Report

IMPORT AN EXCEL FILE WITH PROC PRINT STATEMENT AND MODIFY IT

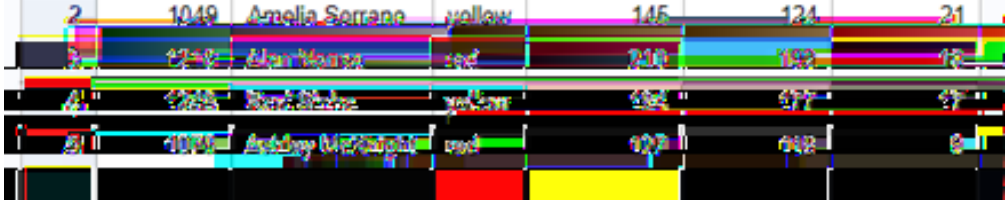
CODE:

```
*Command for importing excel file;

proc import datafile = "/home/u59242808/Data Files/SAS Data1.xlsx"

 out=SASOD_data1 dbms = xlsx replace;

run;

*modifing the given data file (creating variables);

Data SASOD_data1modified;

 set SASOD_data1;

 weightloss=StartWeight - EndWeight;

run;

proc print data=SASOD_data1modified;

run;
```

Copy

OUTPUT:

Obs	IdNumber	Name	Team	StartWeight	EndWeight	weightloss
1	1023	David Shaw	red	189	165	24
2	1049	Amelia Serrano	yellow	145	124	21



EXPLANATION:

· We can add a comment line inside the SAS code starting with star (*) and ending with semi-colon (;) in the same line.

· " **Proc import datafile=** " is used to import a datafle and the **position of the file** which we want to import is written next to it inside " ". We can get the position from **datafile properties**.

· **Out** = we give the name to the datafle.

· **Dbms** = for excel fle we write xlsx, for csv fle we write csv.

· Then we include **replace** so that if by the same name another datafle is saved this new datafle will replace that dataset and then we use semi-colon to end this particular program.

· Then we write **run;** to run this particular lines of codes.

· Then for modifying this data, at frst we give a datafle name with the code **data** new_name;

· Then we call the datafle which we want to modify by the code **set** old_name.

· In this program we had StartWeight and EndWeight columns and we want a new program by the name **weightloss** and then we will simply just put the formula/equation like **weightloss = Startweight – EndWeight.**

· Then we put **run;** and run this particular lines of codes.

· We can also print the data in beautiful form with this code **proc print data = datafile_name** which we want to see.

IMPORT AN EXCEL FILE WITH PROC PRINT STATEMENT AND MODIFY IT

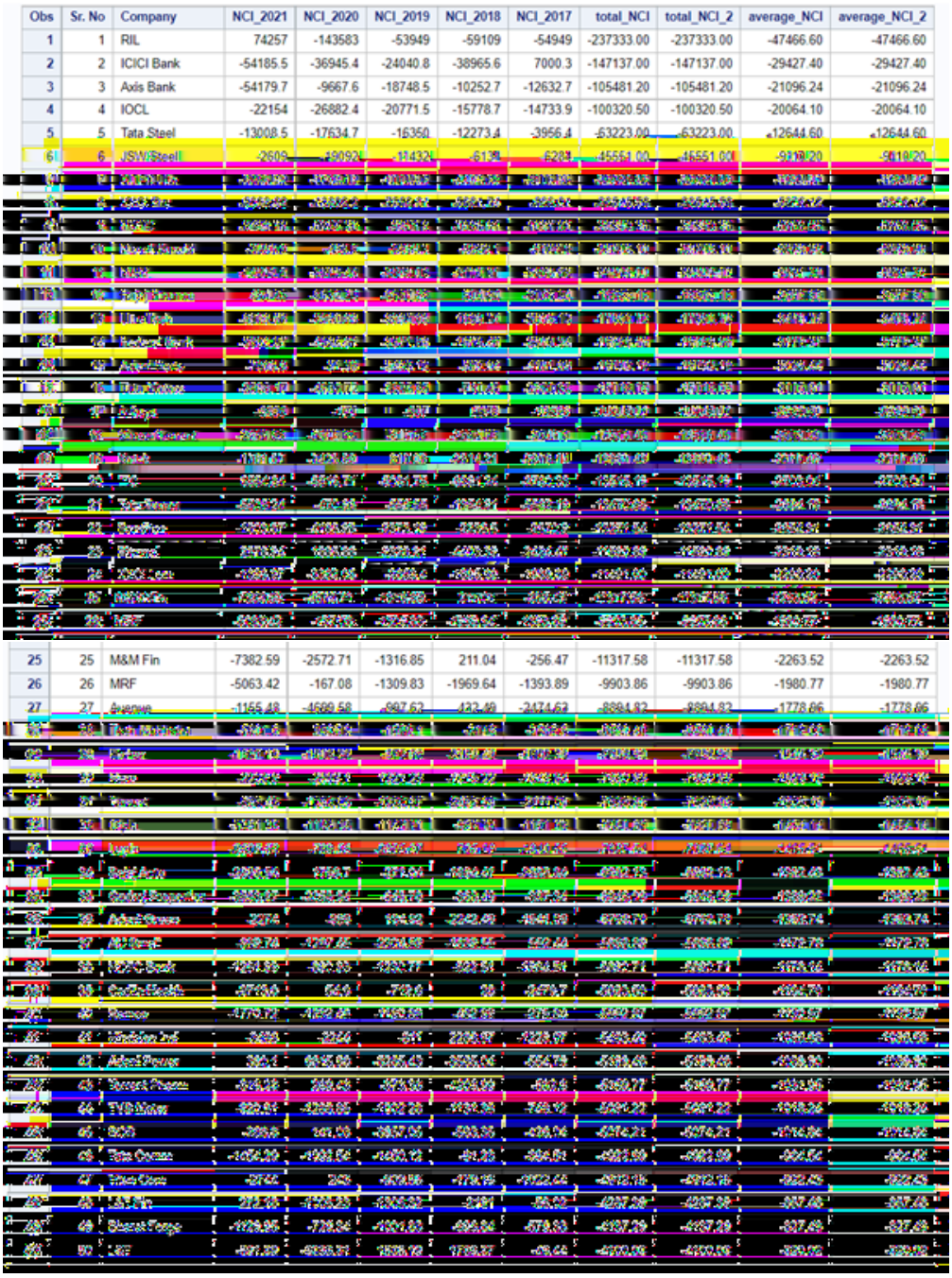
CODE:

```
/* importing file*/
```

```
proc import datafile="/home/u59242808/Data Files/SAS Data2.xlsx"
 out=data2
 dbms=XLSX replace;
run;
/* modifying the data*/
data data2_modified;
 set data2;
 total_NCI = (NCI_2021 + NCI_2020 + NCI_2019 + NCI_2018 + NCI_2017);
 total_NCI_2 = sum(NCI_2021, NCI_2020, NCI_2019, NCI_2018, NCI_2017);
 average_NCI = total_NCI / 5;
 average_NCI_2 = mean(NCI_2021, NCI_2020, NCI_2019, NCI_2018, NCI_2017);
run;
proc print data= data2_modified;
run;
```

Copy

OUTPUT:



Obs	Sr. No	Company	NCI_2021	NCI_2020	NCI_2019	NCI_2018	NCI_2017	total_NCI	total_NCI_2	average_NCI	average_NCI_2
1	1	RIL	74257	-143583	-53949	-59109	-54949	-237333.00	-237333.00	-47466.60	-47466.60
2	2	ICICI Bank	-54185.5	-36945.4	-24040.8	-38965.6	7000.3	-147137.00	-147137.00	-29427.40	-29427.40
3	3	Axis Bank	-54179.7	-9667.6	-18748.5	-10252.7	-12632.7	-105481.20	-105481.20	-21096.24	-21096.24
4	4	IOCL	-22154	-26882.4	-20771.5	-15778.7	-14733.9	-100320.50	-100320.50	-20064.10	-20064.10
5	5	Tata Steel	-13008.5	-17634.7	-16350	-12273.4	-3956.4	-63223.00	-63223.00	-12644.60	-12644.60
6	6	JSW Steel	-2609	-19092	-14432	-6138	6284	-45551.00	-45551.00	-9110.20	-9110.20
25	25	M&M Fin	-7382.59	-2572.71	-1316.85	211.04	-256.47	-11317.58	-11317.58	-2263.52	-2263.52
26	26	MRF	-5063.42	-167.08	-1309.83	-1969.64	-1393.89	-9903.86	-9903.86	-1980.77	-1980.77
27	27	Avenue	-1155.48	-4699.58	-997.62	122.49	-2474.63	-8894.82	-8894.82	-1778.96	-1778.96

EXPLANATION:

In this program the things we learnt new are pointed below:

· In case of modifying a datafle, we can simply put the **new_variable_name = equation** like previous work or we can also use **new_variable_name = formula.**

· Here we used

total_NCI = (NCI_2021 + NCI_2020 + NCI_2019 + NCI_2018 + NCI_2017);

which is the **equation.**

total_NCI_2 = sum(NCI_2021, NCI_2020, NCI_2019, NCI_2018, NCI_2017);

but in this case we simply put the **sum** formula

· Also in case of fnding average we used **mean** formula.

