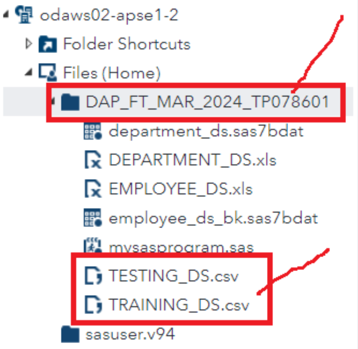
CHAPTER 4: DATA DICTIONARY/ METADATA

Name of Variable	*Description*	*Data Type*	*Length*	*Sample Data*
SME_LOAN_ID_NO	Loan registration number	Char	8	LP001002 LP001003 LP001005 LP001006 LP001008
GENDER	Applicant’s gender; male **or** female	Varchar	6	Male Male Male Male Male
MARITAL_STATUS	Applicant’s marital status; married **or** not married	Varchar	11	Not Married Married Married Married Not Married
FAMILY_MEMBERS	Number of applicant’s family member	Varchar	2	0 2 0 3+ 2
QUALIFICATION	Applicant’s education background; graduate **or** undergraduate	Varchar	14	Under Graduate Graduate Graduate Under Graduate Graduate
EMPLOYMENT	Applicant’s employment status	Varchar	3	Yes No No Yes No
CANDIDATE_INCOME	Applicant’s income	Num	8	3036 4006 12841 3200 2500

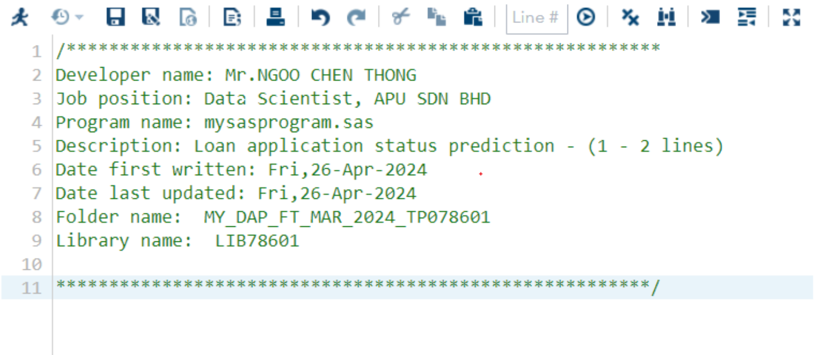
GUARANTEE_INCOME	Applicant's guarantee income	Num	8	2358 0 4196 1516 2504
LOAN_AMOUNT	Loan amount from application (thousands)	Num	8	141 267 95 158 168
LOAN_DURATION	Loan duration (months)	Num	8	360 360 120 360 240
LOAN_HISTORY	Applicant's loan history; yes **or** no	Num	8	1 1 0 1 0
LOAN_LOCATION	Location that applicant applied; city **or** town	Varchar	7	Town City City City Village
LOAN_APPROVAL_STATUS	Approval accept or reject? ;yes **or** no	Char	1	Y N N Y N

4.1 Uploads dataset

Screenshot(s)



odaws02-apse1-2
Folder Shortcuts
Files (Home)
DAP_FT_MAR_2024_TP078601
department_ds.sas7bdat
DEPARTMENT_DS.xls
EMPLOYEE_DS.xls
employee_ds_bk.sas7bdat
mysasprogram.sas
TESTING_DS.csv
TRAINING_DS.csv
sasuser.v94



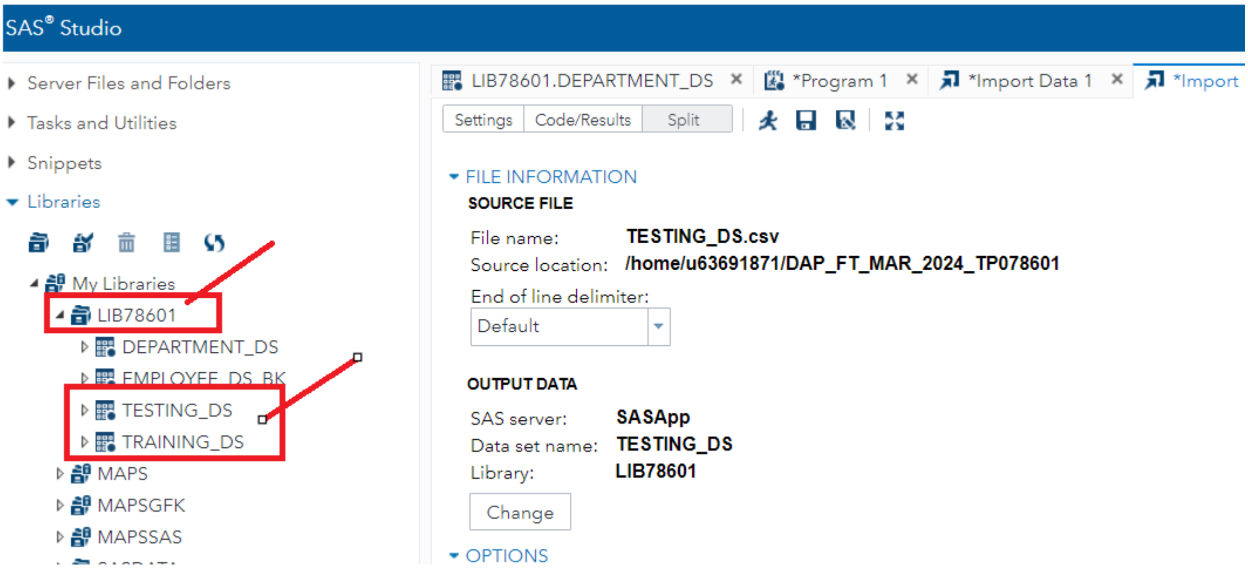
```
/**********************************************************
Developer name: Mr.NGOO CHEN THONG
Job position: Data Scientist, APU SDN BHD
Program name: mysasprogram.sas
Description: Loan application status prediction - (1 - 2 lines)
Date first written: Fri,26-Apr-2024
Date last updated: Fri,26-Apr-2024
Folder name:  MY_DAP_FT_MAR_2024_TP078601
Library name:  LIB78601

**********************************************************/
```

Description

The first thing we have to do is upload the datasets to SAS, the datasets includes training and tesing dataset. As the screenshot shows, a folder named DAP_FT_MAR_TP078601 has been created for stored the relevant documents for this task.

4.2 Transfer the datasets from the Project folder to the newly created permanent SAS library – LIB78601

Screenshot(s)



SOURCE FILE

File name: TESTING_DS.csv

Source location: /home/u63691871/DAP_FT_MAR_2024_TP078601

End of line delimiter: Default

OUTPUT DATA

SAS server: SASApp

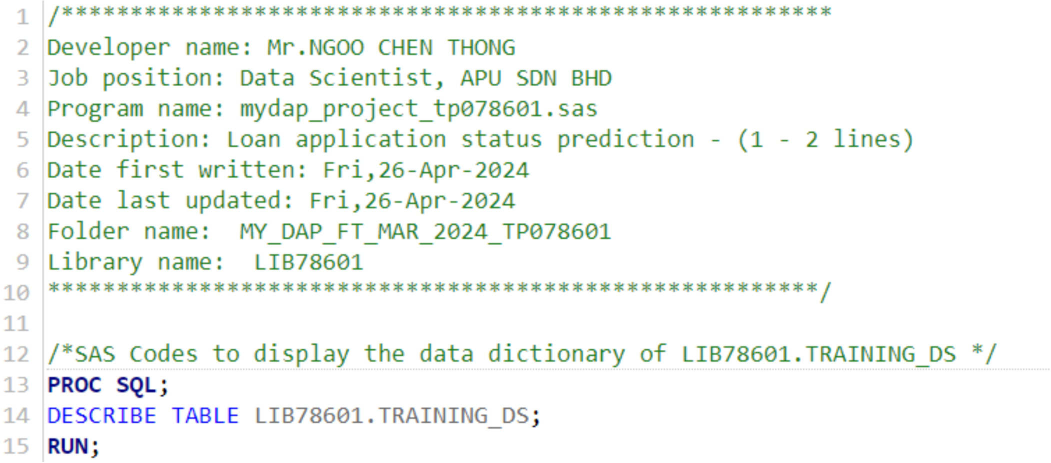
Data set name: TESTING_DS

Library: LIB78601

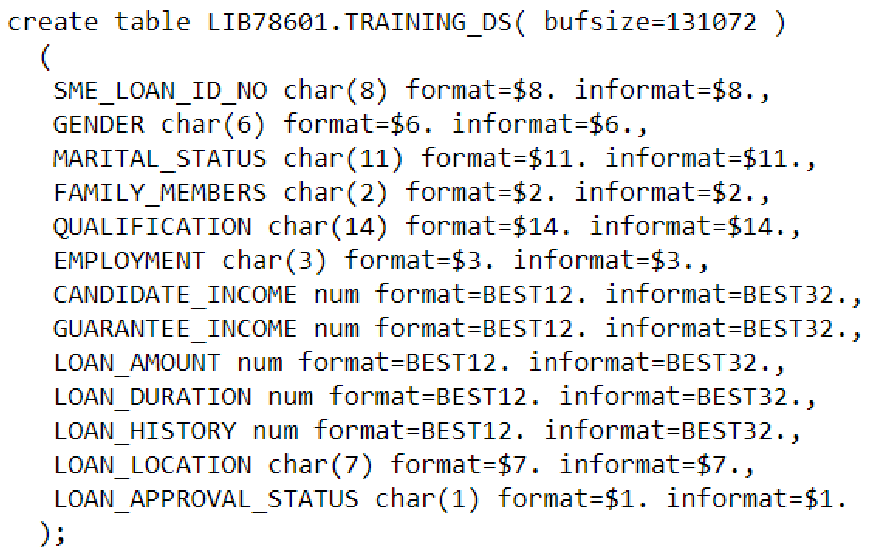
Descriptions

A new library has been created with name LIB78601 to the SAS system, this library mainly purpose is to save the dataset used in this project. By saving these datasets we need to this library, we doesn't need to upload the datasets required for everytime we need it.

4.3 Display the structure of the dataset – LIB78601



```
/*******************************************************
Developer name: Mr.NGOO CHEN THONG
Job position: Data Scientist, APU SDN BHD
Program name: mydap_project_tp078601.sas
Description: Loan application status prediction - (1 - 2 lines)
Date first written: Fri,26-Apr-2024
Date last updated: Fri,26-Apr-2024
Folder name:  MY_DAP_FT_MAR_2024_TP078601
Library name:  LIB78601
*******************************************************/

/*SAS Codes to display the data dictionary of LIB78601.TRAINING_DS */
PROC SQL;
DESCRIBE TABLE LIB78601.TRAINING_DS;
RUN;
```



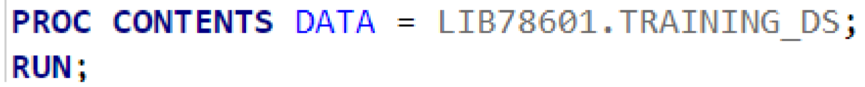
```
create table LIB78601.TRAINING_DS( bufsize=131072 )
  (
   SME_LOAN_ID_NO char(8) format=$8. informat=$8.,
   GENDER char(6) format=$6. informat=$6.,
   MARITAL_STATUS char(11) format=$11. informat=$11.,
   FAMILY_MEMBERS char(2) format=$2. informat=$2.,
   QUALIFICATION char(14) format=$14. informat=$14.,
   EMPLOYMENT char(3) format=$3. informat=$3.,
   CANDIDATE_INCOME num format=BEST12. informat=BEST32.,
   GUARANTEE_INCOME num format=BEST12. informat=BEST32.,
   LOAN_AMOUNT num format=BEST12. informat=BEST32.,
   LOAN_DURATION num format=BEST12. informat=BEST32.,
   LOAN_HISTORY num format=BEST12. informat=BEST32.,
   LOAN_LOCATION char(7) format=$7. informat=$7.,
   LOAN_APPROVAL_STATUS char(1) format=$1. informat=$1.
  );
```

Description

By display the structure of the dataset, we can easily to observe the properties of each attributes including the data type and format.

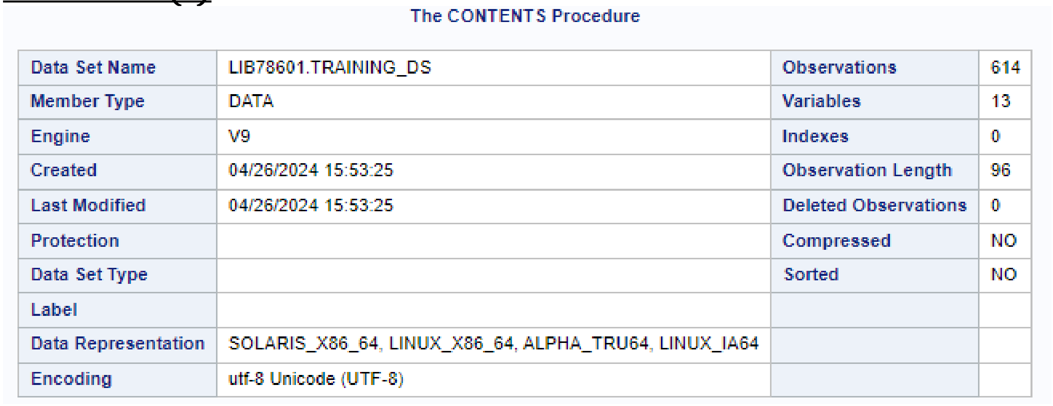
4.4 View the structure of dataset – LIB78601.TESTING_DS

SAS Code



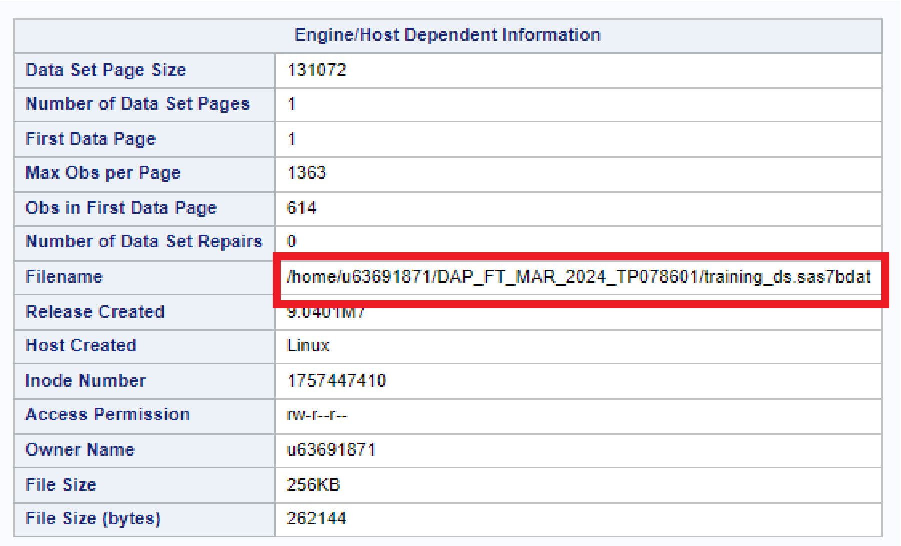
```
PROC CONTENTS DATA = LIB78601.TRAINING_DS;
RUN;
```

Screenshot(s)

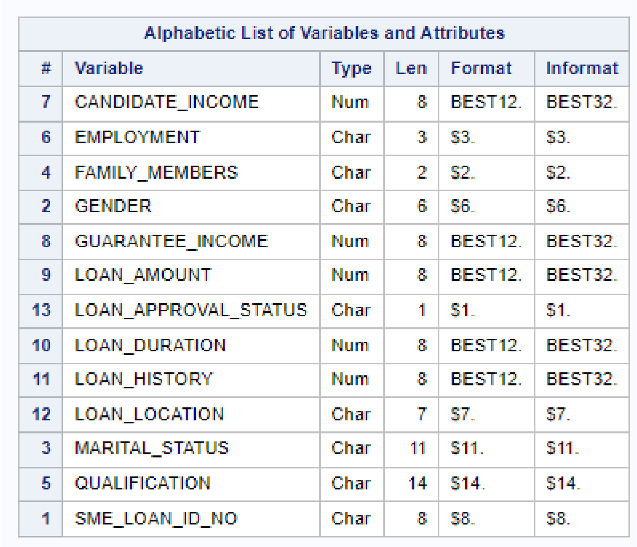


The CONTENTS Procedure

Data Set Name	LIB78601.TRAINING_DS	Observations	614
Member Type	DATA	Variables	13
Engine	V9	Indexes	0
Created	04/26/2024 15:53:25	Observation Length	96
Last Modified	04/26/2024 15:53:25	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		



Engine/Host Dependent Information	
Data Set Page Size	131072
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1363
Obs in First Data Page	614
Number of Data Set Repairs	0
Filename	/home/u63691871/DAP_FT_MAR_2024_TP078601/training_ds.sas7bdat
Release Created	9.0401M7
Host Created	Linux
Inode Number	1757447410
Access Permission	rw-r--r--
Owner Name	u63691871
File Size	256KB
File Size (bytes)	262144



Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat
7	CANDIDATE_INCOME	Num	8	BEST12.	BEST32.
6	EMPLOYMENT	Char	3	$3.	$3.
4	FAMILY_MEMBERS	Char	2	$2.	$2.
2	GENDER	Char	6	$6.	$6.
8	GUARANTEE_INCOME	Num	8	BEST12.	BEST32.
9	LOAN_AMOUNT	Num	8	BEST12.	BEST32.
13	LOAN_APPROVAL_STATUS	Char	1	$1.	$1.
10	LOAN_DURATION	Num	8	BEST12.	BEST32.
11	LOAN_HISTORY	Num	8	BEST12.	BEST32.
12	LOAN_LOCATION	Char	7	$7.	$7.
3	MARITAL_STATUS	Char	11	$11.	$11.
5	QUALIFICATION	Char	14	$14.	$14.
1	SME_LOAN_ID_NO	Char	8	$8.	$8.

Description

By using PROC CONTENTS from SAS, we can see that metadata about the dataset easily. This method helps us to visualize the properties of a dataset in convenient.

CHAPTER 6: ANALYSIS OF THE VARIABLES

6.1 Univariate Analysis of the variables found in the dataset LIB78601.TRAINING_DS

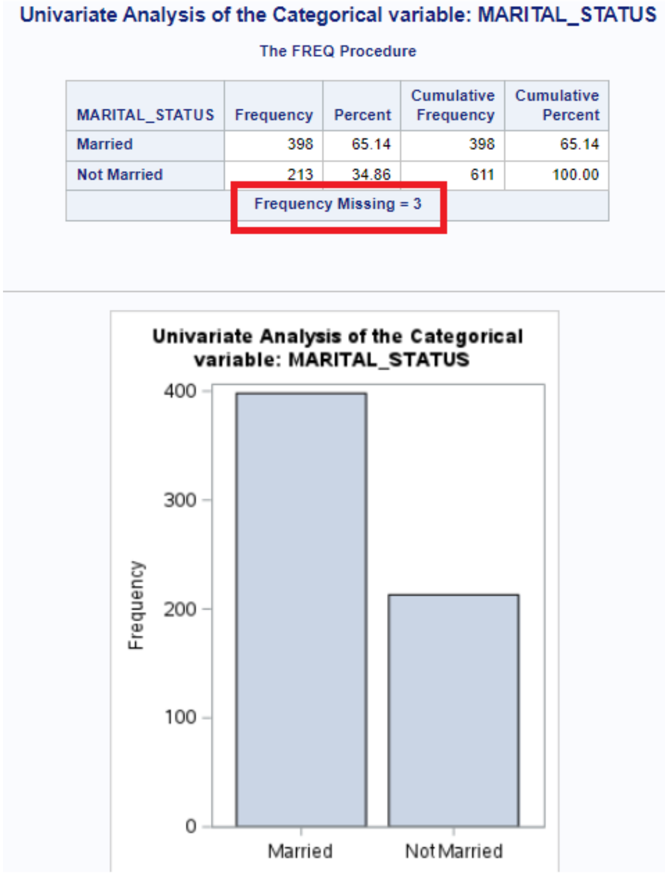
6.1.1 Univariate analysis of the categorical variables found in the dataset.

6.1.1.1 Univariate analysis of the categorical variables – MARITAL_STATUS.

SAS Code

```
TITLE ' Univariate Analysis of the Categorical variable: MARITAL_STATUS';
PROC FREQ DATA = LIB78601.TRAINING_DS;
TABLE marital_status;
RUN;
ODS GRAPHIC / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB78601.TRAINING_DS;
VBAR marital_status;
TITLE ' Univariate Analysis of the Categorical variable: MARITAL_STATUS';
RUN;
```

Screenshot(s)



Univariate Analysis of the Categorical variable: MARITAL_STATUS

The FREQ Procedure

MARITAL_STATUS	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Married	398	65.14	398	65.14
Not Married	213	34.86	611	100.00
Frequency Missing = 3				

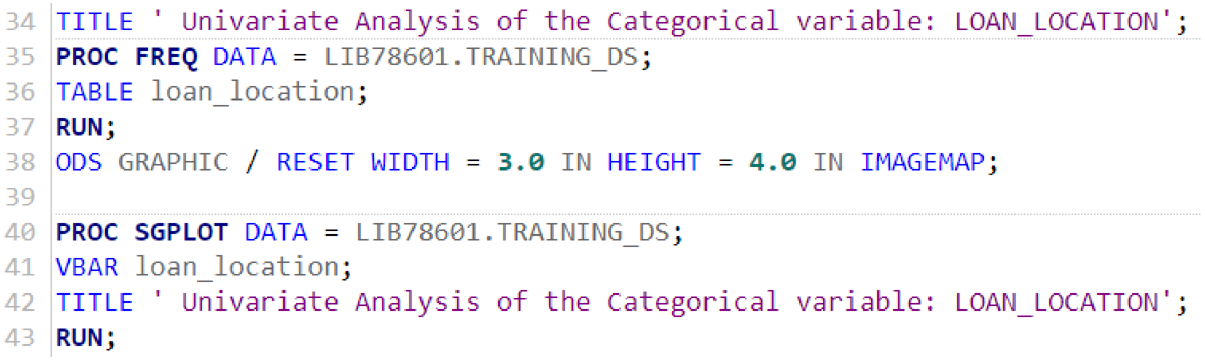
Univariate Analysis of the Categorical variable: MARITAL_STATUS

Description

We can see that most of the loan applicants have married. Furthermore, there is total 3 missing values in the dataset.

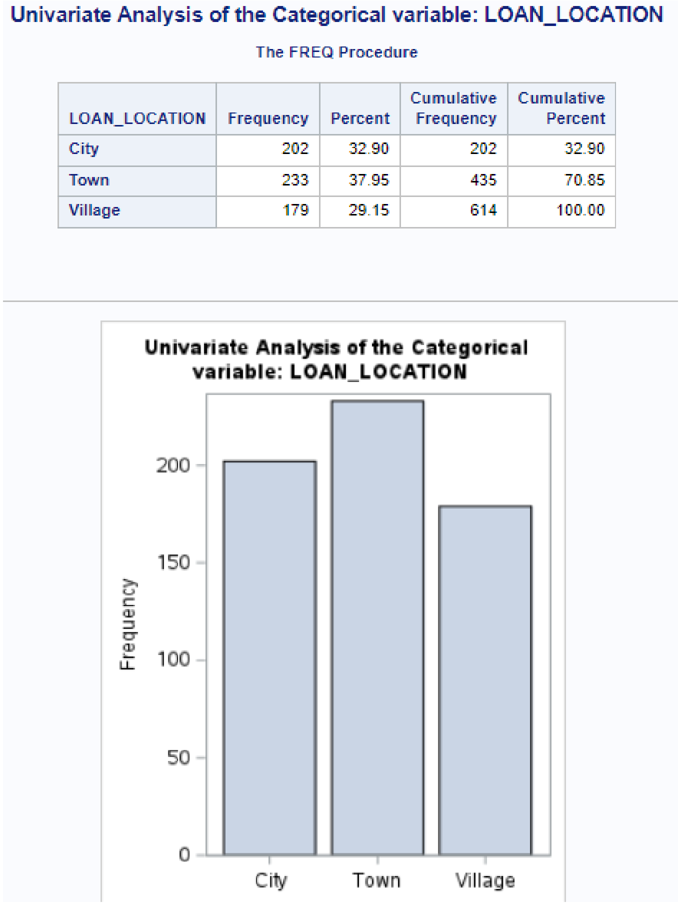
6.1.1.2 Univariate analysis of the categorical variables – LOAN_LOCATION.

