement. Let's have a look at an example of the column input method.

Here is the program:



Output:



In the above program, the input statement displays that the column lengths like Employee Id (EMPID) is 1 to 3 which implies that the EMPID max length is 3. Similarly, the Employee Name (ENAME) begins with the 4th column length and ends at the 12th column. In simple words, the INPUT statement tells that the variable is a character (i.e., ENAME) and seems in columns #4-12. Thus, ENAME is a character variable length is 8.

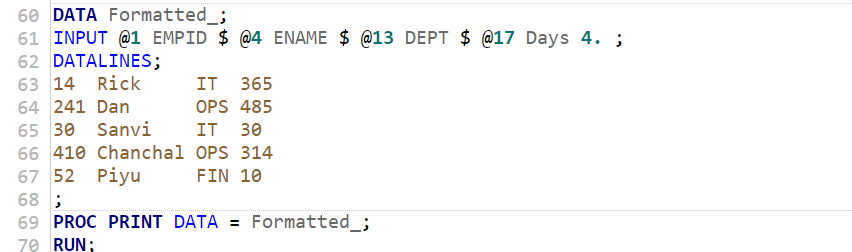
In this method, you have to only take care of the variable’s length and then assign the column length in the input statement. Also, you have to check if it is a character then assign $ and if it is numeric, then assign numeric format like (3.).

*Note: The data values spaces are not compulsory.*

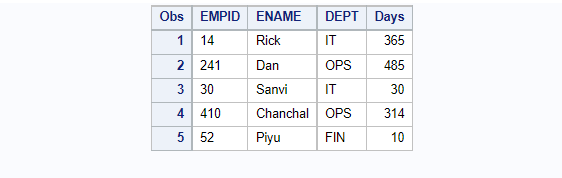
**Formatted Input Method**

The values are read from a set beginning position until a space is met. Each variable has a set beginning position, where the number of columns between any pair of variables equals the width of the initial variable. In short, the input statement format like “@n” character is must specify the initial column position of a variable as the nth column. Let's have a look at an example of the column input method.

Here is the program:



Output:

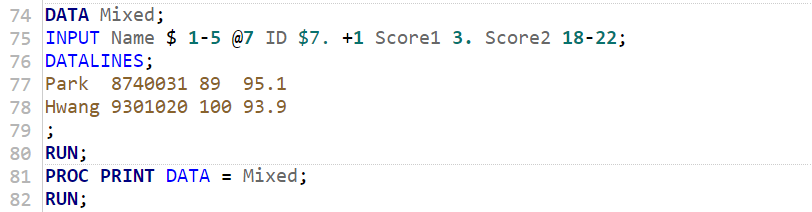


As shown in the above program, the symbol “@” indicates that the initial position of the column.

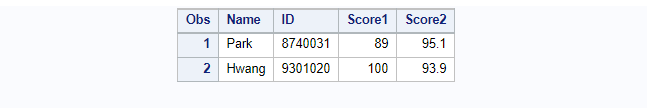
**Note**: For all the input methods, the format for all character variables is “$CHARw.” or “$w.” where the w denotes that the full length of the variable. Whereas, the format of all the numeric variables is “w.d” where the decimal point(s) after the number.

Moreover, it is not the hard and fast rule, you can use multiple input methods in the input statement. It is referred to as the Mixed Input method. Let's have a look at an example of the column input method.

Here is the program of mixed input methods:



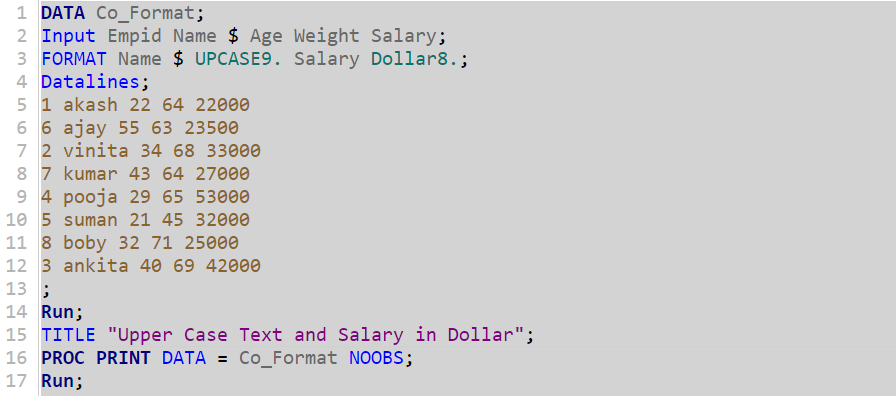
Output:



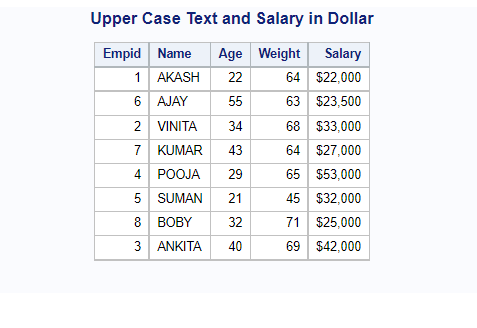
**SAS – Format Data sets**

Format is an in-built feature of SAS where you can easily modify the presence of the data sets and output means that you can customize the data set as you want to like you can add dollar sign, comma with decimal, uppercase and lowercase the text variable, date format of the variable and many more. There are different uses of FORMAT statement, if you apply format statement in DATA step, it will data format of the data set. On the other side, if you apply the format statement in PROC step, then it will define the variables according to the user. Let's have a look at an example of Format statement in Data step.

The program is below:



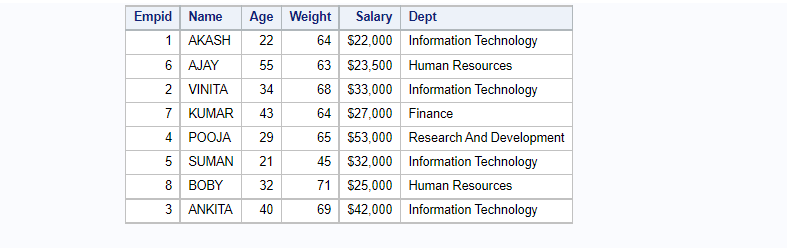
Output:



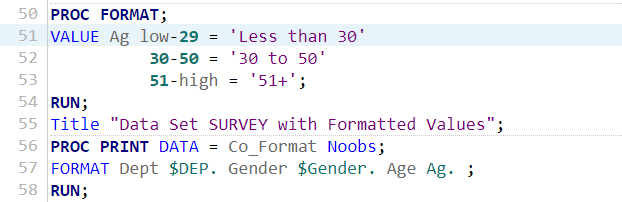
Another example of Format statement for ***character variable*** in Proc step and the program are below:



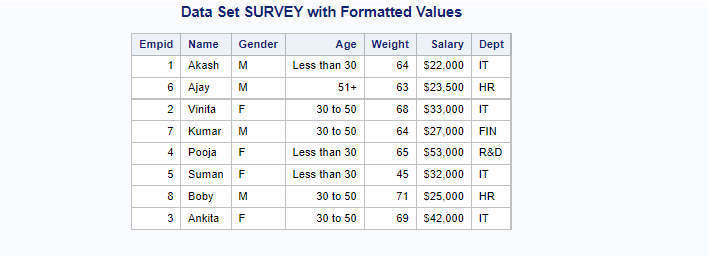
Output:



Another example of Format statement for ***numeric variable*** in Proc step and the program are below:



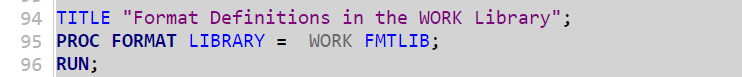
Output:



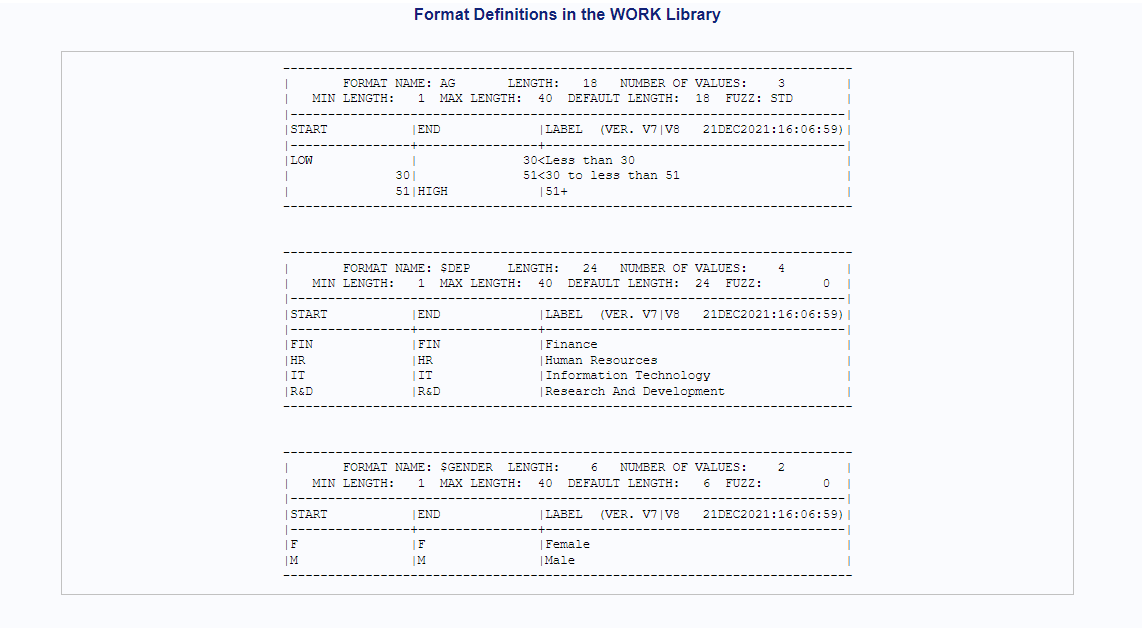
Here, we use the FORMAT statement but the program structure is different (i.e., DATA step and Proc step), the result is different. The format of $Dept. $Gender. Ag. Are used to format variables Dept, Gender, and Age, respectively. If we use Format statement in PROC PRINT, then it will expand the Dept abbreviations. Thus, this is the way to use a format statement. Note that each format is followed by a period (.), as is the case with SAS built-in forms.

Now, the next topic is about how to display your display format definitions.

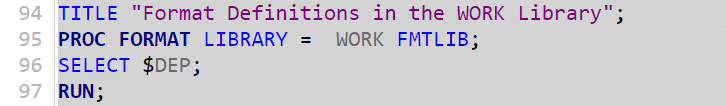
**FMTLIB** is a helpful PROC FORMAT option. This option generates a table with the values and descriptions for each format in the chosen library. To reveal the descriptions of all of the formats in the WORK library, for example, you would proceed to submit code.



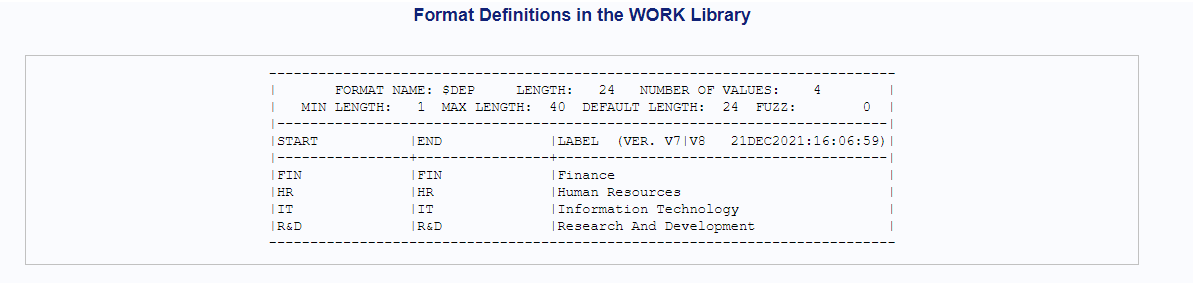
Output appears in a table form like this:



On the other hand, if you want any particular variable format then you can use SAS SQL “SELECT” keyword. You could use the following program as



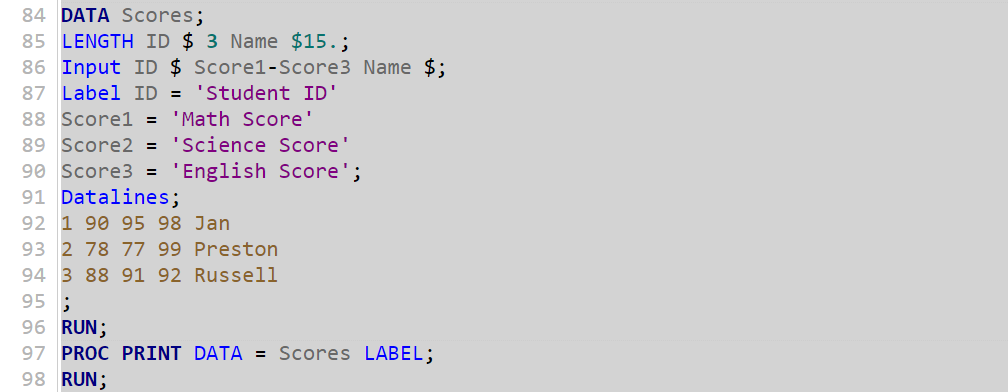
Output:



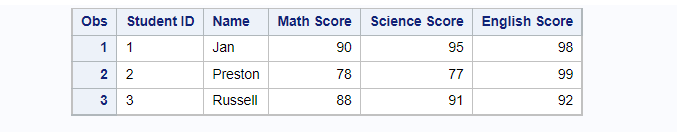
In this way, you can use the SELECT statement in the PROC FORMAT.

**SAS – Labels**

The SAS labels help to enhance the output clearer and more appealing. You can achieve this with the aid of SAS formats and labels. Labels can be created in either a DATA or PROC stage. For example:

****

Output:

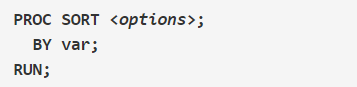
****

We can see that the heading/variable of the output is printed with the defined labels. Thus, SAS labels improve readability.

**SAS – Sort Data sets**

SAS data sets can be sorted based on all the attributes they contain. This seems to be useful for data evaluation as well as other tasks such as merging. Sorting can be done on a single variable or a set of variables. PROC SORT is the SAS procedure that performs the sorting in the SAS database. The sorted result is saved in a new data set, while the original data set is left untouched.

The basic syntax for sorting a dataset is as follows:



The options used are described below.

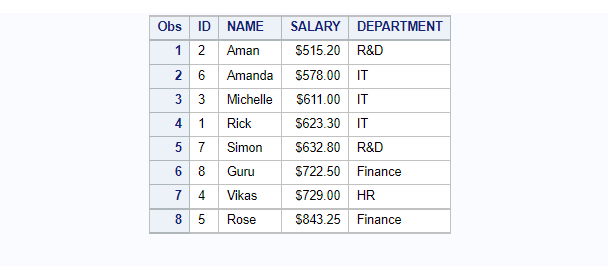
**var** – column’s name on which the sorting occurs.

**<options>** - dataset name that will be sorted.

Here is the program:

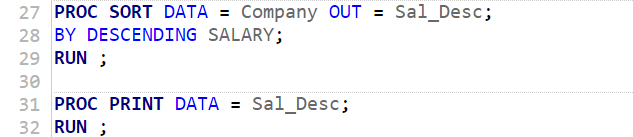


Output:

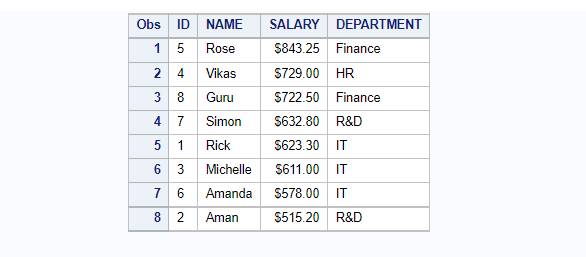


The above data set, which contains the information of the company’s employees. Using the code above, you can sort the dataset by salary.

On the other hand, if you want to sort the data in a reverse way then you just have to add DESCENDING statement before the variable name. Here is the simple line of codes.

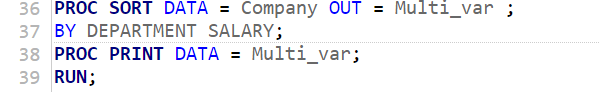


Output:

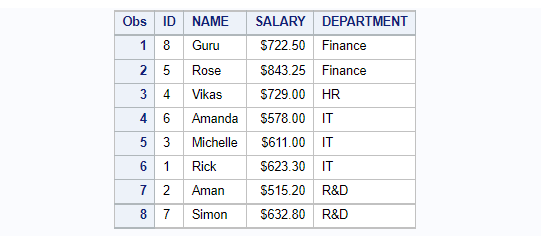


Note: When you want to sort the data set in ASCENDING order, you do not need to add ascending keyword before the variable name because SAS by default arranges the order of data in ascending way.

Additionally, if you want to sort the data set more than one variable, then it is also possible for multiple variables. It is a simple task just like you did for the above program. For this, you require to add those variable names that you want to sort after the BY statement. We display the code below.

****

Output:

****

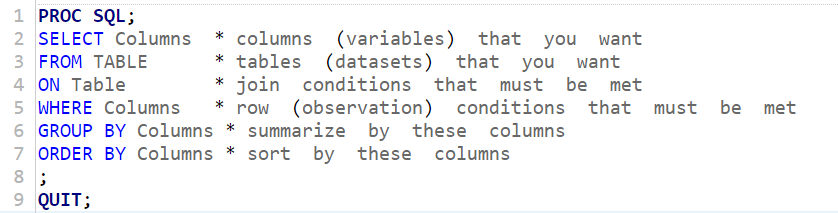
Note that one thing for multiple sorting is that the variables are arranged in order of importance from left to right.

**SAS – SQL**

The SAS® System supports\allows a number of languages, including SQL (i.e., a relational database language). The SAS user gets access to a strong data manipulation and query tool by using PROC SQL statement. PROC SQL can select, subset, sort, grouping, join (merge), summarize data will be addressed without the Data step procedure. This technique can build SAS tables and variables in addition to returning the result of a SQL query.