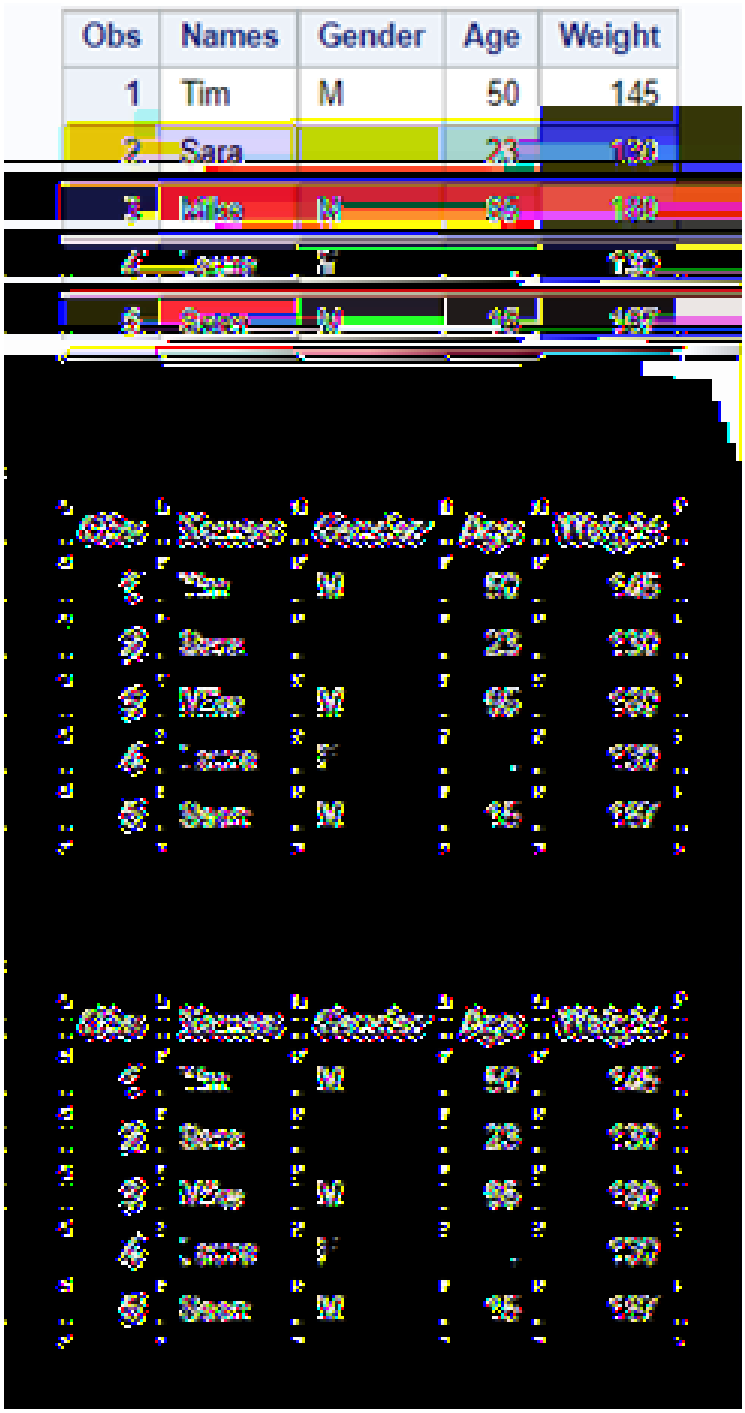
IMPORTING DATAFILE WITH INFILE STATEMENT

CODE:

```
/*reading data separated by blank*/

data sdata_blank;

 infile "/home/u59242808/Data Files/DATA_blanks.txt";

 input Names $ Gender $ Age Weight;

run;

proc print data=sdata_blank;

run;

/*reading Data separated by comma(csv file)*/

Data sdata_commas;

 infile "/home/u59242808/Data Files/DATA_commas.csv" dsd;

 input Names $ Gender $ Age Weight;

run;

proc print data=sdata_commas;
```

```
run;

/*reading data separated by other deliminators like colon in this case*/

data sdata_otherdel;

 infile "/home/u59242808/Data Files/other_del_data.txt" dlm= ":";

 input Names $ Gender $ Age Weight;

run;

proc print data= sdata_otherdel;

run;
```
Copy

OUTPUT:



Obs	Names	Gender	Age	Weight
1	Tim	M	50	145
2	Sara		23	130
3	Mike	M	85	180
4	Jessie	F		130
5	Steve	M	15	167

EXPLANATION:

· At frst, we give the datafle name we want to give the data for this particular notebook as **data** name.

· Next, we write **infile "datafile location"** to import the datafle.

· In case of this datafles imported here the variables names were not defned in the text/csv fles. So, we used **input** and wrote the variable names one by one.

· In case of character type variables we have to use $ sign after the variable name to defne it as character type and in case of numeric values we don't have to put any sign after the variable names.

· After that, when we imported the same datafle from a csv fle instead of a text fle we had to include **dsd** after the infle "location" statement. This dsd specifes that when data values are enclosed in quotation marks, delimiters within the value and are treated as character data. The DSD option changes how SAS treats delimiters when you use LIST input and sets the default delimiter to a comma. When we specify DSD, SAS treats two consecutive delimiters as a missing value and removes quotation marks from character values.

· In case of importing a text fle where the values are separated with any delimiters like semi-colon, comma, colon, or any others then we simply have to defne the delimiter like **dlm = " : "** after the infle "location" statement.

IMPORTING A TEXT FILE DATASET WHERE FIRST ROW IS HEADER BY BOTH INFILE AND PROC PRINT STATEMENT

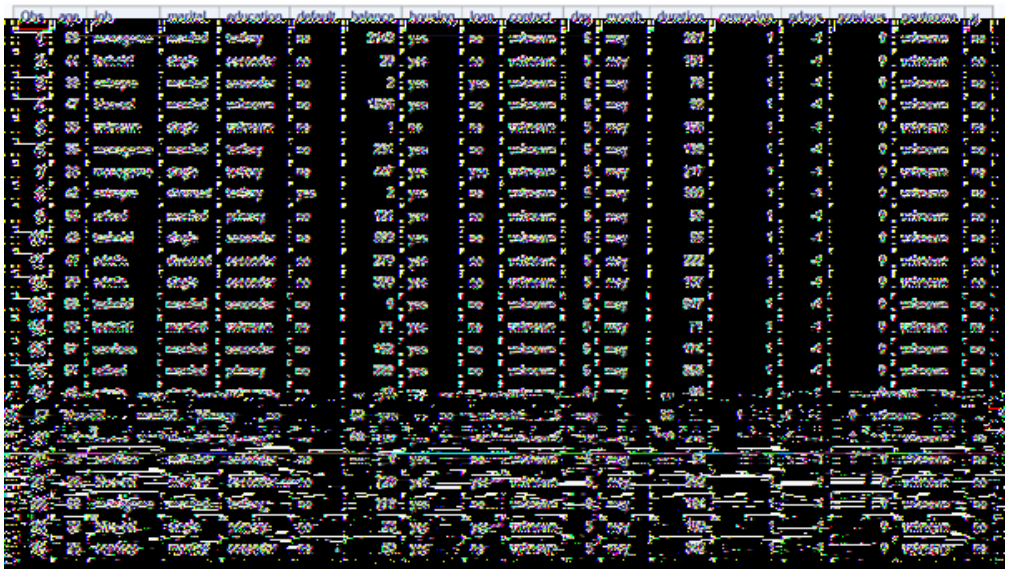
CODES:

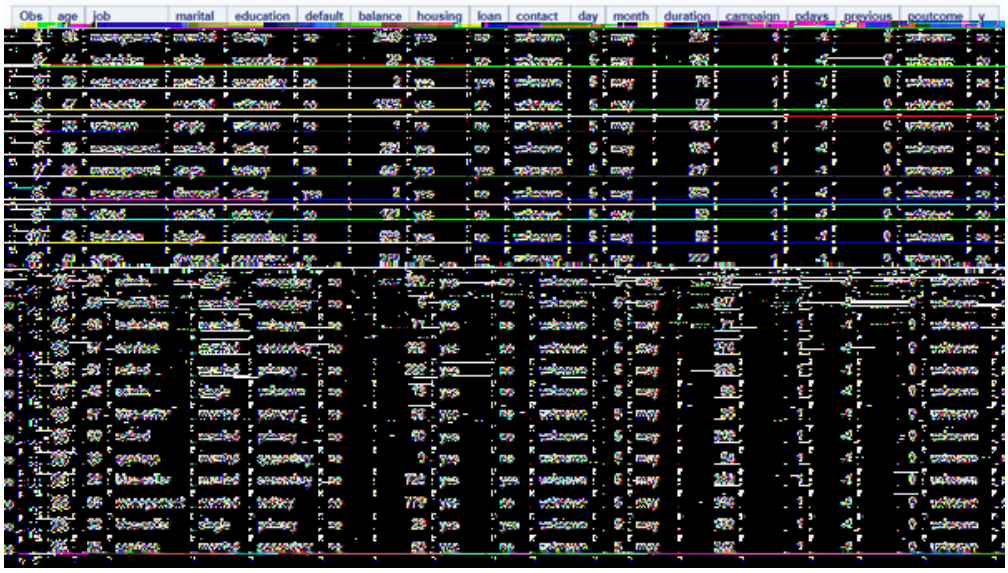
```
/*import txt file with data infile statement(blank delimiter), first row as header*/

data bank_full_blank;

 infile "/home/u59242808/Data Files/Bank_full_Blank.txt" firstobs= 2

 delimiter= " ";

 input age job $ marital $ education $ default $ balance housing $ loan $ contact $ day mont

run;

/*import txt file with proc import (blank delimiter) first row as header*/

proc import datafile="/home/u59242808/Data Files/Bank_full_Blank.txt"
```

```
 out=data_full_blank_2_sasdata dbms=dlm replace;

 delimiter=" ";

 getnames=yes; /*including first line as heading in the data*/

run;
```

Copy

OUTPUT:





EXPLANATION:

· In case of comment lines inside the codes we can also use / *comment* / .

· In case of **infile** statement we have to defne the names of the variables ourselves and as the frst row is our header in the datafle, so we have to defne **firstobs = 2** so that the sas environment takes the values of the variables from the second row ignoring the frst row.

· As there is no delimiter used, we will simply defne **delimiter = " "** putting a blank inside the quotations.

· Then we will have to manually put the variable names in the input statement.

· In case of importing the same fle by proc print statement, we will just simply put a code **getnames = yes;** and the sas environment will understand and take that frst line as heading or simply variable names.