SAS Code



```
TITLE ' Univariate Analysis of the Categorical variable: LOAN_LOCATION';
PROC FREQ DATA = LIB78601.TRAINING_DS;
TABLE loan_location;
RUN;
ODS GRAPHIC / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB78601.TRAINING_DS;
VBAR loan_location;
TITLE ' Univariate Analysis of the Categorical variable: LOAN_LOCATION';
RUN;
```

Screenshot(s)



Univariate Analysis of the Categorical variable: LOAN_LOCATION

The FREQ Procedure

LOAN_LOCATION	Frequency	Percent	Cumulative Frequency	Cumulative Percent
City	202	32.90	202	32.90
Town	233	37.95	435	70.85
Village	179	29.15	614	100.00

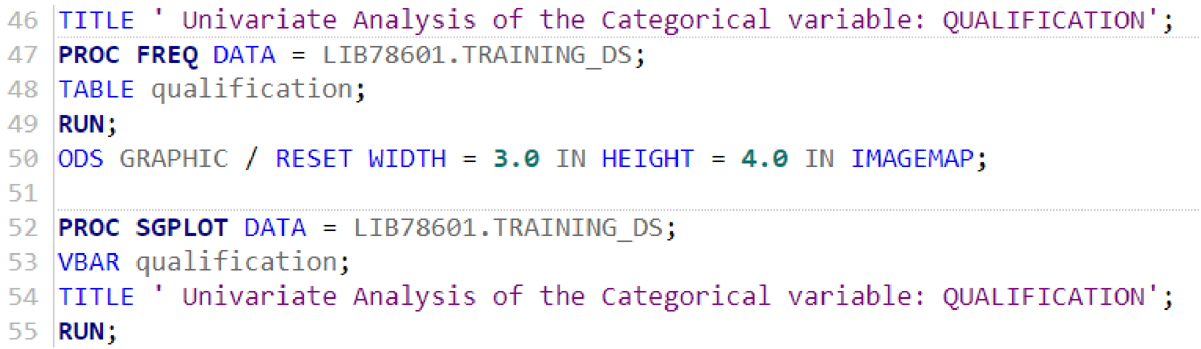
Univariate Analysis of the Categorical variable: LOAN_LOCATION
Frequency
200
150
100
50
0
City
Town
Village

Description

The distribution of loan location for applicants is nearly to identical distribution, means that the applicants are coming from city, town and village, no skew will be found in the visualization. Furthermore, there is no missing value in this dataset.

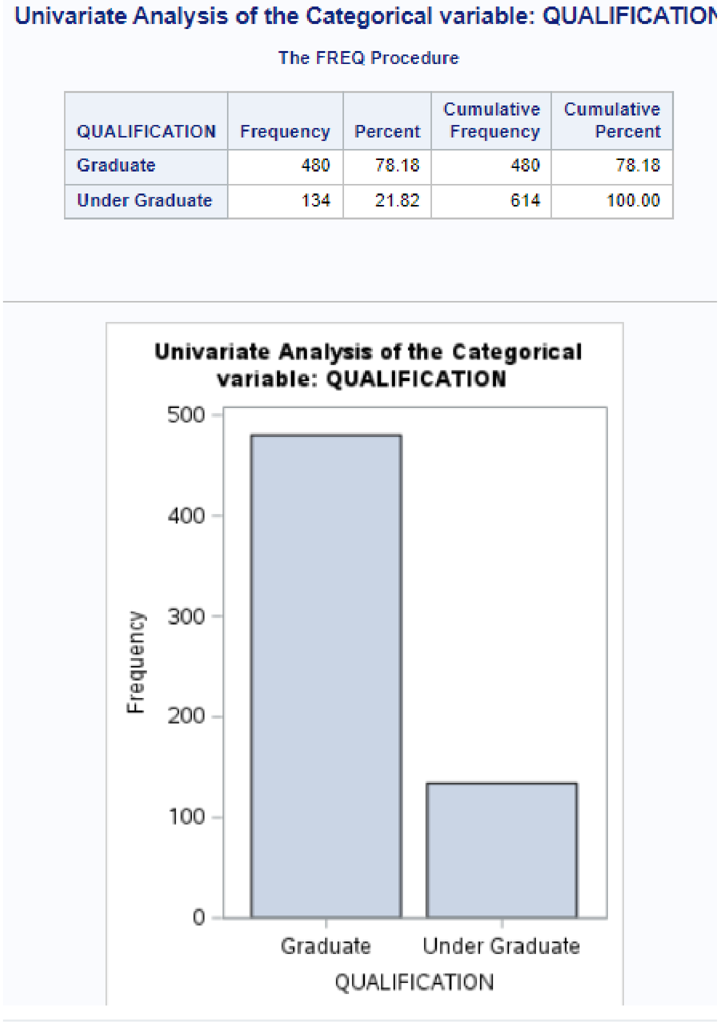
6.1.1.3 Univariate analysis of the categorical variables – QUALIFICATION.

SAS Code



```
TITLE ' Univariate Analysis of the Categorical variable: QUALIFICATION';
PROC FREQ DATA = LIB78601.TRAINING_DS;
TABLE qualification;
RUN;
ODS GRAPHIC / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB78601.TRAINING_DS;
VBAR qualification;
TITLE ' Univariate Analysis of the Categorical variable: QUALIFICATION';
RUN;
```

Screenshot(s)



Univariate Analysis of the Categorical variable: QUALIFICATION

The FREQ Procedure

QUALIFICATION	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Graduate	480	78.18	480	78.18
Under Graduate	134	21.82	614	100.00

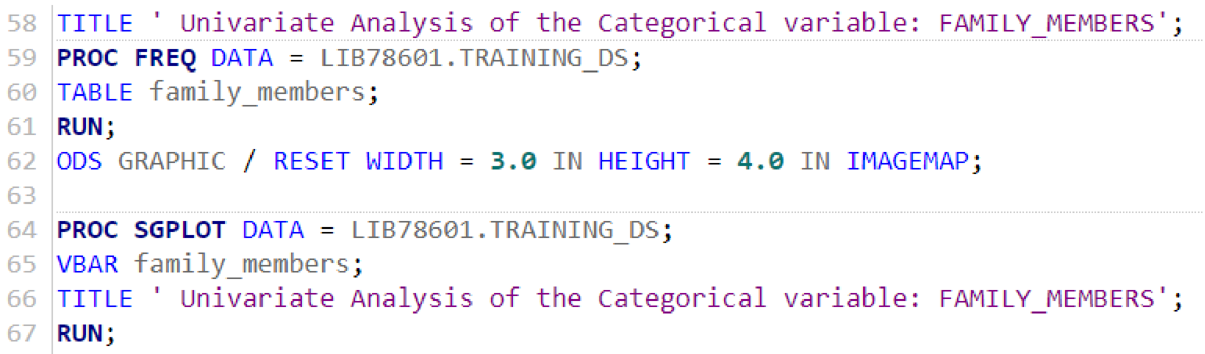
Univariate Analysis of the Categorical variable: QUALIFICATION
Frequency
500
400
300
200
100
0
Graduate
Under Graduate
QUALIFICATION

Description

From the screenshot, we can see that most of the applicants have graduate in qualification, only around 21% of them have undergraduate for qualification. Furthermore, there is no missing value in this dataset.

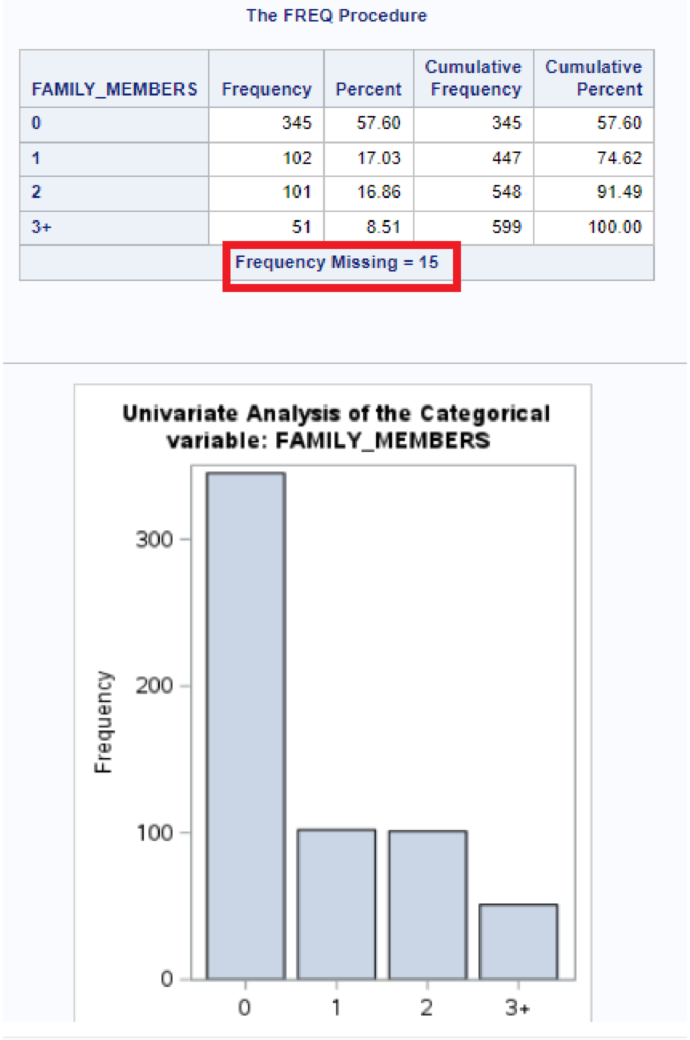
6.1.1.4 Univariate analysis of the categorical variables – FAMILY_MEMBERS.

SAS Code



```
TITLE ' Univariate Analysis of the Categorical variable: FAMILY_MEMBERS';
PROC FREQ DATA = LIB78601.TRAINING_DS;
TABLE family_members;
RUN;
ODS GRAPHIC / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB78601.TRAINING_DS;
VBAR family_members;
TITLE ' Univariate Analysis of the Categorical variable: FAMILY_MEMBERS';
RUN;
```

Screenshot(s)



The FREQ Procedure

FAMILY_MEMBERS	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	345	57.60	345	57.60
1	102	17.03	447	74.62
2	101	16.86	548	91.49
3+	51	8.51	599	100.00
Frequency Missing = 15				

Univariate Analysis of the Categorical variable: FAMILY_MEMBERS
Frequency
300
200
100
0
0
1
2
3+

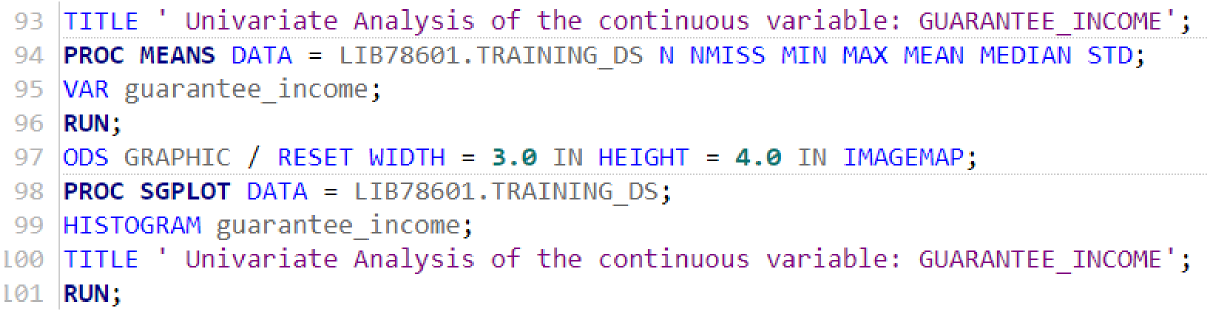
Description

Most of the applicants has 0 for the number of family members, there applicants with 1 member have almost same number with those have family numbers 2. Applicants with 3 or more family members only contribute 8.51% to this dataset. There are 15 missing values in this dataset.

6.1.2 Univariate analysis of the continuous variables found in the dataset – LIB78601.TRAINING_DS.

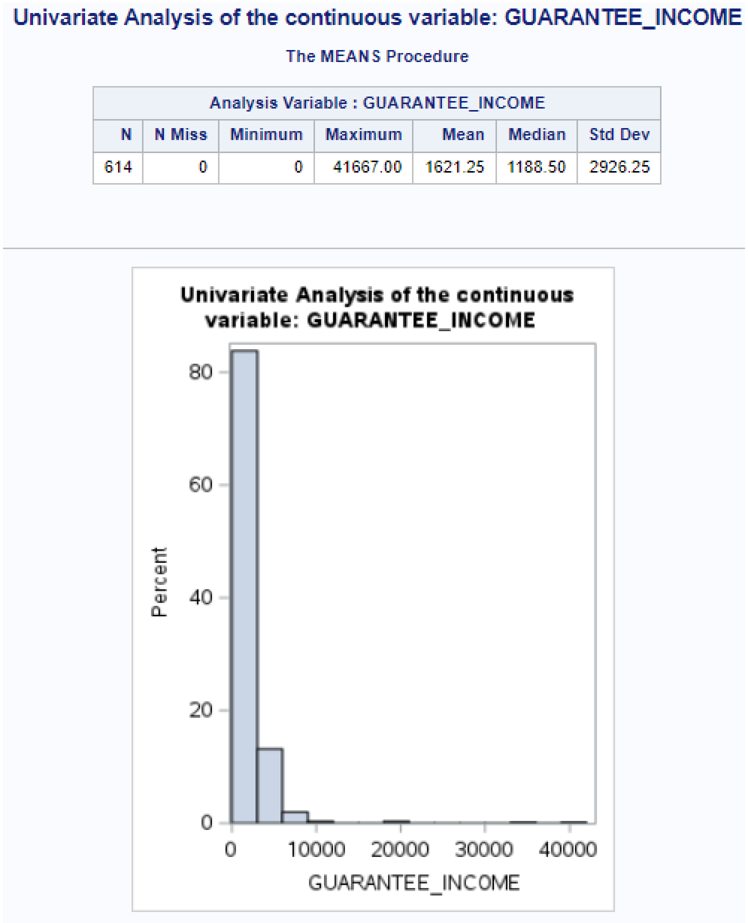
6.1.2.1 Univariate analysis of the continuous variables – GUARANTEE_INCOME.

SAS Code



```
TITLE ' Univariate Analysis of the continuous variable: GUARANTEE_INCOME';
PROC MEANS DATA = LIB78601.TRAINING_DS N NMISS MIN MAX MEAN MEDIAN STD;
VAR guarantee_income;
RUN;
ODS GRAPHIC / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;
PROC SGPLOT DATA = LIB78601.TRAINING_DS;
HISTOGRAM guarantee_income;
TITLE ' Univariate Analysis of the continuous variable: GUARANTEE_INCOME';
RUN;
```

Screenshot(s)



Univariate Analysis of the continuous variable: GUARANTEE_INCOME

The MEANS Procedure

Analysis Variable : GUARANTEE_INCOME						
N	N Miss	Minimum	Maximum	Mean	Median	Std Dev
614	0	0	41667.00	1621.25	1188.50	2926.25

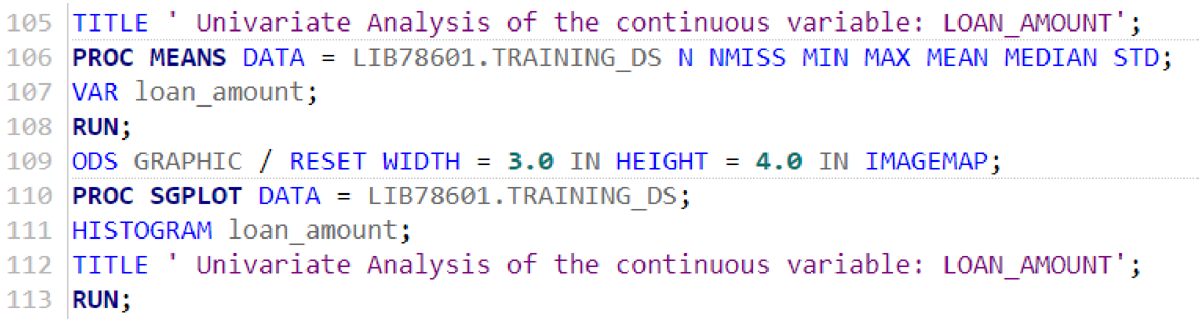
Univariate Analysis of the continuous variable: GUARANTEE_INCOME
Percent
80
60
40
20
0
0
10000
20000
30000
40000
GUARANTEE_INCOME

Description

From the chart, we can easily to see that the guaranteed income from the applicants is concentrated below 10,000. From this view, we can see that people with low income has higher chance to apply loan. Furthermore, there is no missing value in this dataset.

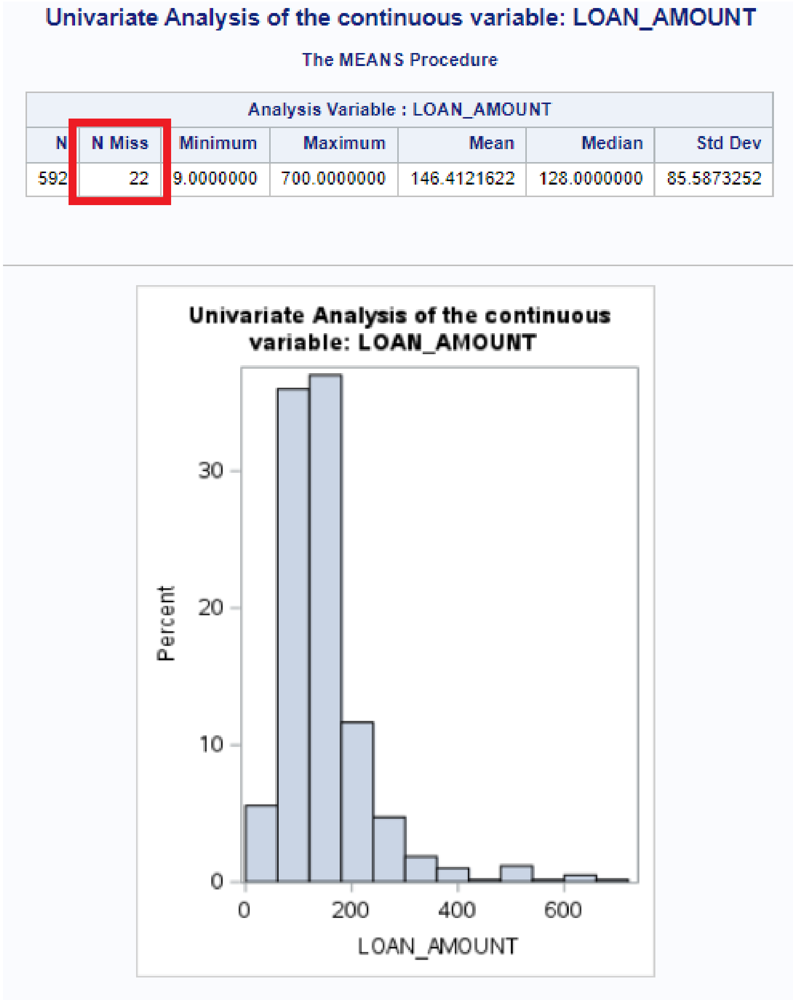
6.1.2.2 Univariate analysis of the continuous variables – LOAN_AMOUNT.

SAS Code



```
TITLE ' Univariate Analysis of the continuous variable: LOAN_AMOUNT';
PROC MEANS DATA = LIB78601.TRAINING_DS N NMISS MIN MAX MEAN MEDIAN STD;
VAR loan_amount;
RUN;
ODS GRAPHIC / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;
PROC SGPLOT DATA = LIB78601.TRAINING_DS;
HISTOGRAM loan_amount;
TITLE ' Univariate Analysis of the continuous variable: LOAN_AMOUNT';
RUN;
```

Screenshot(s)



Univariate Analysis of the continuous variable: LOAN_AMOUNT

The MEANS Procedure

Analysis Variable : LOAN_AMOUNT

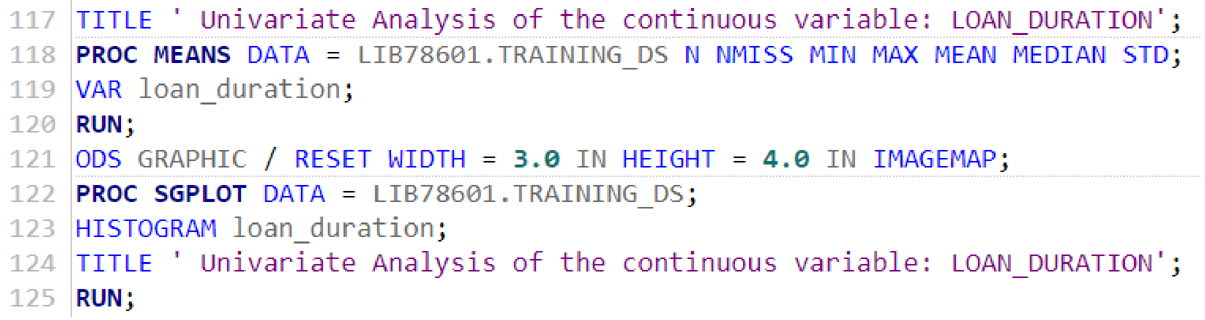
N	N Miss	Minimum	Maximum	Mean	Median	Std Dev
592	22	9.0000000	700.0000000	146.4121622	128.0000000	85.5873252

Description

The distribution for loan amount has right skew distribution, where the mean and median are 146 and 128 respectively. The percentage of loan amount become smaller starts from 300. Furthermore, there are 22 missing values in this dataset.

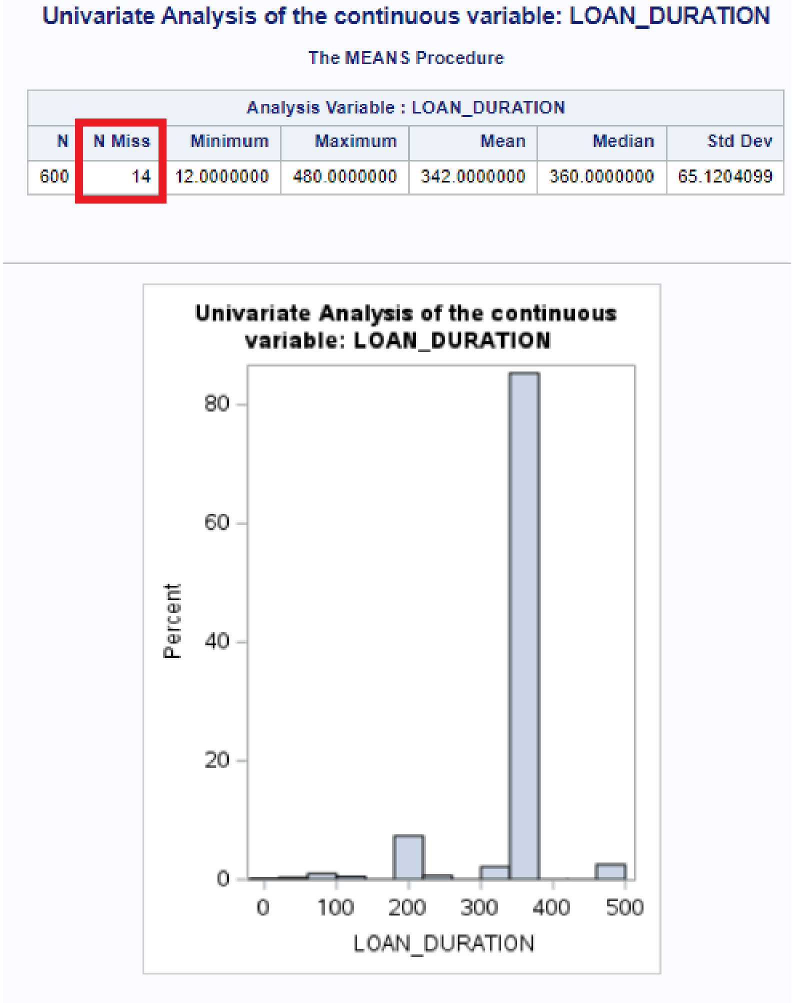
6.1.2.3 Univariate analysis of the continuous variables – LOAN_DURATION.