In this section, you will learn how to use SQL within the SAS system and why PROC SQL is an important tool within the SAS system.

The simplest syntax for using PROC SQL is shown below:



Now, begin with small or basic keywords of SQL language.

* **SELECT statement**

The select statement is useful when you want to select a particular column or a specific variable name. To do this, use Select statement as:

**SELECT** Empid,

Name,

Salary;

In a select statement, you can select as many as a column as per user needed and make sure all variables are separated by commas.

* **FROM clause**

The next step is FROM clause statement which is useful for obtaining the table from the dataset. In simple words, FROM clause is basically used in the select query to find the information from the table. The from clause can be used in views, procedures and tables. Use the FROM clause to perform this:

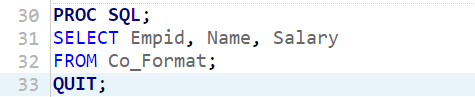
**SELECT** Empid,

Name,

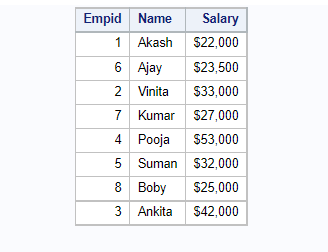
Salary;

**FROM** Co\_Format;

When you use these two aspects (SELECT and FROM), required for a valid or successful SQL query means that the query is complete. We only need to make a couple of changes for SAS to be able to run it.



The output of SELECT and FROM:

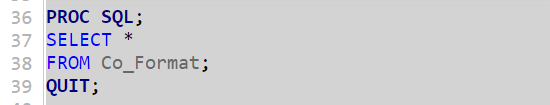


The result of the above query shows that the columns are arranged in the order that they were supplied in the SELECT statement.

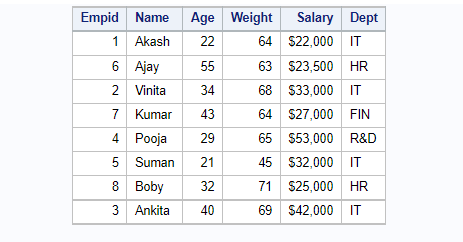
There are a few points to keep in mind with PROC SQL's syntax.

To begin, keep in mind that the complete query (SELECT... FROM...) is handled as a solitary quote. There is just a single semicolon, put toward the finish of the query. This is valid regardless of how complicated the query is or the number of variables it contains. Next, rather than using a RUN line, the action is stopped with a QUIT assertion. Queries are executed quickly when they are finished which indicates that the semicolon (;) on the SELECT statement is encountered. There are a few consequences like the “quit” statement isn't needed for queries to execute.

Moreover, if you want to select all the columns from the table instead of a list if column, then there is a shortcut of selecting all the columns from the table can be using an asterisk (\*). For instance, see below query;



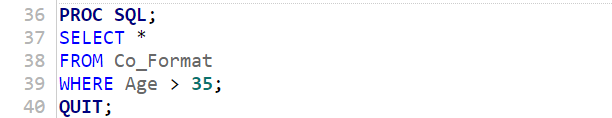
Output:



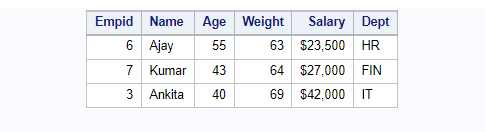
Thus, this is a time saver command.

* **WHERE clause**

Where clause is used in the select query by applying some condition in where clause. It is always used in some specific condition in the select query. The SELECT and FROM statement is mandatory but the WHERE clause is optional. Keep in mind, WHERE always comes immediately after the FROM clause. For instance, see below query;



Output:



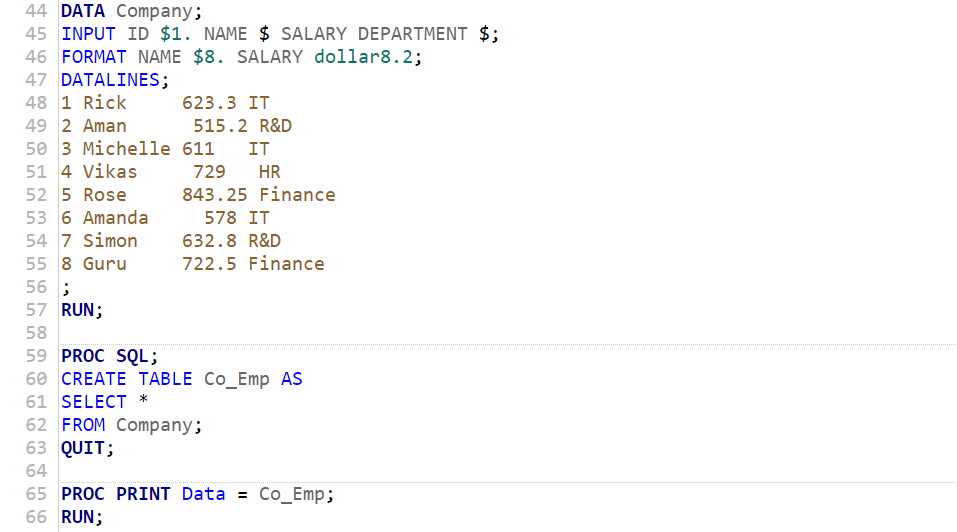
In the above program, the user wants to select all the columns of data, from the dataset (Co\_Format) and where the condition is Age is greater than 35 (Age > 35).

**SQL Operations**

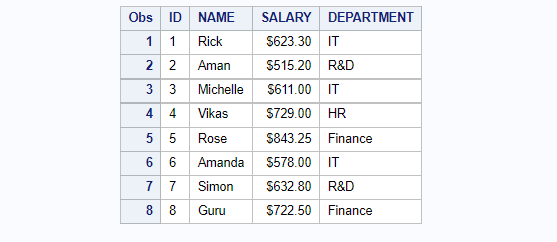
We'll demonstrate how to apply this SAS process in SQL for CRUD (Create, Read, Update, and Delete) actions. We are going to discuss Create, Read, Update, and Delete operations are below

* CREATE Operation

The create operation is useful for creating new data from the existing raw data. For instance, see below query and using the same data set of Company that we used in the section of SAS Sort Data sets:



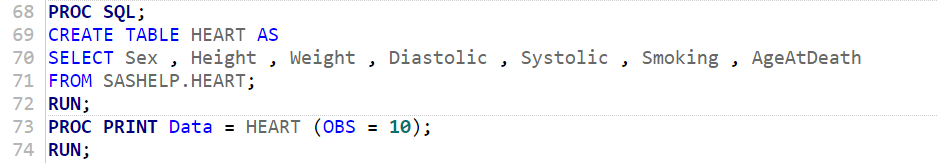
Output:



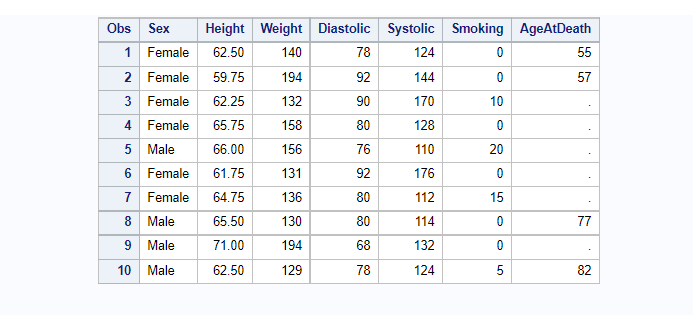
In the above example, we first create a data set called Company that contains the raw data. Thereafter, we create a table from the variables in this data set using a SQL query.

* READ Operation

In this operation, you can read the data from the table by using the SQL SELECT query. This operation is the same as SELECT and FROM, for instance, see below query:



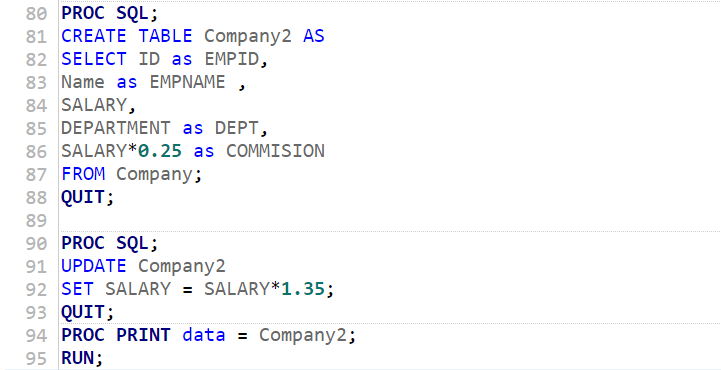
Output:



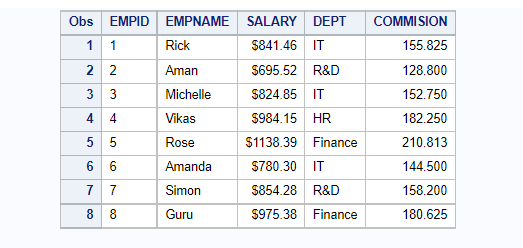
The SAS data set HEART, which is presented in the SASHELP library. The query retrieves some of the data set's columns. Note: In the Age column, the period (.) is treated as a missing value.

* UPDATE Operation

The update operation is useful to update the SAS table statement. For instance, see below query using the same data set of Company that we used above



Output:



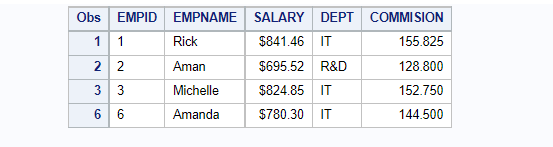
We use UPDATE action for the salary variable in the above program. Firstly, we first create a new table, and with the SELECT clause, we use the “AS” (i.e., ALIAS) option to change the column and then add the commission variable by multiplying 23% of the commission in salary. Thereafter, we update the salary column by a 12.5% increment in the new table.

* DELETE Operation

Using the SQL DELETE command, the delete action in SQL entails eliminating specific values from a database. We'll keep the data from the previous example and eliminate the entries from the database if the employees' salaries are larger than 900. For instance, see the below query using the same data set of Company that we used above.



Output:



From the above result, we can see that the employees whose salary is more than 850 are deleted from the database of Company2.

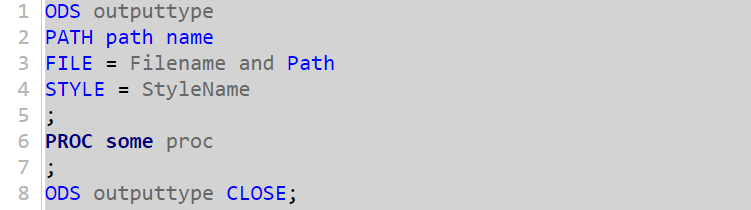
In this section, we learned that the SAS function PROC SQL is the most important and efficient function rather than the DATA step function, because of this you just write a pinpoint line of code. Hence, PROC SQL is a powerful tool in SAS.

**SAS – Output Delivery System (ODS)**

In this section, we'll glance at what SAS ODS is and how to use it to create several forms of output files, including Word output and SAS ODS PDF output to files, using a step-by-step approach with examples.

The SAS ODS allows you more power over how you produce, store, and reproduce SAS procedure and DATA step output, as well as a variety of formats. While using separate operations or the DATA step without ODS, formatting functionality is not supported. Further, SAS ODS was created to address the shortcomings of regular SAS reporting. It allows for the delivery of result/output in a number of formats while also making the styled output accessible. Moreover, HTML, Rich Text Format (RTF), Portable Document Format (PDF), and SAS data sets are among the file types that can be created with ODS.

The syntax of ODS is as following and you can perform the following with ODS:



The following parameters used are described below.

* Path- The path is the location of the file where you want to save the file format.
* Style- It is in-built feature in sas, whatever style format you want to add i.e., BarrettsBlue, Arial, Albany AMT, Helvetica, etc.
* Outputtype – It is the type of the format i.e., HTML, PDF, RTF, etc.

**Creating HTML Output**

The ODS HTML statement is used to generate HTML output.



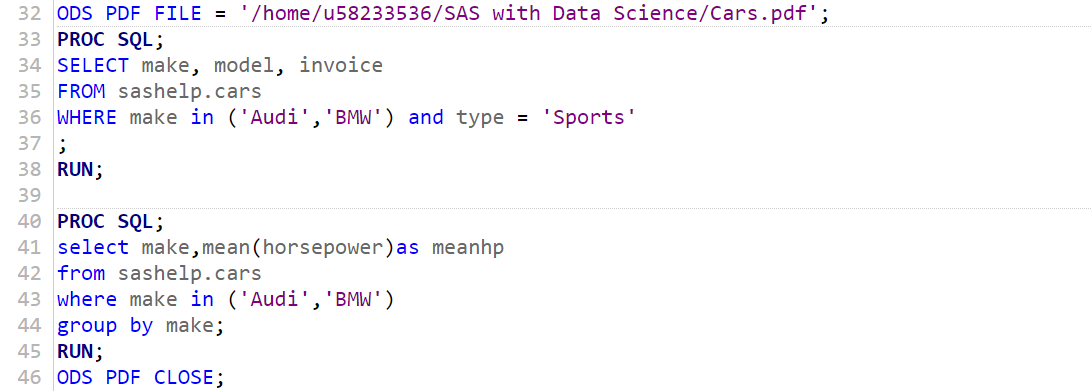
we generate an html file in the specified location. In the above example, we use a style from the library of styles. We can see the executable at the specified location and download it to save it in a location other than the SAS Studio. Kindly note that there are two proc SQL statements in this example, and both of their outputs are saved to a same file.

Output in the form of HTML:



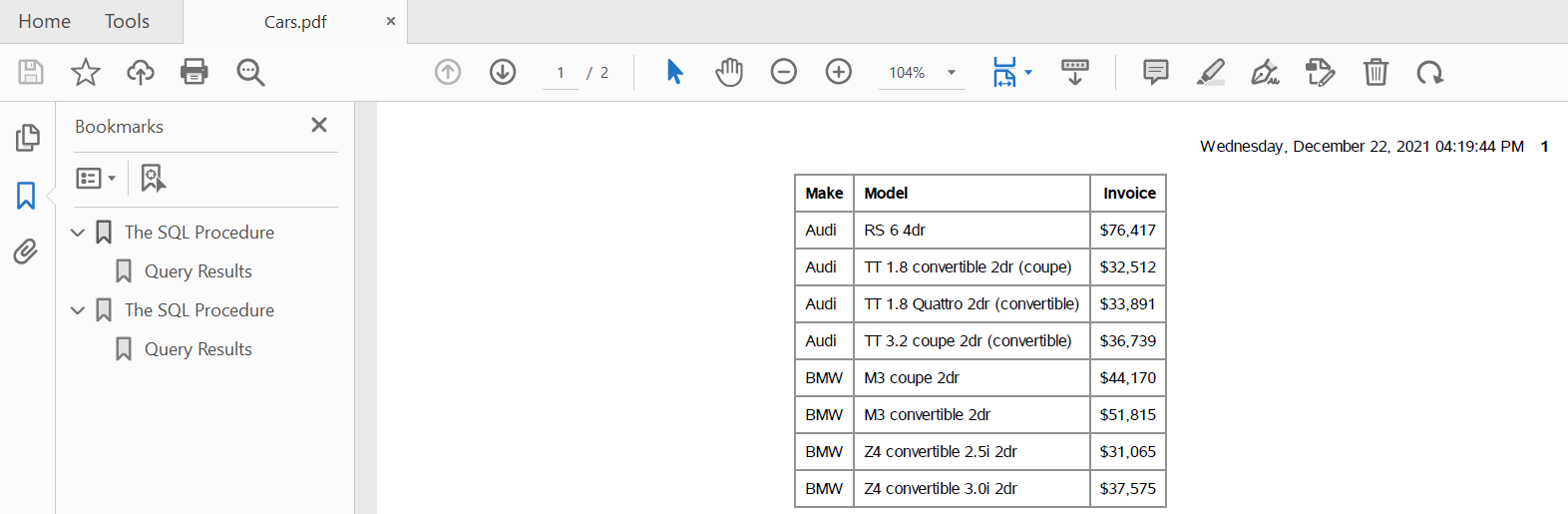
**Creating PDF Output**

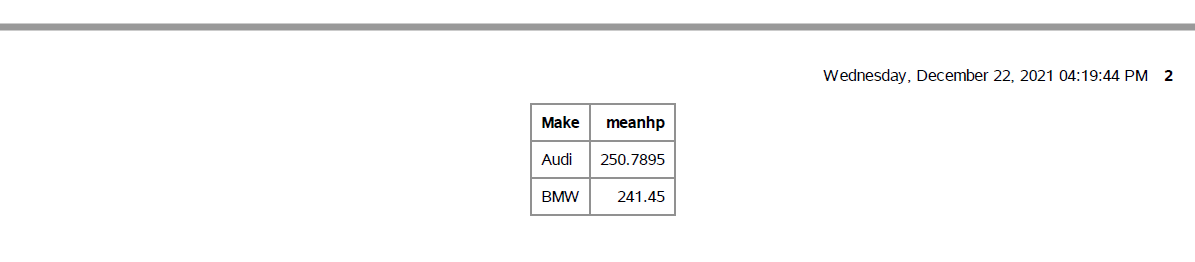
With the help of SAS ODS format, we create a pdf file in our desired location/path. Here, is the program:



We can see that the executable file at the specified location and download it to save it in a location other than the SAS Studio.