READING DATAFILES IN DIFFERENT FORMAT

CODE:

/* Reading datafles in diferent format using informat statement

SAS has diferent formats but we'll work with only two kinds

i. standard numeric which is in the format w.d

ii. character value which is in the format $w.

```
eg, in w.d format, 145 kg is 3. format and 2.1 is 3.1 format*/
data sdata_column;
infle "/home/u59242808/Data Files/DATA_column.txt";
input
@1 Name $5.
@6 Gender $1.
@7 Weight 3.
@10 DOB mmddyy10.;
run;
/ 1 Jan 1960 = 1, 2 Jan 1960 = 2 and so on for date format/
/ display DDMonth YYYY format /
proc print data= sdata_column;
format DOB date9.;
run;
/ display MM/DD/YYYY format /
proc print data= sdata_column;
format DOB mmddyy10.;
run;
```

OUTPUT:

Obs	Name	Gender	Weight	DOB
1	Tim	M	145	21OCT1978
2	Sara		130	20SEP1964
3	Mike	M	180	23NOV1965
4	Laura	F	130	06NOV1980
5	Sean	M	167	07APR2000

Obs	Name	Gender	Weight	DOB
1	Tim	M	145	10/21/1978
2	Sara		130	09/20/1964
3	Mike	M	180	11/23/1965
4	Laura	F	130	11/06/1980
5	Sean	M	167	04/07/2000

EXPLANATION:

· During importing a fle through infle statement from a text fle, if there is no blanks or space or delimiters or the space between one value from the next one is not equal but the variable values are are in same space length then we can read the datafle in **informat statement.**

For eg, if 1-5 available space uses for Name variable and 6th space is available for gender, 7th-9th space takes weight and from 10th till end it takes dob then

we can define them just by using **@1 variable_name $5.** This @1 define that this particular variable will take value from 1st place and $sign will define that it is a character variable and this 5. Will define that it will take 5 places from 1 till 5.

· **Similarly @6 Gender $1.** Will define that Gender variable will start taking values from 6th space till only 1 space from that which means only 6th space and it will be a character type variable.

· **Date** has many formats in SAS environment.

Normally if we don't define any format it will take 1st January 1960 as 1, 2nd January 1960 as 2 and so on for date format.

But, we can change it according to our needs.

Like, here we used 2 formats.

i) **Format DOB date9.** which will show us the date as 01JAN1960 or 11NOV2021 like that.

ii) Format DOB mmddyy10. which will show us the date as 01/01/1960 or 11/11/2021 like that.

IMPORTING DIFFERENT SHEETS FROM EXCEL TO SAS

CODE:

```
/ Importing different sheet other than the first one from Excel to Sas/
proc import datafle= "/home/u59242808/Data Files/excel_data.xlsx"
out= dataset1 dbms= XLSX replace;
sheet= "name_class";
run;
/ Importing some selected ranges of a sheet from Excel to SAS/
proc import datafle= "/home/u59242808/Data Files/excel_data.xlsx"
out= dataset1 dbms= XLSX replace;
sheet= "name_class";
range= "A1:B8";
run;
proc print data= dataset1;
run;
```

/ *by using noobs we are ignoring the obs no column*/

```
proc print data= dataset1 noobs;
run;
```

OUTPUT:

Obs	name	class
1	Tim	D
2	Sharon	D
3	Sean	G
4	Mike	D
5	Tina	D
6	Sara	
7	Severnes	G

EXPLANATION:

· Here, we imported the datafle by the same process and just added an extra code line which is

Sheet= "name_class";

By this way, no matter how many sheets are there in that particular excel fle, only the sheet named as name_class will be imported.

· Similarly, if we want to take only some particular values/observations from a particular sheet of the excel fle we will simply defne the range we want to take like

Range= "A1:B8" like that.

· No matter the dataset, sas environment always put observation number by default in the imported dataset. If we want to ignore that observation number column from our output we will add a command **noobs** after the proc print data= dataset_name. That way, when we print our results at the end, the observation number column is not printed.

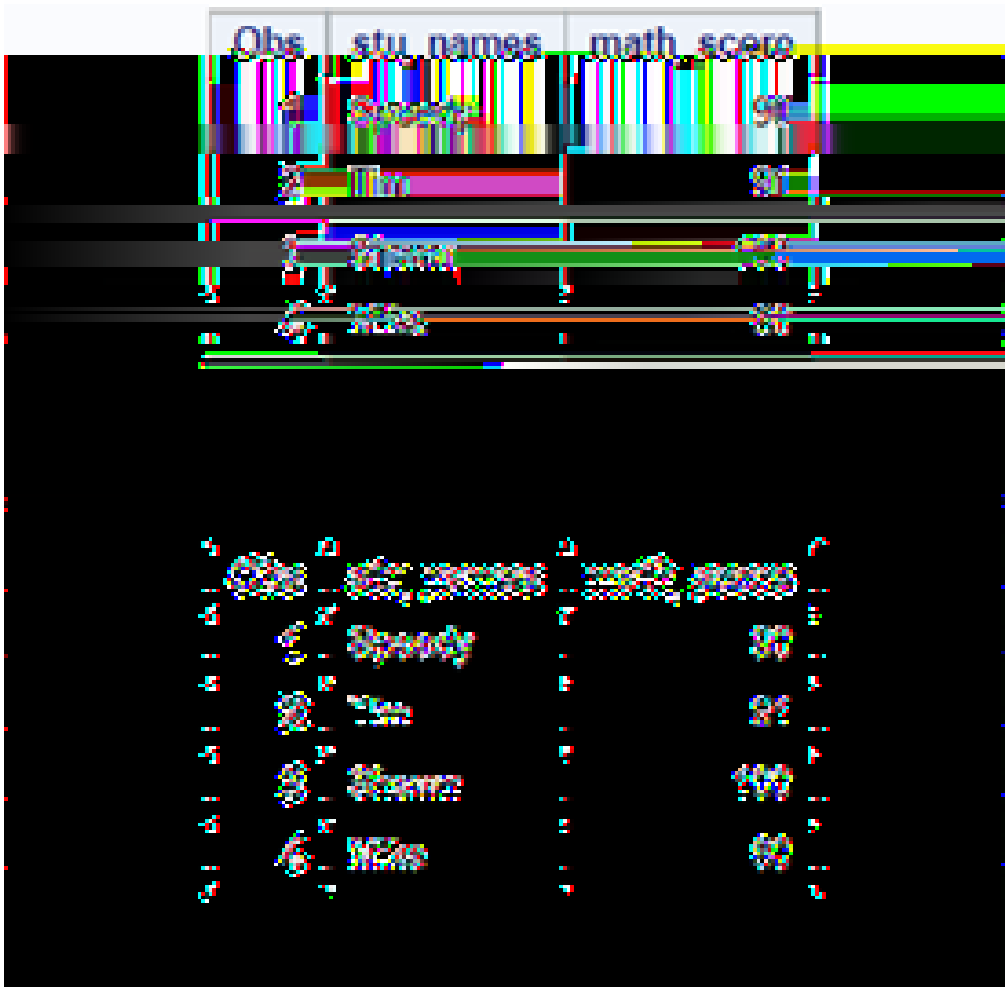
CREATE A DATAFILE

CODE:

```
/ Create datafile/

libname scores "/home/u59242808";

/ Permanent Data File/

data scores.student_math;

input stu_names $ math_score;

datalines;

Speedy 90

Tim 91

Shawn 100

Mike 60

;

run;

proc print data= scores.student_math;

run;

/ Temporary data file (data go away the next time i login)/
```

```
data student_math2;
input stu_names $ math_score;
datalines;
Speedy 90
Tim 91
Shawn 100
Mike 60
;
run;
proc print data = student_math2;
run;
```

OUTPUT:



EXPLANATION:

· We can create a datafle by ourselves in SAS environment as well

i) Permanent datafle

ii) Temporary datafle.

· In case of creating a permanent datafle we will have to save the datafle in a library.

Here, I saved it in my **scores** library by and saved the datafle as **student_math** by using the code data scores.student_math.

· Then, I have to put the variable names as input

· Then, by defning datalines, we will just simply put the values and observations.

· In case of a temporary datafle, we don't have to defne any library name and we can simply just start by defning data dataset name as **data student_math2** and then put the input variable names and observations.

SAS function

CODE:

```
proc import datafle= "/home/u59242808/Data Files/score_data.xlsx"
out= score_data1 dbms= XLSX replace;
run;
/* converting Gender variable from small to upper letter
another function is used to fnd out the length of name variable observations*/
data score_data_new;
set score_data1;
gender_func = upcase(gender);
length_name = length(name);
run;
proc print data= score_data_new;
run;
```

OUTPUT:

Obs	name	score1	score2	score3	gender	gender_func	length_name
1	Tim	88	80	99	m	M	3
2	Sharon	95	90	90	f	F	6
3	Sean	45	78	56	m	M	4
4	Mike	89	.	77	m	M	4
5	Tina	54	69	69	f	F	4

EXPLANATION:

· At frst we imported the data.

· Then we saw that the gender variable was in lowercase we want the uppercase values and we also need the length of the names.

· So, we created a new variable as **gender_func** and used the upcase statement to convert the alphabets present in the gender to uppercase using the statement

Gender_func = upcase(gender);

· We also used length statement to fnd out the length of the name observations in our data and put the output in a new variable by the name length_name like

Length_name = length(name);

IF-THEN STATEMENT

CODE:

/ *using if-then statement*/

data score_data1_ifthen;

set score_data1;

if gender = ”m” then gender_num = 1;

else gender_num = 2;

run;

proc print data= score_data1_ifthen;

run;

OUTPUT:

Obs	name	score1	score2	score3	gender	gender_func	length_name
1	Tim	88	80	99	m	M	3
2	Sharon	95	90	90	f	F	6
3	Sean	45	78	56	m	M	4
4	Mike	89	.	77	m	M	4
5	Tina	54	69	69	f	F	4

EXPLANATION:

· We used the previous dataset here and that's why we didn't have to import the datafle again here.

· Here we want to convert the gender m-f values to binary form.

So, we used if then statement here and made a new variable as gender_num which will give us value 1 if gender is mail and 2 if gender is female.

· By using if then statement we wrote here if gender = "m" then gender_num = 1 else gender_num=2 and then printed the output.