SAS Code



```
TITLE ' Univariate Analysis of the continuous variable: LOAN_DURATION';
PROC MEANS DATA = LIB78601.TRAINING_DS N NMISS MIN MAX MEAN MEDIAN STD;
VAR loan_duration;
RUN;
ODS GRAPHIC / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;
PROC SGPLOT DATA = LIB78601.TRAINING_DS;
HISTOGRAM loan_duration;
TITLE ' Univariate Analysis of the continuous variable: LOAN_DURATION';
RUN;
```

Screenshot(s)



Univariate Analysis of the continuous variable: LOAN_DURATION

The MEANS Procedure

Analysis Variable : LOAN_DURATION						
N	N Miss	Minimum	Maximum	Mean	Median	Std Dev
600	14	12.0000000	480.0000000	342.0000000	360.0000000	65.1204099

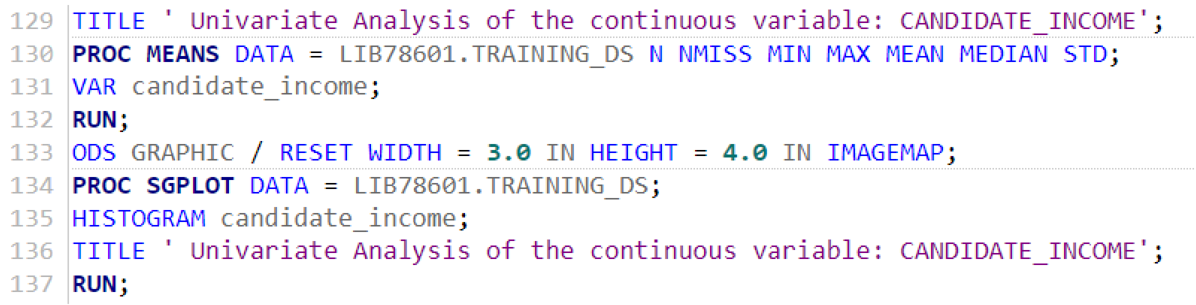
Univariate Analysis of the continuous variable: LOAN_DURATION
Percent
0
20
40
60
80
0
100
200
300
400
500
LOAN_DURATION

Description

By looking the dataset and the visualization for loan duration attribute, we can see that most of the applicants apply for 360 months, which is 30 years. The minimum and maximum loan duration are 12 and 480 months respectively. Furthermore, there are 14 missing values in this dataset.

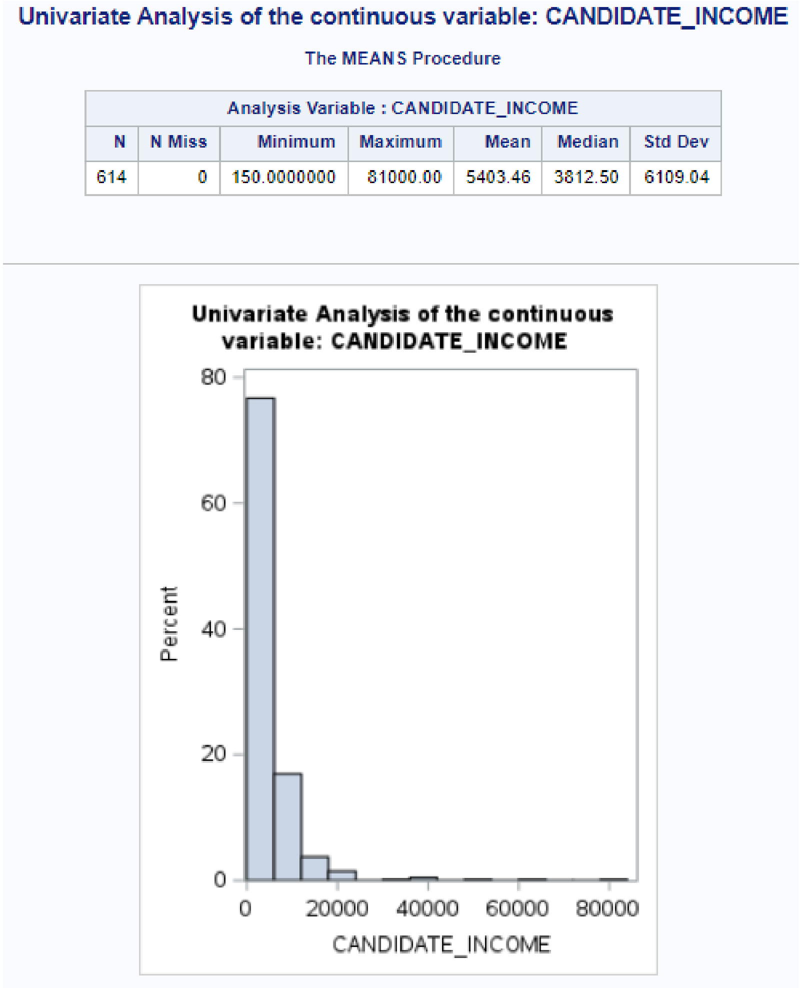
6.1.2.4 Univariate analysis of the continuous variables – CANDIDATE_INCOME.

SAS Code



```
TITLE ' Univariate Analysis of the continuous variable: CANDIDATE_INCOME';
PROC MEANS DATA = LIB78601.TRAINING_DS N NMISS MIN MAX MEAN MEDIAN STD;
VAR candidate_income;
RUN;
ODS GRAPHIC / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;
PROC SGPLOT DATA = LIB78601.TRAINING_DS;
HISTOGRAM candidate_income;
TITLE ' Univariate Analysis of the continuous variable: CANDIDATE_INCOME';
RUN;
```

Screenshot(s)



Univariate Analysis of the continuous variable: CANDIDATE_INCOME

The MEANS Procedure

Analysis Variable : CANDIDATE_INCOME						
N	N Miss	Minimum	Maximum	Mean	Median	Std Dev
614	0	150.0000000	81000.00	5403.46	3812.50	6109.04

Univariate Analysis of the continuous variable: CANDIDATE_INCOME
Percent
80
60
40
20
0
0
20000
40000
60000
80000
CANDIDATE_INCOME

Description

More than 90% of the applicants have income below 10,000, and the percentage become lower when income increase. Furthermore, there is no missing value in this dataset.

6.2 Bivariate analysis of the variables found in the dataset – LIB78601.TRAINING_DS.

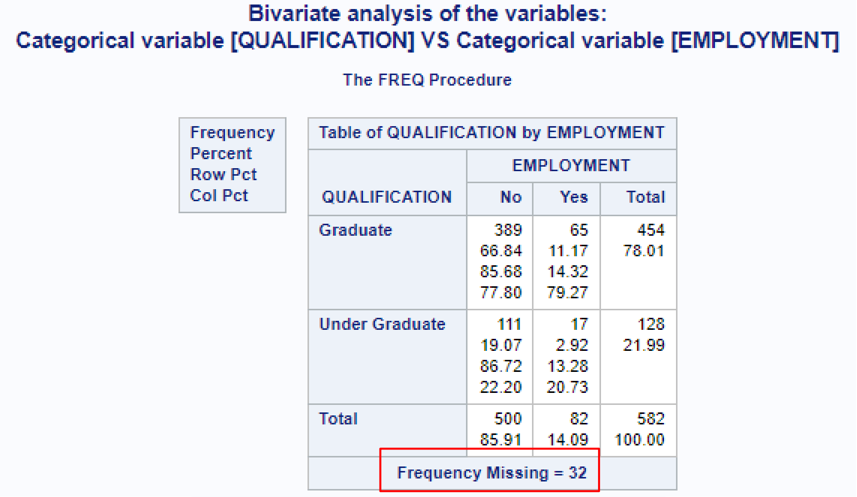
6.2.1 Bivariate analysis of the continuous variables (Categorical vs Categorical).

6.2.1.1 Bivariate analysis of the continuous variables (Categorical - QUALIFICATION vs Categorical - EMPLOYMENT).

SAS Code

```
TITLE1 ' Bivariate analysis of the variables:';
TITLE2 ' Categorical variable [QUALIFICATION] VS Categorical variable [EMPLOYMENT] ';
PROC FREQ DATA = LIB78601.TRAINING_DS;
TABLE qualification*employment/
PLOTS = FREQPLOT(TWOWAY = STACKED SCALE = GROUPPCT);
RUN;
```

Screenshot(s)

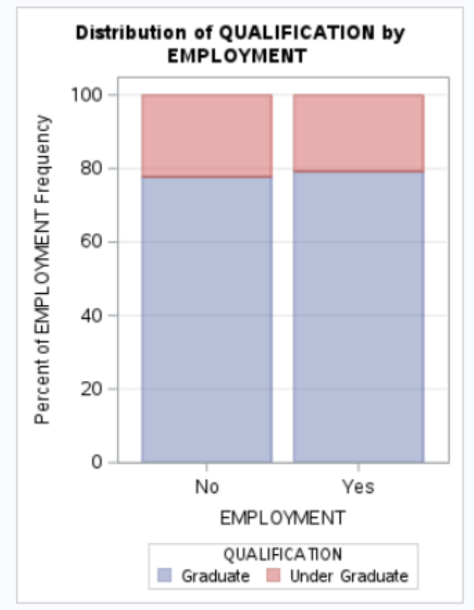


Bivariate analysis of the variables:
Categorical variable [QUALIFICATION] VS Categorical variable [EMPLOYMENT]

The FREQ Procedure

Frequency
Percent
Row Pct
Col Pct

Table of QUALIFICATION by EMPLOYMENT			
	EMPLOYMENT		
QUALIFICATION	No	Yes	Total
Graduate	389 66.84 85.68 77.80	65 11.17 14.32 79.27	454 78.01
Under Graduate	111 19.07 86.72 22.20	17 2.92 13.28 20.73	128 21.99
Total	500 85.91	82 14.09	582 100.00
Frequency Missing = 32			

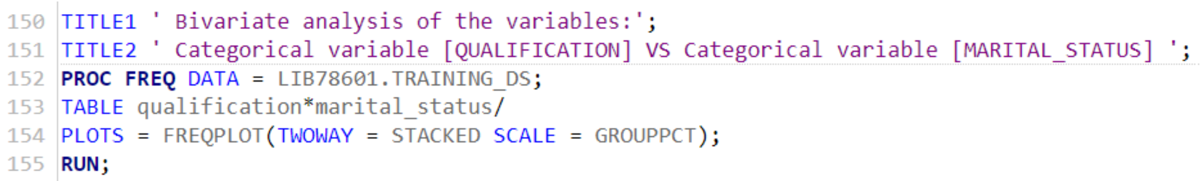


Description

From the table, we can observe that most of the applicants have graduate qualification but didn’t have employment. Another finding from this chart is, graduate and undergraduate qualification applicants have around 80% has no employment. Furthermore, there 32 missing values in this dataset.

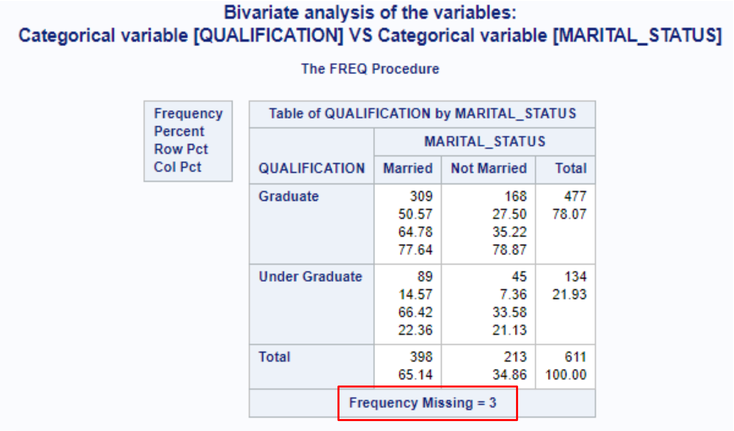
6.2.1.2 Bivariate analysis of the continuous variables (Categorical - QUALIFICATION vs Categorical – MARITAL_STATUS).

SAS Code



```
TITLE1 ' Bivariate analysis of the variables:';
TITLE2 ' Categorical variable [QUALIFICATION] VS Categorical variable [MARITAL_STATUS] ';
PROC FREQ DATA = LIB78601.TRAINING_DS;
TABLE qualification*marital_status/
PLOTS = FREQPLOT(TWOWAY = STACKED SCALE = GROUPPCT);
RUN;
```

Screenshot(s)



Bivariate analysis of the variables:
Categorical variable [QUALIFICATION] VS Categorical variable [MARITAL_STATUS]

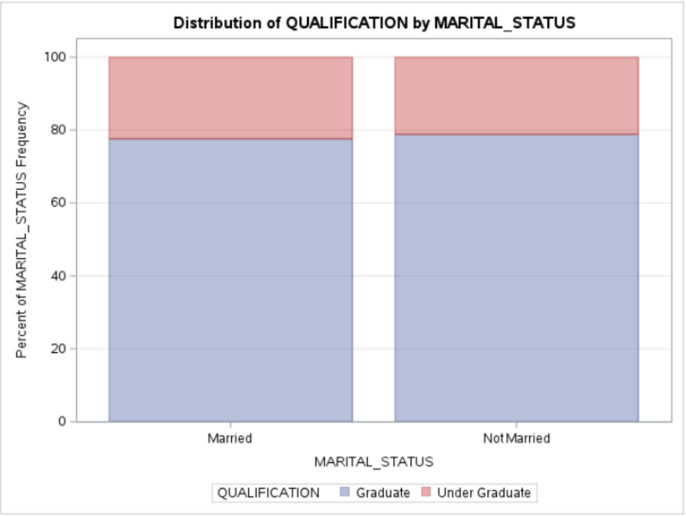
The FREQ Procedure

Frequency
Percent
Row Pct
Col Pct

Table of QUALIFICATION by MARITAL_STATUS

	MARITAL_STATUS		
QUALIFICATION	Married	Not Married	Total
Graduate	309 50.57 64.78 77.64	168 27.50 35.22 78.87	477 78.07
Under Graduate	89 14.57 66.42 22.36	45 7.36 33.58 21.13	134 21.93
Total	398 65.14	213 34.86	611 100.00

Frequency Missing = 3

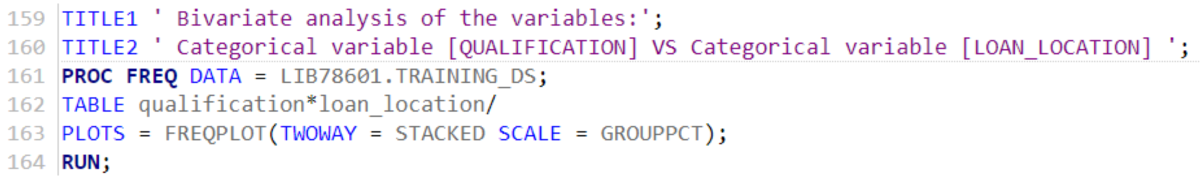


Description

Most of the applicants have married for their marital status, this may be one of the reasons that why they trying to apply bank loan since they need more financial support to cover their daily needs. Furthermore, there are 3 missing values in this dataset.

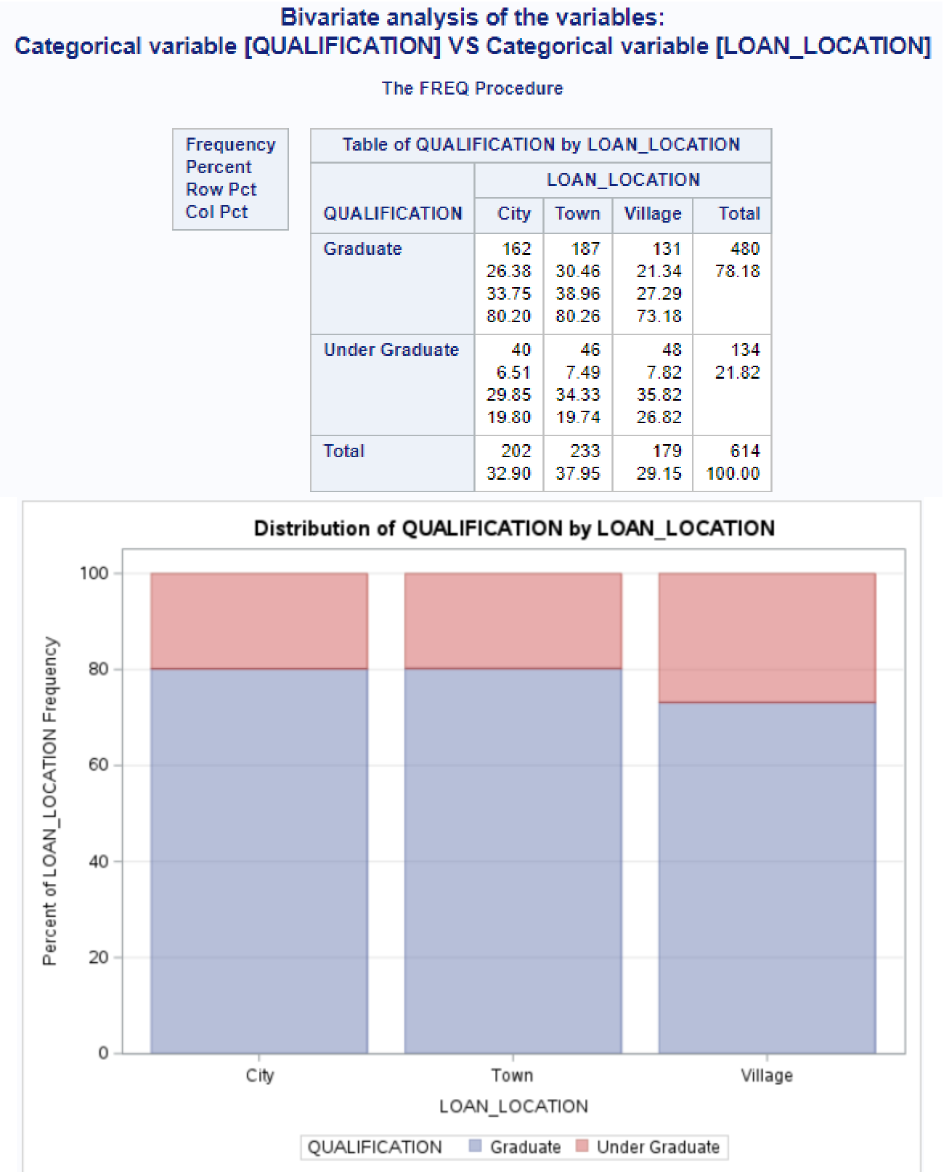
6.2.1.3 Bivariate analysis of the continuous variables (Categorical - QUALIFICATION vs Categorical – LOAN_LOCATION).

SAS Code



```
TITLE1 ' Bivariate analysis of the variables:';
TITLE2 ' Categorical variable [QUALIFICATION] VS Categorical variable [LOAN_LOCATION] ';
PROC FREQ DATA = LIB78601.TRAINING_DS;
TABLE qualification*loan_location/
PLOTS = FREQPLOT(TWOWAY = STACKED SCALE = GROUPPCT);
RUN;
```

Screenshot(s)



Bivariate analysis of the variables:
Categorical variable [QUALIFICATION] VS Categorical variable [LOAN_LOCATION]

The FREQ Procedure

Frequency
Percent
Row Pct
Col Pct

Table of QUALIFICATION by LOAN_LOCATION

	LOAN_LOCATION			
QUALIFICATION	City	Town	Village	Total
Graduate	162 26.38 33.75 80.20	187 30.46 38.96 80.26	131 21.34 27.29 73.18	480 78.18
Under Graduate	40 6.51 29.85 19.80	46 7.49 34.33 19.74	48 7.82 35.82 26.82	134 21.82
Total	202 32.90	233 37.95	179 29.15	614 100.00

Distribution of QUALIFICATION by LOAN_LOCATION
Percent of LOAN_LOCATION Frequency
100
80
60
40
20
0
City
Town
Village
LOAN_LOCATION
QUALIFICATION
Graduate
Under Graduate

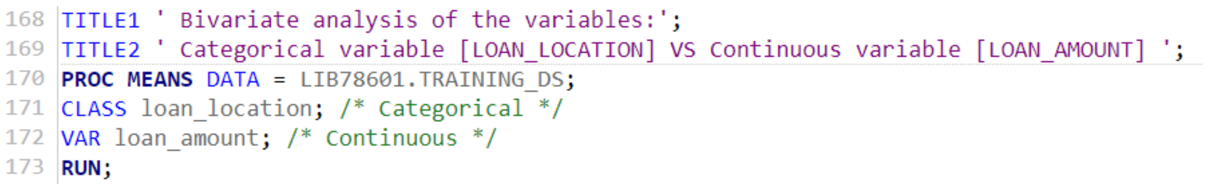
Description

As above, we have seen that the distribution of loan location is quite identically for city, village, and town. The ratio of graduate in village is minor lesser than others, where other two places have around 80% for graduate applicants. Furthermore, there is no missing value in this dataset.

6.2.2 Bivariate analysis of the continuous variables (Categorical vs Continuous).

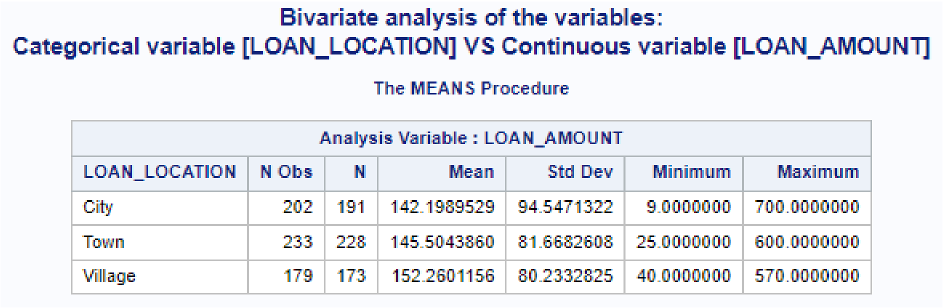
6.2.2.1 Bivariate analysis of the continuous variables (Categorical – LOAN_LOCATION vs Continuous – LOAN_AMOUNT).