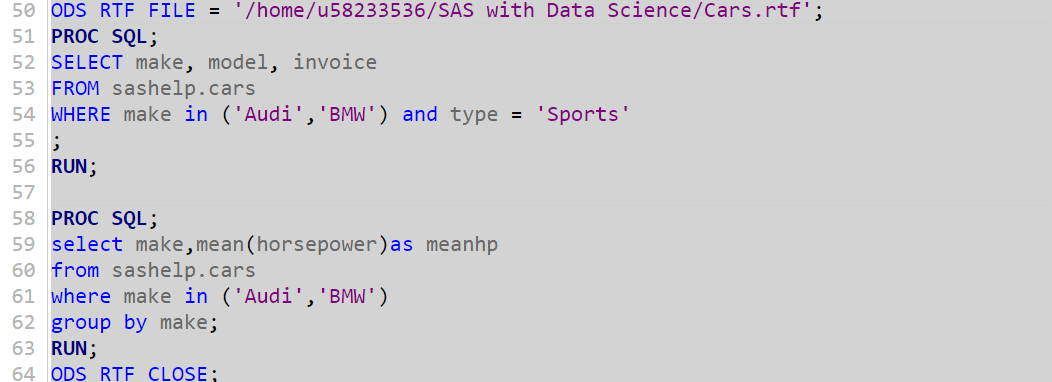
Output in the form of PDF:



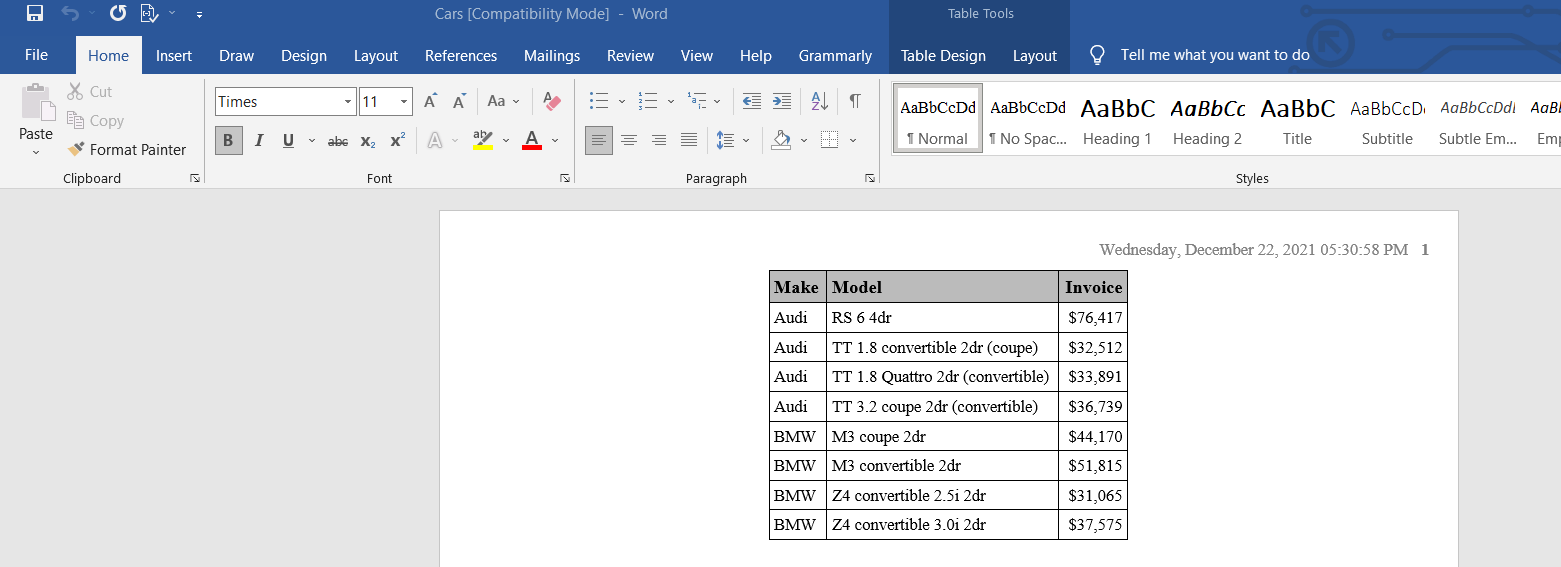


**Creating RTF (Word) Output**

With the help of SAS ODS format, we create a rich text format file in our desired location/path. Here, is the program:



Output in the form of RTF:





After the running the code above, we can see that the executable RTF file at the specified location and you can download it to save it in a location other than the SAS Studio.

Finally, in all the cases, both ODS statements require HTML, RTF and PDF as the output end point, and the filename has a “.html”, “.rtf” and “.pdf” extension, respectively.

**SAS – MACROS**

You can finally quit writing a similar line of SAS code over and over again. It will not be as difficult as you've been making it. The accessibility of SAS Macros might speed up your job as well as save your time. Therefore, Macros are an excellent approach to automate a procedure. In this section, we will give you SAS Macros knowledge of how to deal with the codes in SAS studio. Macros are easy to understand, although they can be puzzling sometimes. Accordingly, readers should pay careful attention to the program and its output. After completing this section, you will know how to develop macros and where they can be utilised.

SAS includes a unique programming feature called Macros that helps us to avoid writing repetitive code and reuse it when necessary. It also facilitates the creation of dynamic parameters inside the program which can take on changing values depending on how the program is performed. In the same way that macro values can be defined for code blocks that will be repeated several times, macros can be defined for lines of code that will be utilized countless times.

There are two sorts of macro variables:

* **Local** - The scope of a macro variable is local if it is defined within a macro code. It'd only be accessible for usage within this macro/script, and it would be withdrawn after the macro was ended.
* **Global**- The definition of a macro variable is global if it is defined outside of a macro code. It could be used anywhere within the SAS programme and is automatically discarded after the process has been completed.

Macros and macro variables are the two primary building blocks of macro code. They are referred to differently in a SAS application as:

* %Name refers to Macro
* &Name refers to Macro Variable

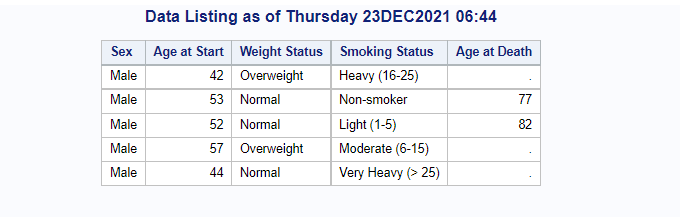
Now, begin with a basic code of macro, in the below example, suppose we want to display a title with the help of macros where the current system date and system time are printed in the title without any values being coded.

Here is the program where an in-built data set will be used and the data name is HEART which is available in the SASHELP library.

Syntax:



After running the above program, the output is:



We can see in the output where the title represents the current date and time. This is the global macro variable program because it is available

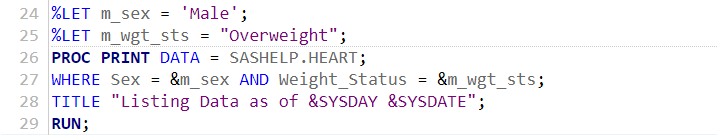
* **%LET Statement**

The use of %LET is to define the macro variable. The following syntax is used to introduce local variables.

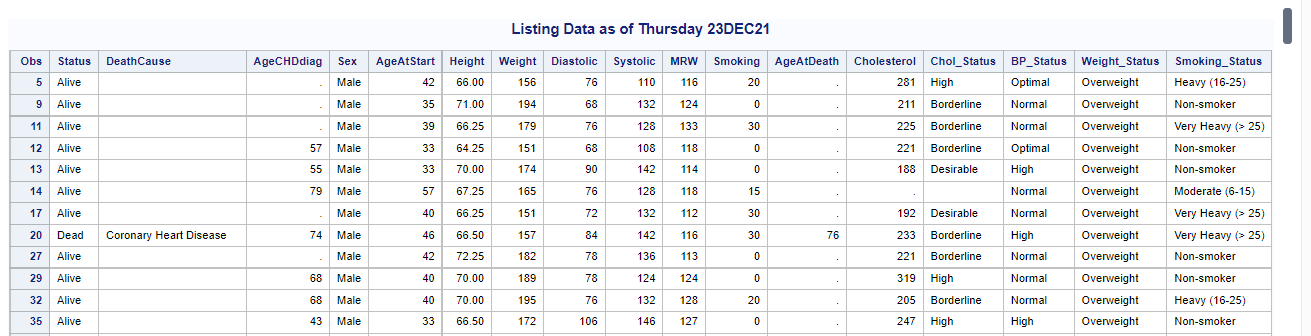


**Note**: The percent sign (%) is to notify the SAS macros. While the ampersand (&) sign is to notify SAS to read the macro variables and its value is always a character. This value may be a variable name, a numeric value, or any text that would replace in the application.

For example,

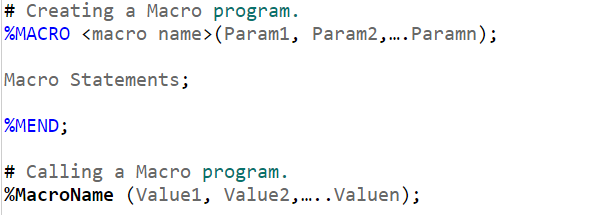


Output:

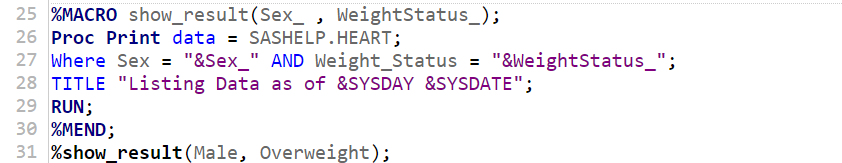


* **%Macro Statement**

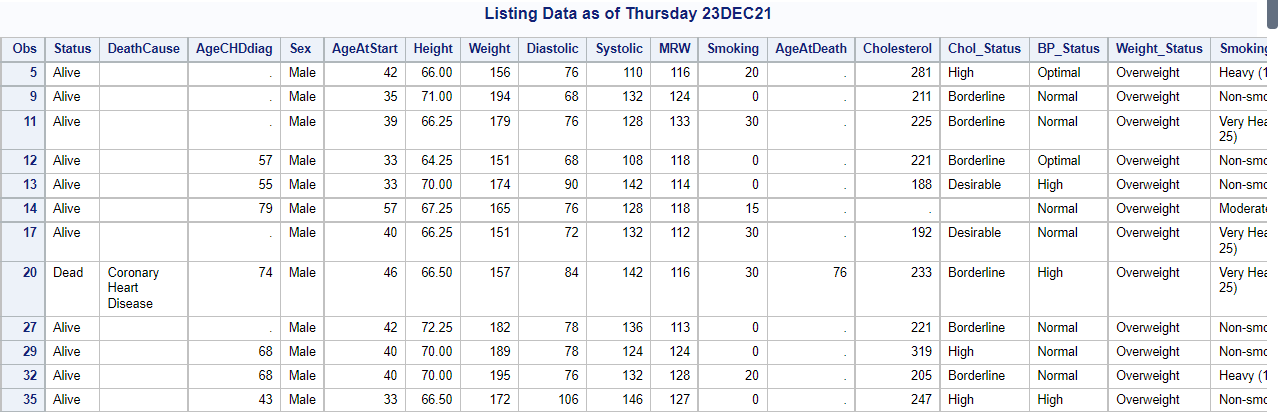
The use of %MACRO is to define the macro program. The following syntax is used to introduce local variables.



For example:

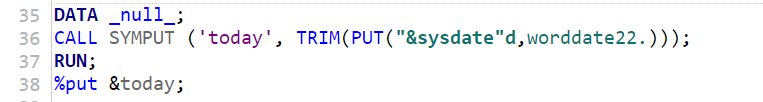


Output:



* **Macro %PUT**

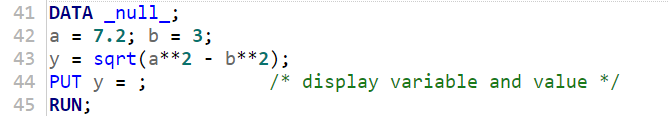
The % PUT macro statements are quite handy in allowing you to represent the values of variables and macro variables, respectively.



Output:



In this above program, we recorded the value of the variable 'today' in the programme log. For simplicity, we are going to take on more examples for %PUT.



Output:

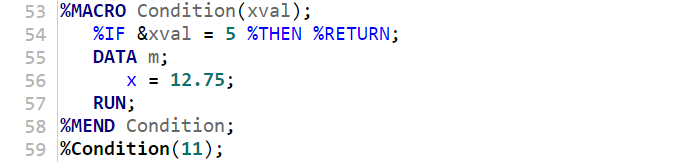


Here, the above statements displays that the variable name that is “y” and value of “y”.

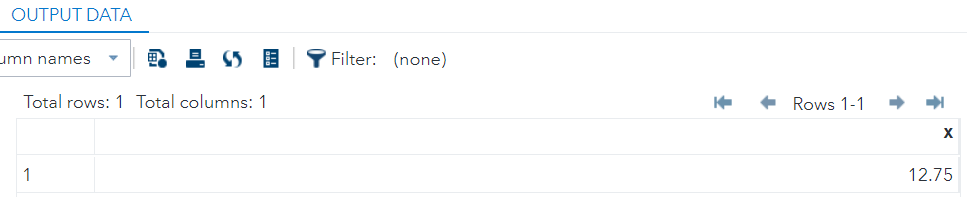
* **Macro %RETURN**

When this macro is run, it causes the currently running macro to terminate normally if a specified condition is met. For better understanding, we provide you with an example that will help you.

Here is the code:



Output;

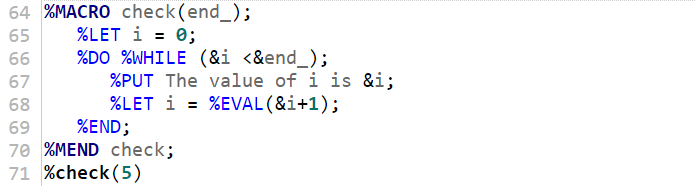


Above program displays that when the value of the variable “**xval**” reaches 5, otherwise it continues.

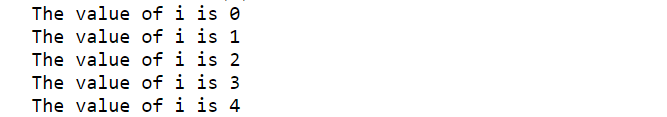
* **Macro %END**

This macro concept combines a % DO % WHILE loop that terminates, as indicated by the % END statement.

Here is the program of %end:



Output:



The above program shows that the check macro takes a user input and uses that value to perform the DO loop. The % end statement marks the conclusion of the DO loop, whereas the % mend statement marks the end of the macro.

**SAS – DATA REPRESENTATION**

There are lots of ways for data representation such as Histogram, Bar Charts, Pie Charts, Scatterplots, Boxplots, etc. Additionally, the term "data visualization" refers to a method of examining quantitative data. A diagram depicts the relationship between facts, ideas, data, and concepts. It is simple to comprehend and one of the most fundamental learning techniques. It is always dependent on the type of data in a specific discipline.

Simply said, it displays that the statistical data in the form of graphical style. SAS, being the premier analytics platform, provides several data visualization tools as a way to view and analyze the trends, patterns, and outliers in data by employing visual elements such as charts, graphs, and maps.

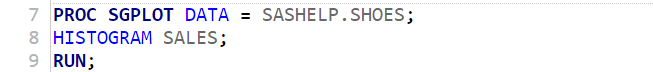
In this section, we will be demonstrating basic and different styles of SAS Graphical Data Representations.

**SAS – Histograms**

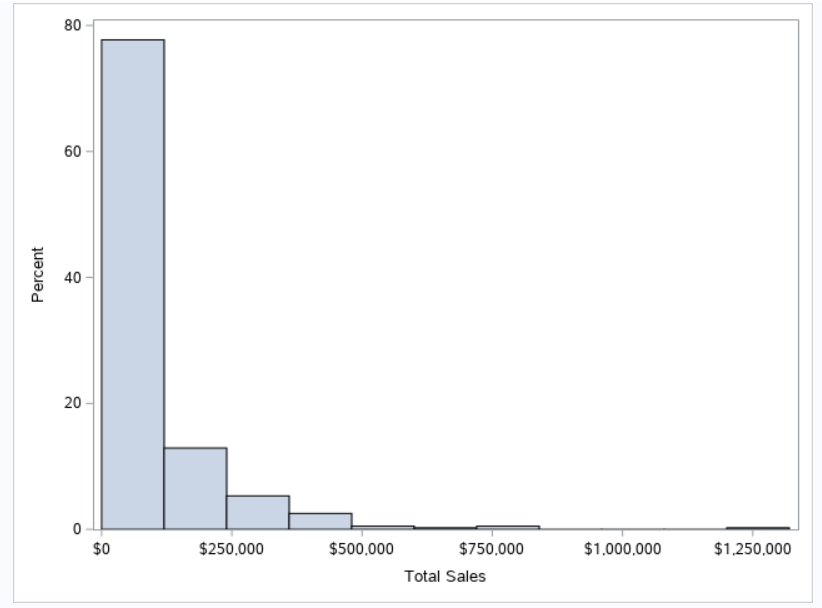
A histogram is a graphical picture that divides a set of data points into ranges defined by the user. It shows that the frequency distribution of a set of data. It provides an "at a glimpse" representation of a distribution pattern, with individual categories depicted. Histograms are mostly used for historical data visualization. In SAS, the histograms can be created utilizing PROC UNIVARIATE, PROC CHART, PROC GCHART, or PROC SGPLOT where SG implies for the statistical graph.

Let’s begin with the examples where you will learn how to construct a histogram by using SGPLOT in SAS. The histogram example below displays the frequency distribution of the SALES variable from the SHOES data set which is available in the SASHELP library.

Here is the code:



Output:

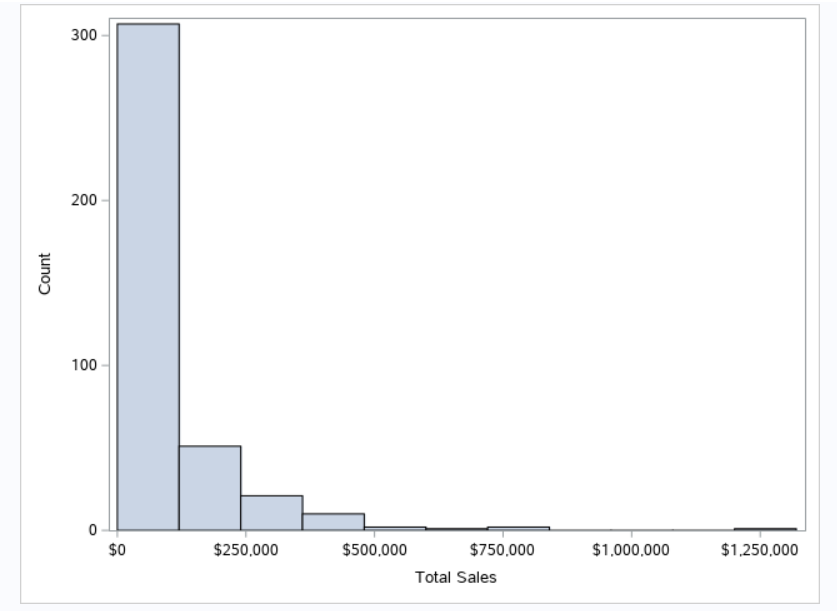


**Y-axis from percentage to count**

In the above chart, the y-axis shows that percentage and we are interested in frequency. So, we will need to change the scale of the **Y-axis** from percentage to frequency. Now, we add a statement in the histogram part see below.



