STAT 425 and STAT 625 Statistical Software

Lecture 2

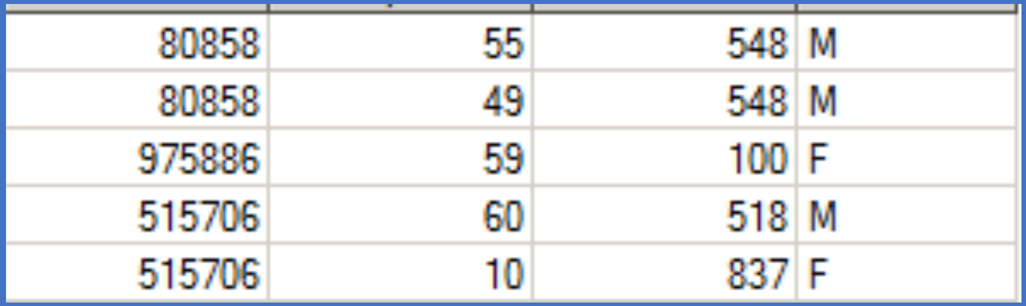
Understanding the Programming Steps

Saida Zardi

A Sample SAS Program

Entering data into SAS:

- Myaccount data have 4 variables: Account, Deposit, Withdrawal and Gender

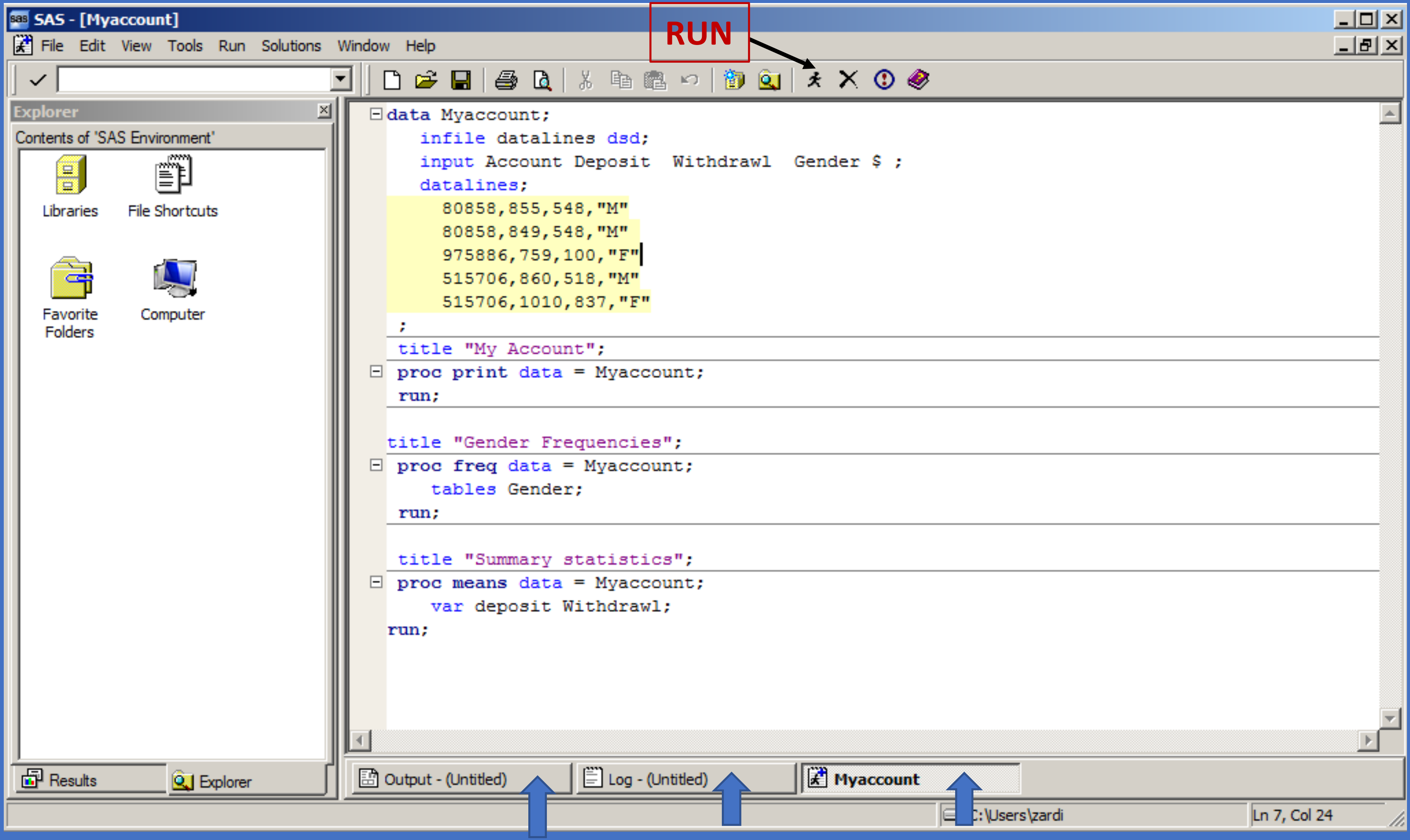


80858	55	548	M
80858	49	548	M
975886	59	100	F
515706	60	518	M
515706	10	837	F

- Each data will be separated from the following one by a comma.
- The purpose of the program is to enter the data and write a program to produce reports.