SAS Code



```
TITLE1 ' Bivariate analysis of the variables:';
TITLE2 ' Categorical variable [LOAN_LOCATION] VS Continuous variable [LOAN_AMOUNT] ';
PROC MEANS DATA = LIB78601.TRAINING_DS;
CLASS loan_location; /* Categorical */
VAR loan_amount; /* Continuous */
RUN;
```

Screenshot(s)



Bivariate analysis of the variables:
Categorical variable [LOAN_LOCATION] VS Continuous variable [LOAN_AMOUNT]

The MEANS Procedure

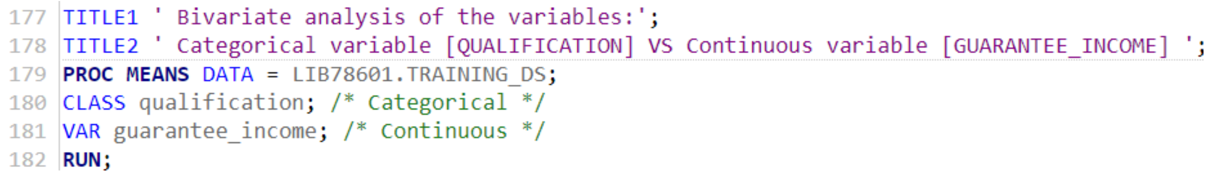
Analysis Variable : LOAN_AMOUNT						
LOAN_LOCATION	N Obs	N	Mean	Std Dev	Minimum	Maximum
City	202	191	142.1989529	94.5471322	9.0000000	700.0000000
Town	233	228	145.5043860	81.6682608	25.0000000	600.0000000
Village	179	173	152.2601156	80.2332825	40.0000000	570.0000000

Description

For loan amount, applicants from village have the highest mean in these 3 loan locations, where the locations that have highest standard deviation will be city, this means that loan amount from city is wider in the distribution chart.

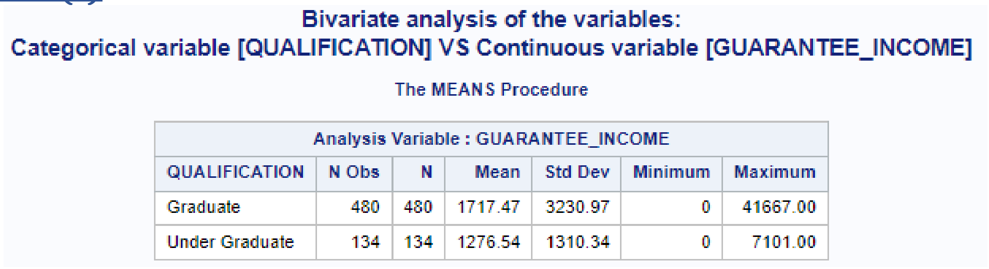
6.2.2.2 Bivariate analysis of the continuous variables (Categorical - QUALIFICATION vs Continuous – GUARANTEE_INCOME).

SAS Code



```
TITLE1 ' Bivariate analysis of the variables:';
TITLE2 ' Categorical variable [QUALIFICATION] VS Continuous variable [GUARANTEE_INCOME] ';
PROC MEANS DATA = LIB78601.TRAINING_DS;
CLASS qualification; /* Categorical */
VAR guarantee_income; /* Continuous */
RUN;
```

Screenshot(s)



Bivariate analysis of the variables:
Categorical variable [QUALIFICATION] VS Continuous variable [GUARANTEE_INCOME]

The MEANS Procedure

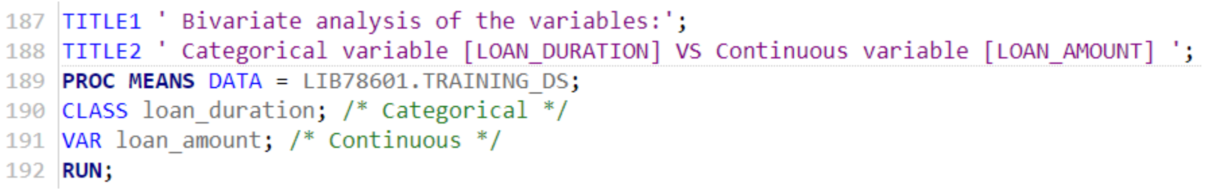
Analysis Variable : GUARANTEE_INCOME						
QUALIFICATION	N Obs	N	Mean	Std Dev	Minimum	Maximum
Graduate	480	480	1717.47	3230.97	0	41667.00
Under Graduate	134	134	1276.54	1310.34	0	7101.00

Description

We can see thar either graduate or undergraduate qualifications, there still have some applicants have 0 income, this means that some of them have totally no income or not employment. But for those who have income, the mean of income from graduate qualifications is higher than undergraduate.

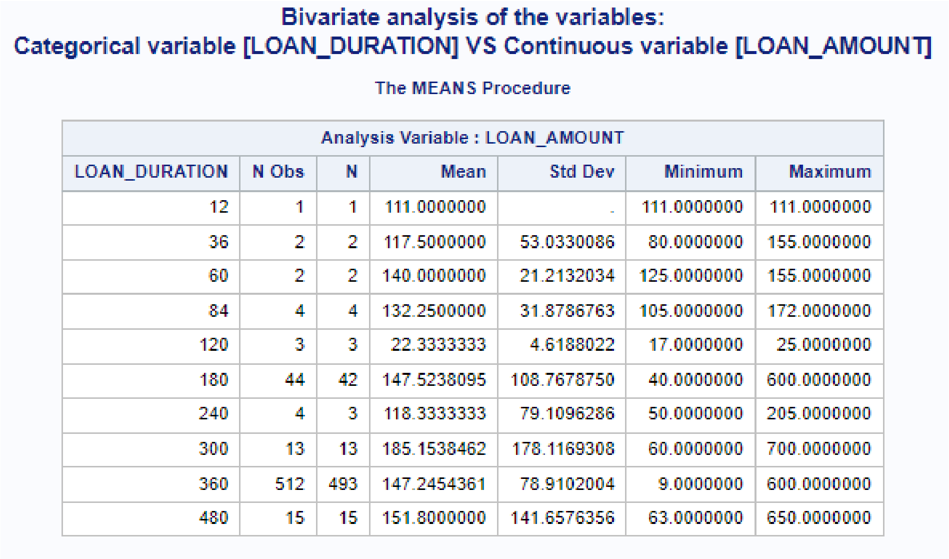
6.2.2.3 Bivariate analysis of the continuous variables (Categorical – LOAN_DURATION vs Continuous – LOAN_AMOUNT).

SAS Code



```
TITLE1 ' Bivariate analysis of the variables:';
TITLE2 ' Categorical variable [LOAN_DURATION] VS Continuous variable [LOAN_AMOUNT] ';
PROC MEANS DATA = LIB78601.TRAINING_DS;
CLASS loan_duration; /* Categorical */
VAR loan_amount; /* Continuous */
RUN;
```

Screenshot(s)



Bivariate analysis of the variables:
Categorical variable [LOAN_DURATION] VS Continuous variable [LOAN_AMOUNT]

The MEANS Procedure

Analysis Variable : LOAN_AMOUNT

LOAN_DURATION	N Obs	N	Mean	Std Dev	Minimum	Maximum
12	1	1	111.0000000	.	111.0000000	111.0000000
36	2	2	117.5000000	53.0330086	80.0000000	155.0000000
60	2	2	140.0000000	21.2132034	125.0000000	155.0000000
84	4	4	132.2500000	31.8786763	105.0000000	172.0000000
120	3	3	22.3333333	4.6188022	17.0000000	25.0000000
180	44	42	147.5238095	108.7678750	40.0000000	600.0000000
240	4	3	118.3333333	79.1096286	50.0000000	205.0000000
300	13	13	185.1538462	178.1169308	60.0000000	700.0000000
360	512	493	147.2454361	78.9102004	9.0000000	600.0000000
480	15	15	151.8000000	141.6576356	63.0000000	650.0000000

Description

The highest option chosen for loan duration will be 360 months, and the option with second highest chosen was 180 months, but the difference between them is large. The mean of highest loan amount from loan duration was 300 months with mean 185,000 and the standard deviation of loan duration with 120 months has only 4.61, this means that the distribution of this group of applicants is heavily concentrate.

6.3 Analysis of the variables found in the dataset LIB78601.TESTING_DS

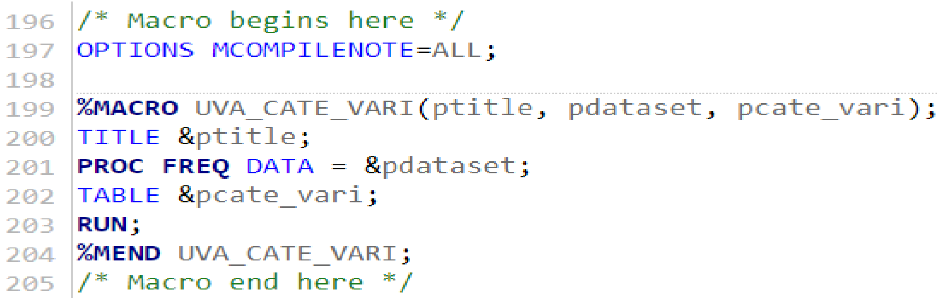
6.3.1 Univariate analysis of the variables found in the dataset – LIB78601.TESTING_DS.

6.3.1.1 Univariate analysis of the categorical variables found in the dataset using SAS MACRO.

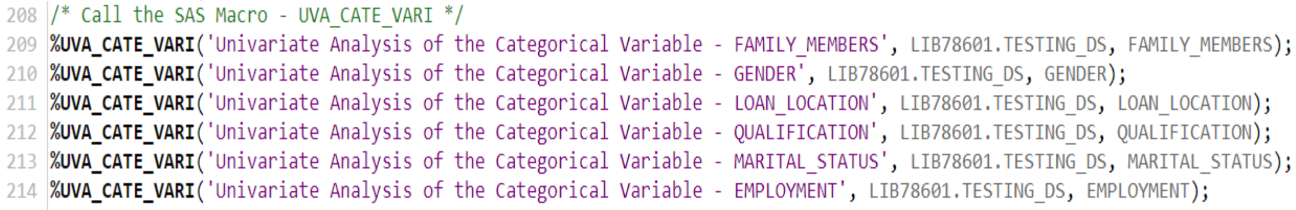
Introduction to SAS MACRO

SAS MACRO is a feature in SAS programming that makes it possible to automate and customize repetitive tasks. Reusable code blocks can be formed by using the %macro and %mend statements to define macros. Their support for parameterization, conditional logic, looping, and the usage of macro variables makes SAS programming simpler as well as efficient Macros are a useful tool for SAS developers and analysts because they improve code readability, expedite procedures, and allow code reuse.

SAS Code

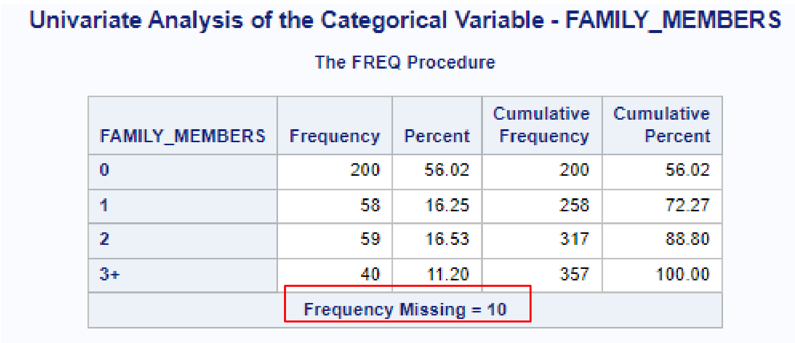


```
/* Macro begins here */
OPTIONS MCOMPILENOTE=ALL;

%MACRO UVA_CATE_VARI(ptitle, pdataset, pcate_vari);
TITLE &ptitle;
PROC FREQ DATA = &pdataset;
TABLE &pcate_vari;
RUN;
%MEND UVA_CATE_VARI;
/* Macro end here */
```



```
/* Call the SAS Macro - UVA_CATE_VARI */
%UVA_CATE_VARI('Univariate Analysis of the Categorical Variable - FAMILY_MEMBERS', LIB78601.TESTING_DS, FAMILY_MEMBERS);
%UVA_CATE_VARI('Univariate Analysis of the Categorical Variable - GENDER', LIB78601.TESTING_DS, GENDER);
%UVA_CATE_VARI('Univariate Analysis of the Categorical Variable - LOAN_LOCATION', LIB78601.TESTING_DS, LOAN_LOCATION);
%UVA_CATE_VARI('Univariate Analysis of the Categorical Variable - QUALIFICATION', LIB78601.TESTING_DS, QUALIFICATION);
%UVA_CATE_VARI('Univariate Analysis of the Categorical Variable - MARITAL_STATUS', LIB78601.TESTING_DS, MARITAL_STATUS);
%UVA_CATE_VARI('Univariate Analysis of the Categorical Variable - EMPLOYMENT', LIB78601.TESTING_DS, EMPLOYMENT);
```

Screenshot(s)

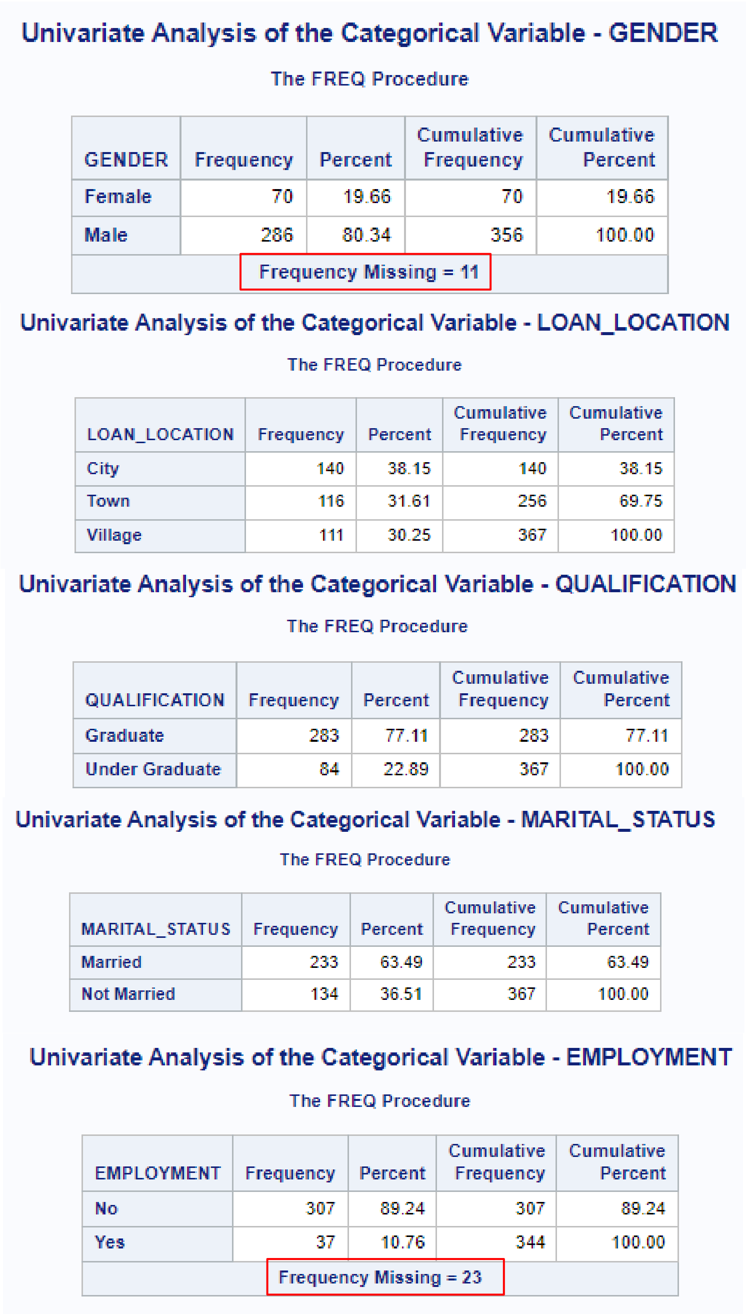


Univariate Analysis of the Categorical Variable - FAMILY_MEMBERS

The FREQ Procedure

FAMILY_MEMBERS	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	200	56.02	200	56.02
1	58	16.25	258	72.27
2	59	16.53	317	88.80
3+	40	11.20	357	100.00

Frequency Missing = 10



Univariate Analysis of the Categorical Variable - GENDER

The FREQ Procedure

GENDER	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Female	70	19.66	70	19.66
Male	286	80.34	356	100.00

Frequency Missing = 11

Univariate Analysis of the Categorical Variable - LOAN_LOCATION

The FREQ Procedure

LOAN_LOCATION	Frequency	Percent	Cumulative Frequency	Cumulative Percent
City	140	38.15	140	38.15
Town	116	31.61	256	69.75
Village	111	30.25	367	100.00

Univariate Analysis of the Categorical Variable - QUALIFICATION

The FREQ Procedure

QUALIFICATION	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Graduate	283	77.11	283	77.11
Under Graduate	84	22.89	367	100.00

Univariate Analysis of the Categorical Variable - MARITAL_STATUS

The FREQ Procedure

MARITAL_STATUS	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Married	233	63.49	233	63.49
Not Married	134	36.51	367	100.00

Univariate Analysis of the Categorical Variable - EMPLOYMENT

The FREQ Procedure

EMPLOYMENT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
No	307	89.24	307	89.24
Yes	37	10.76	344	100.00

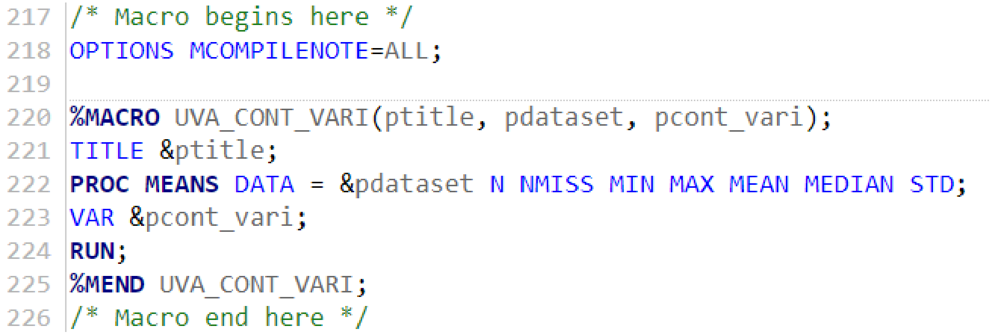
Frequency Missing = 23

Description

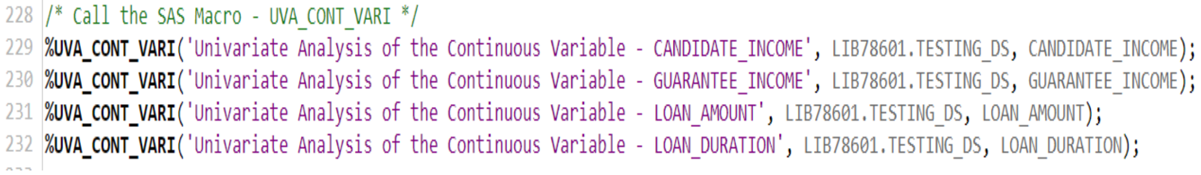
We can observe the data of univariate analysis to categorical data using SAS Macro easily. The missing values for variables family members, gender, and employment are 10, 11, and 23 respectively. For family members, 56% of the applicants didn't have any family members. Furthermore, the distribution of male is greater than female. In addition, the number of graduate applicants is more than undergraduate. Around 63% of the applicants is married.

6.3.1.2 Univariate analysis of the continuous variables found in the dataset using SAS MACRO.

SAS Code

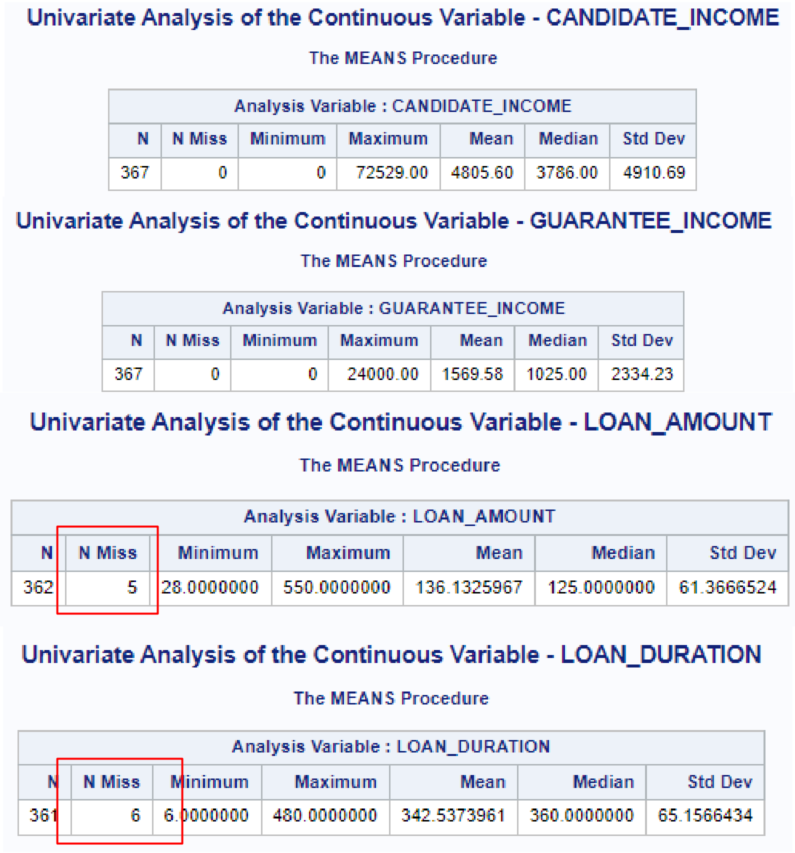


```
/* Macro begins here */
OPTIONS MCOMPILENOTE=ALL;

%MACRO UVA_CONT_VARI(ptitle, pdataset, pcont_vari);
TITLE &ptitle;
PROC MEANS DATA = &pdataset N NMISS MIN MAX MEAN MEDIAN STD;
VAR &pcont_vari;
RUN;
%MEND UVA_CONT_VARI;
/* Macro end here */
```



```
/* Call the SAS Macro - UVA_CONT_VARI */
%UVA_CONT_VARI('Univariate Analysis of the Continuous Variable - CANDIDATE_INCOME', LIB78601.TESTING_DS, CANDIDATE_INCOME);
%UVA_CONT_VARI('Univariate Analysis of the Continuous Variable - GUARANTEE_INCOME', LIB78601.TESTING_DS, GUARANTEE_INCOME);
%UVA_CONT_VARI('Univariate Analysis of the Continuous Variable - LOAN_AMOUNT', LIB78601.TESTING_DS, LOAN_AMOUNT);
%UVA_CONT_VARI('Univariate Analysis of the Continuous Variable - LOAN_DURATION', LIB78601.TESTING_DS, LOAN_DURATION);
```

Screenshot(s)



Univariate Analysis of the Continuous Variable - CANDIDATE_INCOME

The MEANS Procedure

Analysis Variable : CANDIDATE_INCOME

N	N Miss	Minimum	Maximum	Mean	Median	Std Dev
367	0	0	72529.00	4805.60	3786.00	4910.69

Univariate Analysis of the Continuous Variable - GUARANTEE_INCOME

The MEANS Procedure

Analysis Variable : GUARANTEE_INCOME

N	N Miss	Minimum	Maximum	Mean	Median	Std Dev
367	0	0	24000.00	1569.58	1025.00	2334.23

Univariate Analysis of the Continuous Variable - LOAN_AMOUNT

The MEANS Procedure

Analysis Variable : LOAN_AMOUNT

N	N Miss	Minimum	Maximum	Mean	Median	Std Dev
362	5	28.0000000	550.0000000	136.1325967	125.0000000	61.3666524

Univariate Analysis of the Continuous Variable - LOAN_DURATION

The MEANS Procedure

Analysis Variable : LOAN_DURATION

N	N Miss	Minimum	Maximum	Mean	Median	Std Dev
361	6	6.0000000	480.0000000	342.5373961	360.0000000	65.1566434

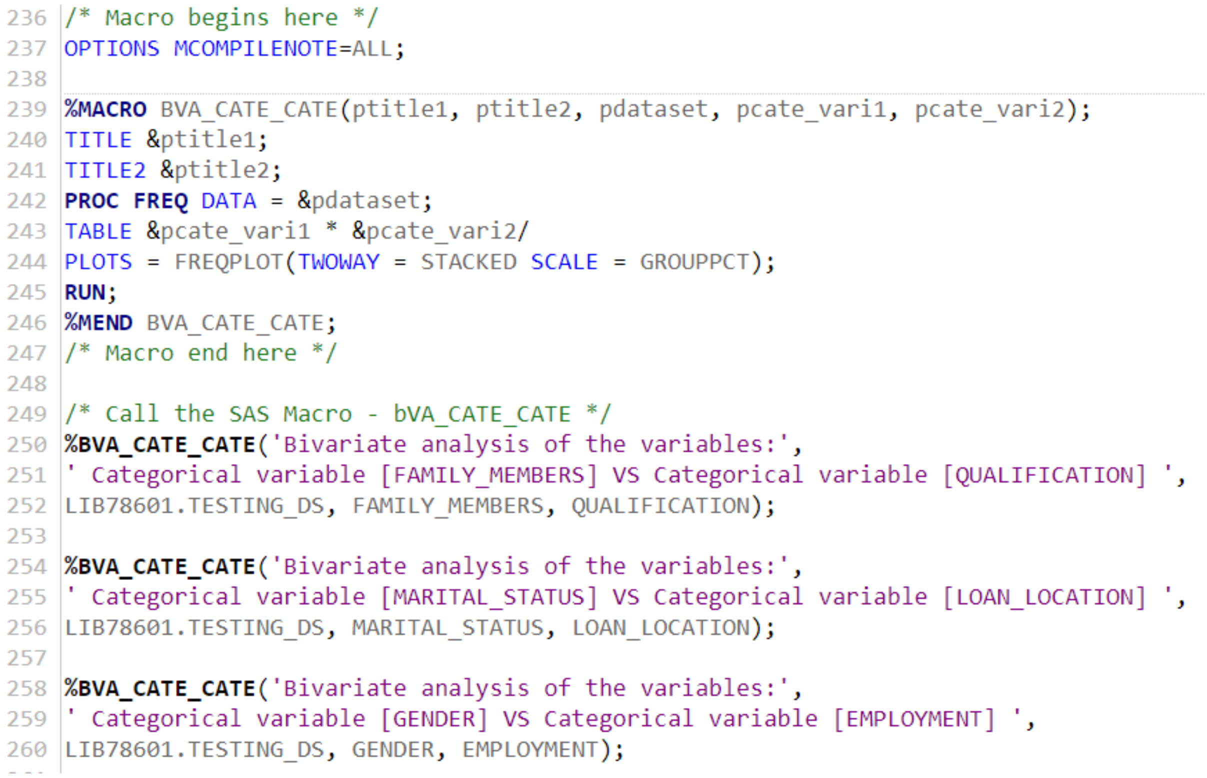
Description

We can observe the data of univariate analysis to continuous data using SAS Macro easily. The missing values for variables loan amount and loan duration are 5 and 6 respectively. There is total 367 of observations record in this dataset. For applicants’ income, the mean and standard deviation value are 4,805.60 and 4,910.69 respectively. The maximum of loan amount in this dataset will be 550 thousand.