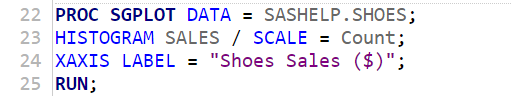
Output:



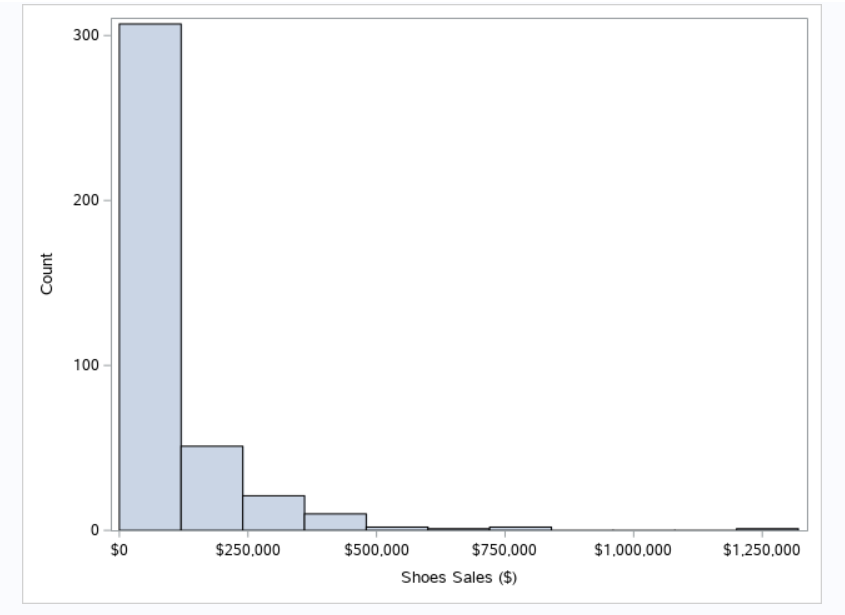
Note: There are many histogram statements of SCALE = options such as percent, count, density, proportion.

**X-axis Label**

As we know that, labels increase the readability of the graph and we can see that in the graph, there is no label on the axis. If you want to add the label of the x-axis in the graph, then you have to use the **XAXIS statement** with the label option. Let's have a look at the below program.



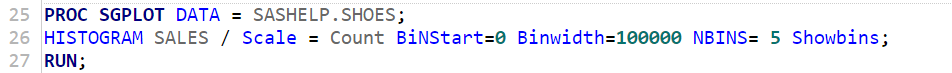
Output:



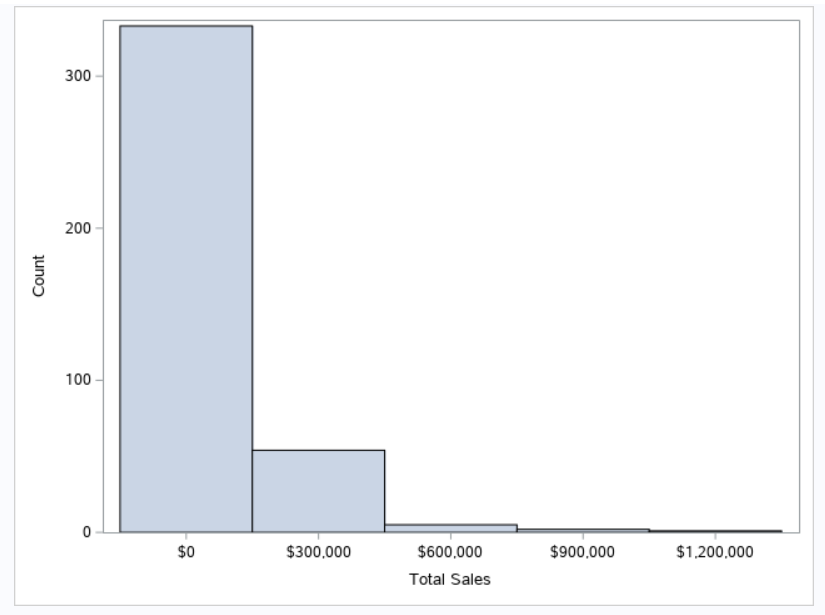
In this graph, we can see that the x-axis is labelled with “Shoes Sales ($). In this way, you can add labels in graph.

**Bin Size and Number of Bins**

Additionally, bin size is more important in histogram. In the above programs, the bin size is not set, however, SGPLOT procedure estimates by default the best bin size and number of bins. This estimate is best but not always. So, if you want to change the bins size and number of bins, then follow the below program’s optional argument which helps you to change the size.



Output:

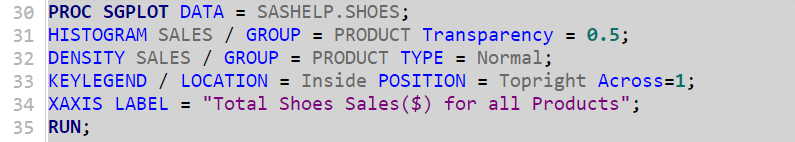


For this graph, we specify all options i.e., binstart, binwidth, and showbins in the above code.

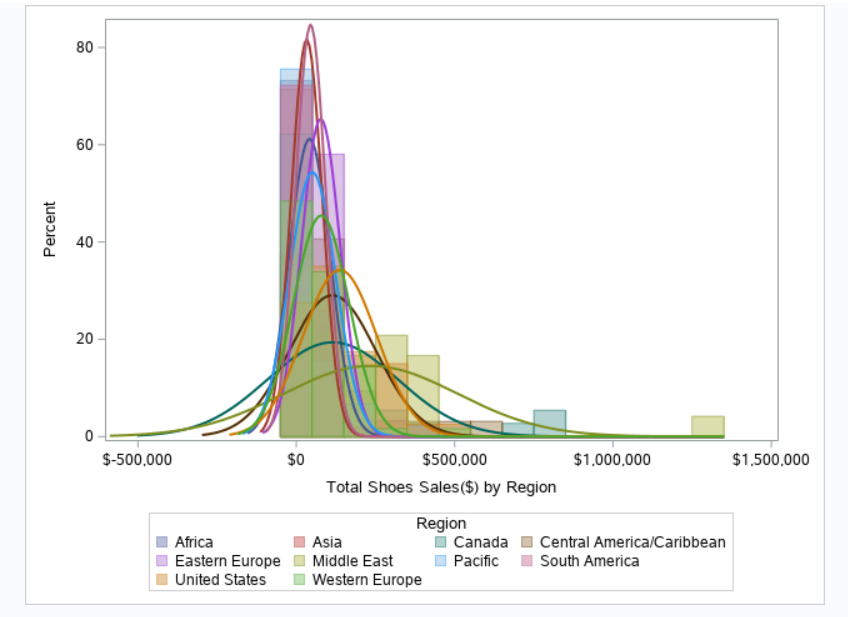
**Create Histogram by Group with Curve Fitting**

If your data can be divided into categories, a histogram with several groups can be useful. Additional choices can be used to fit some distribution curves into the histogram. To construct a single plot containing histograms by group, use the GROUP=option. The HISTOGRAM statement's GROUP=option is an optional argument. On the other hand, the density statement displays the fitting curve.

Here is the program:



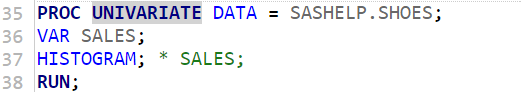
Output:



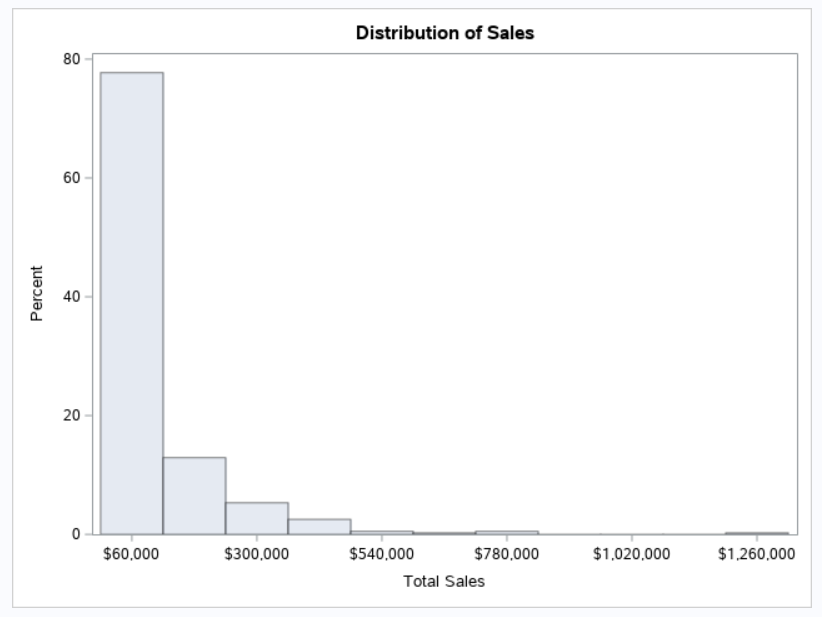
. The graph shows that each group is represented by a different shade with fitting curves.

Now, we will take same data (i.e., shoes data in-built SASHELP library) where you will learn how to construct a histogram by using PROC UNIVARIATE in SAS.

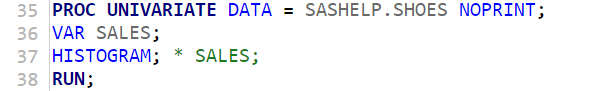
Here is the code of UNIVARIATE:



Output:

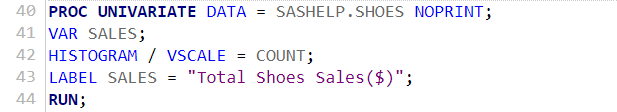


To begin, use the DATA=option to specify the data set which is used in the analysis (such as SASHELP.SHOES). The “VAR” statement is then used to define the variable you would like to display. Next, HISTOGRAM command is used to present the histogram graph. Thus, the above output which is produced by UNIVARIATE command. Apart from that, when we run the code above, we'll see that the PROC UNIVARIATE process additionally creates a report with numerous statistics. In the event that you're just inspired by the histogram, you can add the NOPRINT option later the DATA=option see below.

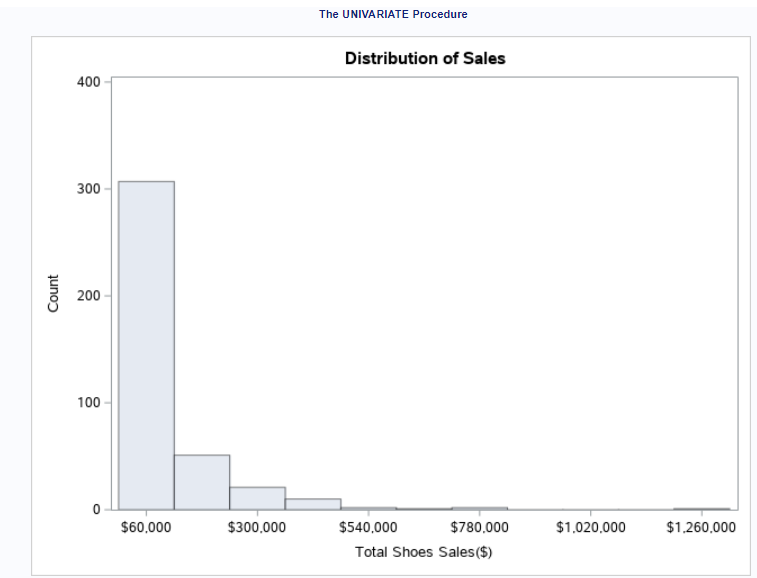


**Y-axis from Percent to Count and Label X-axis**

The **y-axis** shows the percent and we want to change it count, then we have to set the option to COUNT. Additionally, there is no label in the x-axis. So, we have to use label statement for adding x-axis which begin with the LABEL keyword. See below program:

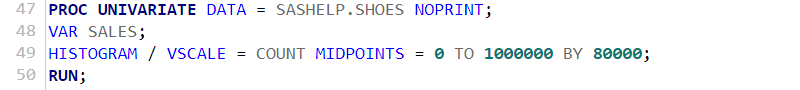


Output:

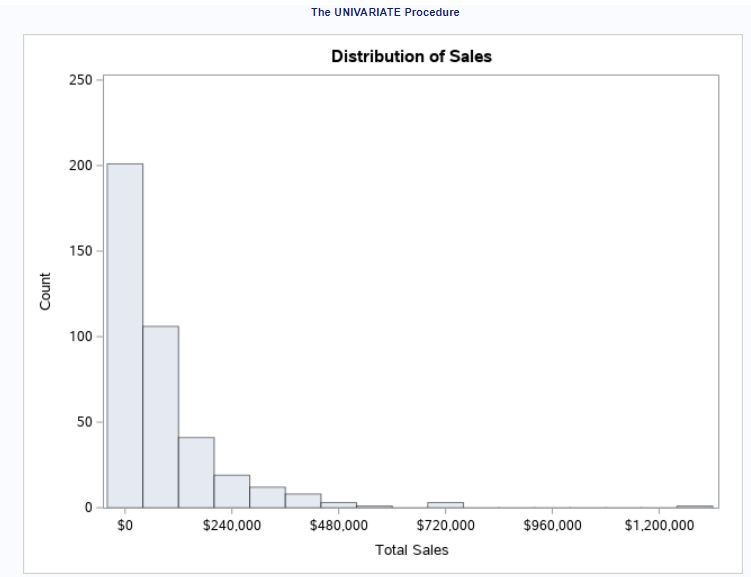


**Bin Size and Number of Bins**

By choosing a range of midpoints, the number of bins and bin size can be determined. To begin, you must define the first and last bin's midpoints. The step size (or bins width) of each bin is then defined.

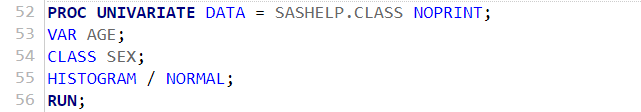


Output:

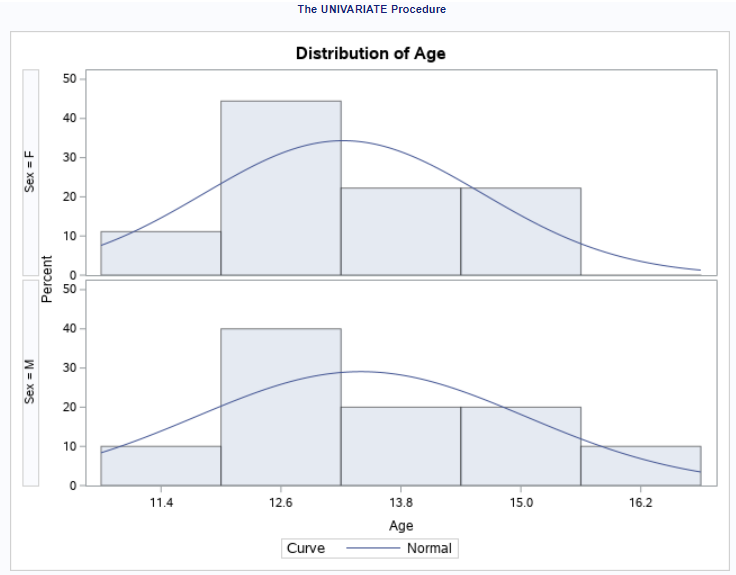


**Create Histogram by Group with Normal Curve Fitting**

For creating a histogram by group, we require to use the CLASS statement. We use other data set i.e.; CLASS available in SASHELP library. See below the code.



Output:

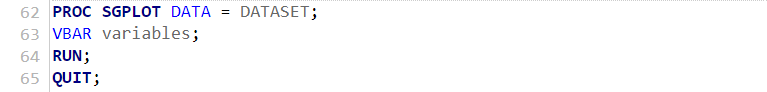


Thus, the PROC UNIVARIATE procedure create more than one histogram within one image, however, in PROC SGPLOT procedure it is not possible. In the graph above, the CLASS statement is used to build a histogram for each class of sex (Male and Female).

**SAS – Bar Charts**

The bar charts represent a graphical picture of data, and it is better to use when information is used to compare two groups/categories of data. It displays the data in rectangular bars forms. To make bar charts, SAS needs the PROC SGPLOT method. In a bar chart, we can create both simple and stacked bars.

Basic Syntax:



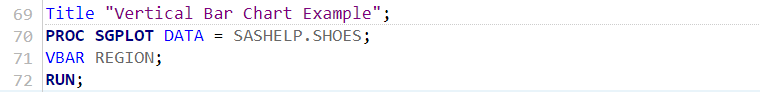
The following is a list of the parameters that were utilised.

DATASET – Name of the dataset which is used.

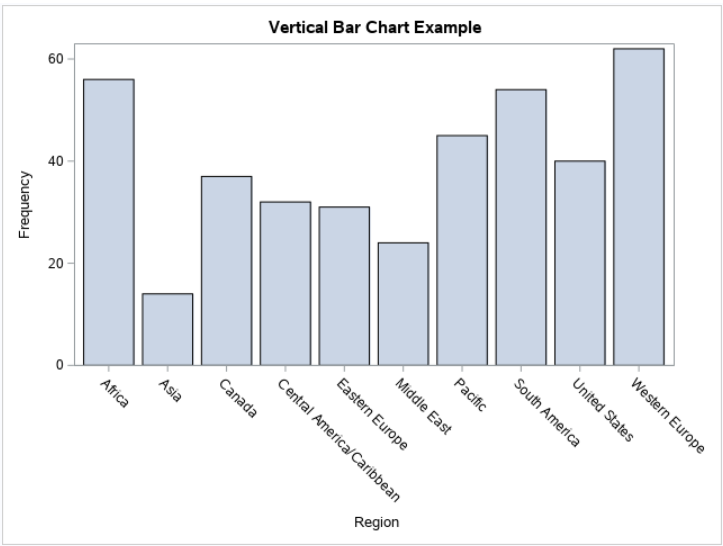
Variables – Name of the variable which you want to plot the graph.