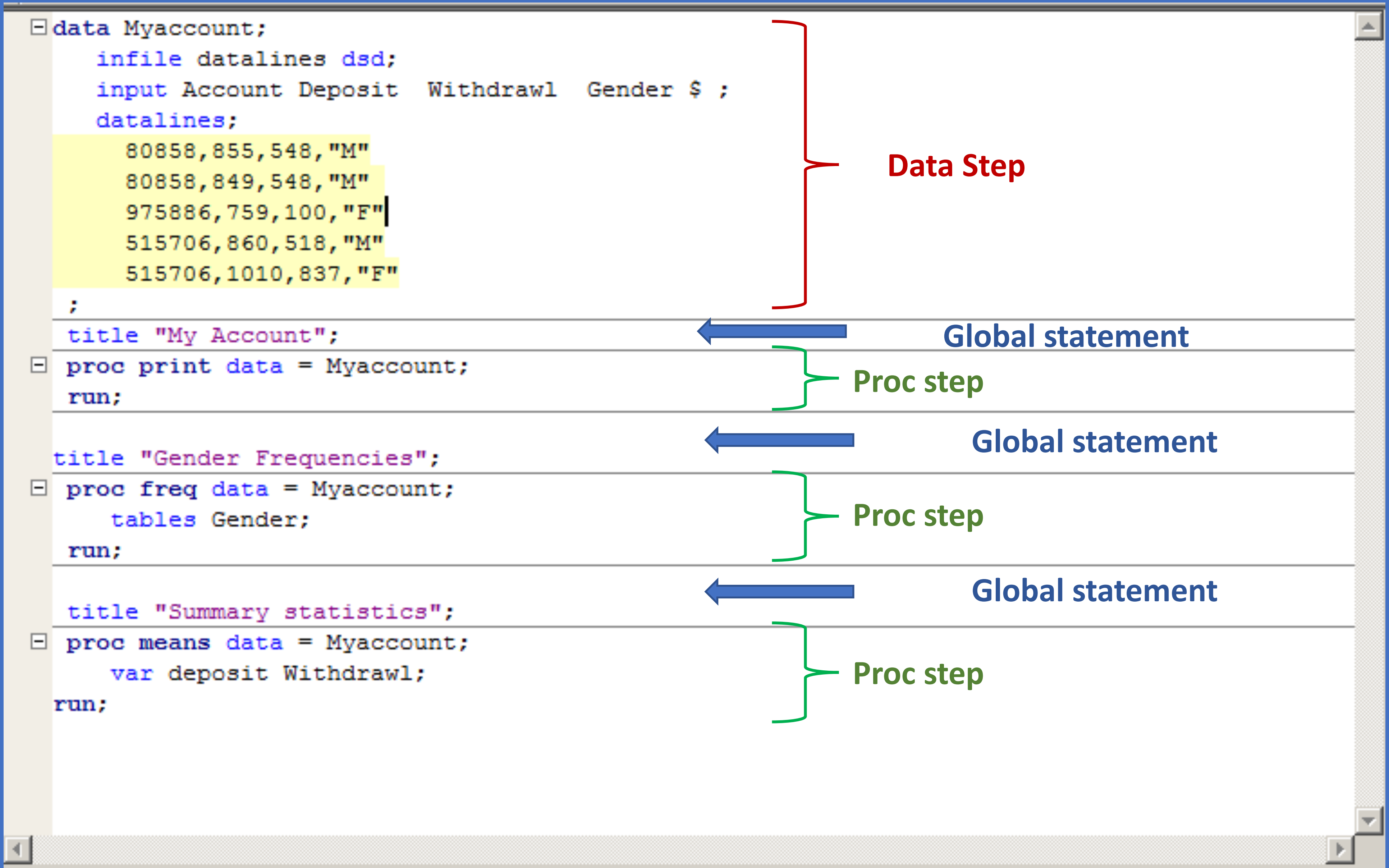
A Sample Program

- Here is the program, written in the Editor Window



```
data Myaccount;
   infile datalines dsd;
   input Account Deposit  Withdrawl  Gender $ ;
   datalines;
     80858,855,548,"M"
     80858,849,548,"M"
     975886,759,100,"F"
     515706,860,518,"M"
     515706,1010,837,"F"
 ;
 title "My Account";
 proc print data = Myaccount;
 run;

title "Gender Frequencies";
 proc freq data = Myaccount;
    tables Gender;
 run;

 title "Summary statistics";
 proc means data = Myaccount;
    var deposit Withdrawl;
run;
```

Editor Widow



```
data Myaccount;
   infile datalines dsd;
   input Account Deposit  Withdrawl  Gender $ ;
   datalines;
     80858,855,548,"M"
     80858,849,548,"M"
     975886,759,100,"F"
     515706,860,518,"M"
     515706,1010,837,"F"
 ;
 title "My Account";
 proc print data = Myaccount;
 run;

title "Gender Frequencies";
 proc freq data = Myaccount;
    tables Gender;
 run;

 title "Summary statistics";
 proc means data = Myaccount;
    var deposit Withdrawl;
run;
```