6.3.2 Bivariate analysis of the variables found in the dataset – LIB78601.TESTING_DS.

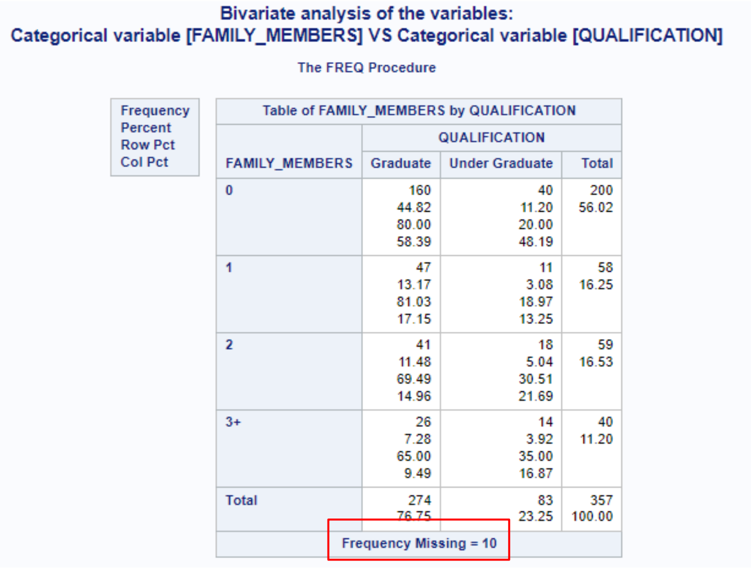
6.3.2.1 Bivariate analysis of the variables (Categorical vs Categorical) using the SAS MACRO.

SAS Code



```
/* Macro begins here */
OPTIONS MCOMPILENOTE=ALL;

%MACRO BVA_CATE_CATE(ptitle1, ptitle2, pdataset, pcate_vari1, pcate_vari2);
TITLE &ptitle1;
TITLE2 &ptitle2;
PROC FREQ DATA = &pdataset;
TABLE &pcate_vari1 * &pcate_vari2/
PLOTS = FREQPLOT(TWOWAY = STACKED SCALE = GROUPPCT);
RUN;
%MEND BVA_CATE_CATE;
/* Macro end here */

/* Call the SAS Macro - bVA_CATE_CATE */
%BVA_CATE_CATE('Bivariate analysis of the variables:',
' Categorical variable [FAMILY_MEMBERS] VS Categorical variable [QUALIFICATION] ',
LIB78601.TESTING_DS, FAMILY_MEMBERS, QUALIFICATION);

%BVA_CATE_CATE('Bivariate analysis of the variables:',
' Categorical variable [MARITAL_STATUS] VS Categorical variable [LOAN_LOCATION] ',
LIB78601.TESTING_DS, MARITAL_STATUS, LOAN_LOCATION);

%BVA_CATE_CATE('Bivariate analysis of the variables:',
' Categorical variable [GENDER] VS Categorical variable [EMPLOYMENT] ',
LIB78601.TESTING_DS, GENDER, EMPLOYMENT);
```

Screenshot(s)



Bivariate analysis of the variables:
Categorical variable [FAMILY_MEMBERS] VS Categorical variable [QUALIFICATION]

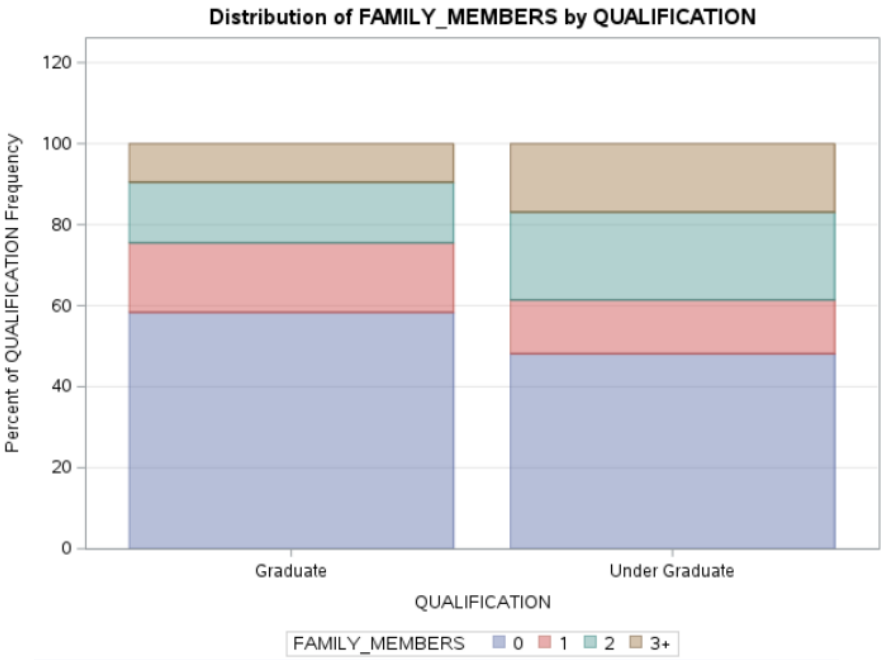
The FREQ Procedure

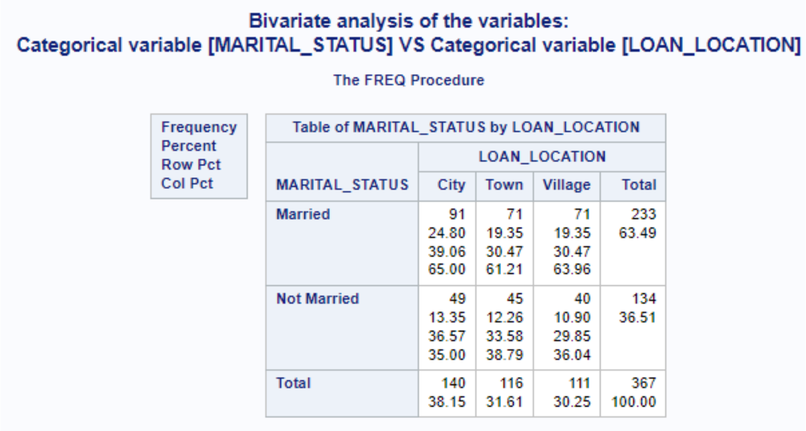
Frequency
Percent
Row Pct
Col Pct

Table of FAMILY_MEMBERS by QUALIFICATION

	QUALIFICATION		
FAMILY_MEMBERS	Graduate	Under Graduate	Total
0	160 44.82 80.00 58.39	40 11.20 20.00 48.19	200 56.02
1	47 13.17 81.03 17.15	11 3.08 18.97 13.25	58 16.25
2	41 11.48 69.49 14.96	18 5.04 30.51 21.69	59 16.53
3+	26 7.28 65.00 9.49	14 3.92 35.00 16.87	40 11.20
Total	274 76.75	83 23.25	357 100.00

Frequency Missing = 10





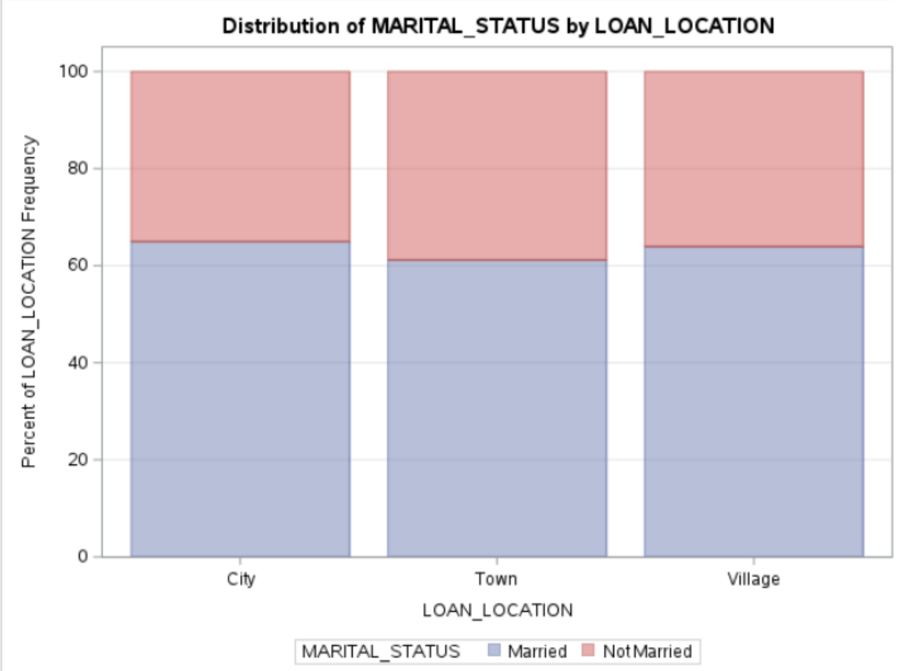
Bivariate analysis of the variables:
Categorical variable [MARITAL_STATUS] VS Categorical variable [LOAN_LOCATION]

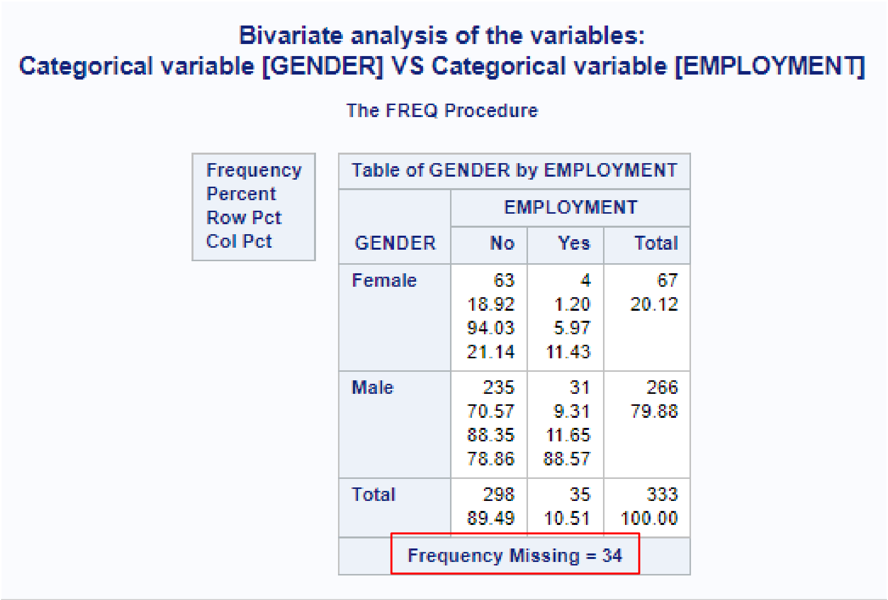
The FREQ Procedure

Frequency Percent Row Pct Col Pct

Table of MARITAL_STATUS by LOAN_LOCATION

	LOAN_LOCATION			
MARITAL_STATUS	City	Town	Village	Total
Married	91 24.80 39.06 65.00	71 19.35 30.47 61.21	71 19.35 30.47 63.96	233 63.49
Not Married	49 13.35 36.57 35.00	45 12.26 33.58 38.79	40 10.90 29.85 36.04	134 36.51
Total	140 38.15	116 31.61	111 30.25	367 100.00



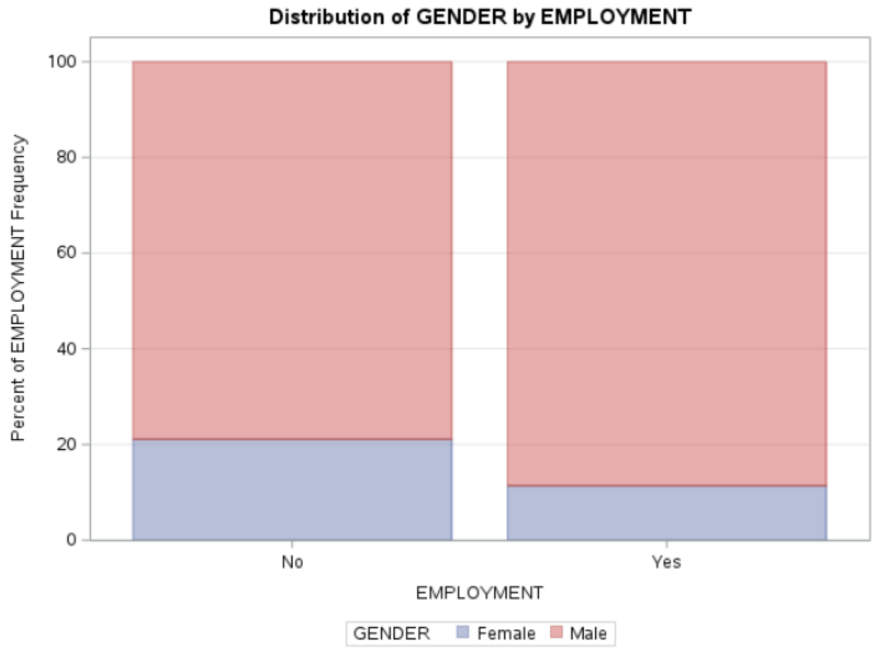


Bivariate analysis of the variables:
Categorical variable [GENDER] VS Categorical variable [EMPLOYMENT]

The FREQ Procedure

Frequency Percent Row Pct Col Pct

Table of GENDER by EMPLOYMENT			
	EMPLOYMENT		
GENDER	No	Yes	Total
Female	63 18.92 94.03 21.14	4 1.20 5.97 11.43	67 20.12
Male	235 70.57 88.35 78.86	31 9.31 11.65 88.57	266 79.88
Total	298 89.49	35 10.51	333 100.00
Frequency Missing = 34			

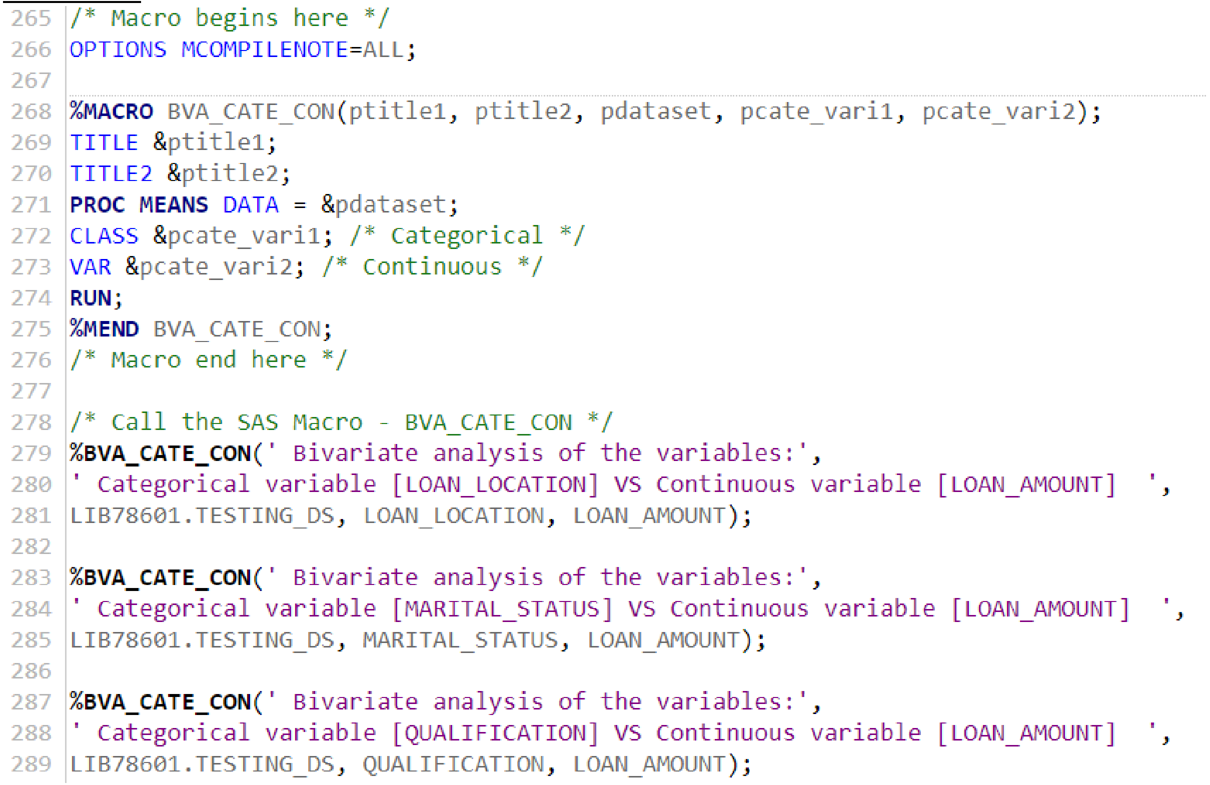


Description

We can observe the data of bivariate analysis to categorical data VS categorical data using SAS Macro easily. The missing values for first and third comparison are 10 and 34 respectively. From the first chart, we can observe than no matter applicants are married or not married, both groups of people have 0 family members. From the last chart in this output, we can see that most of the applicants are male, but male has higher employment percentage than female, vice versa.

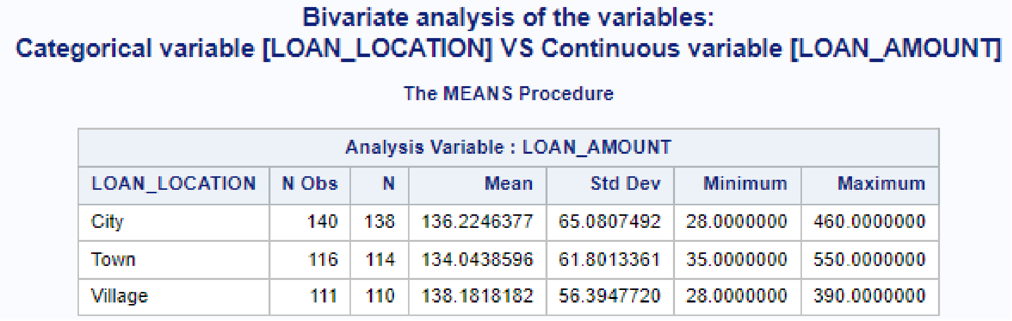
6.3.2.2 Bivariate analysis of the variables (Categorical vs Continuous) using the SAS MACRO.