**Simple Bar Chart (Vertical and Horizontal)**

PROC SGPLOT is used to construct a category variable type vertical or horizontal bar chart, as shown below the program:

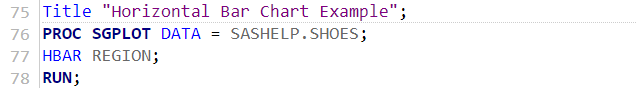


Output:

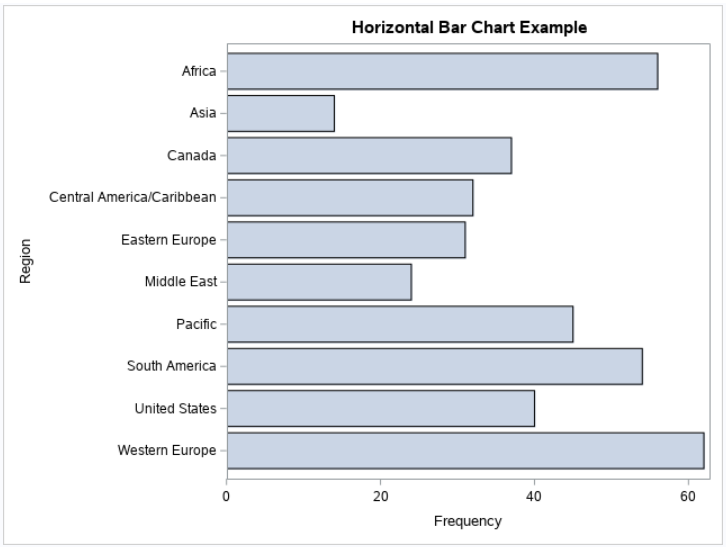


As we see that the height of the bar charts shows the frequency counts (see y-axis). This is the vertical bars, but if you want to construct the horizontal bars then you have to insert HBAR statement instead of VBAR statement, as shown in the below program:

Note: VBAR refers vertical bars and HBAR refers horizontal bar.

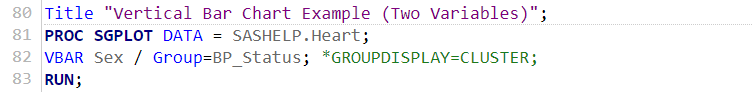


Output:

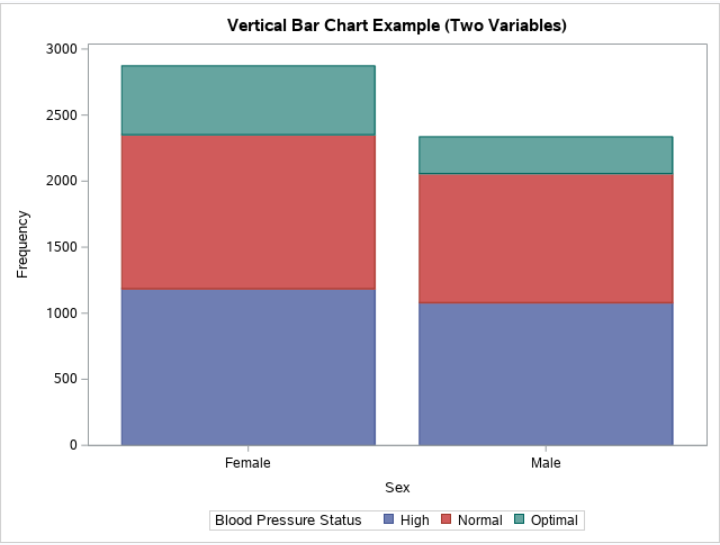


**Stacked Bar chart**

A stacked bar chart is better to use when you have to compare more than one categorical variable because it split down pieces of a larger picture where segments representing various categories. Suppose you want to look at frequencies of blood pressure status for females and males. See below the program and we use the HEART data set from SASHELP library.

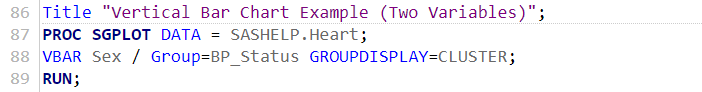


Output:

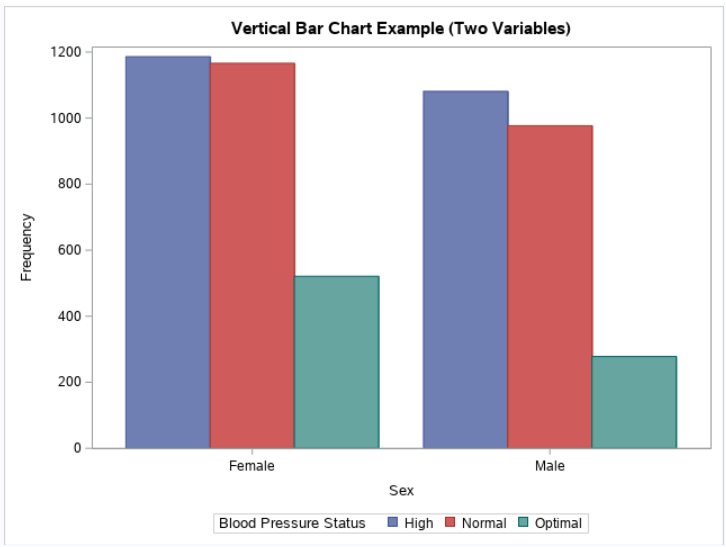


**Clustered Bar Chart/** **Column Chart**

The clustered bar chart is used to display that how a variable's values are distributed across cultures. In this chart, if we would like to see the side-by-side groups rather than stacked groups, then we just need to introduce the option GROUPDISPLAY=CLUSTER to the VBAR line on the right side. For Example:



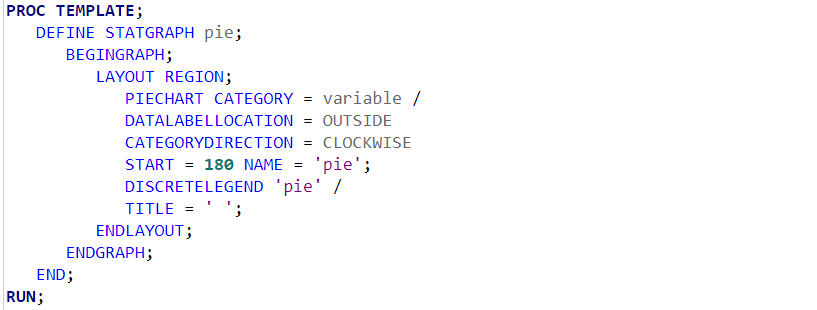
Output:



**SAS – Pie Charts**

A pie chart is a visual depiction of values as coloured segments of a circle. The segments are identified, and the graphic also shows the numbers that correlate to every segment. SAS Pie Chart provides basic, grouped, or stacked charts that display data as segments of a pie to show the variables that contribute of the components to all this. Every piece shows the data class. The proportion of the dataset to the total chart statistics is represented by the magnitude of a segment. It is constructed in SAS using PROC TEMPLATE, which accepts inputs for controlling percentage, labelling, colour, and headline, among other things.

Basic syntax of pie-chart is as follows:



The following parameter that was utilised.

Variables – Name of the variable which you want to plot pie-chart.

**Simple Pie Chart**

First of all, we will start with the basic pie chart implies that the variable is single to create the pie chart where value of the slices refers to the ratio/proportion in relation to the total value. Let's start with a sample of a built-in dataset for SHOES. The arguments used in the above syntax will subsequently be examined in depth.

In the below example each segment signifies the portion of the category of shoes product from the total number of shoes.



After execution of the above code, the output is shown below:

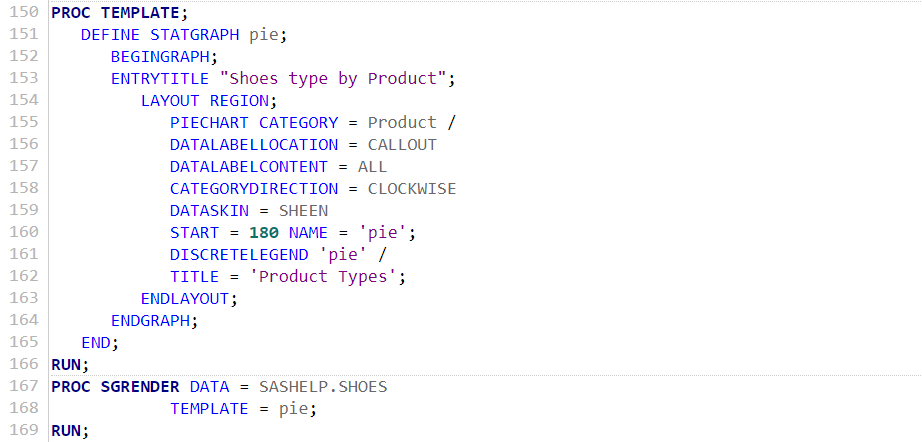


**SAS Pie Chart with Data Labels**

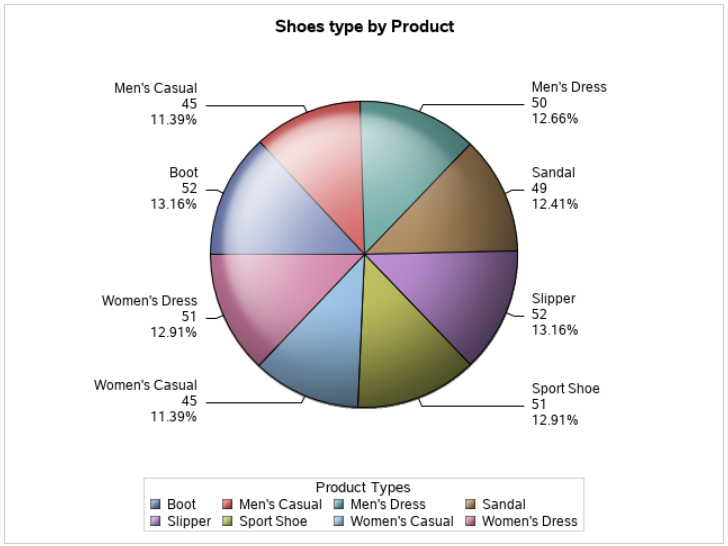
For each portion of the pie chart, we can show the proportional value as well as the percentage of the value. We likewise change the area of the name to be inside the diagram. The style of appearance of the graph is changed by utilizing the DATASKIN option. It utilizes one of the inbuilt styles, accessible in the SAS.

*Note*: There are different options for DATALABELLOCATION = “INSIDE”, “OUTSIDE”, “AUTO”, “CALLOUT”. It depends on the user which type of option his/her want in the pie circumference. Additionally, the graphical look of the completed pie pieces is improved by using DATASKIN=options (that is None, Sheen, etc.).

In the below example, we improved the pie chart each segment by using the DATALABELLOCATION and DATASKIN.



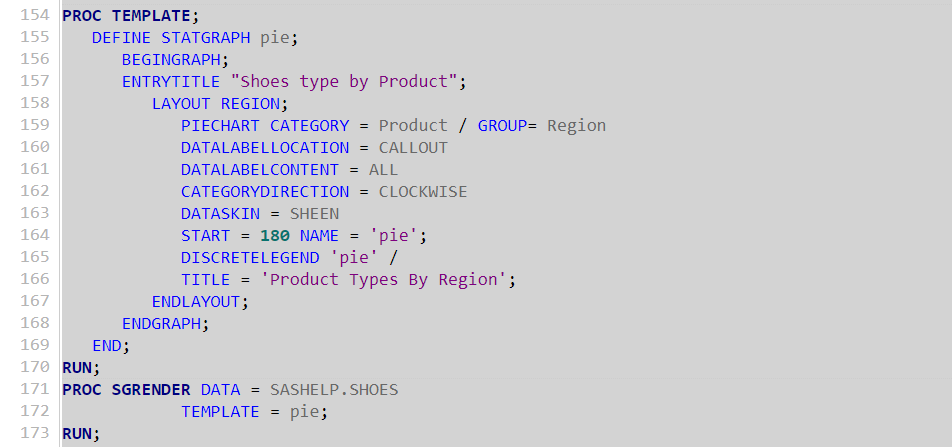
After execution of the above code, the output is shown below:



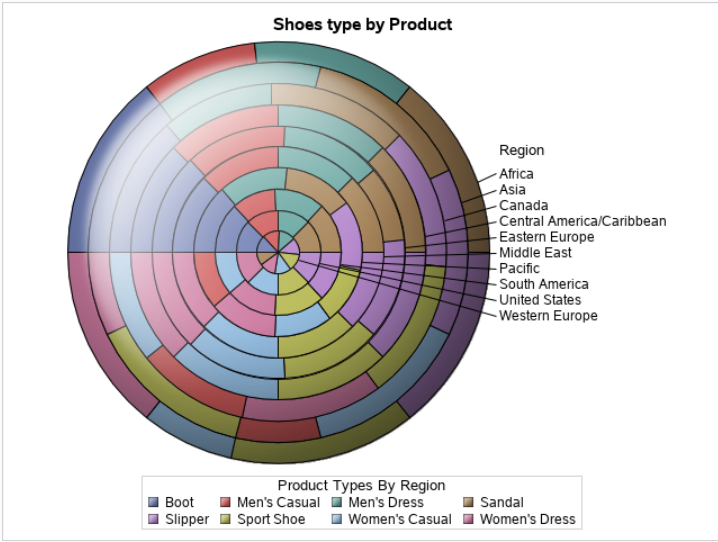
**Grouped Pie Chart**

The result of the graphed variable is clustered with regard to some other variable from the same data set in this pie chart. Thus, every group is turned into a circle, and the graph will have as many geometrical shapes as there are groups.

Here is the program where we group the graph with respect to the variable are as follows:



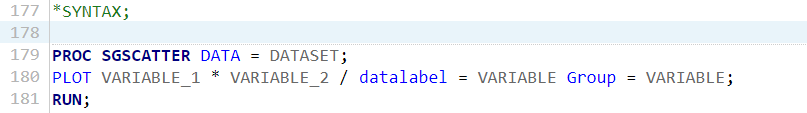
After execution of the above code, the output is shown below:



**SAS – Scatterplots**

Scatter plot observes and represents the relationship between two quantitative variables. The dots or separate data points of the plot reports the direction or pattern of the data. It is a type (i.e., plot or graph) of SAS programming language which displays the values of two different variables using cartesian coordinates.

Syntax of scatterplot is as:

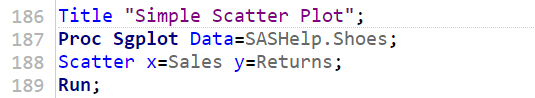
****

The description of the parameters that were utilised.

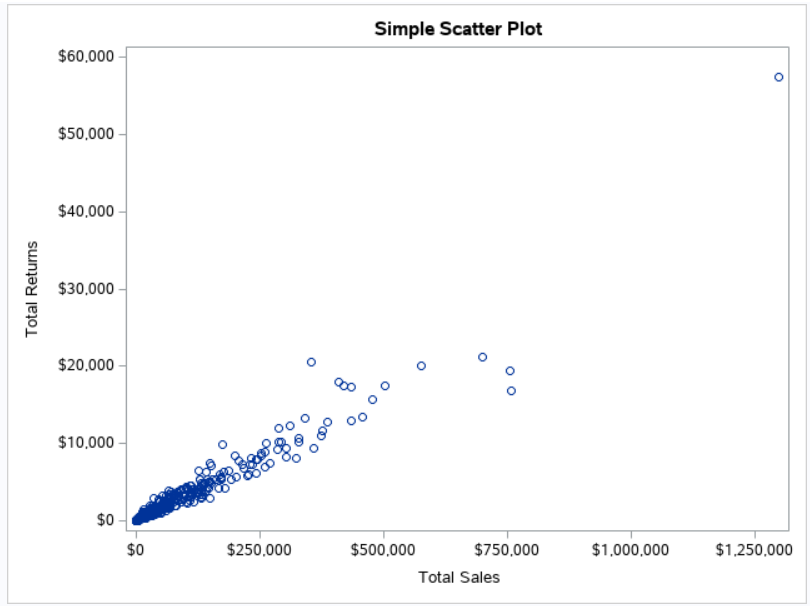
Dataset – Name of the dataset.

Variables – Name of the variable which you want to plot pie-chart.

We use the SHOES data set from the SASHELP library. You can use SCATTER statement for plotting the graph along with the options i.e., x equals and y equals where you can specify the variable names on y and x axis. Here is the program where we plot the returns on y-axis and sales on x-axis:

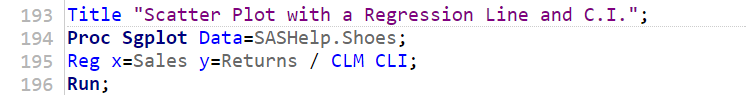


After execution of the above code, the output is shown below:

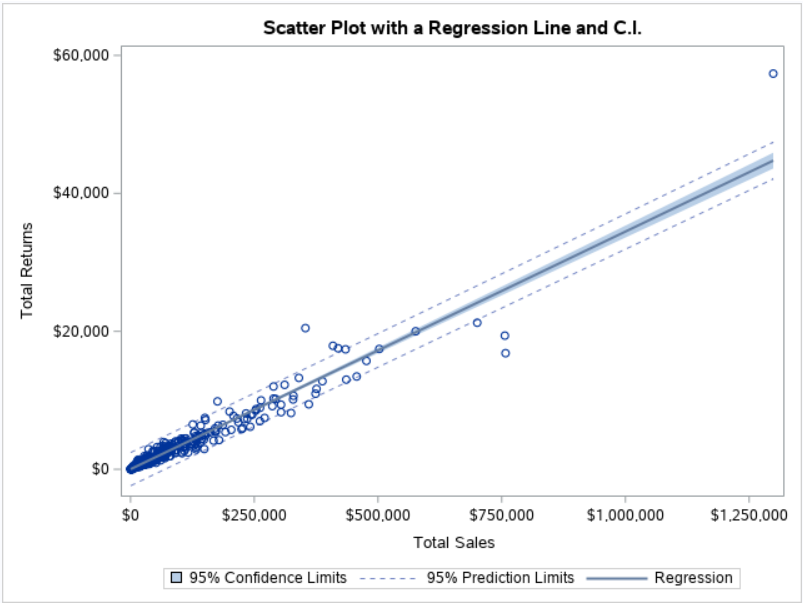
****

An examination of this plot might lead you to believe that sales and returns have a strong relationship. Statistical techniques like correlation and regression could be used to validate this.