Data Step

Global statement

Proc step

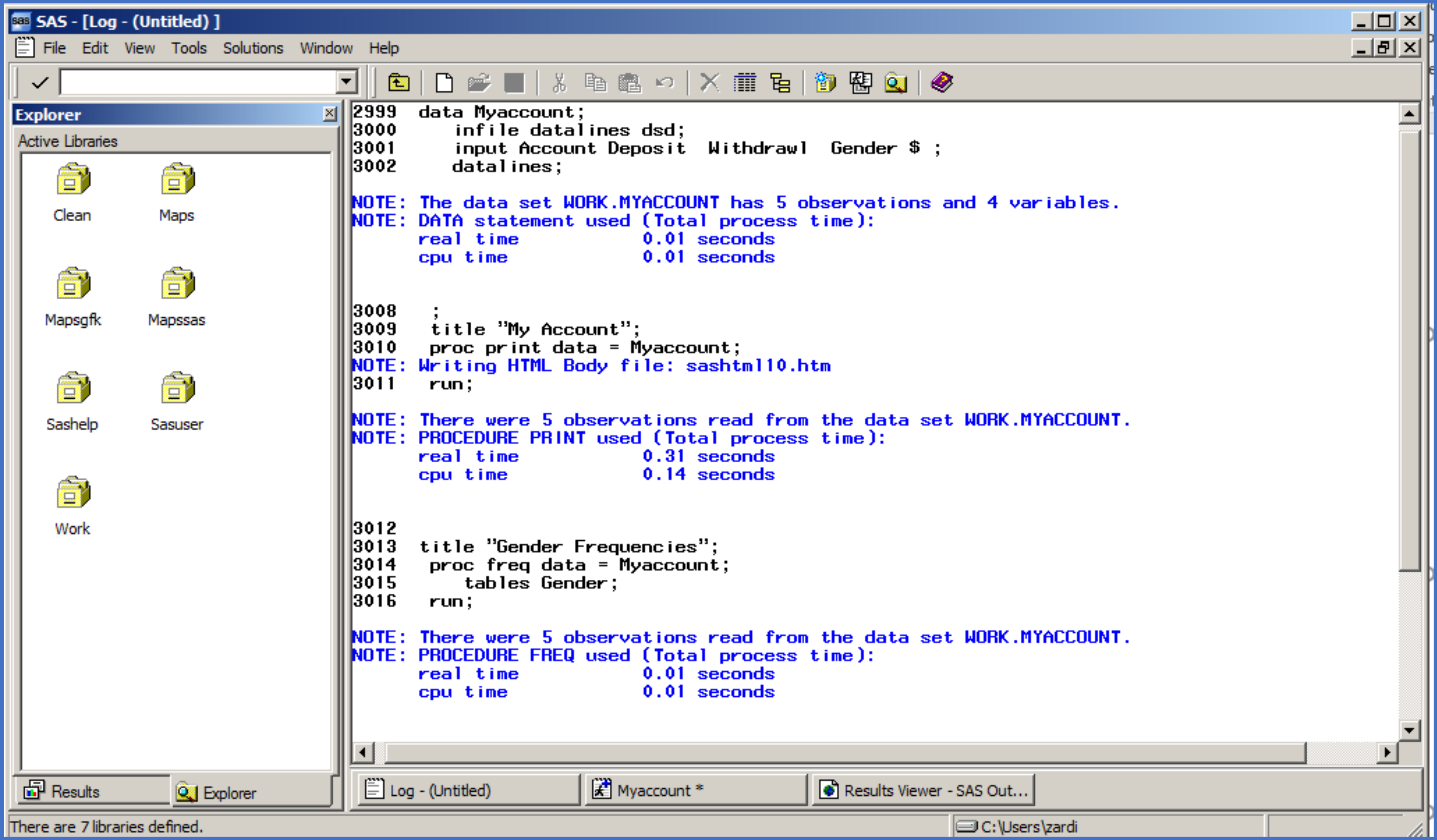
Global statement

Proc step

Global statement

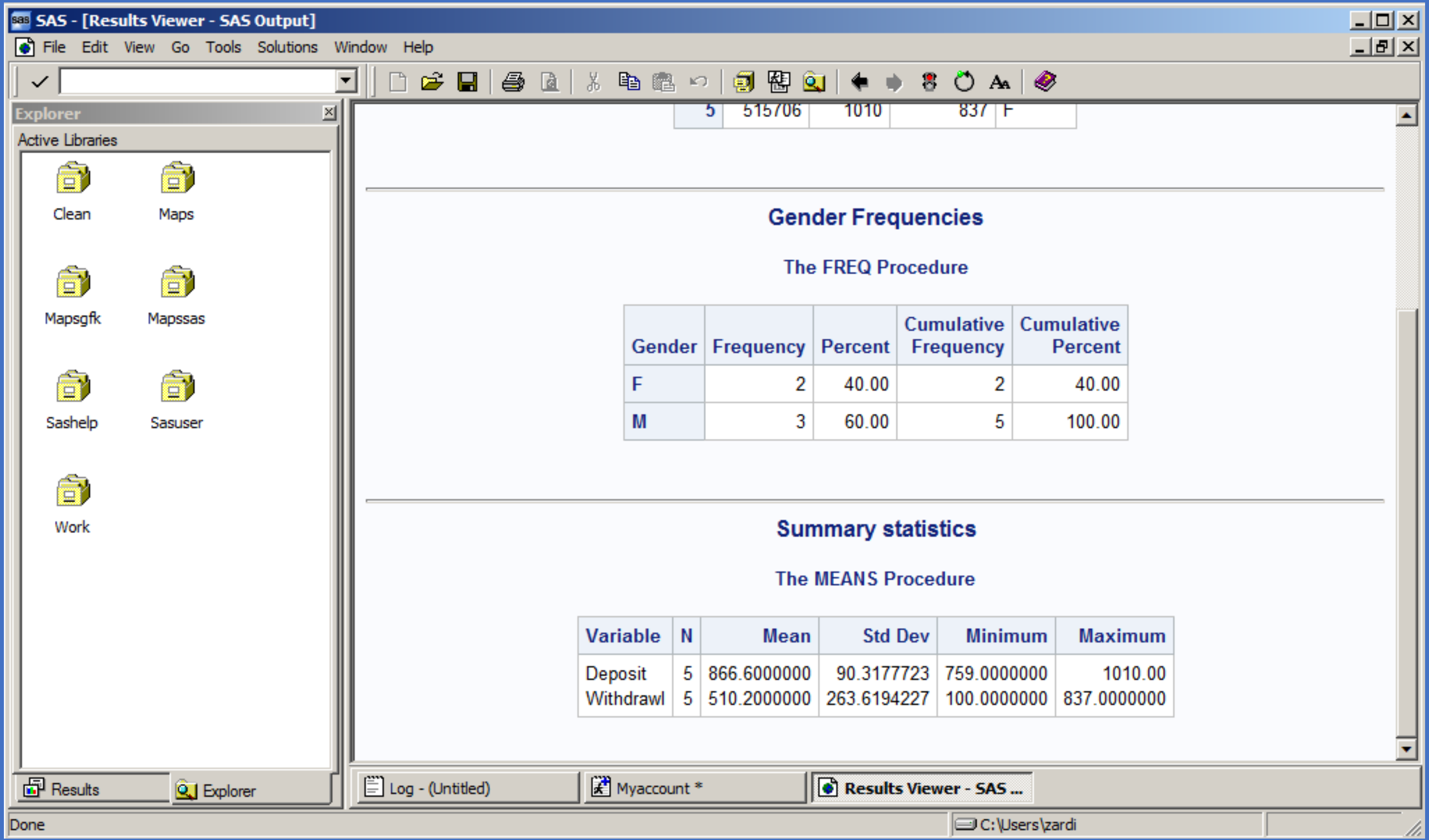
Proc step

Log Window



```
2999  data Myaccount;
3000     infile datalines dsd;
3001     input Account Deposit  Withdrawl  Gender $ ;
3002    datalines;

NOTE: The data set WORK.MYACCOUNT has 5 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