SAS Code



```
/* Macro begins here */
OPTIONS MCOMPILENOTE=ALL;

%MACRO BVA_CATE_CON(ptitle1, ptitle2, pdataset, pcate_vari1, pcate_vari2);
TITLE &ptitle1;
TITLE2 &ptitle2;
PROC MEANS DATA = &pdataset;
CLASS &pcate_vari1; /* Categorical */
VAR &pcate_vari2; /* Continuous */
RUN;
%MEND BVA_CATE_CON;
/* Macro end here */

/* Call the SAS Macro - BVA_CATE_CON */
%BVA_CATE_CON(' Bivariate analysis of the variables:',
' Categorical variable [LOAN_LOCATION] VS Continuous variable [LOAN_AMOUNT]  ',
LIB78601.TESTING_DS, LOAN_LOCATION, LOAN_AMOUNT);

%BVA_CATE_CON(' Bivariate analysis of the variables:',
' Categorical variable [MARITAL_STATUS] VS Continuous variable [LOAN_AMOUNT]  ',
LIB78601.TESTING_DS, MARITAL_STATUS, LOAN_AMOUNT);

%BVA_CATE_CON(' Bivariate analysis of the variables:',
' Categorical variable [QUALIFICATION] VS Continuous variable [LOAN_AMOUNT]  ',
LIB78601.TESTING_DS, QUALIFICATION, LOAN_AMOUNT);
```

Screenshot(s)

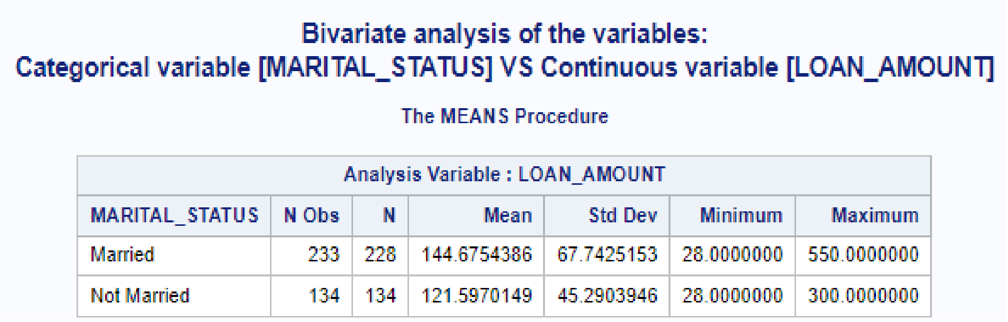


Bivariate analysis of the variables:
Categorical variable [LOAN_LOCATION] VS Continuous variable [LOAN_AMOUNT]

The MEANS Procedure

Analysis Variable : LOAN_AMOUNT

LOAN_LOCATION	N Obs	N	Mean	Std Dev	Minimum	Maximum
City	140	138	136.2246377	65.0807492	28.0000000	460.0000000
Town	116	114	134.0438596	61.8013361	35.0000000	550.0000000
Village	111	110	138.1818182	56.3947720	28.0000000	390.0000000

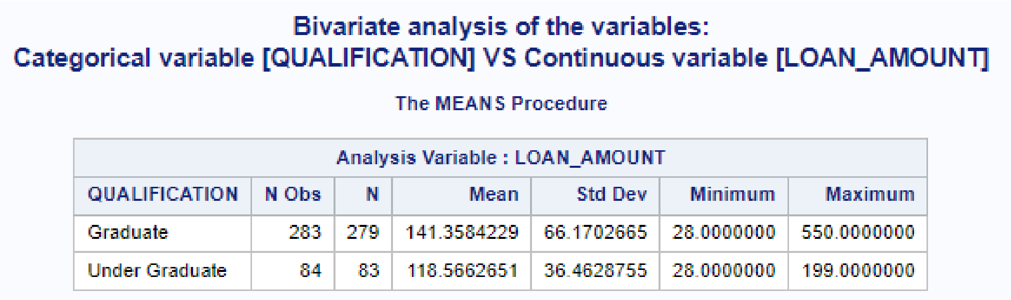


Bivariate analysis of the variables:
Categorical variable [MARITAL_STATUS] VS Continuous variable [LOAN_AMOUNT]

The MEANS Procedure

Analysis Variable : LOAN_AMOUNT

MARITAL_STATUS	N Obs	N	Mean	Std Dev	Minimum	Maximum
Married	233	228	144.6754386	67.7425153	28.0000000	550.0000000
Not Married	134	134	121.5970149	45.2903946	28.0000000	300.0000000



Bivariate analysis of the variables:
Categorical variable [QUALIFICATION] VS Continuous variable [LOAN_AMOUNT]

The MEANS Procedure

Analysis Variable : LOAN_AMOUNT						
QUALIFICATION	N Obs	N	Mean	Std Dev	Minimum	Maximum
Graduate	283	279	141.3584229	66.1702665	28.0000000	550.0000000
Under Graduate	84	83	118.5662651	36.4628755	28.0000000	199.0000000

Description

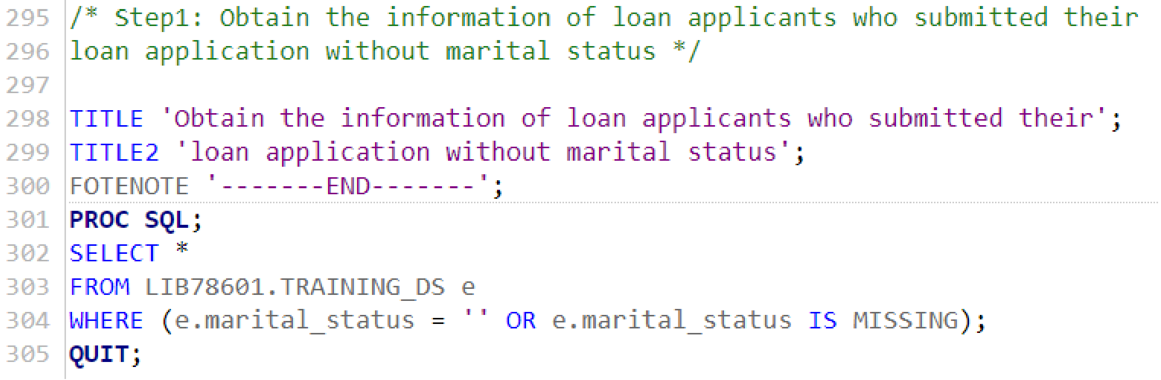
Here is the bivariate analysis of categorical variable and continuous variable. From the loan location VS loan amount, we can see that the mean value of each category has no very huge difference, but the standard deviation for each category has difference. For marital status VS loan amount, the mean value and standard for both group of applicants have big difference. This issue also happened for qualification VS loan amount.

6.4 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TRAINING_DS.

6.4.1 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TRAINING_DS – [MARITAL_STATUS]

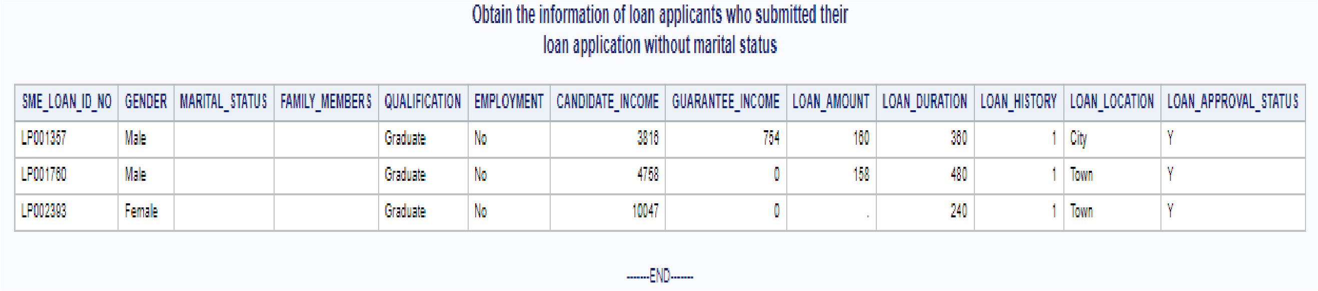
Step1: Obtain the information.

SAS Code



```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without marital status */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without marital status';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.marital_status = '' OR e.marital_status IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without marital status

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001357	Male			Graduate	No	3816	754	160	360	1	City	Y
LP001760	Male			Graduate	No	4758	0	158	480	1	Town	Y
LP002393	Female			Graduate	No	10047	0	.	240	1	Town	Y

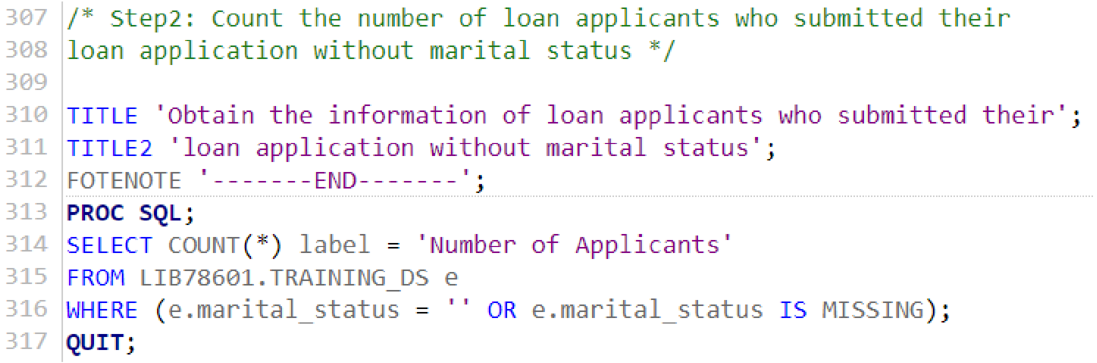
-------END-------

Description

From the screenshot, we can easily to see the full details of applicants that have missing values for marital status.

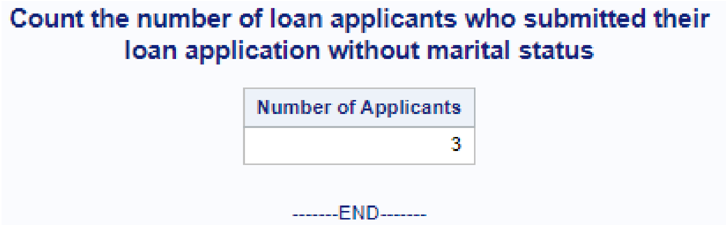
Step2: Count the number of missing values.

SAS Code



```
/* Step2: Count the number of loan applicants who submitted their
loan application without marital status */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without marital status';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.marital_status = '' OR e.marital_status IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their loan application without marital status

Number of Applicants
3

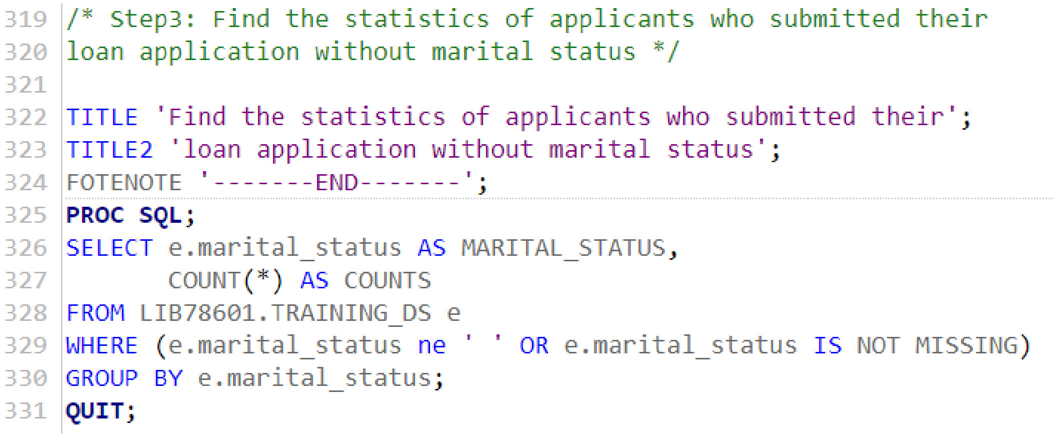
-------END-------

Description

Clearly, we can see that there is total 3 missing values for this variable, we will treat this issue in the coming steps.

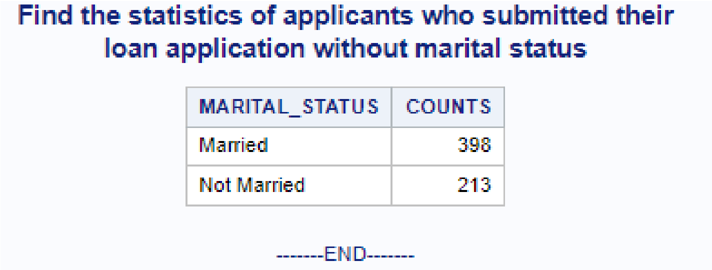
Step3: Statistics about missing values.

SAS Code



```
/* Step3: Find the statistics of applicants who submitted their
loan application without marital status */

TITLE 'Find the statistics of applicants who submitted their';
TITLE2 'loan application without marital status';
FOTENOTE '-------END-------';
PROC SQL;
SELECT e.marital_status AS MARITAL_STATUS,
       COUNT(*) AS COUNTS
FROM LIB78601.TRAINING_DS e
WHERE (e.marital_status ne ' ' OR e.marital_status IS NOT MISSING)
GROUP BY e.marital_status;
QUIT;
```

Screenshot(s)



Find the statistics of applicants who submitted their loan application without marital status

MARITAL_STATUS	COUNTS
Married	398
Not Married	213

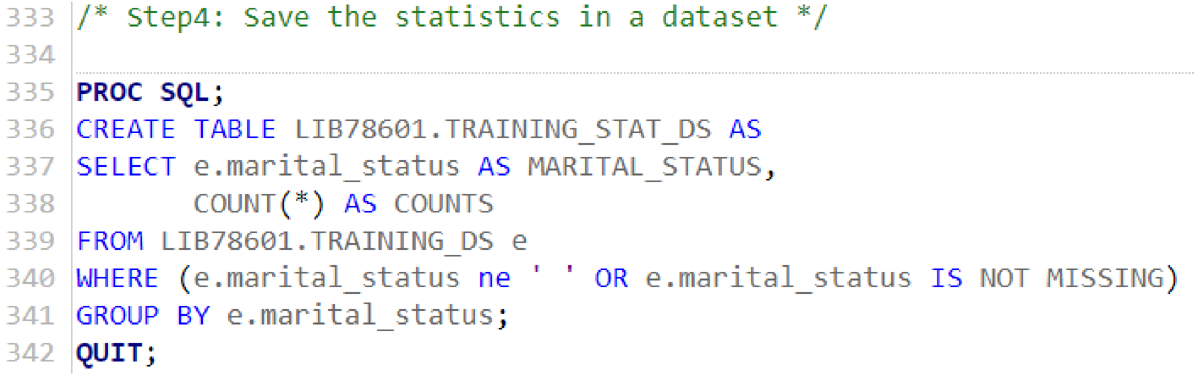
-------END-------

Description

We can observe that the number of each category for variable marital status, this information is useful for choosing treatment method for impute missing values.

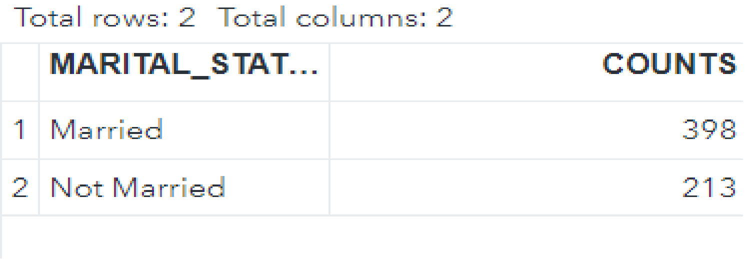
Step4: Save the statistics in a dataset.

SAS Code



```
/* Step4: Save the statistics in a dataset */

PROC SQL;
CREATE TABLE LIB78601.TRAINING_STAT_DS AS
SELECT e.marital_status AS MARITAL_STATUS,
       COUNT(*) AS COUNTS
FROM LIB78601.TRAINING_DS e
WHERE (e.marital_status ne ' ' OR e.marital_status IS NOT MISSING)
GROUP BY e.marital_status;
QUIT;
```

Screenshot(s)



Total rows: 2 Total columns: 2

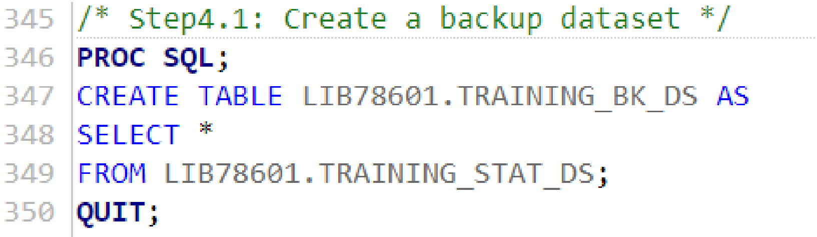
	MARITAL_STAT...	COUNTS
1	Married	398
2	Not Married	213

Description

Save the output from last output as a table format, this will help for selecting specific value for impute missing value for specific variable in general.

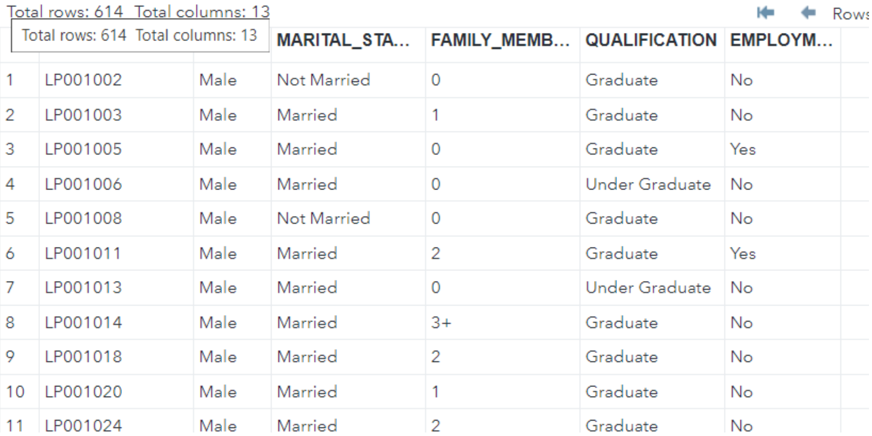
Step4.1: Backup the existing dataset.

SAS Code



```
/* Step4.1: Create a backup dataset */
PROC SQL;
CREATE TABLE LIB78601.TRAINING_BK_DS AS
SELECT *
FROM LIB78601.TRAINING_STAT_DS;
QUIT;
```

Screenshot(s)



Total rows: 614 Total columns: 13

Total rows: 614 Total columns: 13

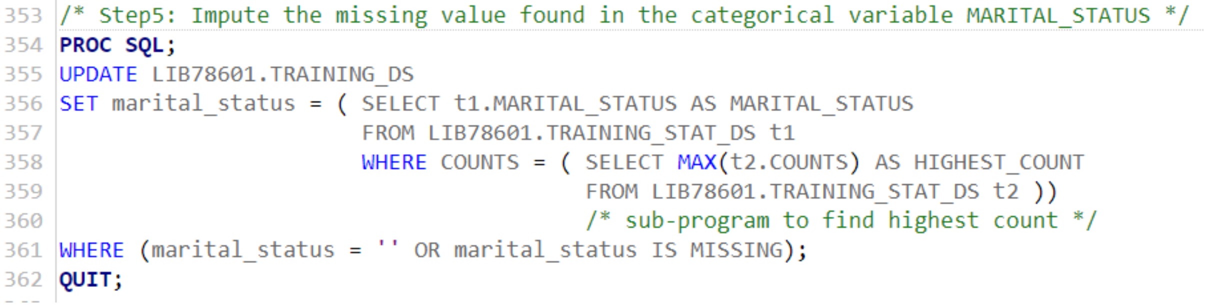
			MARITAL_STA...	FAMILY_MEMB...	QUALIFICATION	EMPLOYM...
1	LP001002	Male	Not Married	0	Graduate	No
2	LP001003	Male	Married	1	Graduate	No
3	LP001005	Male	Married	0	Graduate	Yes
4	LP001006	Male	Married	0	Under Graduate	No
5	LP001008	Male	Not Married	0	Graduate	No
6	LP001011	Male	Married	2	Graduate	Yes
7	LP001013	Male	Married	0	Under Graduate	No
8	LP001014	Male	Married	3+	Graduate	No
9	LP001018	Male	Married	2	Graduate	No
10	LP001020	Male	Married	1	Graduate	No
11	LP001024	Male	Married	2	Graduate	No

Description

Create a table for duplicate the table before imputing the value for missing values cell. This action will guarantee that we will not corrupt the existing table.

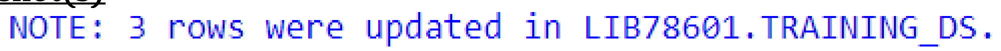
Step5: Impute the missing values.

SAS Code



```
/* Step5: Impute the missing value found in the categorical variable MARITAL_STATUS */
PROC SQL;
UPDATE LIB78601.TRAINING_DS
SET marital_status = ( SELECT t1.MARITAL_STATUS AS MARITAL_STATUS
                       FROM LIB78601.TRAINING_STAT_DS t1
                       WHERE COUNTS = ( SELECT MAX(t2.COUNTS) AS HIGHEST_COUNT
                                        FROM LIB78601.TRAINING_STAT_DS t2 ))
                                        /* sub-program to find highest count */
WHERE (marital_status = '' OR marital_status IS MISSING);
QUIT;
```

Screenshot(s)



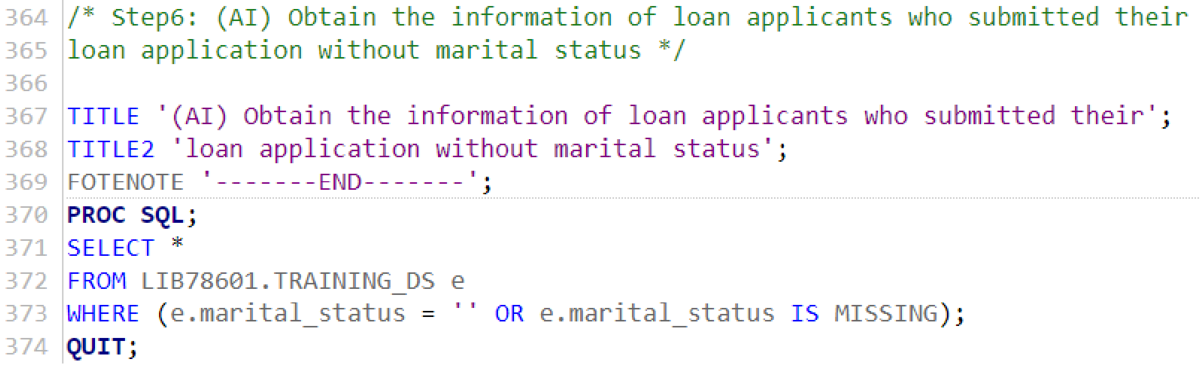
```
NOTE: 3 rows were updated in LIB78601.TRAINING_DS.
```

Description

Impute the missing values by the mode of the observation from its variable. From the screenshot, we can see that 3 rows were updated.

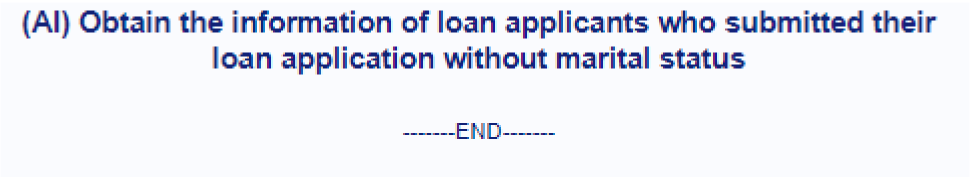
Step6: Ensure that the missing values was imputed.

SAS Code



```
/* Step6: (AI) Obtain the information of loan applicants who submitted their
loan application without marital status */

TITLE '(AI) Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without marital status';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.marital_status = '' OR e.marital_status IS MISSING);
QUIT;
```

Screenshot(s)



(AI) Obtain the information of loan applicants who submitted their
loan application without marital status

-------END-------

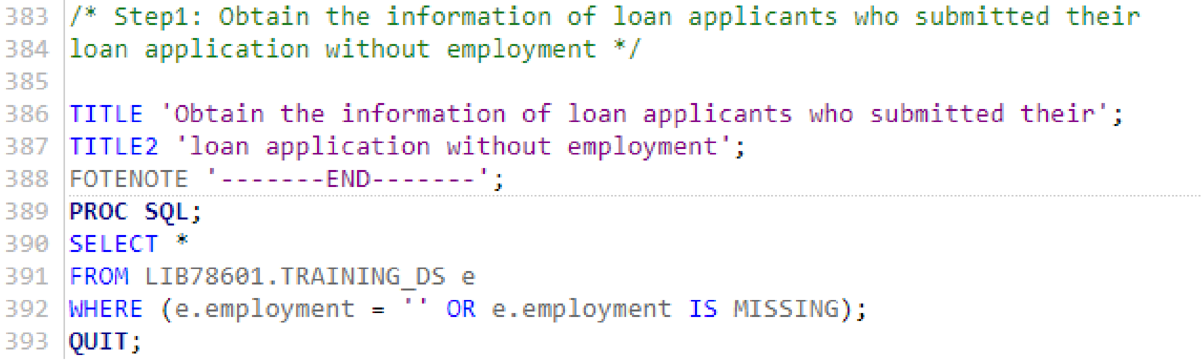
Description

We checked that there is not any missing value for this variable after impute the existing missing values.