Now, we are going to **adding a regression line and confidence limits** to the plot.



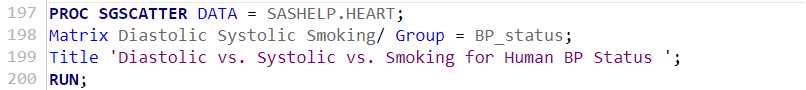
Output:



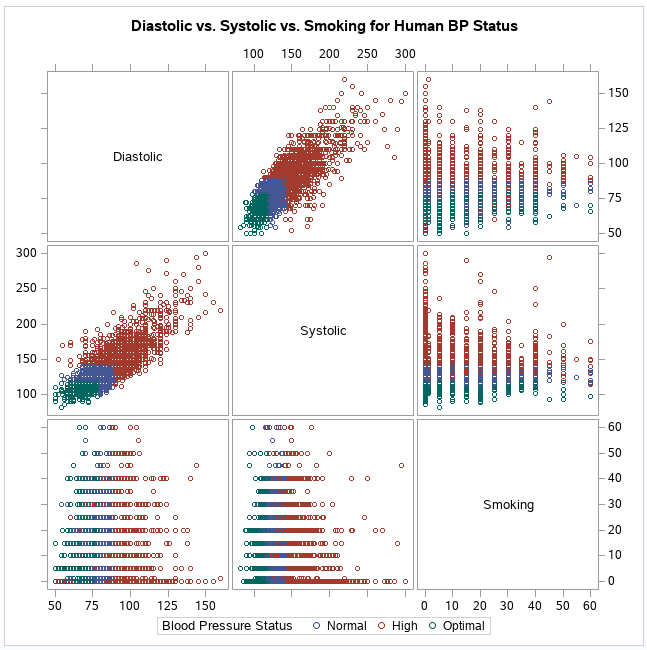
In the above output, we can see that the separate data points along with line i.e., regression line and two confidence limits i.e., upper limit and lower limit.

**Scatter Matrix**

In this plot, we can include more than two variables by combining the variables into pairs. We pick three variables and create a scatter plot matrix in the example below. We receive three pairs of matrices as a result.

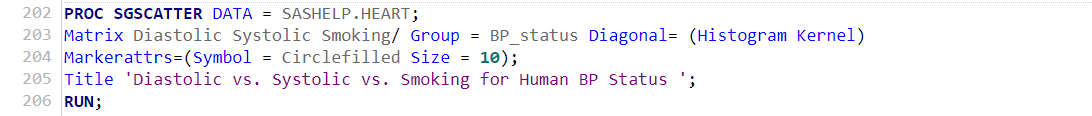


Output:

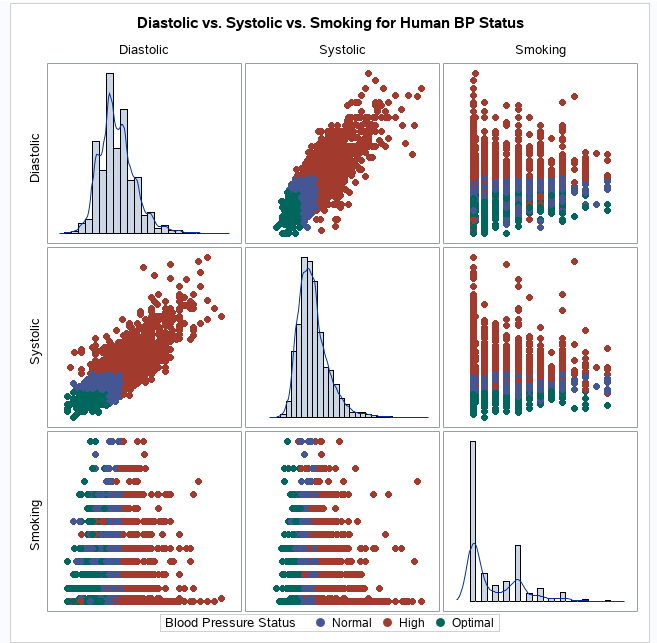


The above output is made up of numerous pairwise scatter plots that are displayed as a matrix. The matrix indicates whether there is a positive or negative connection between various variables. They assist us in determining whether or not there is a correlation among numerous factors.

Additionally, you can add diagonal statement which tell us that the data is bell shaped or non-bell shaped. See below the program:

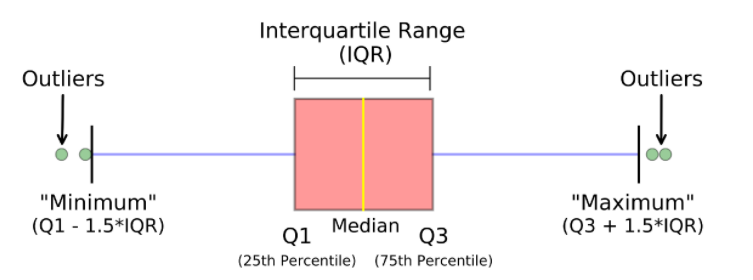


Output:



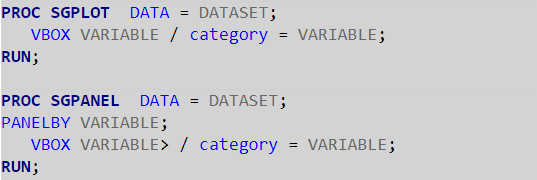
**SAS – Boxplot**

The boxplot (commonly termed as a box and whisker plot) represents the 5 number data summary which includes minimum, first (or lower) quartile (Q1), median (Q2), third (or upper) quartile (Q3), and maximum. The “whisker” (include minimum or maximum) is a parallel single (blue) line and the box part refer to the IQR- interquartile range (see below). All the portion of the box plot equals to 25%. If the data point lies outside of the box plot, then it will be treated as an outlier (abnormal value as compared to the other values).

****

It is a better way to show the visual representation of data distribution through quartiles. This type of technique could be referred as EDA- exploratory data analysis. In SAS, PROC SGPLOT is used to make a simple Boxplot, and PROC SGPANEL is used to build a panelled Boxplot.

Basic SAS Syntax of Boxplot:



The description of parameters that were utilised.

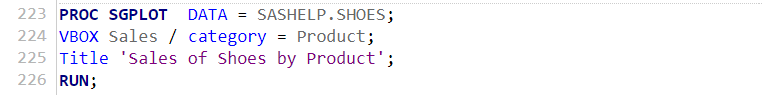
Dataset – Name of the data set.

Variables – Name of the variable which you want to plot.

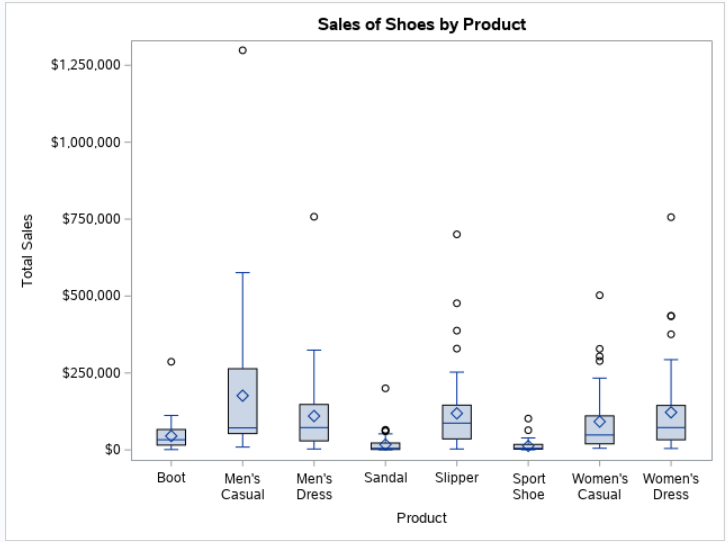
**Simple Box Plot**

* **Vertical Box**

For the simple box plot, we select one categorical variable and another variable is from the data set. Here is the program:

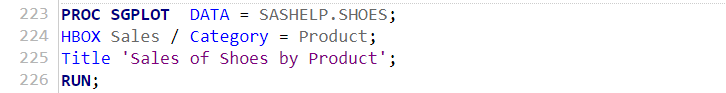
****

Output:

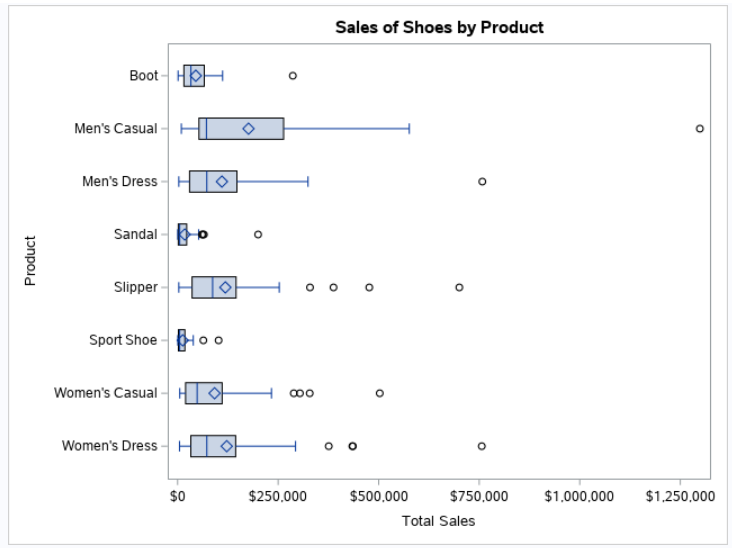


* **Horizontal Box**

The above box plot is vertical because we use the “VBOX” command, if we use “HBOX” then it will produce horizontal box plot, same concept like horizontal and vertical bar graph. See below the code:

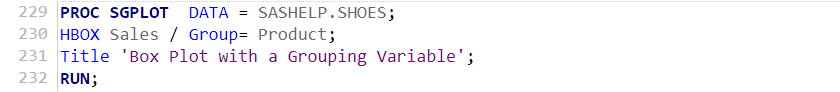


Output:

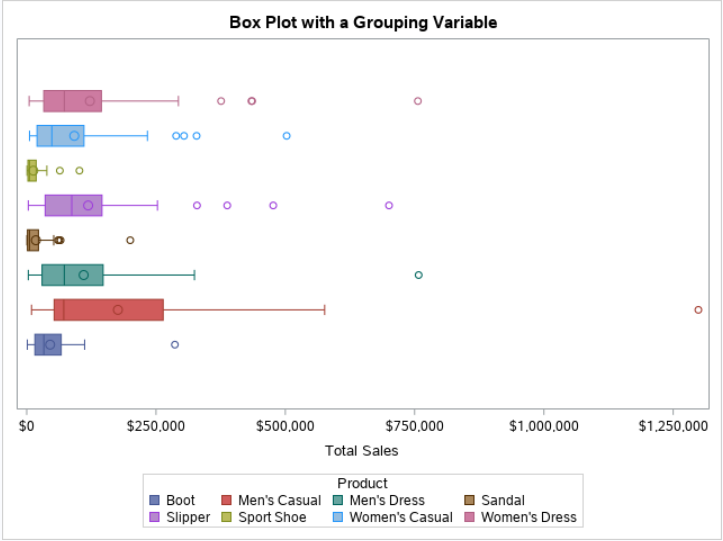


**Box Plot with a Grouping Variable**

Add a GROUP=option to the VBOX/HBOX statement to construct a box plot for each value of a clustering variable. The following example creates a Shoes Sales box plot for each type of product:



Output:



This type of plot provides a great visual representation of the distribution of one variable for every class of some other variable.

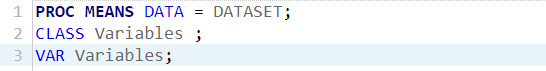
**SAS – Statistical Procedure**

A statistical procedure is a convenient way of interpreting or analyzing the statistical data, a technique for computing a statistic implies that a technique of doing anything, especially one that is systematic; indicates a logical and organised structure (usually in steps). In this section, we are going to introduce and you will learn some SAS statistical tools and procedures.

**Arithmetic Mean (A.M.)**

The arithmetic mean or average is the most basic and extensively used method of calculating a mean or average. The way of calculating the mean is as simple as adding up a bunch of numbers and then divide by the total number of observations in the variable. **PROC MEANS** is used in SAS to calculate the arithmetic mean. You can calculate a single, some or all variables of the mean.

Here is the syntax of A.M.:



The description of parameters that were utilised.

Dataset – Name of the data set.

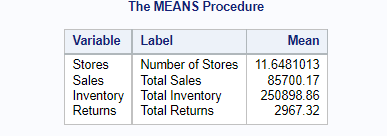
Variables – Name of the variable which you want to find out the mean.

* **Mean of all variables**

In the below program, we evaluate the mean of the dataset where all the variables belong to quantitate or numeric type and the data set named SHOES.

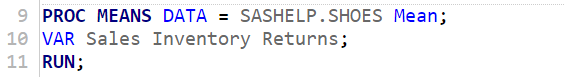


Output:

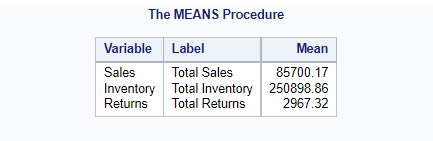


* **Mean of selected variables**

Above output displays mean of all the variables of a Shoes data set. However, you want to display “sum” of all the variables then you have to add “SUM” after the MEAN statement. Thus, these is the case of mean of all the variables. But, in case, we want some selected variables mean then follow the below program:



Output:

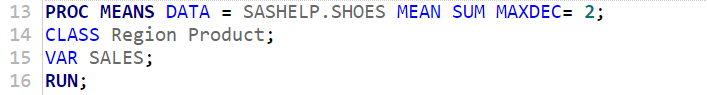


This can be done with the help of “VAR” statement. You just need to add selected variables after the VAR statement.

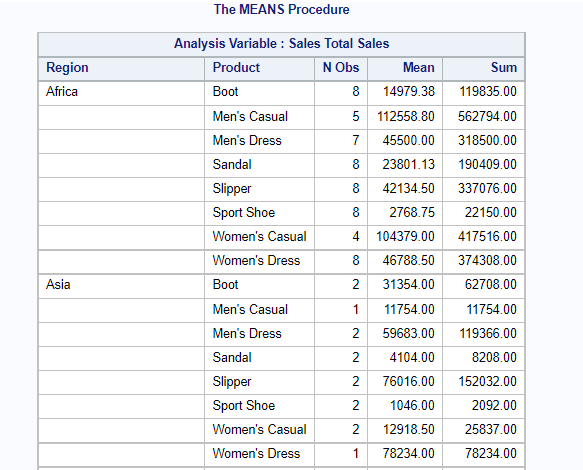
* **Mean by class**

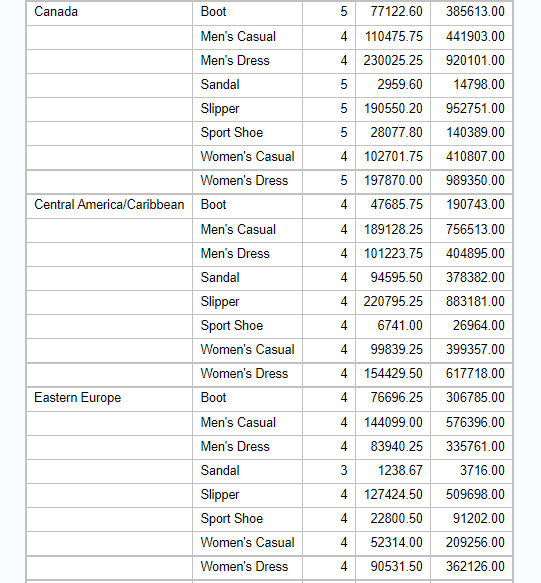
Additionally, the mean of quantitative variables by grouping them with the categorical variable. In this case, the CLASS or BY statement will better to use because it expresses the subgroups of the data. Notice one thing, the class statement variable refers to categorical variable.

Here is the program:



Output:





**Note:** All the statistics is informed that there are two decimal points because of MAXDEC=2. It controls the decimal point.