3008   ;
3009   title "My Account";
3010   proc print data = Myaccount;
NOTE: Writing HTML Body file: sashtml10.htm
3011   run;

NOTE: There were 5 observations read from the data set WORK.MYACCOUNT.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.31 seconds
      cpu time            0.14 seconds


3012
3013  title "Gender Frequencies";
3014   proc freq data = Myaccount;
3015      tables Gender;
3016   run;

NOTE: There were 5 observations read from the data set WORK.MYACCOUNT.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Output – Results Window



5	515706	1010	837	F

Gender Frequencies

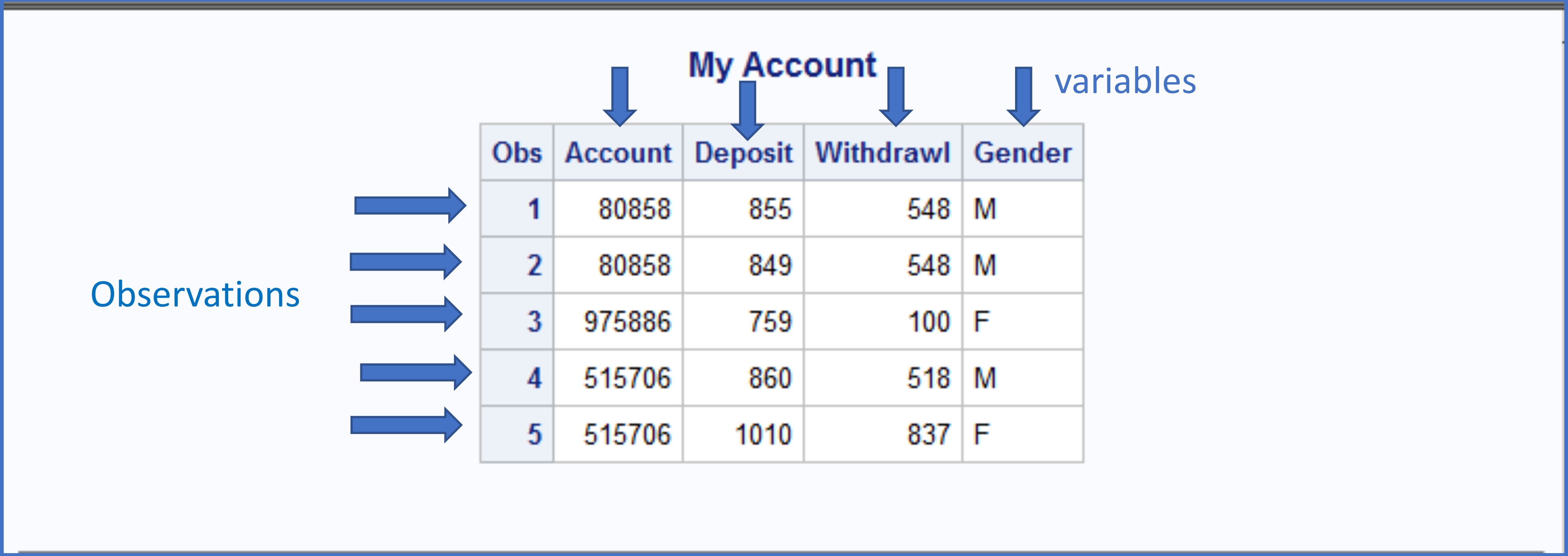
The FREQ Procedure

Gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	2	40.00	2	40.00
M	3	60.00	5	100.00

Summary statistics