6.4.2 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TRAINING_DS – [EMPLOYMENT]

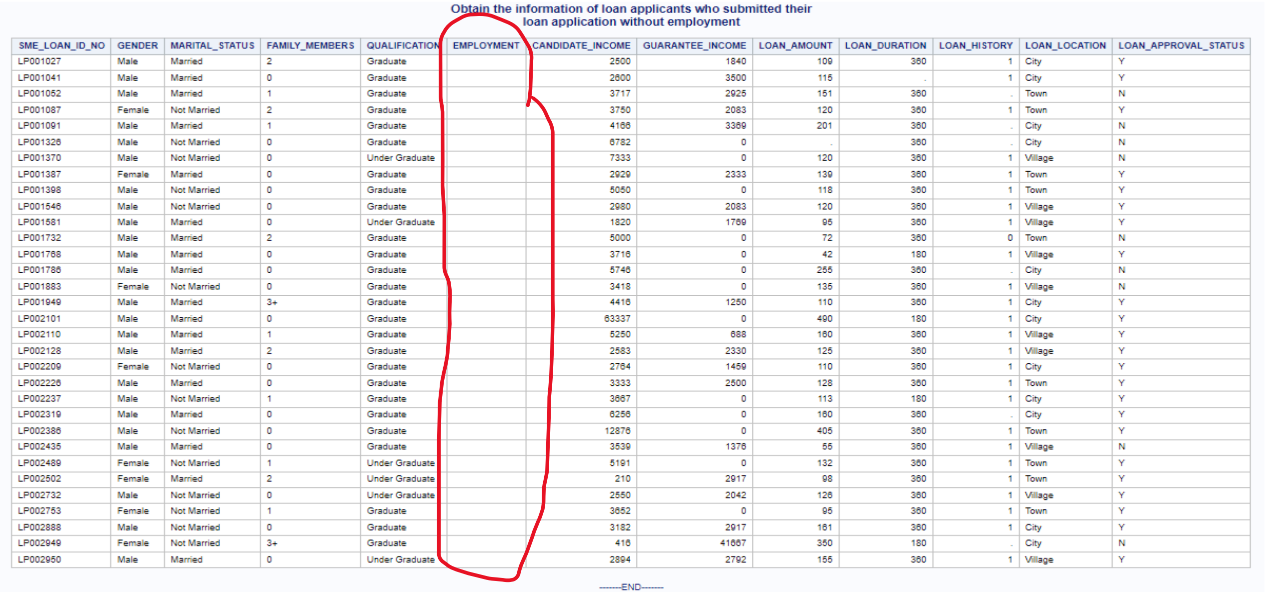
Step1: Obtain the information.

SAS Code



```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without employment */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without employment';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.employment = '' OR e.employment IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without employment

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001027	Male	Married	2	Graduate		2500	1840	109	360	1	City	Y
LP001041	Male	Married	0	Graduate		2600	3500	115	.	1	City	Y
LP001052	Male	Married	1	Graduate		3717	2925	151	360	.	Town	N
LP001087	Female	Not Married	2	Graduate		3750	2083	120	360	1	Town	Y
LP001091	Male	Married	1	Graduate		4166	3369	201	360	.	City	N
LP001326	Male	Not Married	0	Graduate		6782	0	.	360	.	City	N
LP001370	Male	Not Married	0	Under Graduate		7333	0	120	360	1	Village	N
LP001387	Female	Married	0	Graduate		2929	2333	139	360	1	Town	Y
LP001398	Male	Not Married	0	Graduate		5050	0	118	360	1	Town	Y
LP001546	Male	Not Married	0	Graduate		2980	2083	120	360	1	Village	Y
LP001581	Male	Married	0	Under Graduate		1820	1769	95	360	1	Village	Y
LP001732	Male	Married	2	Graduate		5000	0	72	360	0	Town	N
LP001768	Male	Married	0	Graduate		3716	0	42	180	1	Village	Y
LP001786	Male	Married	0	Graduate		5746	0	255	360	.	City	N
LP001883	Female	Not Married	0	Graduate		3418	0	135	360	1	Village	N
LP001949	Male	Married	3+	Graduate		4416	1250	110	360	1	City	Y
LP002101	Male	Married	0	Graduate		63337	0	490	180	1	City	Y
LP002110	Male	Married	1	Graduate		5250	688	160	360	1	Village	Y
LP002128	Male	Married	2	Graduate		2583	2330	125	360	1	Village	Y
LP002209	Female	Not Married	0	Graduate		2764	1459	110	360	1	City	Y
LP002226	Male	Married	0	Graduate		3333	2500	128	360	1	Town	Y
LP002237	Male	Not Married	1	Graduate		3667	0	113	180	1	City	Y
LP002319	Male	Married	0	Graduate		6256	0	160	360	.	City	Y
LP002386	Male	Not Married	0	Graduate		12876	0	405	360	1	Town	Y
LP002435	Male	Married	0	Graduate		3539	1376	55	360	1	Village	N
LP002489	Female	Not Married	1	Under Graduate		5191	0	132	360	1	Town	Y
LP002502	Female	Married	2	Under Graduate		210	2917	98	360	1	Town	Y
LP002732	Male	Not Married	0	Under Graduate		2550	2042	126	360	1	Village	Y
LP002753	Female	Not Married	1	Graduate		3652	0	95	360	1	Town	Y
LP002888	Male	Not Married	0	Graduate		3182	2917	161	360	1	City	Y
LP002949	Female	Not Married	3+	Graduate		416	41667	350	180	.	City	N
LP002950	Male	Married	0	Under Graduate		2894	2792	155	360	1	Village	Y

--------END--------

Description

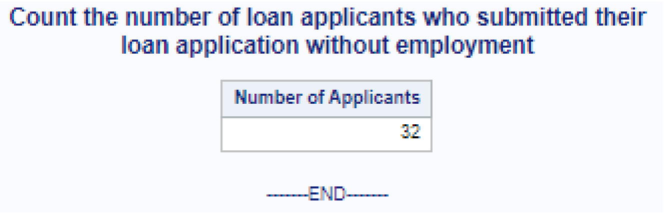
From the screenshot, we can easily to see the full details of applicants that have missing values for employment.

Step2: Count the number of missing values.

SAS Code

```
/* Step2: Count the number of loan applicants who submitted their
loan application without employment */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without employment';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.employment = '' OR e.employment IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their
loan application without employment

Number of Applicants
32

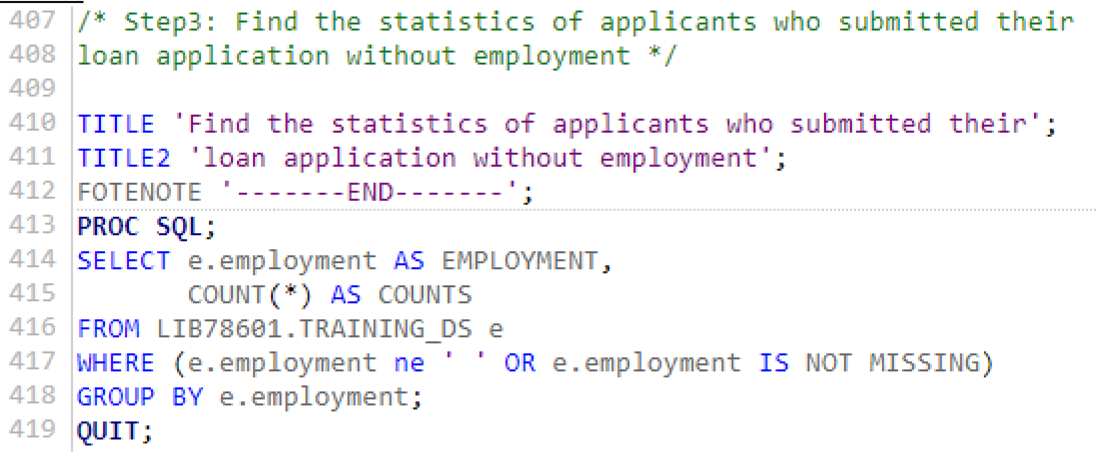
--------END--------

Description

Clearly, we can see that there is total 3 missing values for this variable, we will treat this issue in the coming steps.

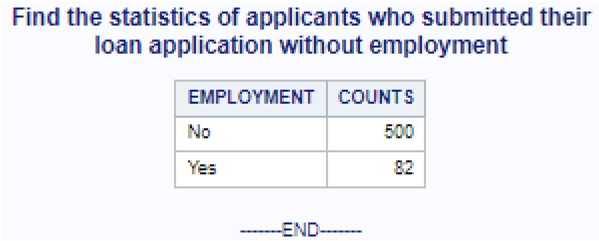
Step3: Statistics about missing values.

SAS Code



```
/* Step3: Find the statistics of applicants who submitted their
loan application without employment */

TITLE 'Find the statistics of applicants who submitted their';
TITLE2 'loan application without employment';
FOTENOTE '-------END-------';
PROC SQL;
SELECT e.employment AS EMPLOYMENT,
       COUNT(*) AS COUNTS
FROM LIB78601.TRAINING_DS e
WHERE (e.employment ne ' ' OR e.employment IS NOT MISSING)
GROUP BY e.employment;
QUIT;
```

Screenshot(s)



Find the statistics of applicants who submitted their
loan application without employment

EMPLOYMENT	COUNTS
No	500
Yes	82

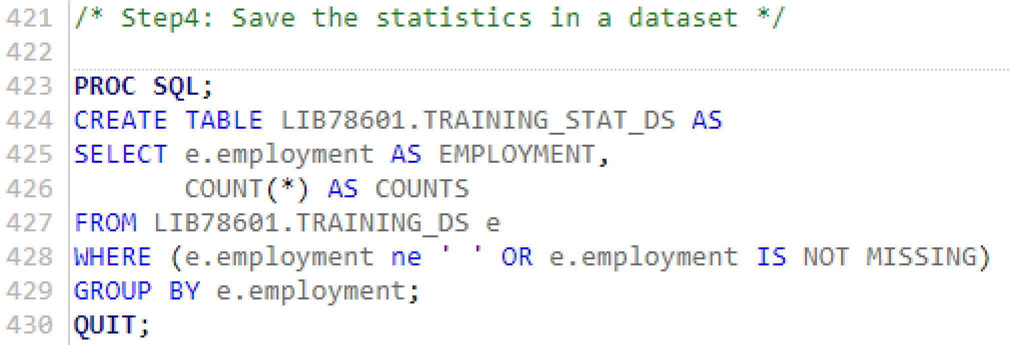
-------END-------

Description

We can observe that the number of each category for variable employment, this information is useful for choosing treatment method for impute missing values.

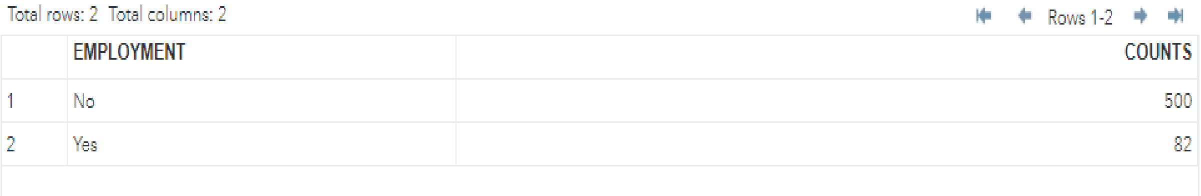
Step4: Save the statistics in a dataset.

SAS Code



```
/* Step4: Save the statistics in a dataset */

PROC SQL;
CREATE TABLE LIB78601.TRAINING_STAT_DS AS
SELECT e.employment AS EMPLOYMENT,
       COUNT(*) AS COUNTS
FROM LIB78601.TRAINING_DS e
WHERE (e.employment ne ' ' OR e.employment IS NOT MISSING)
GROUP BY e.employment;
QUIT;
```

Screenshot(s)



Total rows: 2 Total columns: 2

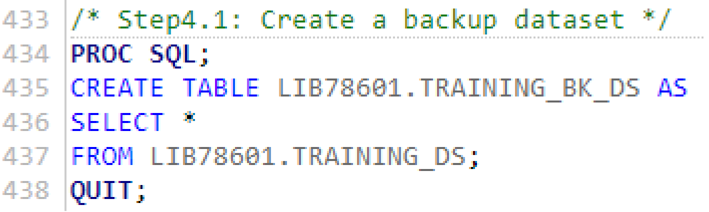
	EMPLOYMENT	COUNTS
1	No	500
2	Yes	82

Description

Save the output from last output as a table format, this will help for selecting specific value for impute missing value for specific variable in general.

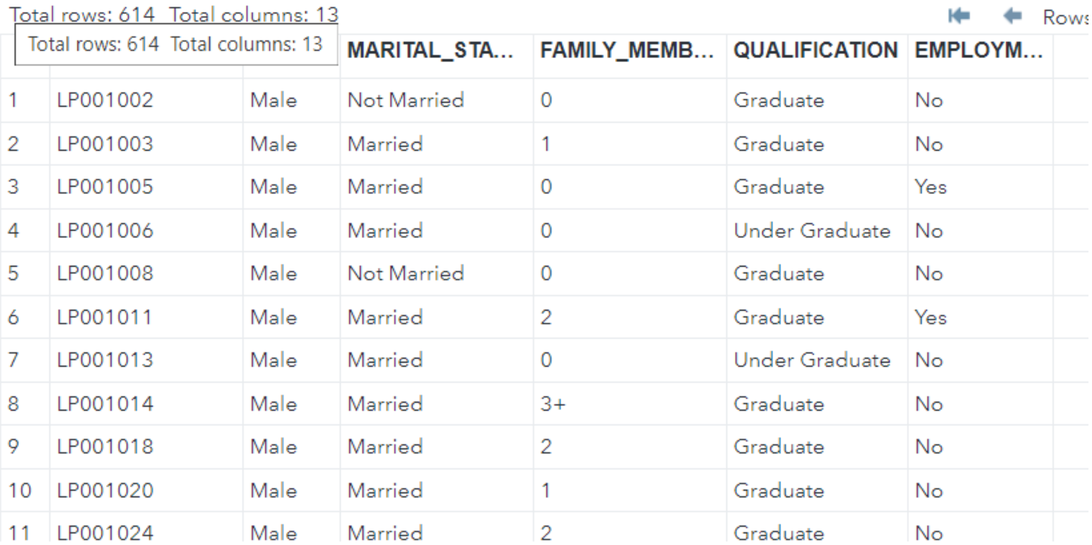
Step4.1: Backup the existing dataset.

SAS Code



```
/* Step4.1: Create a backup dataset */
PROC SQL;
CREATE TABLE LIB78601.TRAINING_BK_DS AS
SELECT *
FROM LIB78601.TRAINING_DS;
QUIT;
```

Screenshot(s)



Total rows: 614 Total columns: 13

Total rows: 614 Total columns: 13

Rows

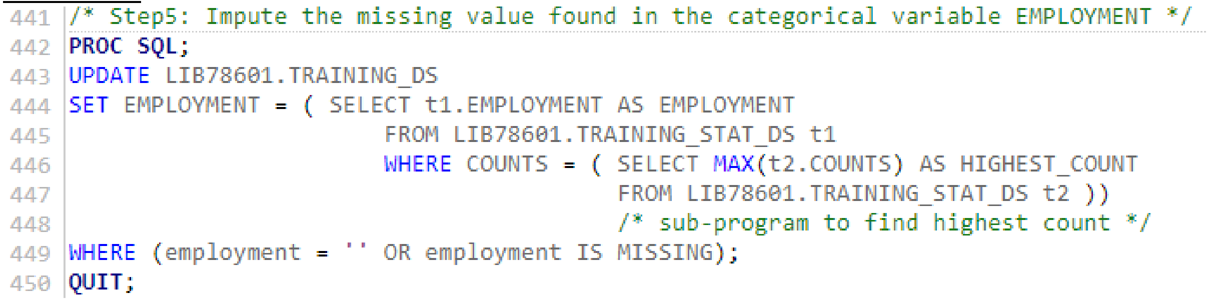
			MARITAL_STA...	FAMILY_MEMB...	QUALIFICATION	EMPLOYM...
1	LP001002	Male	Not Married	0	Graduate	No
2	LP001003	Male	Married	1	Graduate	No
3	LP001005	Male	Married	0	Graduate	Yes
4	LP001006	Male	Married	0	Under Graduate	No
5	LP001008	Male	Not Married	0	Graduate	No
6	LP001011	Male	Married	2	Graduate	Yes
7	LP001013	Male	Married	0	Under Graduate	No
8	LP001014	Male	Married	3+	Graduate	No
9	LP001018	Male	Married	2	Graduate	No
10	LP001020	Male	Married	1	Graduate	No
11	LP001024	Male	Married	2	Graduate	No

Description

Create a table for duplicate the table before imputing the value for missing values cell. This action will guarantee that we will not corrupt the existing table.

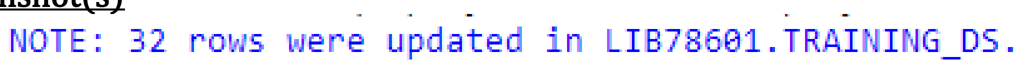
Step5: Impute the missing values.

SAS Code



```
/* Step5: Impute the missing value found in the categorical variable EMPLOYMENT */
PROC SQL;
UPDATE LIB78601.TRAINING_DS
SET EMPLOYMENT = ( SELECT t1.EMPLOYMENT AS EMPLOYMENT
                       FROM LIB78601.TRAINING_STAT_DS t1
                       WHERE COUNTS = ( SELECT MAX(t2.COUNTS) AS HIGHEST_COUNT
                                        FROM LIB78601.TRAINING_STAT_DS t2 ))
                                        /* sub-program to find highest count */
WHERE (employment = '' OR employment IS MISSING);
QUIT;
```

Screenshot(s)



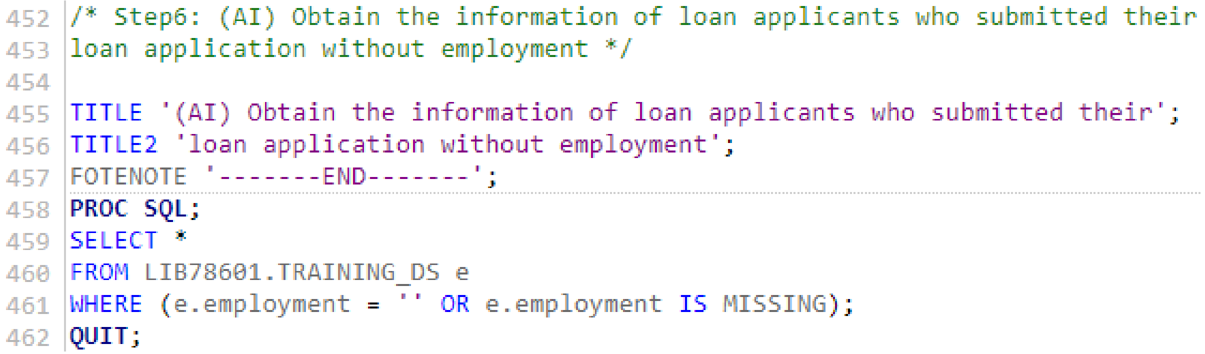
```
NOTE: 32 rows were updated in LIB78601.TRAINING_DS.
```

Description

Impute the missing values by the mode of the observation from its variable. From the screenshot, we can see that 32 rows were updated.

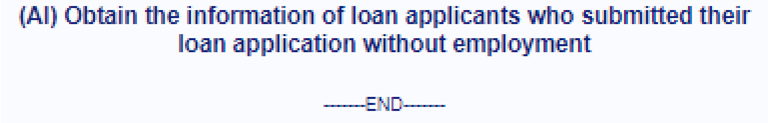
Step6: Ensure that the missing values was imputed.

SAS Code



```
/* Step6: (AI) Obtain the information of loan applicants who submitted their
loan application without employment */

TITLE '(AI) Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without employment';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.employment = '' OR e.employment IS MISSING);
QUIT;
```

Screenshot(s)



(AI) Obtain the information of loan applicants who submitted their
loan application without employment

-------END-------

Description

We checked that there is not any missing value for this variable after impute the existing missing values.

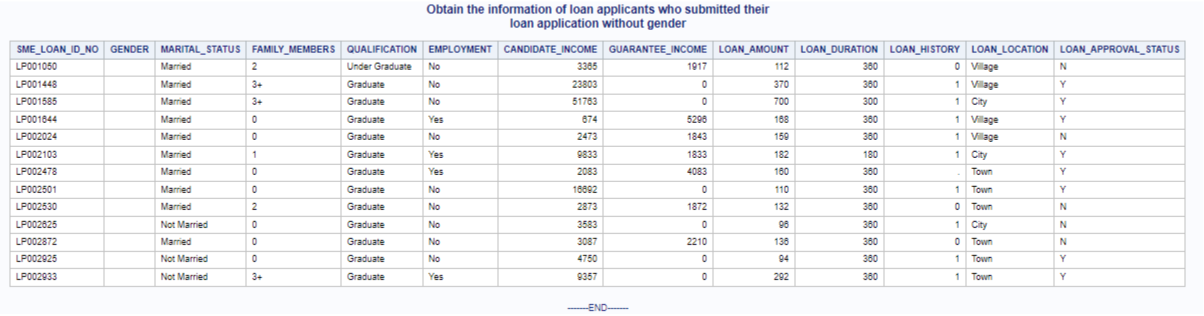
6.4.3 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TRAINING_DS – [GENDER]

Step1: Obtain the information.

SAS Code

```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without gender */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without gender';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.gender = '' OR e.gender IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without gender

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001050		Married	2	Under Graduate	No	3365	1917	112	360	0	Village	N
LP001448		Married	3+	Graduate	No	23803	0	370	360	1	Village	Y
LP001585		Married	3+	Graduate	No	51763	0	700	300	1	City	Y
LP001644		Married	0	Graduate	Yes	674	5296	168	360	1	Village	Y
LP002024		Married	0	Graduate	No	2473	1843	159	360	1	Village	N
LP002103		Married	1	Graduate	Yes	9833	1833	182	180	1	City	Y
LP002478		Married	0	Graduate	Yes	2083	4083	160	360	.	Town	Y
LP002501		Married	0	Graduate	No	16692	0	110	360	1	Town	Y
LP002530		Married	2	Graduate	No	2873	1872	132	360	0	Town	N
LP002625		Not Married	0	Graduate	No	3583	0	96	360	1	City	N
LP002872		Married	0	Graduate	No	3087	2210	136	360	0	Town	N
LP002925		Not Married	0	Graduate	No	4750	0	94	360	1	Town	Y
LP002933		Not Married	3+	Graduate	Yes	9357	0	292	360	1	Town	Y

-------END-------

Description

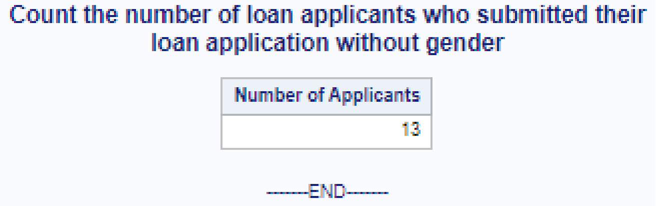
From the screenshot, we can easily to see the full details of applicants that have missing values for gender.

Step2: Count the number of missing values.

SAS Code

```
/* Step2: Count the number of loan applicants who submitted their
loan application without gender */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without gender';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.gender = '' OR e.gender IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their loan application without gender

Number of Applicants
13

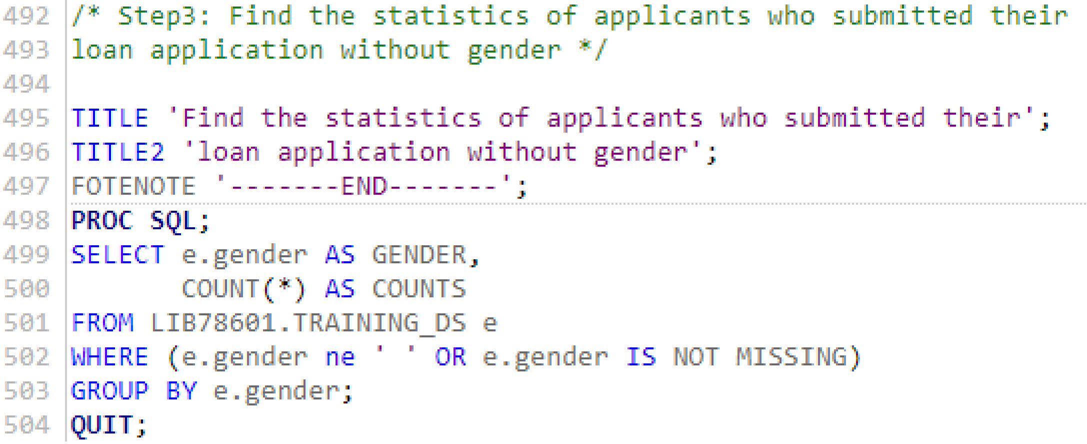
-------END-------

Description

Clearly, we can see that there is total 3 missing values for this variable, we will treat this issue in the coming steps.