IF ELSE THEN STATEMENT

CODE:

proc import datafle= "/home/u59242808/Data Files/score_data_miss.xlsx"

out= score_data_miss dbms= xlsx replace;

run;

data score_data_miss_cond;

set score_data_miss;

```
if gender = ”m” then gender_num = 1;
else if gender = ”f” then gender_num = 2;
else gender_num = .;
avg_marks = mean(score1, score2, score3);
run;
proc print data= score_data_miss_cond;
run;
data score_data_miss_cond2;
set score_data_miss_cond;
length take $12;
if avg_marks >= 90 then do
grade = ”A”;
result = ”Pass”;
end;
else if avg_marks >= 80 then do
grade = ”B”;
result = ”Pass”;
end;
else if avg_marks >= 70 then do
grade = ”C”;
result = ”Pass”;
end;
else if avg_marks >= 60 then do
grade = ”D”;
result = ”Pass”;
end;
else if 0 <= avg_marks < 60 then do
grade = ”F”;
result = ”Fail”;
end;
else do
```

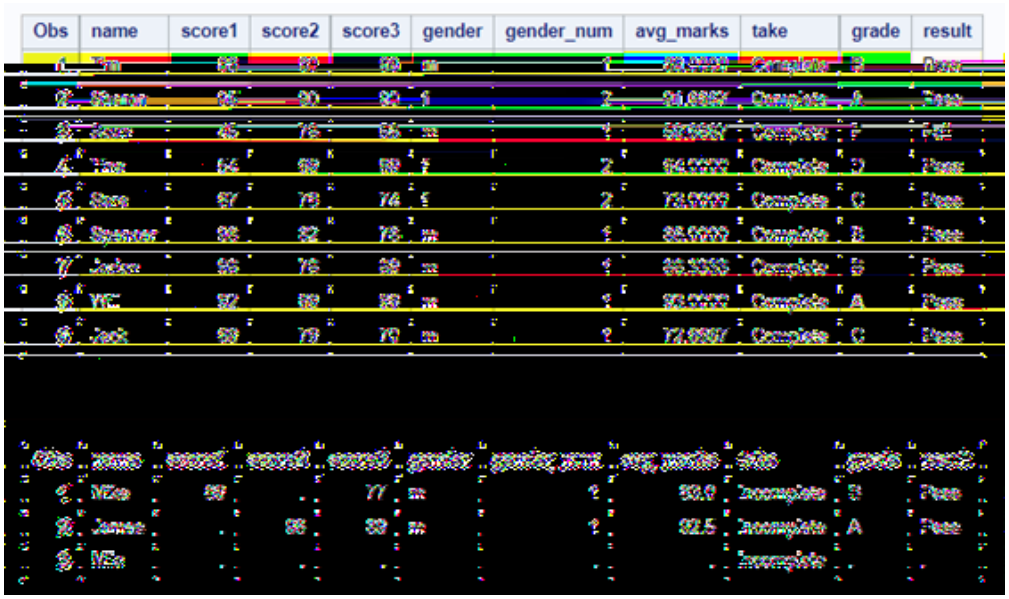
```
grade = ".";
result = ".";
end;
/ create two variables i) complete when all the marks are given and ii) pass if the average score is >= 60/
if score1 ~=. and score2 ~=. and score3 ~=. then take = "Complete";
else take = "Incomplete";
run;
proc print data= score_data_miss_cond2;
run;
/ Divide the file into two parts using if statement where part 1= who complete the assignment and 2= incomplete/
data subset_1;
set score_data_miss_cond2;
if take = "Complete";
run;
data subset_2;
set score_data_miss_cond2;
if take = "Incomplete";
run;
proc print data= subset_1;
proc print data= subset_2;
run;
/ Divide the file into two parts using delete statement where part 1= who complete the assignment and 2= incomplete/
data subset_3;
set score_data_miss_cond2;
if take = "Incomplete" then delete;
run;
data subset_4;
set score_data_miss_cond2;
if take = "Complete" then delete;
```

```
run;
proc print data = subset_3;
proc print data = subset_4;
run;
```

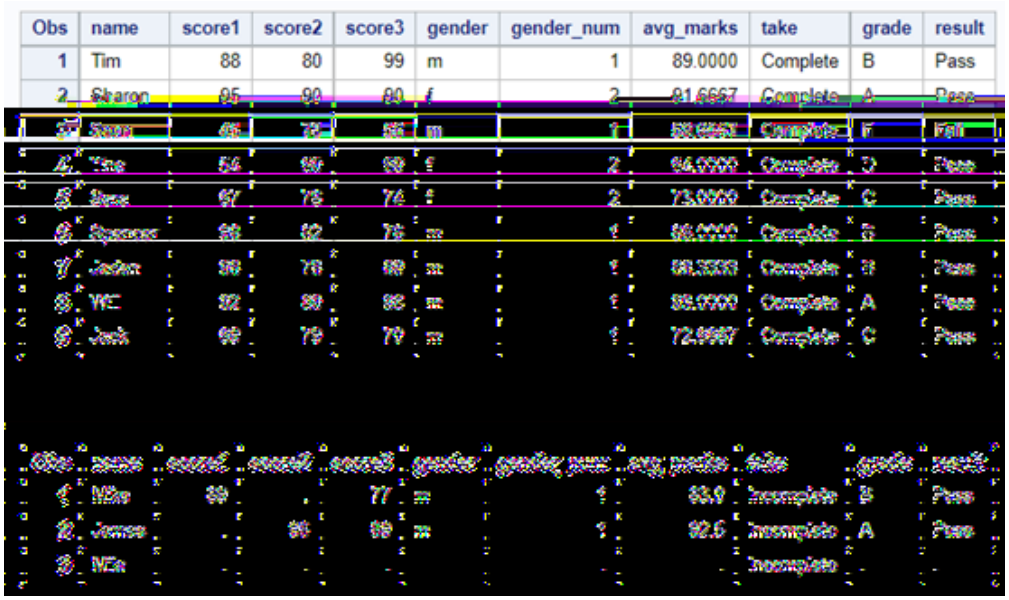
OUTPUT:

Obs	name	score1	score2	score3	gender	gender_num	avg_marks
1	Tim	88	80	99	m	1	89.0000
2	Sharon	95	90	90	f	2	91.6667
3	Sean	45	78	56	m	1	59.6667
4	Mike	89	.	77	m	1	83.0000
5	Tina	54	69	69	f	2	64.0000
6	Sara	67	78	74	f	2	73.0000
7	Spencer	55	82	78	m	1	85.
8	James	95	89	89	m	1	92.
9	Jaden	88	78	83	m	1	83.
10	WE	92	89	96	m	1	92.
11	Jack	69	79	76	m	1	74.
12	WEe						

Obs	name	score1	score2	score3	gender	gender_num	avg_marks	take	grade	result
1	Tim	88	80	99	m	1	89.0000	Complete	B	Pass
2	Sharon	95	90	90	f	2	91.6667	Complete	A	Pass
3	Sean	45	78	56	m	1	59.6667	Complete	F	Fail
4	Mike	89	.	77	m	1	83.0000	Incomplete	B	Pass
5	Tina	54	69	69	f	2	64.0000	Complete	D	Pass
6	Sara	67	78	74	f	2	73.0000	Complete	C	Pass
7	Spencer	55	82	78	m	1	85.0000	Complete	B	Pass
8	James	95	89	89	m	1	92.0000	Incomplete	A	Pass
9	Jaden	88	78	83	m	1	83.3333	Complete	B	Pass
10	WE	92	89	96	m	1	92.0000	Complete	A	Pass
11	Jack	69	79	76	m	1	74.6667	Complete	C	Pass
12	WEe							Incomplete		



Obs	name	score1	score2	score3	gender	gender_num	avg_marks	take	grade	result
1	Tim	88	80	99	m	1	89.0000	Complete	B	Pass
2	Sharon	95	90	90	f	2	91.6667	Complete	A	Pass
3	Sam	45	75	85	m	1	68.3333	Complete	F	Fail
4	Tina	56	60	60	f	2	58.6667	Complete	F	Fail
5	Sam	67	75	74	f	2	72.0000	Complete	C	Pass
6	Spencer	88	82	75	m	1	81.6667	Complete	B	Pass
7	Jaden	66	75	69	m	1	70.0000	Complete	C	Pass
8	Will	92	89	88	m	1	89.6667	Complete	A	Pass
9	Jack	69	75	79	m	1	74.3333	Complete	C	Pass
10	Mike	89		77	m	1	83.0	Incomplete	B	Pass
11	James		85	99	m	1	92.5	Incomplete	A	Pass
12	Max							Incomplete		



Obs	name	score1	score2	score3	gender	gender_num	avg_marks	take	grade	result
1	Tim	88	80	99	m	1	89.0000	Complete	B	Pass
2	Sharon	95	90	90	f	2	91.6667	Complete	A	Pass
3	Sam	45	75	85	m	1	68.3333	Complete	F	Fail
4	Tina	56	60	60	f	2	58.6667	Complete	F	Fail
5	Sam	67	75	74	f	2	72.0000	Complete	C	Pass
6	Spencer	88	82	75	m	1	81.6667	Complete	B	Pass
7	Jaden	66	75	69	m	1	70.0000	Complete	C	Pass
8	Will	92	89	88	m	1	89.6667	Complete	A	Pass
9	Jack	69	75	79	m	1	74.3333	Complete	C	Pass
10	Mike	89		77	m	1	83.0	Incomplete	B	Pass
11	James		85	99	m	1	92.5	Incomplete	A	Pass
12	Max							Incomplete		

EXPLANATION:

· Here, at frst we imported the datafle which is the frst output.

· Then, we used if then statement to convert gender to binary in a new variable **gender_num** and made another variable avg_marks which implies the average of score1, score2, score3 and printed the frst output.

· Then, we made 3 new variables take, grade and result.

We take the length of take variable as 12 as by default it takes only 8 length.

We defned if avg_marks >= 90 then grade will show A and result will show pass

If avg_marks >= 80 then grade will show B and result will show pass and so on.

If avg_marks lies between 0 to 60 then grade will show F and result will show Fail and at last, when there is no avg_marks given, then grade and result will also show blank.

Later, in case of take variable where there is atleast 1 missing value in score1, score2, score3, take will show incomplete and if for any student, all 3 marks are given, it will show complete.

And this result is shown in output2.