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Deposit	5	866.6000000	90.3177723	759.0000000	1010.00
Withdrawl	5	510.2000000	263.6194227	100.0000000	837.0000000

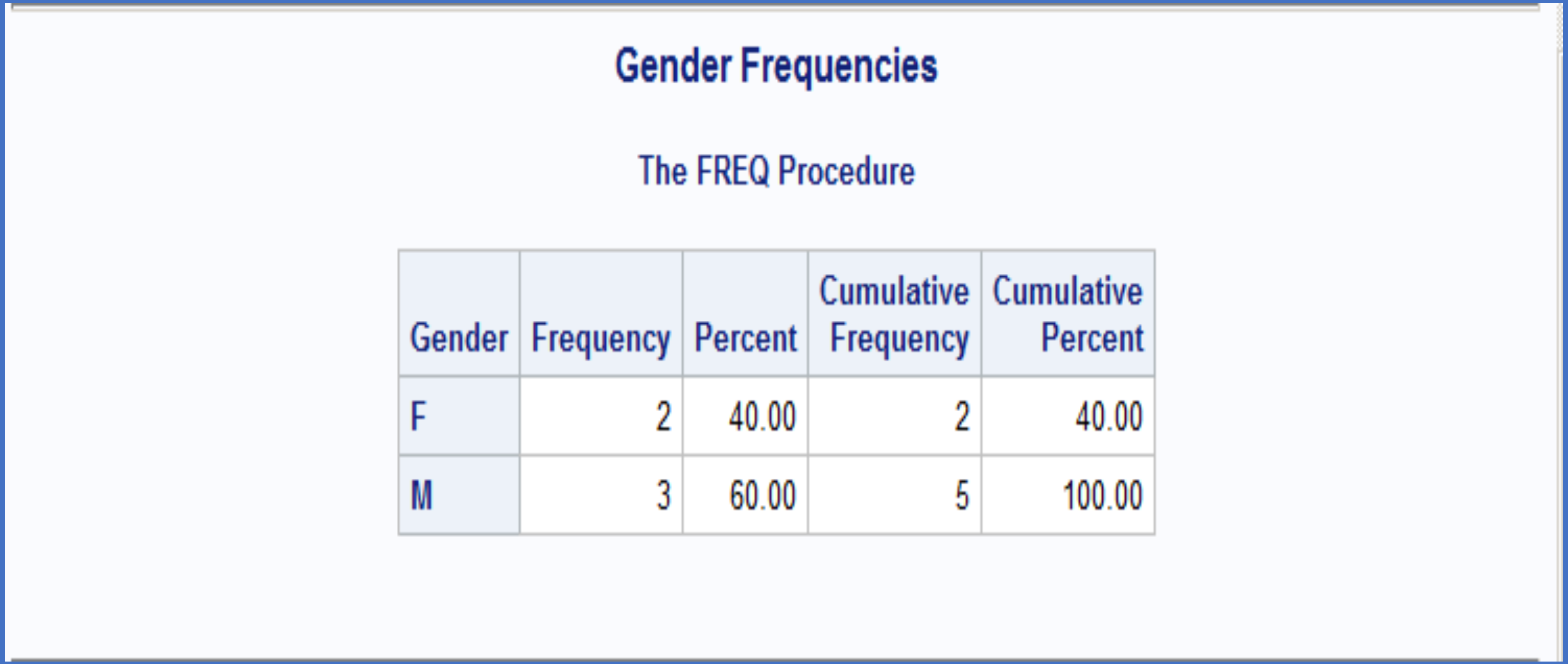


variables

Observations

My Account

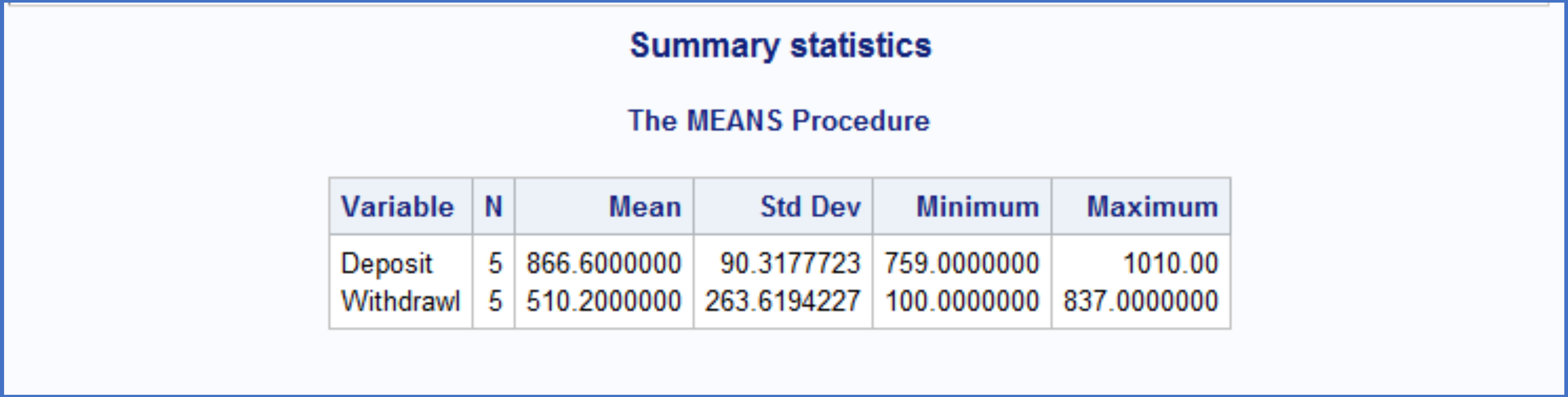
Obs	Account	Deposit	Withdrawl	Gender
1	80858	855	548	M
2	80858	849	548	M
3	975886	759	100	F
4	515706	860	518	M
5	515706	1010	837	F



Gender Frequencies