**Standard Deviation (S.D.)**

The S.D. measure the dispersion of the data in the data set. Mathematically, it is the indicator of how mean value is closer or far to the data set. When the S.D. is lower or close to zero which implies that the data observations are close to the data set’s mean. On the other side, when the S.D. is high or far from zero suggest that the data observations are dispersed across a greater range of values. In SAS, the statement PROC MEAN and PROC SURVEYMEANS are being used to calculate S.D. statistics. Also, we have to use STD option in the PROC step. Thus, S.D. values for each numeric variable are displayed. Basis Syntax of S.D. is:

**Note**: In PROC MEANS statement, lots of statistics options such as MEAN, STD, VAR, N, NMISS, MIN, MAX, MEDIAN, so on.

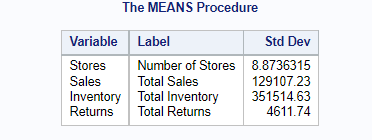
* **STD Option**



Example for STD option by calculating standard deviation:

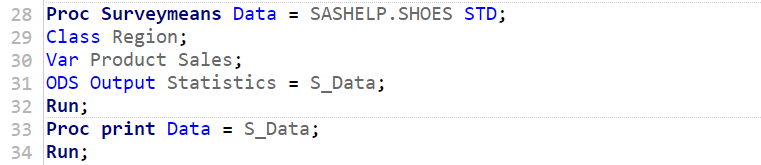


Output:

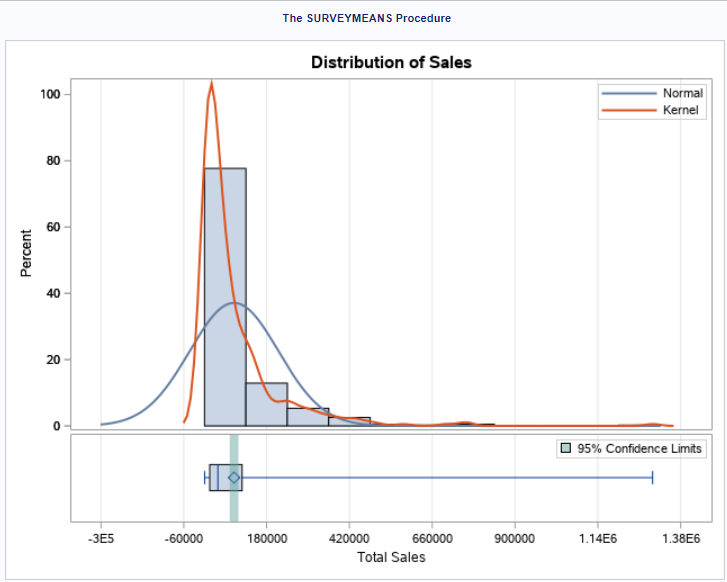


* **SURVEYMEANS statement**

In this statement, you could estimate S.D., provides estimates in variance and some advanced capabilities such as measuring SD for categorical data.



Output:

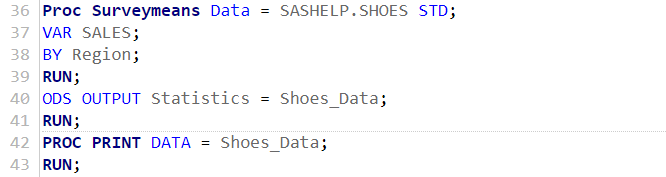




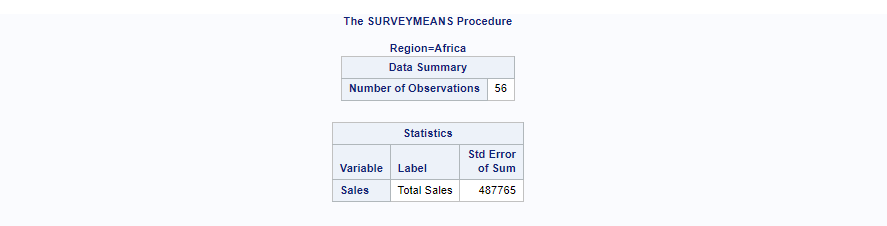
In the above graph, it depicts that the data distribution for the variable 'Sales' with a confidence interval of 95 percent, indicating that the data spread is likely to be within this range.

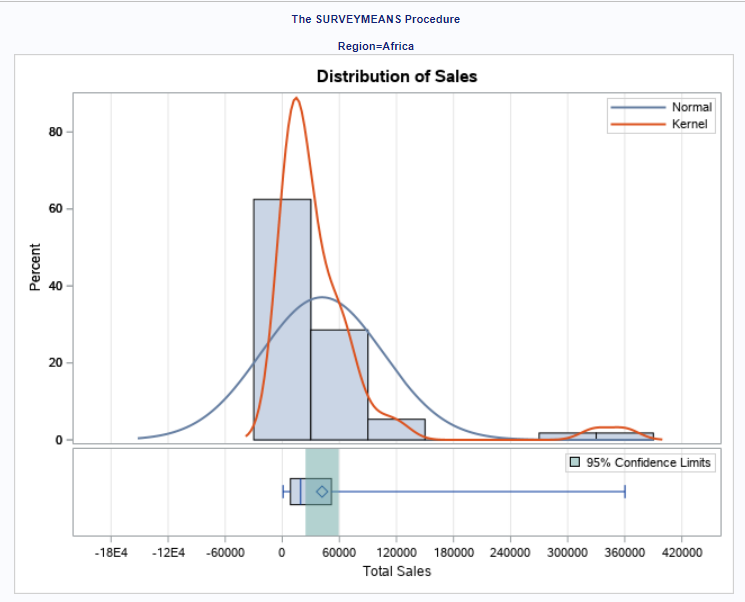
* **BY option**

The result of the by option is grouped each value. Look at the program below where we use by option.



Output:

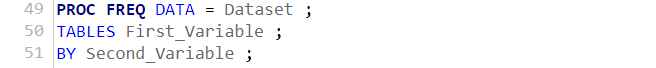




Here, we paste a single group output. The output produced all group of regions.

**Frequency Distributions**

It shows that table of frequency or count which indicates that how frequently data points in a data set occur, in the given data table each data points describes the interval between the data points and it help us to summarize the description of the data. To find out the frequency data points in sas, you have to use PROC FREQ. The syntax is given below:



The description of parameters that were utilised.

Dataset – Name of the data set.

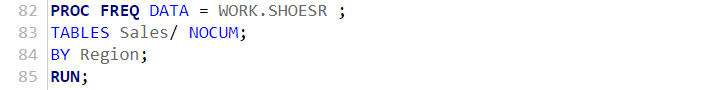
First\_Variable – Name of the variables which you require to find out the frequency distribution.

Second\_Variable – Name of the variables that were used to categorize the frequency distribution's outcome.

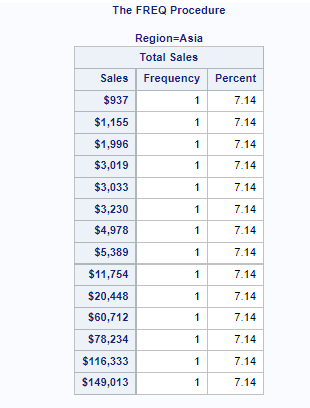
* **Selecting single variable for PROC FREQ**

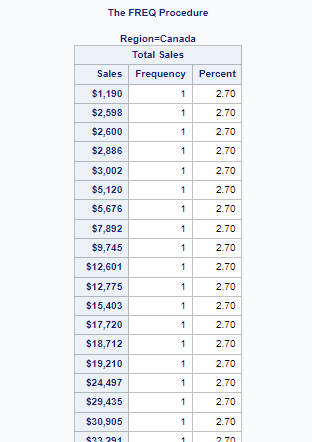
When we selecting a single variable to find the frequency distribution of the data points. In this scenario, outcome will be displayed of frequency of each value of the variable. Along with that, it will show that the cumulative frequency, and cumulative percentage in the output.

Here is the program, where we evaluate the frequency distribution of “sales” variable from the dataset named SHOESR which is refashioned data available in the library.



Output:

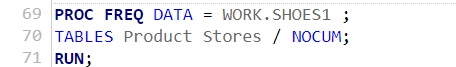




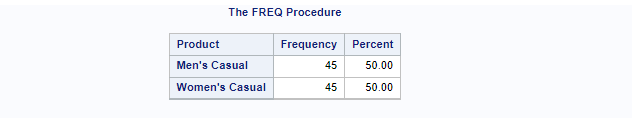
From the above result, we can observe that the results are separated into two groups meaning that the sales of shoes by region is divided into two groups that is Asia and Canada.

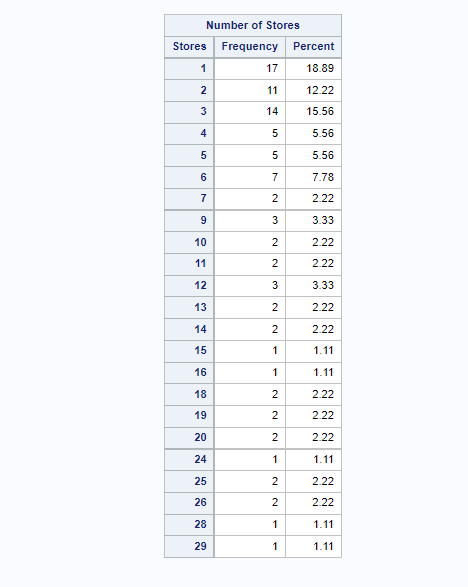
* **Selecting Multiple variables for PROC FREQ**

You must use a TABLES statement to identify the variables for which you wish to compute frequencies. We wish to selected some variable not all because you will hardly require to compute frequencies to all the variables in a dataset. Apart from that, you could also wish to get rid of the cumulative columns, which are rarely used for analysis purpose. The below program picks Product and Stores from the SHOESP which is refashioned from the library SASHELP.SHOES and removes the cumulative statistics from the result.



Output:





In the program above, we determine the frequency analysis for a shoes product when grouped by shoes sales, as well as the frequency distribution for each shoes sales when grouped by product. Additionally, if users would not like to compute percentages, then they can also include the NOPERCENT option in the TABLES statement.

**Cross Tabulations**

The cross tabulations (in short cross tab) help to define the cross tables (or contingent tables) combinations between the two or more categorical variables. The table displays all possible combinations. In SAS, we require PROC FREQ with the TABLES option to generate cross tables.

Syntax:



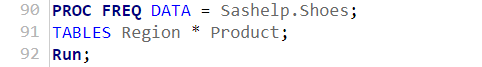
Description of the parameters are as follows:

Dataset – Name of the data set.

First\_Variable and Second\_Variable – Name of the variables which you wish to find out the frequency distribution.

* **Crosstab of two variables**

Take a look at the following program, which combine the categorical variables "Region" and "Product".



Output:

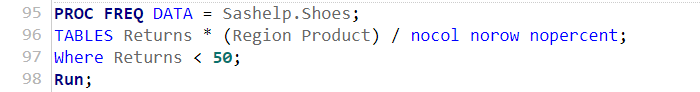


The above result display that the shoes product types are presented under each region using the dataset using SASHELP.Shoes. In this output, we see that there is a separate frequency values all over the region and shoes product. We observe that there are four results for all rows and columns and the dimension of the table is 10x8 because row variables are 10 and the column variables are 8. Besides, if you need only frequency distribution in the output then you can easily eliminate the other additional statistics by using the slash, thereafter, select the options that you want to avoid from the output (see below output).

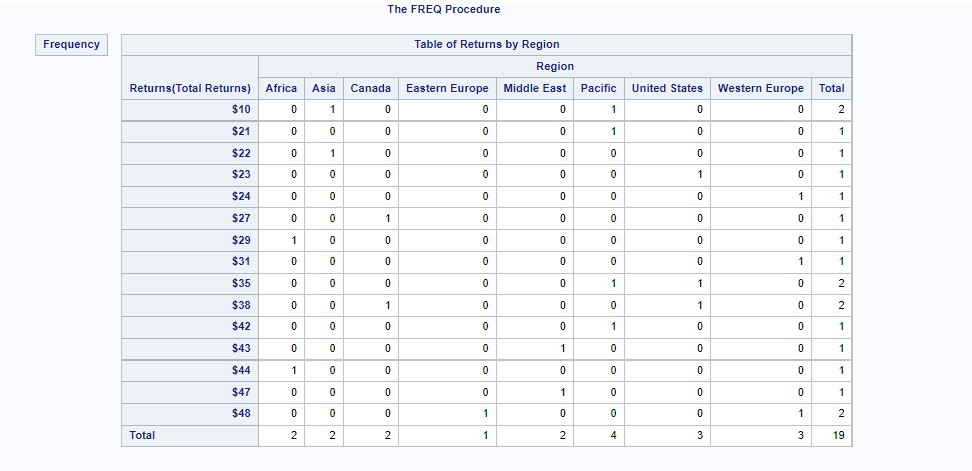
* **Crosstab of three variables**

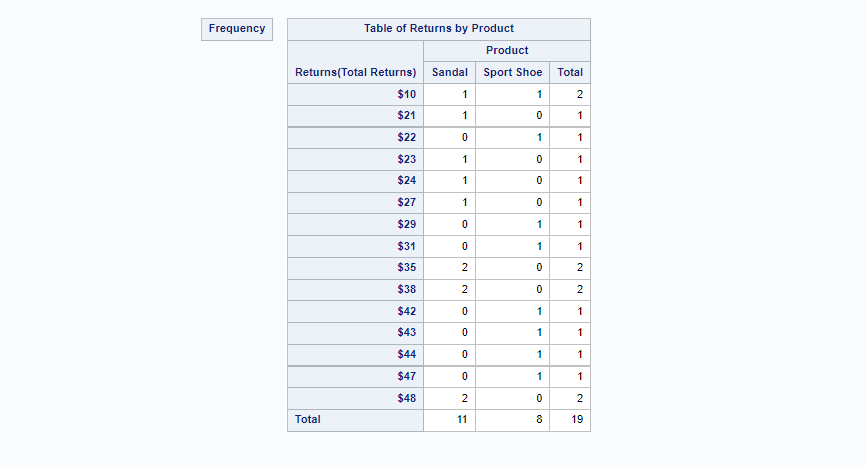
Take a look at the following program, which combine the two variables and cross-tabulate each of them with the third variable. As a result, we’ve two contingent tables.

Here is the program:



Output:





We observe the output above shows that the shoes “returns” variable is divided into regions and products variables. Additionally, we eliminate the extra statistics from the output. Moreover, you can increase the variables for paired combinations by using the correct use of asterisk between the variables.

**Correlation Analysis**

The correlation analysis is related to the relationship (or strength or direction) between the two numeric variables. The symbol of correlation coefficient denotes by “r” when it belongs to sample otherwise population i.e., ρ. The range of r lies between the minus 1 to plus 1. In SAS, you have to use PROC CORR statement for calculating the correlation coefficient between the two variables in a data set.

Basis Syntax:



Description of the parameters are as follows:

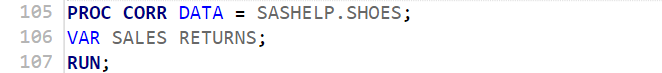
Dataset – Name of the data set.

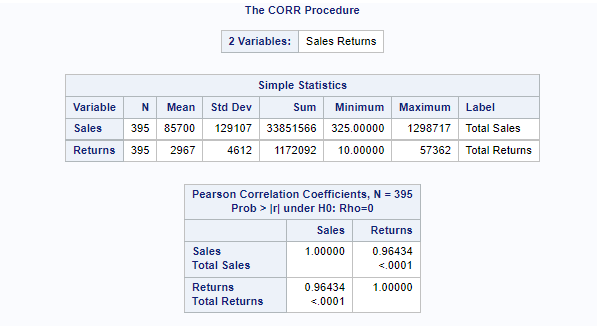
Variable – Name of the variable that you wish to find the correlation.

Options – Extra options for producing such as matrix etc.

* **Correlation between two variables**

Here is the program for correlation:





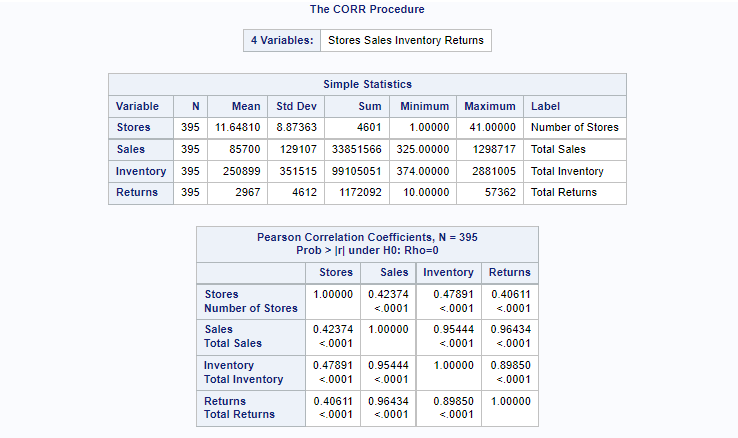
As a result, the correlation between the sales and returns of shoes is 0.96, which indicates that there is a strongly positive relationship among them.

* **Correlation between all variables**

Here is the program for correlation among all variables:



Output:



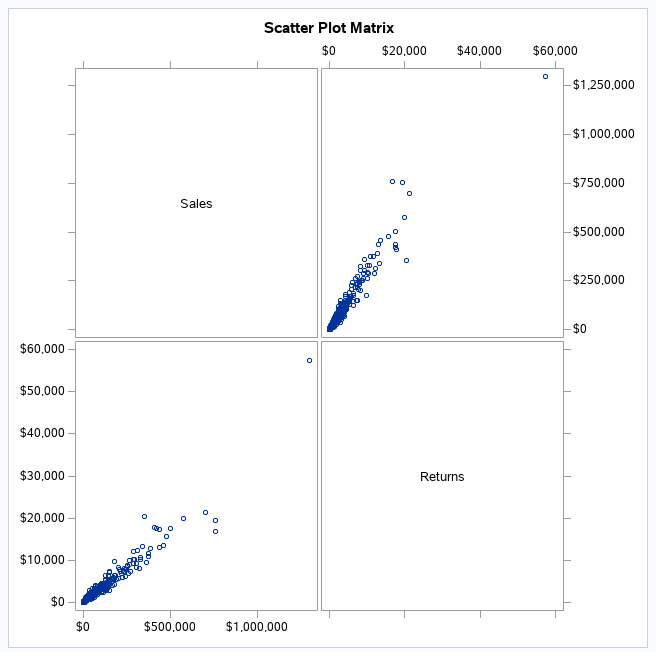
We see the difference above in the both programs i.e., if we wish to find out all the correlation among the variables then there is no need to add “VAR” statement, but in case if you want some selected variables then you have to use VAR statement.

* **Correlation Matrix**

For getting the correlation matrix among the variables, we have to select option to plot matrix in the PROC command. The matrix between sales and returns is shown in the program below.



Output:



From the above result, you can observe the relationship between the two variables from the scatter matrix plot. There is a positive strength between the sales and returns, meaning when one variable increase while the other increases and vice-versa.

**Hypothesis Testing**

The hypothesis testing is the statistical method uses a sample data to determine the population parameter or probability on a given hypothesis is true. There are two types of hypothesis that’s is null hypothesis (H0, where statement has no effect) and alternative hypothesis (H1, where statement has an effect or difference).

Here are the few steps of hypothesis testing:

Step 1: Construct a null and alternative hypothesis.