· Then we made 2 subset of this output by 2 diferent processes where 1 subset will show all the student who has take = complete and another output will show who has take = incomplete.

i) We simply took the dataset and checked if take = complete given, then it will be printed in the subset1.

ii) If take = incomplete, then those observations will be printed on subset2

iii) Then again we took the entire dataset and checked if take = incomplete then deleted those values and saved it as subset3.

iv) Lastly, if take = complete, then deleted those observations and saved it as subset4.

1D ARRAY

CODE:

- ARRAY;

```
proc import datafle= "/home/u59242808/Data Files/score_data_miss999.xlsx"
out= score_data_miss999 dbms=xlsx replace;
run;
proc print data = score_data_miss999;
run;
```

- One-dimensional ARRAY, converting the 999 values to period(. / space);

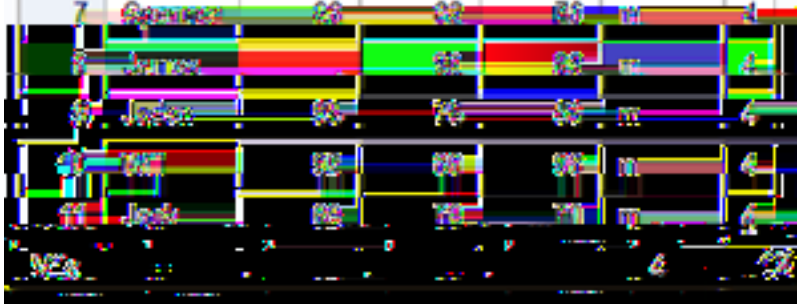
```
data dataset1;
```

```
set score_data_miss999;
array score_var(3) score1 score2 score3;
do i = 1 to 3;
if score_var(i) = 999 then score_var(i) = .;
end;
run;
proc print data = dataset1;
run;
```

OUTPUT:

Obs	name	score1	score2	score3	gender
1	Tim	88	80	99	m
2	Sharon	95	90	90	f
3	Sean	45	78	56	m
4	Mike	89	999	77	m
5	Tina	54	69	69	f
6	Sara	67	78	74	f
7	Spencer	98	82	78	m
8	James	999	96	89	m
9	Jaden	98	78	89	m
10	Will	92	89	98	m
11	Jack	69	79	70	m
12	Mila	999	999	999	

Obs	name	score1	score2	score3	gender	i
1	Tim	88	80	99	m	4
2	Sharon	95	90	90	f	4
3	Sean	45	78	56	m	4
4	Mike	89	.	77	m	4
5	Tina	54	69	69	f	4
6	Sara	67	78	74	f	4
7	[illegible]					



EXPLANATION:

· At frst, we imported the datafle.

· As we can see that there are some 999 values there which are wrong and we want to convert those to . null values and we used 1Dimensional array here for that purpose.

· We defned a array as score_var here which has 3 variables score1, score2, score3.

· Then for every ith variable of score_var we checked if any value is 999 or not and if it is 999 then we changed it to "." (null value) and if it is not 999 then we stopped for that particular value and checked the next one.

GIVING TITLE TO A DATASET

CODE:

```
proc import datafle= "/home/u59242808/Data Files/score_data_miss777.xlsx"
dbms = xlsx out = score_data0 replace;
run;
proc print data= score_data0;
title "missing values are shown as 777";
run;
```

OUTPUT:

missing values are shown as 777

Obs	name	score1	score2	score3	gender
1	Tim	88	80	99	m
2	Sharon	95	90	90	f
3	Sean	45	78	56	m
4	Mike	89	777	77	m
5	Tina	54	69	69	f
6	Sara	67	78	74	f
7	Spencer	98	82	78	m
8	James	777	96	89	m
9	Jaden	98	78	89	m
10	Will	92	89	777	m
11	Jack	69	79	70	m
12	Mila	.	.	.	

EXPLANATION:

· Here, the dataset is imported through proc print statement.

· Then while print the output data we used a **title** code after the proc print data statement and before running it. That way we can give any title to any dataset.

ADDING LABELS TO THE VARIABLES

· We generally add labels for listing and reporting purpose.

· Syntax for label:

label var_name = ”Variable name”;

or

Obs	name	Math score	Stats score	English score	gender
1	Tim	88	80	99	m
2	Sharon	95	90	90	f
3	Sean	45	78	56	m
4	Mike	89	.	77	m
5	Tina	54	69	69	f
6	Sara	67	78	74	f
7	Spencer	98	82	78	m
8	James	.	96	89	m
9	Jaden	98	78	89	m
10	Will	92	89	98	m
11	Jack	69	79	70	m

Obs	Students name	Math score	Stats score	English score	gender
1	Tim	88	80	99	m
2	Sharon	95	90	90	f
3	Sean	45	78	56	m
4	Mike	89	.	77	m
5	Tina	54	69	69	f
6	Sara	67	78	74	f
7	Spencer	98	82	78	m
8	James	.	96	89	m
9	Jaden	98	78	89	m
10	Will	92	89	98	m
11	Jack	69	79	70	m

Obs	name	score1	score2	score3	gender
1	Tim	88	80	99	m
2	Sharon	95	90	90	f
3	Sean	45	78	56	m
4	Mike	89	.	77	m
5	Tina	54	69	69	f
6	Sara	67	78	74	f
7	Spencer	98	82	78	m
8	James	.	96	89	m
9	Jaden	98	78	89	m
10	Will	92	89	98	m
11	Jack	69	79	70	m

Obs	student name	Math score	Stats score	English score	gender
1	Tim	88	80	99	m
2	Sharon	95	90	90	f
3	Sean	45	78	56	m
4	Mike	89	.	77	m
5	Tina	54	69	69	f
6	Sara	67	78	74	f
7	Spencer	98	82	78	m
8	James	.	96	89	m
9	Jaden	98	78	89	m
10	Will	92	89	98	m
11	Jack	69	79	70	m

EXPLANATION:

· At frst, we imported the data.

· Then we used the label syntax for sas in our program and gave the label for Math score, Stats score, English score for score1, score2, score3 respectively through the datastep and printed it keeping in mind that we have to add LABEL statement in while printing the dataset.

· Later, we gave the label for name as Students name and printed it.

· Later, we only printed the data not using the LABEL statement in the proc print statement and got the original variable names in our output.

· We can also split the label name in two different lines in the variable column by defning some concatenation as label split and using it in the label name code itself. We used *as concatenation here and used label name = student*name and got student and name splitted in two different lines in the variable name space.

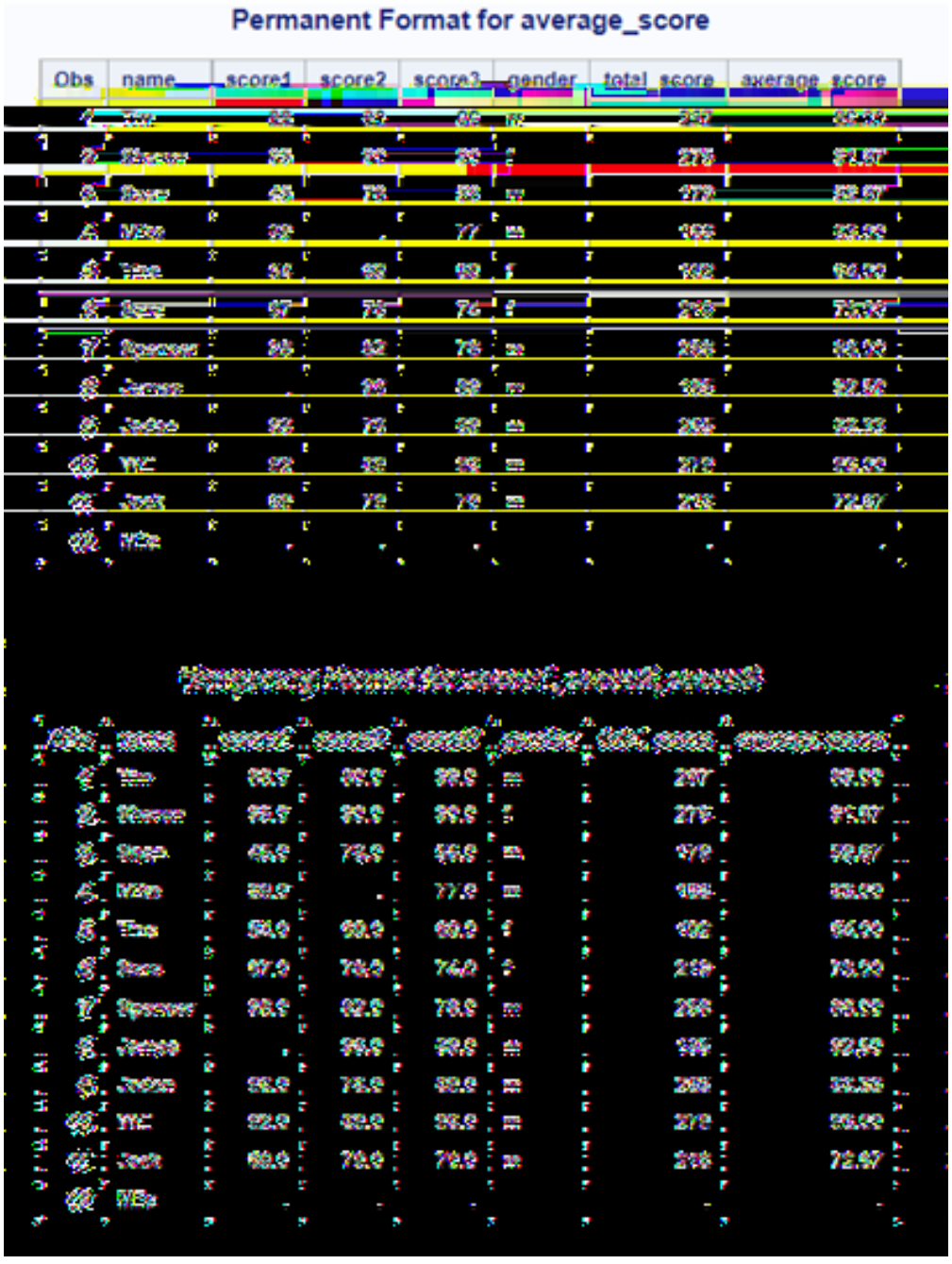
ASSIGNING FORMAT INTO VARIABLE

CODE:

```
proc import datafle= "/home/u59242808/Data Files/score_data_miss.xlsx"
out = data0 dbms = xlsx replace;
run;
data dataset1;
set data0;
total_score = sum(score1, score2, score3);
average_score = mean(score1, score2, score3);
run;
data dataset2;
set dataset1;
format average_score 5.2;
run;
proc print data = dataset2;
title "Permanent Format for average_score";
run;
/ Using temporary format in proc print statement /
proc print data= dataset2;
format score1 4.1;
format score2 4.1;
```

```
format score3 4.1;
title ”Temporary Format for score1, score2, score3”;
RUN;
```

OUTPUT:



EXPLANATION:

· At frst, we imported the dataset in SAS environment.