The FREQ Procedure

Gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	2	40.00	2	40.00
M	3	60.00	5	100.00

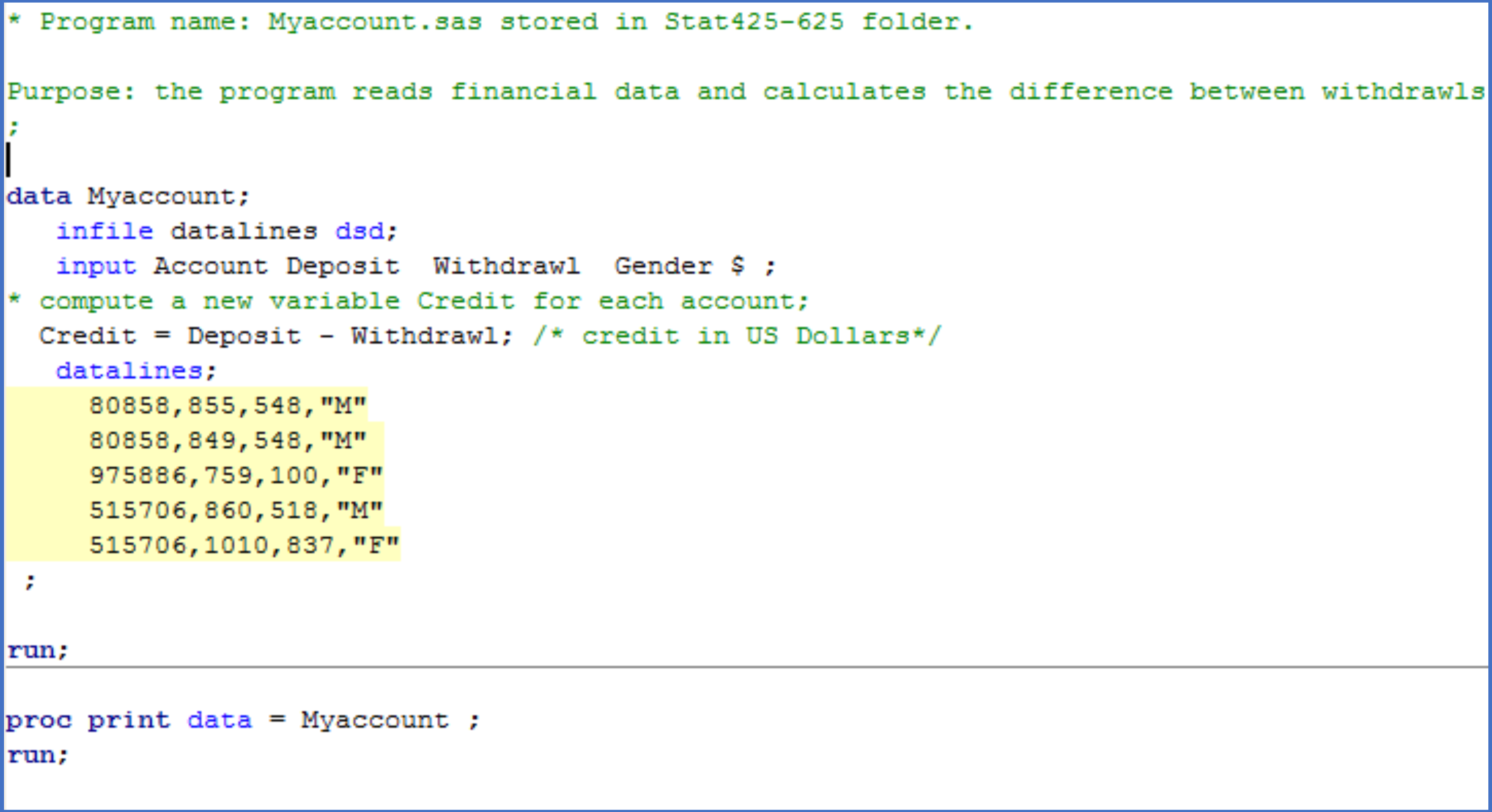


Summary statistics

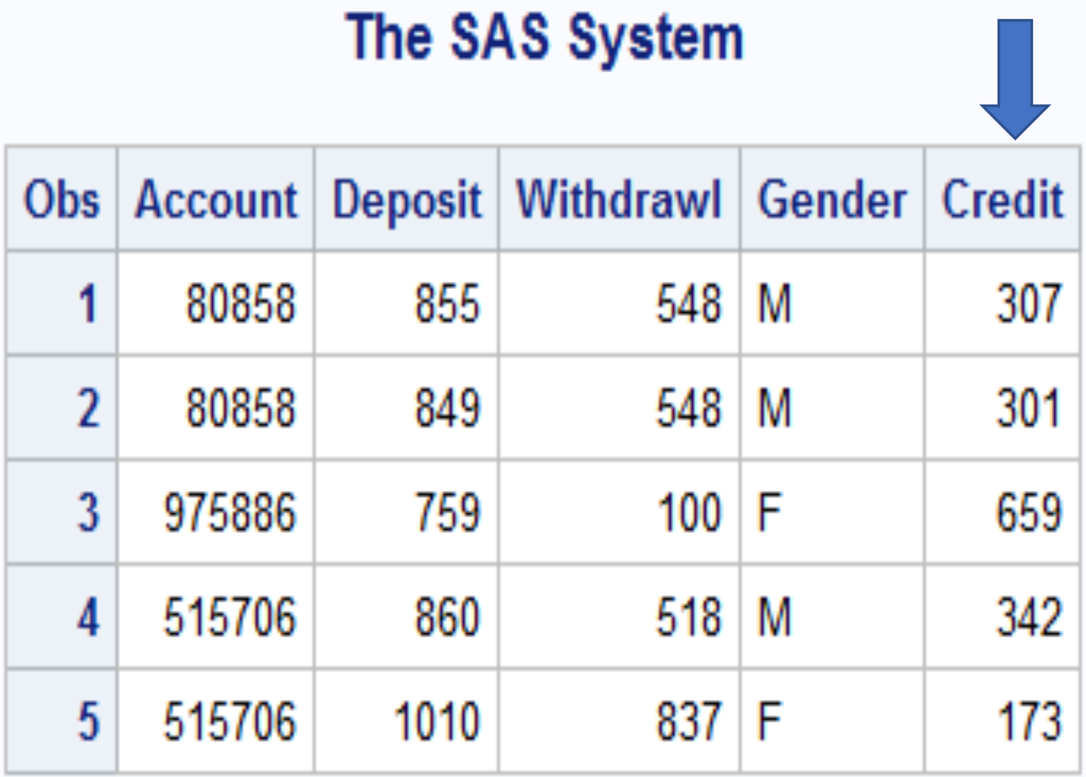
The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Deposit	5	866.6000000	90.3177723	759.0000000	1010.00
Withdrawl	5	510.2000000	263.6194227	100.0000000	837.0000000

Enhancing the Program