Step 2: Set the level of significance i.e., alpha ( meaning if we set alpha at 5% which implies that you will not reject your H1, when you H0 is actually true.

Step 3: Select and compute test statistics and p-value.

Step 4: Compare the p-value, if the p-value is more than the alpha level (0.05), do not reject the null hypothesis, whereas, if p-value is less than the significance level, reject the null hypothesis. In short, p-value < 0.05, reject H0, otherwise do not reject H0.

These are the basic steps of hypothesis testing. In sas, we have an options in the statement for hypothesis testing such PROC TEST H0=0. If we do not set null hypothesis then it will by default choose H0=0.

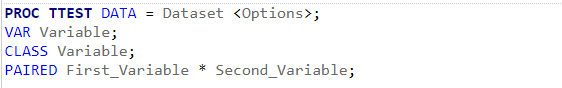
**T-Tests or Student’s T-Test**

A t-test is a statistical/analytical test in SAS programming which is used for comparing the means of two groups or check there is a significant difference between two groups. It is mostly use in hypothesis testing when analyst wants to know that whether two groups are different from one another. In SAS, the PROC TTEST is a procedure for doing t-tests on a single variable or a set of variables. The formula of t-test is as:



Where, is a sample mean, is a population mean difference, s( is a sample difference S.D., and n is sample size (or no. of observations).

General syntax for using PROC TTEST are as follows:



Description of the parameters are as follows:

Dataset – Name of the data set.

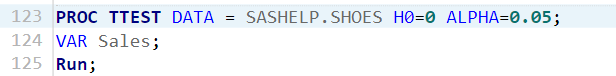
First\_Variable and Second\_Variable – Name of the variables that you want to applying the t-test.

Options – Additional options helps fixed some values such null hypothesis, alpha value, etc.

* **One-Sample T-Test**

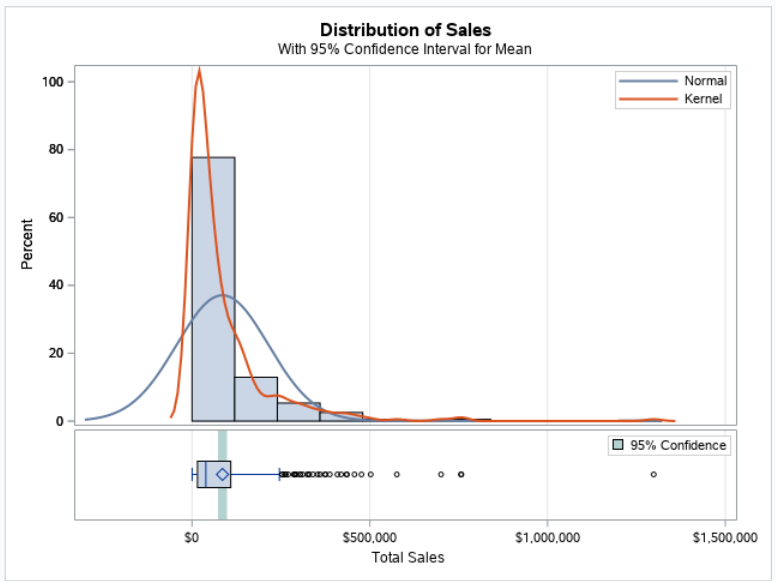
We provide you one example where one sample t-test finds the t-test estimation of variable “Sales” with 5% level of significance. We again choose the SASHELP.SHOES dataset.

Here is the program:



Output of t-test:



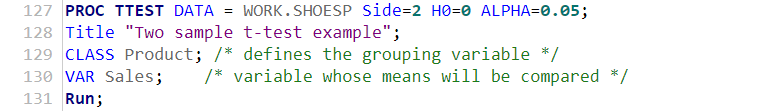


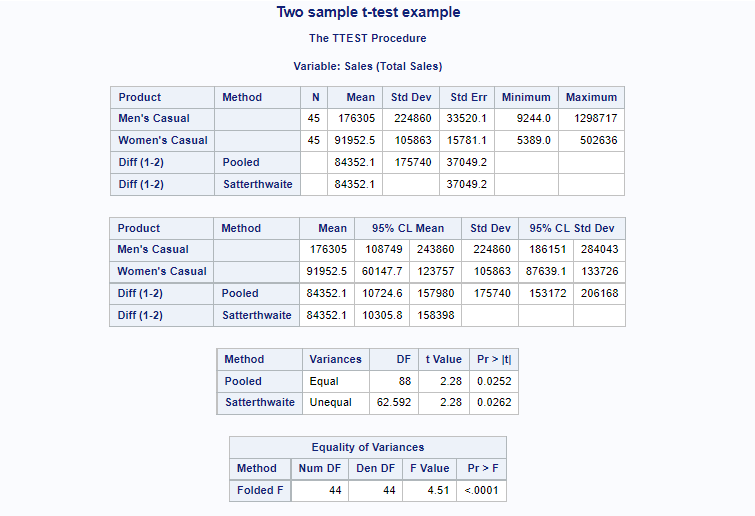
From the above result, the t-test value is 13.19 with degree of freedom is 394 where the p-value is less than 0.05 which indicates that the sales is statistically significant. Hence, t-test determine the total shoes sales of population is significantly different from the mean test value.

* **Independent Two-Sample T-Test**

The two-sample t-test is used to measure the relationship between two groups means of the same variable. In this scenario, we are comparing the mean of the variable sales between two different product types ("Men's Casual" and " Women's Casual").

Here is the program:

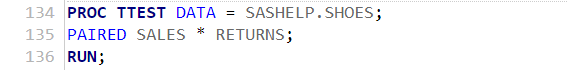




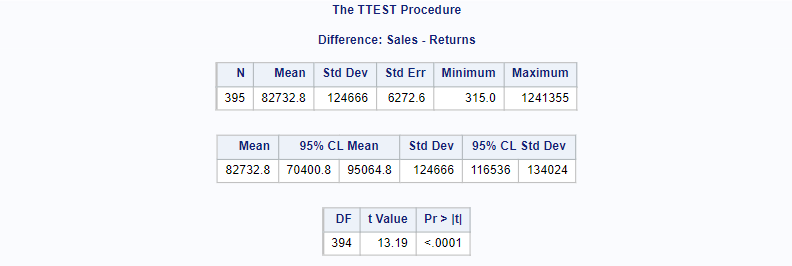
From the above equality of variances, a p-value is close to zero which is less than level of significance i.e., 0.05, so variances are significantly different. The result also displays that the t-value is 2.28 with 88 degree of freedoms and the p-value is 0.03 which is less than the alpha value i.e., 0.05, this means that the we reject the null hypothesis and support the alternative hypothesis. Thus, total sales for Men's Casual and Women's Casual is statistically significant different (t = 2.28, p = 0.03). In other word, we can conclude that product type of Men's Casual (176305) have a statistically significantly higher mean sales than Women's Casual (91952.5)

* **Paired T-Test**

The paired t-test is used to determine whether two dependent variables are statistically distinct from each other or not. We use the paired T test, as illustrated below, the sales and returns of shoes are dependent on each other. Here is the program:



Output:



The above result displays that the t-value is 13.19 with 394 degree of freedoms and the p-value is close to zero which is less than the alpha value i.e., 0.05, this means that the we reject the null hypothesis and support the alternative hypothesis. Thus, the sales and returns variable are statistically not different from each other.

**Z-Test**

A Z-test is an analytical test that uses a normal distribution and is mostly used to find solutions with large samples where the number of observations is higher than or equal to 30. Thus, the key identification of when to use z-test is that the population standard deviation is recognized and sample size is greater than 30. As we know that sas PROC TTEST is taken into account sample size, which returns results accordingly. In SAS, there is no isolated code for the z test.

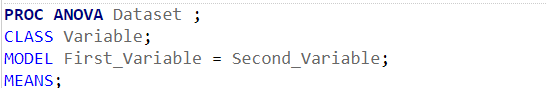
**Analysis of Variance (ANOVA)**

ANOVA is a statistical model which helps or used to analyze the differences between groups of means or data are statistically significant.

**One-way ANOVA**

It conducts data analysis on a variety of experimental designs. In this analysis, there is a dependent (continuous) variable which is determined by independent variables (also known as classification variables). in SAS, it is perform using the PROC ANOVA statement. We use the proc anova, as illustrated below, to test whether the mean of total sales of shoes differs between the product types which is the categorical variable.

Syntax of ANOVA:



Description of the parameters are as follows:

Dataset – Name of the data set.

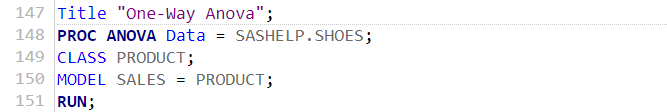
Class – Name of the categorical variable

Model – it specifies which variables from the dataset will be used to fit the model.

First\_Variable and Second\_Variable – Name of the variables that you want to use in analysis.

Means – it is a term that describes the method of computing and comparing means.

Here is the program:



Output of ANOVA:



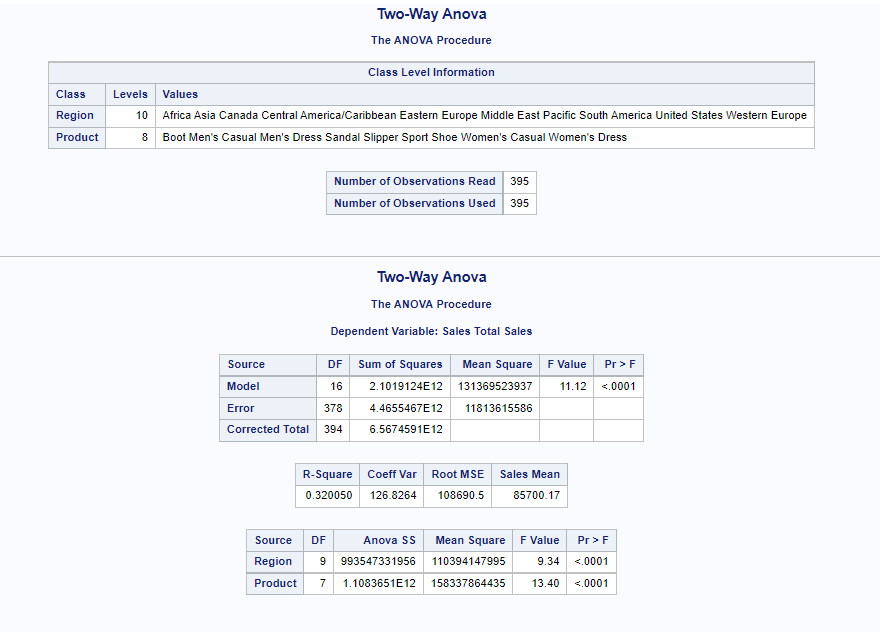
In this output, we observe that the shoes product is a categorical variable, so we consider it as class variable and numeric variable is total sales. In the model statement, we use both variables. The result of the output is that the mean of the dependent variable (sales) differs significantly among the products type.

**Two-Way ANOVA**

In this method, we require one continuous response variable and two categorical controlled variables. With the help of below program, we would like to analyse sales differs between the product and region.



Output:

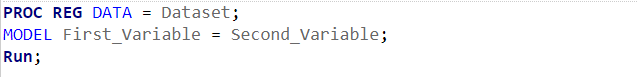


The above output point that the overall model is statistically significant (F = 11.12 and p < 0.0001). The variables region and product are statistically significant (F = 9.34 and F = 13.40, respectively and p value is same for both i.e., p = 0.0001).

**Linear Regression**

The Linear regression (or Linear model) is one in which the input variables (x) and the single output variable (y) are assumed to have a linear relationship (y). The procedure is known as simple linear regression (SLR) when there is only one explanatory/input variable (x), whereas, if the is more than one explanatory variable then it is known as multiple linear regression (MLR). Also, linear regression is a technique for determining the connection between one or more explanatory/independent variables and a dependent variable. A relationship framework is designed, and parameter values estimations are used to create an estimated regression equation. This model is based on OLS (ordinary least square) assumptions such as regression model is linear, etc. Further, there are various tests which determine the model is good fit/satisfactory or not. In SAS, the PROC REG statement is used to apply for regression model.

Basic Syntax:



Description of the parameters are as follows:

Dataset – Name of the data set.

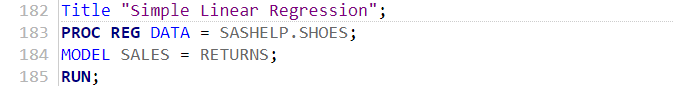
First\_Variable – Name of the single output variable or dependent variable.

Second\_Variable – Name of the input variable(s) or explanatory variable(s).

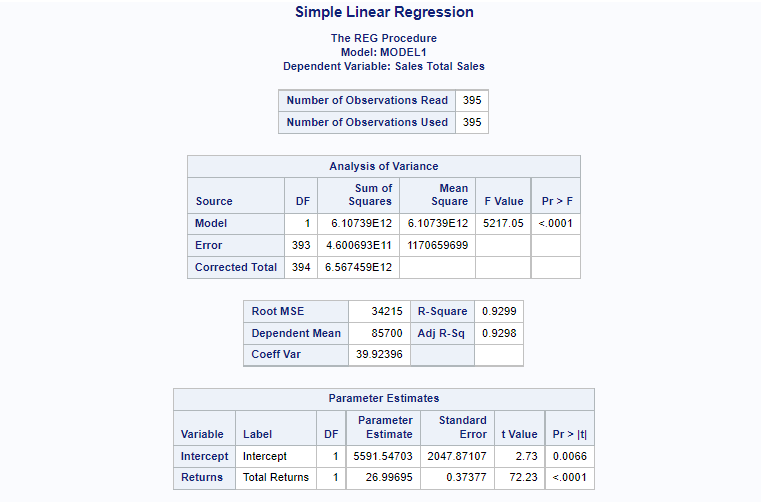
**Simple Linear Regression (SLR)**

The below program determines that the relationship between the independent variable (returns) and dependent variable (sales) by using PROC REG. Thus, we get the intercept values from the result which can be developed to create the regression model equation.

Here is the program of SLR:



After running the regression program, we get the result below:



The intercept term is 5591.55 and the coefficient of returns is +27. Therefore, the simple linear regression equation is as follows

Next, you will learn how to interpret the regression model results.

Sales-intercept, , implies that when the value of X = 0, the mean value of Y is 5591.55. On the other hand, the slope of = 27, implies that for each increase of 1 product is return, the value of total sales is estimated to increase by 27.

*How to diagnose the model?*

You can diagnose the model with the help of p-value. The general rule of p-value is when the p-value is less than the level of significance, then model is significant otherwise model is insignificant. Thus, based on the regression result the overall model is statistically significant, p-value is 0.0001 < 0.05.

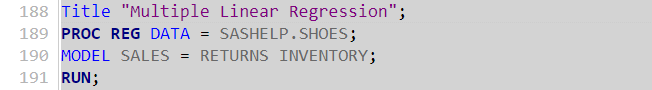
Also, the R-square of the regression model is 0.92 (or 92%) which indicates that the variation in the dependent variable (sales) can be explained by the variation in the independent variable (returns). The higher R-square generally indicates a better fit for the data or model.

**Multiple Linear Regression (MLR)**

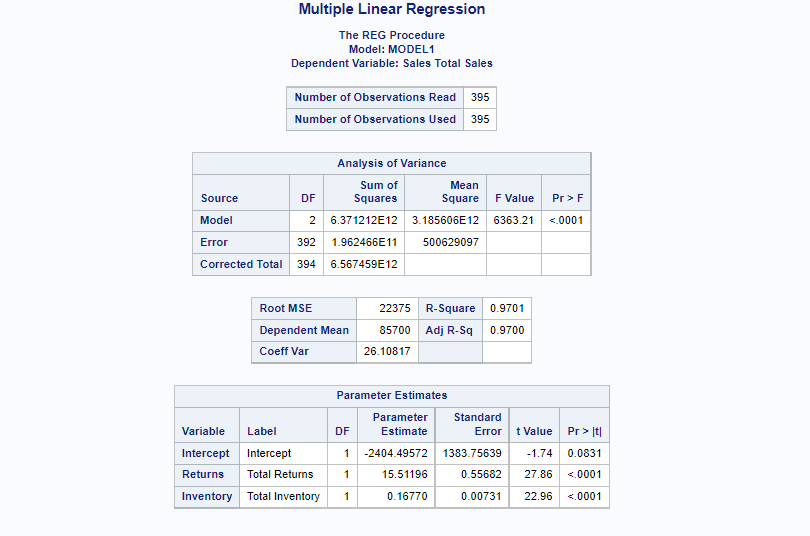
This method determines the connection between more than one explanatory/independent variables and a single dependent variable.

The below program determines that the relationship between the independent variables (returns and inventory) and dependent variable (sales) by using PROC REG. In short, we just increase one independent variable, rest things are same.

Here is the program of MLR:



After running the regression program, we get the result below:



The intercept term is -2404.5. The coefficient of returns and inventory is 15.51 and 0.1677, respectively. Therefore, the multiple linear regression equation is as follows

In this case, we can see that the R-square is increased. The R-square of the regression model is 0.97 (or 97%) which indicates that the variation in the dependent variable (sales) can be explained by the variation in the independent variables (returns and inventory). Hence, the higher R-square indicates a better fit for the data or model.

# **References**

First, S. S. T. e. a., n.d. *SAS Support.* [Online]   
Available at: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fsupport.sas.com%2Fresources%2Fpapers%2Fproceedings%2Fproceedings%2Fsugi30%2F242-30.pdf&clen=44377&chunk=true

Ron Cody, 2018. *Learning SAS® by Example: A Programmer's Guide.* Second Edition ed. United States of America.: Cary, NC: SAS Institute Inc..

S. G., 2017. *Applying Data Science: Business Case Studies Using SAS®.* United States of America: Cary, NC: SAS Institute Inc..

Zhao, Y., 2008. *Lexjansen.* [Online]   
Available at: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fwww.lexjansen.com%2Fnesug%2Fnesug08%2Fbb%2Fbb04.pdf&clen=57259&chunk=true  
[Accessed 2008].

Documentations are available online, at <http://support.sas.com/documentation/onlinedoc/index.html>.

SAS Institute Home Page: <http://www.sas.com>