· Later, from score1, score2, score3 we made 2 new variables as total_score and average_score which defnes the sum and mean of these 3 score values respectively.

· There we got some average values like 91.66666667, 59.6666667 like that and that's why we used formatting here.

· At frst, we permanently formatted the average_score in 5.2 rule by which it will take only 2 decimal places and can take 5 values before the point and printed the frst dataset with a title permanent format for average_score.

· Later, we temporary formatted the 3 scores available (score1, score2, score3) in our dataset by 4.1 rule by which it can take 4 number values before point and can take only 1 decimal place and printed it with the title temporary format for score1, score2, score3.

USER DEFINE FORMAT

CODE:

```
proc import datafle= '/home/u59242808/Data Files/score_data_miss.xlsx'
out= score_data0 dbms= xlsx replace;
run;
data score_data1;
set score_data0;
total_score = sum(score1, score2, score3);
average_score = mean(score1, score2, score3);
run;
proc format;
value $genderf 'f' = 'female'
'm' = 'male';
/ this $genderf variable will be created having the values female and male from f and m respectively/
value as_group 0 - <60 = 'F'
60 - <70 = 'D'
70 - <80 = 'C'
80 - <90 = 'B'
90 - High = 'A'
```

other = 'Missing';

run;

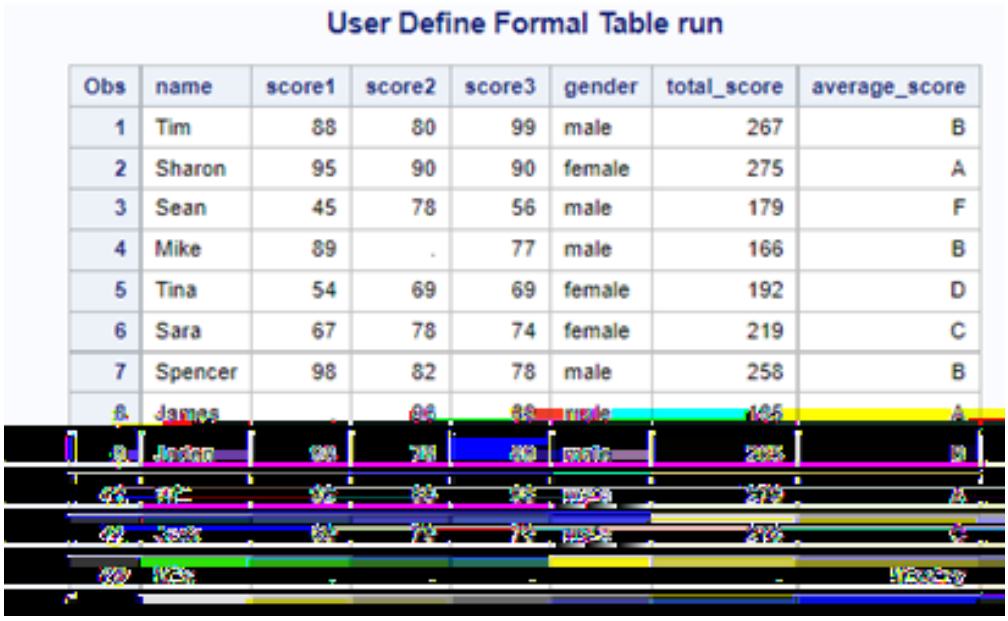
proc print data = score_data1;

format gender $genderf. average_score as_group.;

title "User Defne Formal Table"

run;

OUTPUT:



User Define Formal Table run

Obs	name	score1	score2	score3	gender	total_score	average_score
1	Tim	88	80	99	male	267	B
2	Sharon	95	90	90	female	275	A
3	Sean	45	78	56	male	179	F
4	Mike	89	.	77	male	166	B
5	Tina	54	69	69	female	192	D
6	Sara	67	78	74	female	219	C
7	Spencer	98	82	78	male	258	B
8	James						

EXPLANATION:

· At frst, we imported the dataset and fnd out the total score and average score in numerical values.

· Now, we had 'm' and 'f' values in gender variable and numerical mean values in average_score variable.

· We used proc format here and defned gender f value as female and m value as male.

· Then we took value from average_score and took them in group values like if the average_score is between 0 to 60 then it will return F, if its between 60 to 70 it will return D, like that we converted the average scores to grades and if there is any null value in average_score it will return missing.

· Then we printed our data with proc print statement and used format statement to change these given gender values to formatted genderf character values and

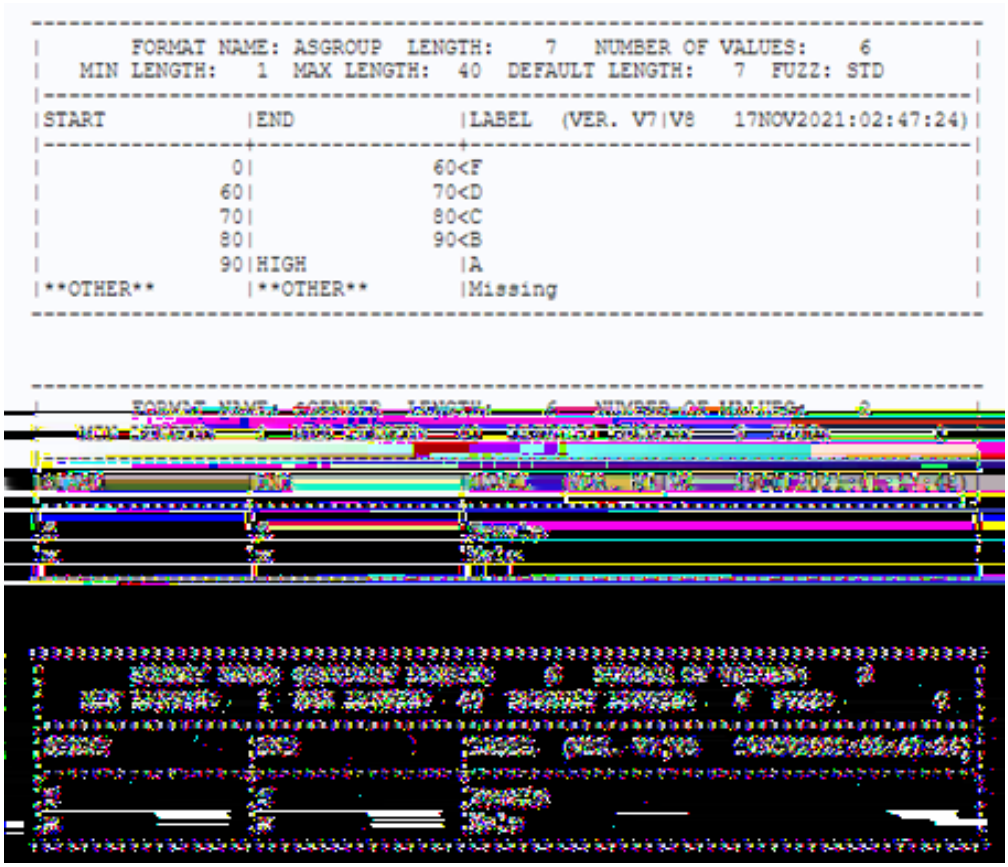
changed average_score to as_group which we defned and also gave a title to our dataset and printed it.

STORING AND REFERING AND USING USER DEFINE FORMAT

CODE:

```
libname myfmts "/home/u59242808/SAS_code";
proc format library= myfmts;
value $genderf 'm' = 'Male'
'f' = 'Female';
value asgroup 0-<60 = 'F'
60-<70 = 'D'
70-<80 = 'C'
80-<90 = 'B'
90- High = 'A'
other = 'Missing';
run;
proc format library= myfmts fmtlib;
run;
proc import datafle= "/home/u59242808/Data Files/score_data_miss.xlsx"
out = scoredata0 dbms= xlsx replace;
run;
data scoredata1;
set scoredata0;
total_score = sum(score1, score2, score3);
avg_score = mean(score1, score2, score3);
libname myfmts "/home/u59242808/SAS_code";
option fmtsearch = (myfmts work library);
proc print data= scoredata1;
format gender $genderf. avg_score asgroup.;
run;
```

OUTPUT:



FORMAT NAME: ASGROUP LENGTH: 7 NUMBER OF VALUES: 6
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 7 FUZZ: STD

START	END	LABEL (VER. V7|V8 17NOV2021:02:47:24)
0	60<F	
60	70<D	
70	80<C	
80	90<B	
90	HIGH	A
OTHER	**OTHER**	Missing

Obs	name	score1	score2	score3	gender	total_score	avg_score
1	Tim	88	80	99	Male	267	B
2	Sharon	95	90	90	Female	275	A
3	Sean	45	78	56	Male	179	F
4	Mike	89	.	77	Male	166	B
5	Tina	54	69	69	Female	192	D
6	Sara	67	78	74	Female	219	C
7	Spencer	98	82	78	Male	258	B
8	James	.	96	89	Male	185	A
9	Jaden	98	78	89	Male	265	B
10	Will	92	89	98	Male	279	A
11	Jack	69	79	70	Male	218	C
12	Mila	.	.	.		.	Missing

EXPLANATION:

· If we have a lot of datasets and there are some primary formatting which has to been done in all of these datasets, we can use this method of storing the user defne format and can use for all of the datasets in very simple 2 steps and don't have to defne again and again.

· For this we have to call out our library to store the data and have to make a library for storing it.

· Then we can defne the values and grades as we did previously.

· Then we printed the stored data in which we can check what exactly we defned.