```
* Program name: Myaccount.sas stored in Stat425-625 folder.

Purpose: the program reads financial data and calculates the difference between withdrawls
;

data Myaccount;
   infile datalines dsd;
   input Account Deposit  Withdrawl  Gender $ ;
* compute a new variable Credit for each account;
  Credit = Deposit - Withdrawl; /* credit in US Dollars*/
   datalines;
     80858,855,548,"M"
     80858,849,548,"M"
     975886,759,100,"F"
     515706,860,518,"M"
     515706,1010,837,"F"
 ;

run;

proc print data = Myaccount ;
run;
```



The SAS System

Obs	Account	Deposit	Withdrawl	Gender	Credit
1	80858	855	548	M	307
2	80858	849	548	M	301
3	975886	759	100	F	659
4	515706	860	518	M	342
5	515706	1010	837	F	173

Operator	Description	Priority
+	Addition	Lowest
-	Subtraction	Lowest
*	Multiplication	Next Highest
/	Division	Next Highest
**	Exponiation	Highest
-	Negation	Highest

A look inside SAS: How does it work?

- SAS processes DATA steps in two ages: a **compile stage** and an **execution stage**.
- In the compile stage:
  - It prepares an area to store the SAS data set
  - Checks the INPUT file (or datalines)
  - Determines the various attributes
  - Sets aside a place in memory called: the ***input buffer***
  - Reads each line of the program, checks for invalid syntax, and determines the name of all variables in the data set.
  - Depending on the INPUT statement, SAS determines whether the variable is character or numeric and the storage length of each variable. This constitutes the *descriptor portion* of the data set.

In the compile stage no data is read and no logical statement is evaluated.

Each line is processed in order: top to bottom; left to right.

In our sample program, SAS sees the four variables in the INPUT statement and sets the storage length of each of them. Unless a length is specified, SAS gives the default length of 8 bytes.

Account Numeric 8 bytes	Deposit Numeric 8 bytes	Withdrawal Numeric 8 bytes	Gender Character 8 bytes

Then SAS sees the assignment statement for Credit. This variable is defined by an arithmetic operation, so SAS understands that it’s a numeric variable and uses the default storage of 8 bytes then adds it to the reserved area of memory called Program Data Vector (PVD)

Account Numeric 8 bytes	Deposit Numeric 8 bytes	Withdrawal Numeric 8 bytes	Gender Character 8 bytes	Credit Numeric 8 bytes

SAS reached the bottom of the DATA step. The compile stage is complete.

The Execution Stage

- SAS_sets all the values in the PDV to a missing value: uses a blank for a character value and a period for a numeric value:

Account Numeric 8 bytes	Deposit Numeric 8 bytes	Withdrawal Numeric 8 bytes	Gender Character 8 bytes	Credit Numeric 8 bytes
.	.	.		.

- The first line of data from the input is copied to the input buffer

80858	855	548	M

- An internal pointer, keeping track of the current record of the input , moves to the following line.

- SAS reads each value until it finds a delimiter and then moves along until it find the next value.

Account **Numeric** **8 bytes**	**Deposit** **Numeric** **8 bytes**	**Withdrawal** **Numeric** **8 bytes**	**Gender** **Character** **8 bytes**	**Credit** **Numeric** **8 bytes**
80858	855	548	M	**.**

- Next the Credit is evaluated and the value is added to the PDV

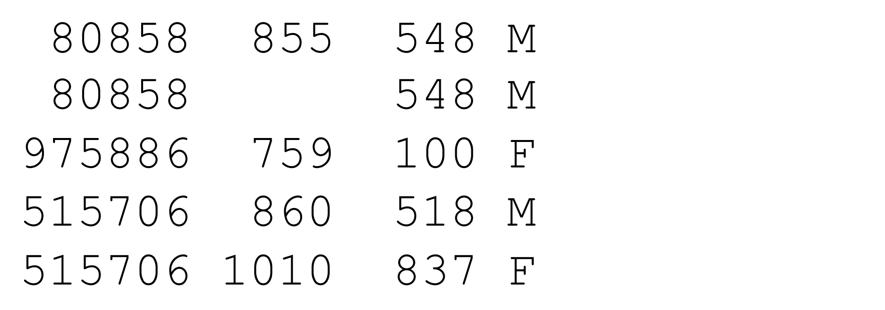
Account **Numeric** **8 bytes**	**Deposit** **Numeric** **8 bytes**	**Withdrawal** **Numeric** **8 bytes**	**Gender** **Character** **8 bytes**	**Credit** **Numeric** **8 bytes**
80858	855	548	M	**307**

SAS has reached the bottom of the data set. At this point the values of the PDV are written in the SAS data set (Myaccount) forming the first observation.

SAS returns back to the top of the DATA step , sees that there are more lines of data to read and repeats the same stages.

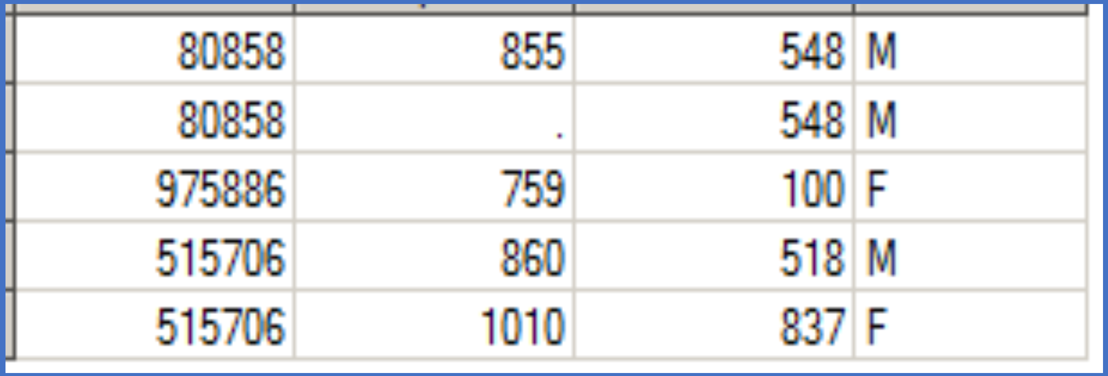
Missing Data

Table 1



```
 80858  855  548 M
 80858       548 M
975886  759  100 F
515706  860  518 M
515706 1010  837 F
```

Table 2



80858	855	548	M
80858	.	548	M
975886	759	100	F
515706	860	518	M
515706	1010	837	F