· Later, we imported a data and fnd out total and mean of the dataset.

· Later, we called out the data where we stored our user defned format and refered to it.

· Then we printed our data and just used the format statement so that sas environment can understand exactly in which dataset we want to use the use defne format and for exactly which variables we want to use it and got our output.

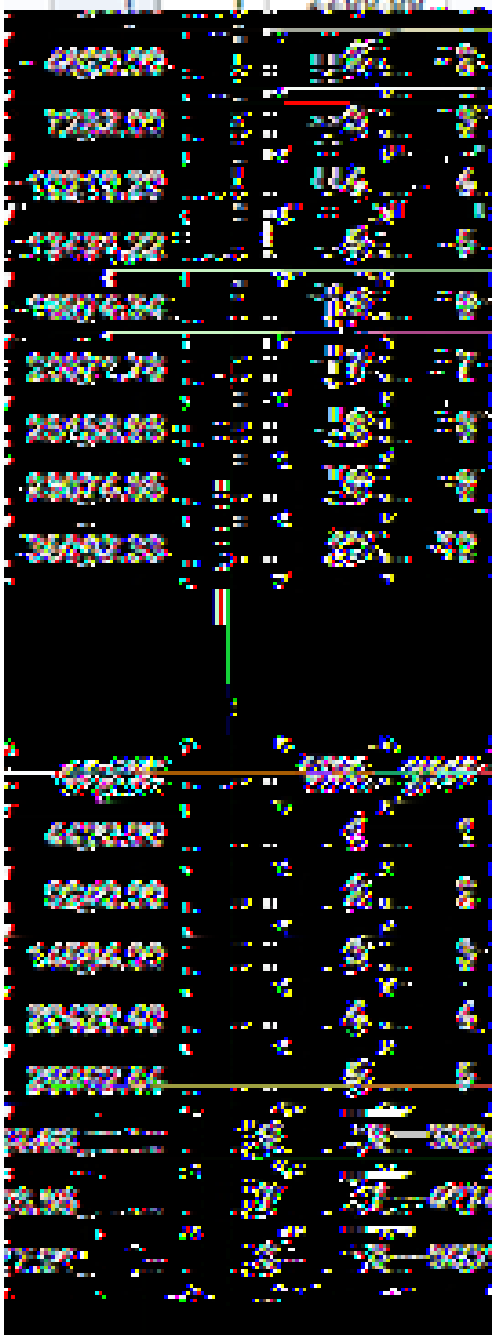
USING CONDITIONAL CLAUSE WHILE ITTERATING DO-WHILE STATEMENT

CODE:

```
sas
/ Using conditional clause while itterating do-while statement /

/* In this dataset suppose we want to limit the number of years to invest in the capital to 10 years

and add the UNTIL expression to determine years it take for investment to reach 50 thousand to control the

number of years */

data invest;

do year = 1 to 10 until (capital >= 50000);

capital + 2000;

capital + capital 0.10; / if the interest rate is 10% */

output;

end;

/ if year = 11 then year = 10; /

run;

data invest;

do year = 1 to 10 until (capital >= 50000);

capital + 4000;

capital + capital 0.10; / if the interest rate is 10% */

output;

end;

if year = 11 then year = 10;

run;
```

OUTPUT:

Obs	year	capital
1	1	2200.00



EXPLANATION:

· Here we tried to fnd out at 10% interest rate after investing some exact amount each year + gained amount how long will it take to reach that capital amount to 50000 in maximum of 10 years time.

· Here at frst out investment amount was 2000 each year and we can see that we couldn't make 50000 at the end of 10th year and it seems like we need more time.

· So, later we tried to invest 4000 each year and got the result with the help of do-while loop that we can gain 50000 capital plus some amount at the end of 8th year and don't have to wait till 10th year.

CREATING VARIABLES IN ARRAY STATEMENT

· array array_name {dimension}

· if variables are not specifed then new variables name will be created like array_name1, array_name2 ... till dimension specifcation

· if we want to specify the character array then syntax will be:

array array_name {5}$;

· if we don't specify the length of the character it becomes 8

· if we want to increase the length of the asssigned variable then syntax will be:

array array_name {5}$ 24;

· Assigning initial value in array & creating temporary array element Syntax:

array Goal{3} G1 G2 G3 (50 32 41)

· if array is string,

then, array string{3} $ str1-str3 ('red','green','blue')

· temporary array element which is not written in output,

array var{4}*temporary*(40 41 42 43) */

CODE:

```
proc import datafle= "/home/u59242808/Data Files/score_data_miss999.xlsx"
dbms = xlsx out = data0 replace;
run;
data score_data1(drop= i);
set data0;
```

```
array score_variable(3) score1 score2 score3;
do i = 1 to 3;
if score_variable(i) = 999 then score_variable(i) = .;
end;
average_score = mean(score1, score2, score3);
run;
proc print data= score_data1;
run;
data score_data2(drop= i);
set score_data1;
array score(3) score1 score2 score3;
array score_dif(3);
do i = 1 to 3;
score_dif{i} = score{i} - average_score;
end;
run;
proc print data= score_data2;
title ”Creating variables in an Array statement”;
run;
proc means data= score_data1;
var score1 score2 score3;
run;
data score_data3(drop= i);
set score_data1;
array score(3) score1 score2 score3;
array avg{3} (79.5 81.9 80.8); /Assigning inititial values/
array score_dif(3);
do i = 1 to 3;
score_dif{i} = score{i} - AVG{i};
end;
run;
```

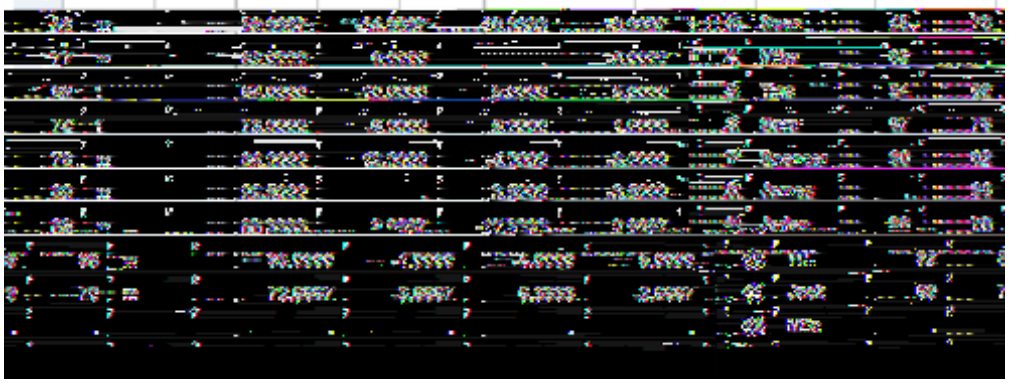
```
proc print data= score_data3 (keep= name score_dif1 score_dif2
score_dif3);
title ”Finding our X - Xbar using arrays”;
run;
```

OUTPUT:

Obs	name	score1	score2	score3	gender	average_score
1	Tim	88	80	99	m	89.0000
2	Sharon	95	90	90	f	91.6667
3	Sean	45	78	56	m	59.6667
4	Mike	89	.	77	m	83.0000
5	Tina	54	69	69	f	64.0000
6	Sara	67	78	74	f	73.0000
7	Spencer	98	82	78	m	86.0000
8	James	.	96	89	m	92.5000
9	Jaden	98	78	89	m	88.3333
10	Will	92	89	98	m	93.0000
11	Jack	69	79	70	m	72.6667
12	Mila	.	.	.		.

Creating variables in an Array statement

Obs	name	score1	score2	score3	gender	average_score	score_diff1	score_diff2	score_diff3
1	Tim	88	80	99	m	89.0000	-1.0000	-9.0000	10.0000
2	Sharon	95	90	90	f	91.6667	3.3333	-1.6667	-1.6667



Creating variables in an Array statement

The MEANS Procedure

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
score1	score1	10	79.5000000	19.2945012	45.0000000	98.0000000
score2	score2	10	81.9000000	7.7380733	69.0000000	96.0000000
score3	score3	11	80.8181818	13.3627705	56.0000000	99.0000000

Finding our X - Xbar using arrays

Obs	name	score_diff1	score_diff2	score_diff3
1	Tim	8.5	-1.9	18.2
2	Sharon	15.5	8.1	9.2
3	Sean	-34.5	-3.9	-24.8
4	Mike	9.5	.	-3.8
5	Tina	-25.5	-12.9	-11.8
6	Sara	-12.5	-3.9	-6.8
7	Spencer	18.5	0.1	-2.8
8	James	.	14.1	8.2
9	Jaden	18.5	-3.9	8.2
10	Will	12.5	7.1	17.2
11	Jack	-10.5	-2.9	-10.8
12	Mila	.	.	.

EXPLANATION:

· At frst, we imported the datafle and converted all the 999 values to missing values (.) and also found out the average score of all the students.

· Then we again used array statement and found out the difference between each score and the average (per person wise) score for those individual students.

· Then we found out the subject wise number of available values of students, average score, std deviation, minimum and maximum values of all the 3 subjects available.

· Then we again used array statement and found out the difference between each score and the average (per subject wise) score for those students.

· Also, while printing this last output we ignored to print the rest of the columns

like score1, score2, score3, gender, and person wise average score.

CREATING TEMPORARY ARRAY ELEMENTS WHICH WE DONOT NEED TO PRINT IN THE FINAL RESULT

CODE:

```
/ Creating temporary array elements which need not be printed in the final output/

proc import datafle="/home/u59242808/Data Files/score_data_miss999.xlsx"

out=data dbms=xlsx replace;

run;

data data1 (drop = i);

set data;

array score_array(3) score1 score2 score3;

do i= 1 to 3;

if score_array(i)=999 then score_array(i)= . ;

end;

average_score = mean(score1,score2, score3);

run;

data data2 (drop = i);

set data1;

array score(3) score1 score2 score3;

array average{3}temporary(79.5 81.9 80.8); /*We are assigning values to average
1 2 and 3 which is stored

temporarily and does not require in fnal output*/

/ array average{3}temporary(79.5 81.9 80.8); /

array score_dif(3);

do i = 1 to 3;

score_dif{i} = score{i} - average{i};

end;

run;
```

proc print data=data2;

title ”Assingning temporary array elements”;

run;

OUTPUT:

Assingning temporary array elements

Obs	name	score1	score2	score3	gender	average_score	score_diff1	score_diff2	score_diff3
1	Tim	88	80	99	m	89.0000	8.5	-1.9	18.2
2	Sharon	95	90	90	f	91.6667	15.5	8.1	9.2
3	Sean	45	78	56	m	59.6667	-34.5	-3.9	-24.8
4	Mike	89	.	77	m	83.0000	9.5	.	-3.8
5	Tina	54	69	69	f	64.0000	-25.5	-12.9	-11.8
6	Sara	67	78	74	f	73.0000	-12.5	-3.9	-6.8
7	Spencer	98	82	78	m	86.0000	18.5	0.1	-2.8
8	James	.	96	89	m	92.5000	.	14.1	8.2
9	Jaden	98	78	89	m	88.3333	18.5	-3.9	8.2
10	Will	92	89	98	m	93.0000	12.5	7.1	17.2
11	Jack	69	79	70	m	72.6667	-10.5	-2.9	-10.8
12	Mila	.	.	.		.	.	.	.

EXPLANATION:

· At frst we imported the dataset and found out the person wise average score in the 3 scores given.

· Then we fnd out the subject wise average score but didn’t store it permanently in the sas dataset but rather found out the diference in subject wise average score and the scores obtained by the students with the help of array statement.

MERGING TWO DATASETS (ONE TO ONE)

CODE:

/ *Combining SAS dataset (one-one mail merging)* /

/ *one-one merging* /

proc import datafle= ”/home/u59242808/Data Files/score_data_id_partial.xlsx”

out= data_a dbms= xlsx replace;

run;

proc import datafle= ”/home/u59242808/Data Files/score_data_id.xlsx”

out= data_b dbms= xlsx replace;

```
run;
data one2one;
set data_a;
set data_b;
run;
proc print data= one2one;
title ”One to One marging”;
run;
proc print data= data_a;
title ”Partial dataset”;
proc print data= data_b;
title ”Complete dataset”;
run;
```

OUTPUT:

One to One marging

Obs	name	score1	score2	gender	stu_id	score3
1	Tim	88	80	m	1	99
2	Sharon	95	90	f	2	90
3	Sean	45	78	m	3	56
4	Mike	89	.	m	4	77
5	Tina	54	69	f	5	69
6	Sara	67	78	f	6	74

Partial dataset

Obs	name	score1	score2	gender	stu_id
1	Tim	88	80	m	1
2	Sharon	95	90	f	2
3	Sean	45	78	m	3
4	Mike	89	.	m	4
5	Tina	54	69	f	5
6	Sara	67	78	f	6

Complete dataset

Obs	name	score1	score2	score3	gender	stu_id
1	Tim	88	80	99	m	1
2	Sharon	95	90	90	f	2
3	Sean	45	78	56	m	3
4	Mike	89	.	77	m	4
5	Tina	54	69	69	f	5
6	Sara	67	78	74	f	6
7	Sam	98	82	78	m	7
8	James	.	96	89	m	8
9	Jaden	98	78	89	m	9
10	Jack	69	79	70	m	11
11	Will	92	89	98	m	10

EXPLANATION:

· There are 2 datasets given.

i) Full complete dataset given.

ii) Partial dataset given, like details of only 6 students are given out of 11 and score3 variables scores are not present in the dataset.

· At frst, we imported both of these 2 datasets.

· Then just made a new dataset and set both of the datasets in there and simply we got the one-to-one merged dataset which was required here.

· For convenience, the 2 given datasets are also printed here.

CONCATENATING TWO DATASETS

· **general form of CONCATENATING (syntax):**

data <fle name>;

set <fle name1> <fle name 2>;

run;

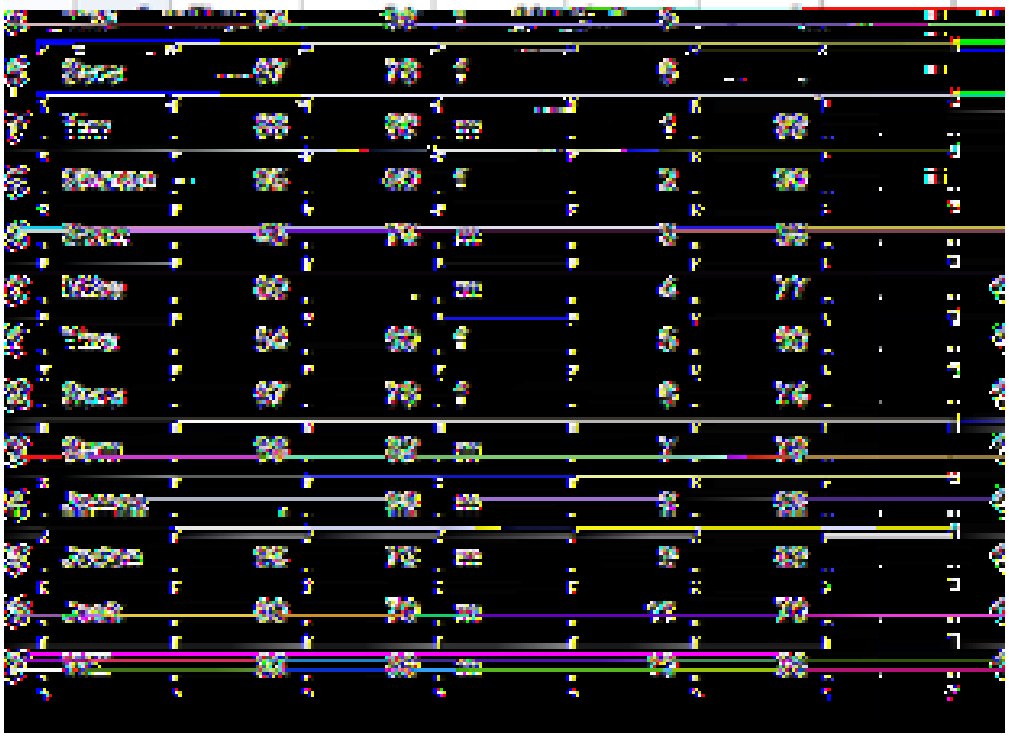
CODE:

```
/ Concatenating (combining)/
proc import datafle= "/home/u59242808/Data Files/score_data_id_partial.xlsx"
out= data_a dbms= xlsx replace;
run;
proc import datafle= "/home/u59242808/Data Files/score_data_id.xlsx"
out= data_b dbms= xlsx replace;
run;
data concat;
set data_a data_b;
run;
proc print data= concat;
title "Concatenated dataset";
run;
proc print data= data_a;
title "partial dataset";
proc print data= data_b;
title "complete dataset";
run;
```

OUTPUT:

Concatenated dataset

Obs	name	score1	score2	gender	stu_id	score3
1	Tim	88	80	m	1	.
2	Sharon	95	90	f	2	.
3	Sean	45	78	m	3	.
4	Mike	89	.	m	4	.



partial dataset

Obs	name	score1	score2	gender	stu_id
1	Tim	88	80	m	1
2	Sharon	95	90	f	2
3	Sean	45	78	m	3
4	Mike	89	.	m	4
5	Tina	54	69	f	5
6	Sara	67	78	f	6

complete dataset

Obs	name	score1	score2	score3	gender	stu_id
1	Tim	88	80	99	m	1
2	Sharon	95	90	90	f	2
3	Sean	45	78	56	m	3
4	Mike	89	.	77	m	4
5	Tina	54	69	69	f	5
6	Sara	67	78	74	f	6
7	Sam	98	82	78	m	7
8	James	.	96	89	m	8
9	Jaden	98	78	89	m	9
10	Jack	69	79	70	m	11
11	Will	92	89	98	m	10

EXPLANATION:

· There are 2 datasets given.

i) Full complete dataset given.

ii) Partial dataset given, like details of only 6 students are given out of 11 and score3 variables scores are not present in the dataset.

· At frst, we imported both of these 2 datasets.

· Then just made a new dataset and set both of the datasets in there in the same line instead of taking 2 datasets in 2 diferent sets in 2 diferent lines and simply we got the concatenated (combined) dataset which was required here.

· For convenience, the 2 given datasets are also printed here.

APPENDING TWO DATASETS

· **syntax:**

```
proc append base = <flename_1>
data = <flename_2>;
run;
```

· Base is the dataset to which observations are added.

· Data is the name of the dataset containing observation that have to be added in base dataset.

· We use force command where there is unlike structure of datasets.

CODE:

```
/ Appending /
proc import datafle= "/home/u59242808/Data Files/score_data_id_partial.xlsx"
out= data_a dbms= xlsx replace;
run;
proc import datafle= "/home/u59242808/Data Files/score_data_id.xlsx"
out= data_b dbms= xlsx replace;
run;
proc append base= data_a
data= data_b force;
run;
proc print data= data_a;
title "Appending dataset";
run;
```

OUTPUT:

Appending dataset

Obs	name	score1	score2	gender	stu_id
1	Tim	88	80	m	1
2	Sharon	95	90	f	2
3	Sean	45	78	m	3
4	Mike	89	.	m	4
5	Tina	54	69	f	5
6	Sara	67	78	f	6
7	Tim	88	80	m	1
8	Sharon	95	90	f	2
9	Sean	45	78	m	3
10	Mike	89	.	m	4
11	Tina	54	69	f	5
12	Sara	67	78	f	6
13	Sam	98	82	m	7
14	James	.	96	m	8
15	Jaden	98	78	m	9
16	Jack	69	79	m	11
17	Will	92	89	m	10

EXPLANATION:

· There are 2 datasets given.

i) Full complete dataset given.

ii) Partial dataset given, like details of only 6 students are given out of 11 and score3 variables scores are not present in the dataset.

· At frst, we imported both of these 2 datasets.

· Here the appending was not so simple as merging and concatenating.

· Here, we have to use append command in proc statement taking frst data as the base data and second data to add it after the base data and then printed it and got the appending data which was required here.

· As there is unlike structure of datasets, that's why we had to use the force command or else there would be the observations from the second dataset used in the command here as there is an extra variable here then the base dataset.