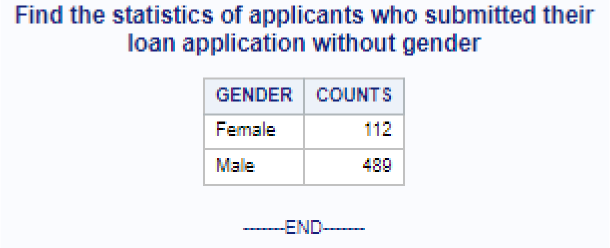
Step3: Statistics about missing values.

SAS Code



```
/* Step3: Find the statistics of applicants who submitted their
loan application without gender */

TITLE 'Find the statistics of applicants who submitted their';
TITLE2 'loan application without gender';
FOTENOTE '-------END-------';
PROC SQL;
SELECT e.gender AS GENDER,
       COUNT(*) AS COUNTS
FROM LIB78601.TRAINING_DS e
WHERE (e.gender ne ' ' OR e.gender IS NOT MISSING)
GROUP BY e.gender;
QUIT;
```

Screenshot(s)



Find the statistics of applicants who submitted their loan application without gender

GENDER	COUNTS
Female	112
Male	489

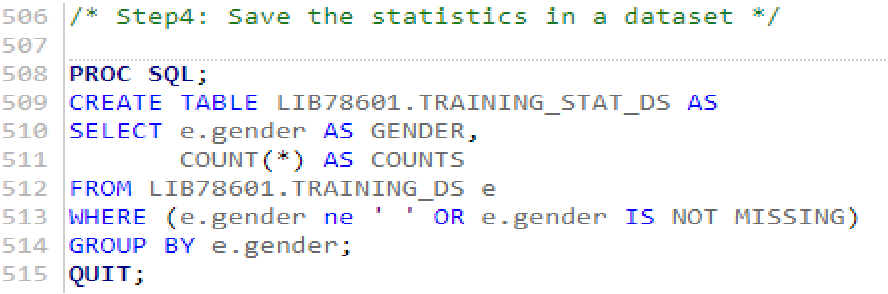
-------END-------

Description

We can observe that the number of each category for variable gender, this information is useful for choosing treatment method for impute missing values.

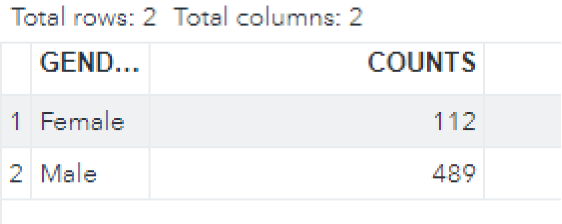
Step4: Save the statistics in a dataset.

SAS Code



```
/* Step4: Save the statistics in a dataset */

PROC SQL;
CREATE TABLE LIB78601.TRAINING_STAT_DS AS
SELECT e.gender AS GENDER,
       COUNT(*) AS COUNTS
FROM LIB78601.TRAINING_DS e
WHERE (e.gender ne ' ' OR e.gender IS NOT MISSING)
GROUP BY e.gender;
QUIT;
```

Screenshot(s)



Total rows: 2 Total columns: 2

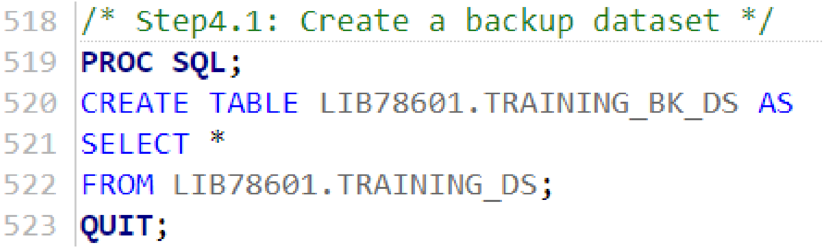
	GEND...	COUNTS
1	Female	112
2	Male	489

Description

Save the output from last output as a table format, this will help for selecting specific value for impute missing value for specific variable in general.

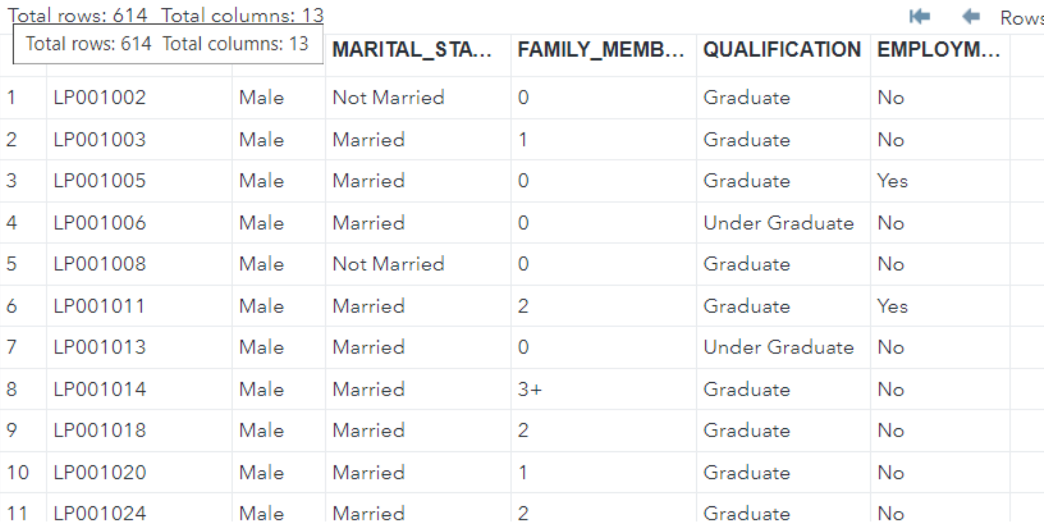
Step4.1: Backup the existing dataset.

SAS Code



```
/* Step4.1: Create a backup dataset */
PROC SQL;
CREATE TABLE LIB78601.TRAINING_BK_DS AS
SELECT *
FROM LIB78601.TRAINING_DS;
QUIT;
```

Screenshot(s)



Total rows: 614 Total columns: 13

Total rows: 614 Total columns: 13

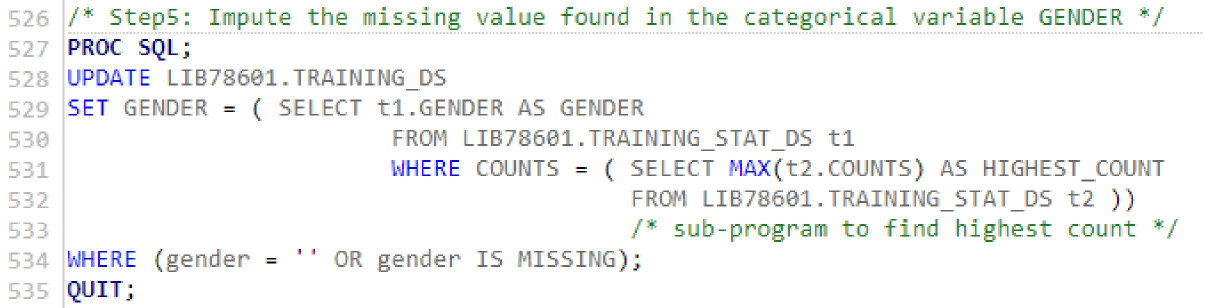
			MARITAL_STA...	FAMILY_MEMB...	QUALIFICATION	EMPLOYM...
1	LP001002	Male	Not Married	0	Graduate	No
2	LP001003	Male	Married	1	Graduate	No
3	LP001005	Male	Married	0	Graduate	Yes
4	LP001006	Male	Married	0	Under Graduate	No
5	LP001008	Male	Not Married	0	Graduate	No
6	LP001011	Male	Married	2	Graduate	Yes
7	LP001013	Male	Married	0	Under Graduate	No
8	LP001014	Male	Married	3+	Graduate	No
9	LP001018	Male	Married	2	Graduate	No
10	LP001020	Male	Married	1	Graduate	No
11	LP001024	Male	Married	2	Graduate	No

Description

Create a table for duplicate the table before imputing the value for missing values cell. This action will guarantee that we will not corrupt the existing table.

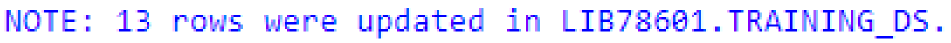
Step5: Impute the missing values.

SAS Code



```
/* Step5: Impute the missing value found in the categorical variable GENDER */
PROC SQL;
UPDATE LIB78601.TRAINING_DS
SET GENDER = ( SELECT t1.GENDER AS GENDER
                       FROM LIB78601.TRAINING_STAT_DS t1
                       WHERE COUNTS = ( SELECT MAX(t2.COUNTS) AS HIGHEST_COUNT
                                        FROM LIB78601.TRAINING_STAT_DS t2 ))
                                        /* sub-program to find highest count */
WHERE (gender = '' OR gender IS MISSING);
QUIT;
```

Screenshot(s)



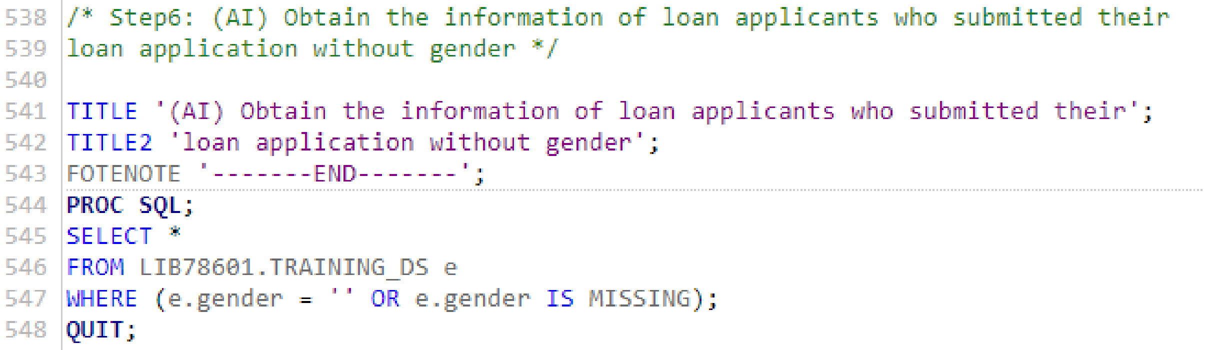
```
NOTE: 13 rows were updated in LIB78601.TRAINING_DS.
```

Description

Impute the missing values by the mode of the observation from its variable. From the screenshot, we can see that 13 rows were updated.

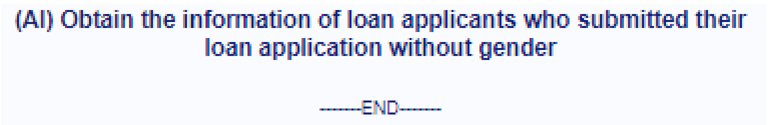
Step6: Ensure that the missing values was imputed.

SAS Code



```
/* Step6: (AI) Obtain the information of loan applicants who submitted their
loan application without gender */

TITLE '(AI) Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without gender';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.gender = '' OR e.gender IS MISSING);
QUIT;
```

Screenshot(s)



(AI) Obtain the information of loan applicants who submitted their
loan application without gender

-------END-------

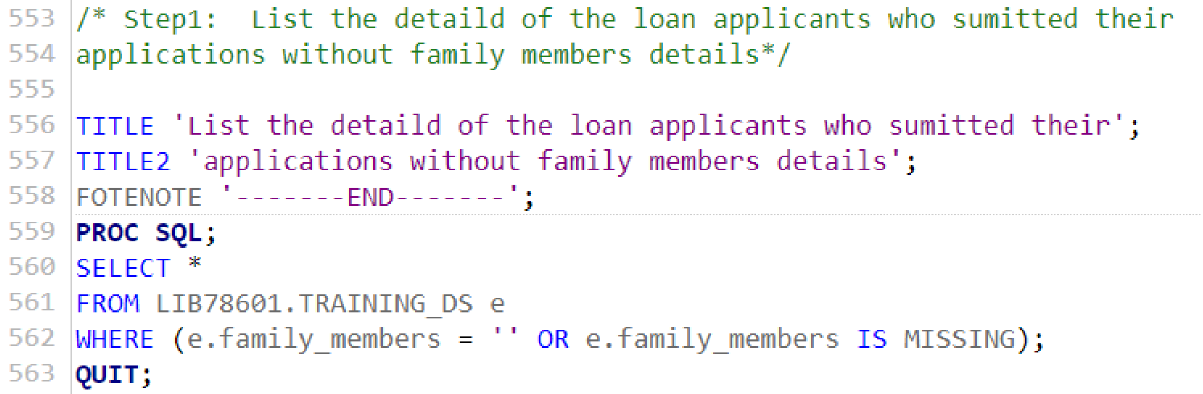
Description

We checked that there is not any missing value for this variable after imputing the existing missing values.

6.4.4 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TRAINING_DS – [FAMILY_MEMBERS]

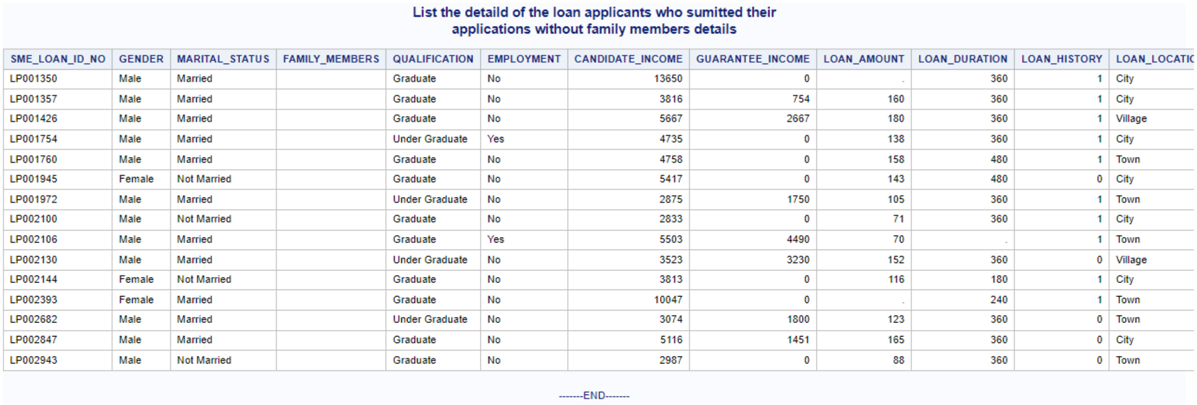
Step1: Obtain the information.

SAS Code



```
/* Step1:  List the detaild of the loan applicants who sumitted their
applications without family members details*/

TITLE 'List the detaild of the loan applicants who sumitted their';
TITLE2 'applications without family members details';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.family_members = '' OR e.family_members IS MISSING);
QUIT;
```

Screenshot(s)



List the detaild of the loan applicants who sumitted their
applications without family members details

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATIC
LP001350	Male	Married		Graduate	No	13650	0	.	360	1	City
LP001357	Male	Married		Graduate	No	3816	754	160	360	1	City
LP001426	Male	Married		Graduate	No	5667	2667	180	360	1	Village
LP001754	Male	Married		Under Graduate	Yes	4735	0	138	360	1	City
LP001760	Male	Married		Graduate	No	4758	0	158	480	1	Town
LP001945	Female	Not Married		Graduate	No	5417	0	143	480	0	City
LP001972	Male	Married		Under Graduate	No	2875	1750	105	360	1	Town
LP002100	Male	Not Married		Graduate	No	2833	0	71	360	1	City
LP002106	Male	Married		Graduate	Yes	5503	4490	70	.	1	Town
LP002130	Male	Married		Under Graduate	No	3523	3230	152	360	0	Village
LP002144	Female	Not Married		Graduate	No	3813	0	116	180	1	City
LP002393	Female	Married		Graduate	No	10047	0	.	240	1	Town
LP002682	Male	Married		Under Graduate	No	3074	1800	123	360	0	Town
LP002847	Male	Married		Graduate	No	5116	1451	165	360	0	City
LP002943	Male	Not Married		Graduate	No	2987	0	88	360	0	Town

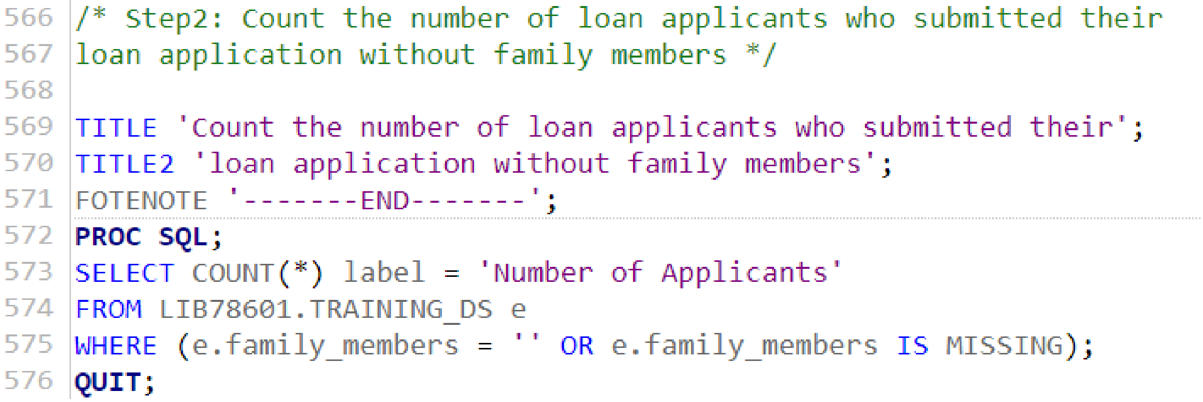
-------END-------

Description

From the screenshot, we can easily to see the full details of applicants that have missing values for family members.

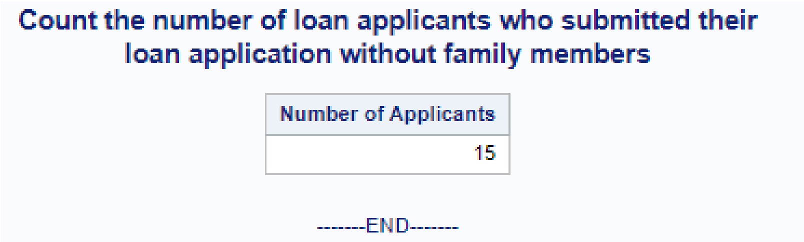
Step2: Count the number of missing values.

SAS Code



```
/* Step2: Count the number of loan applicants who submitted their
loan application without family members */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without family members';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.family_members = '' OR e.family_members IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their loan application without family members

Number of Applicants
15

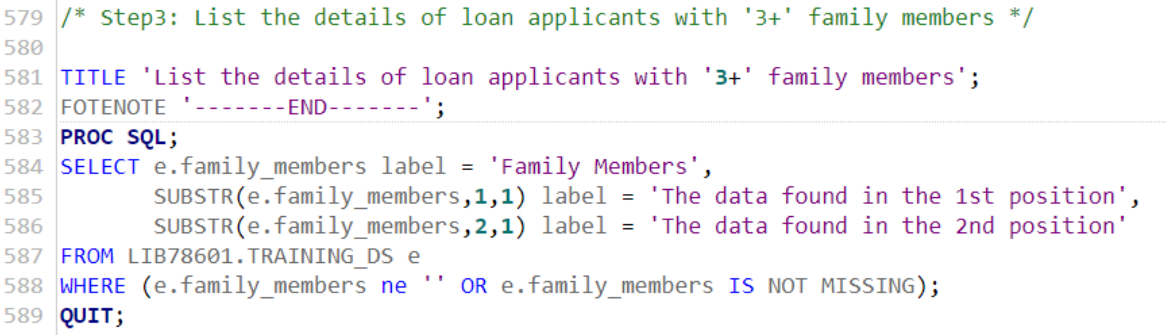
-------END-------

Description

Clearly, we can see that there is total 15 missing values for this variable, we will treat this issue in the coming steps.

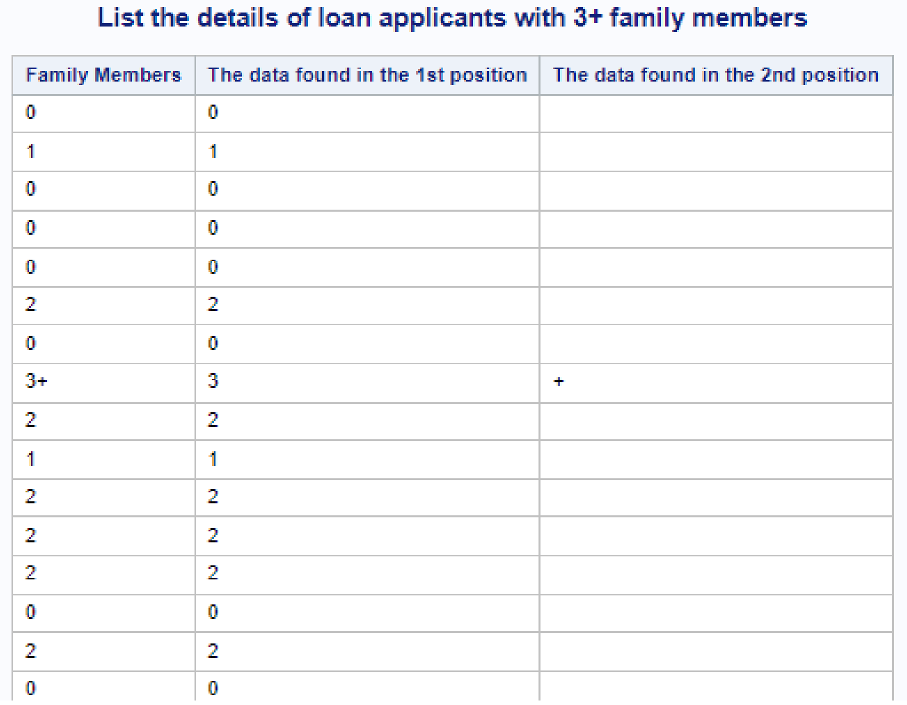
Step3: List the details of loan applicants with '3+' family members.

SAS Code



```
/* Step3: List the details of loan applicants with '3+' family members */

TITLE 'List the details of loan applicants with '3+' family members';
FOTENOTE '-------END-------';
PROC SQL;
SELECT e.family_members label = 'Family Members',
       SUBSTR(e.family_members,1,1) label = 'The data found in the 1st position',
       SUBSTR(e.family_members,2,1) label = 'The data found in the 2nd position'
FROM LIB78601.TRAINING_DS e
WHERE (e.family_members ne '' OR e.family_members IS NOT MISSING);
QUIT;
```

Screenshot(s)



List the details of loan applicants with 3+ family members

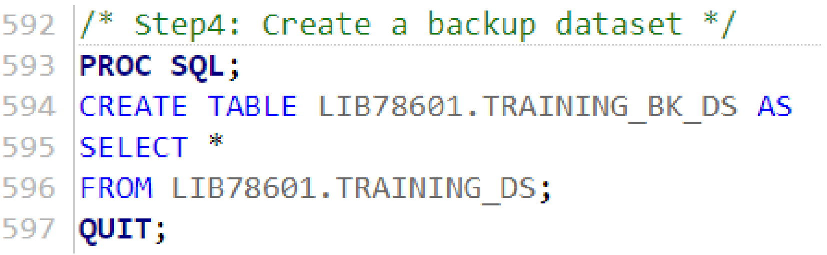
Family Members	The data found in the 1st position	The data found in the 2nd position
0	0	
1	1	
0	0	
0	0	
0	0	
2	2	
0	0	
3+	3	+
2	2	
1	1	
2	2	
2	2	
2	2	
0	0	
2	2	
0	0	

Description

By observing the data, we found that some of the observations of family members column are recorded as ‘3+’. Before doing analysis, performing separation the symbol ‘+’ to ensure that only numeric value remains is needed.

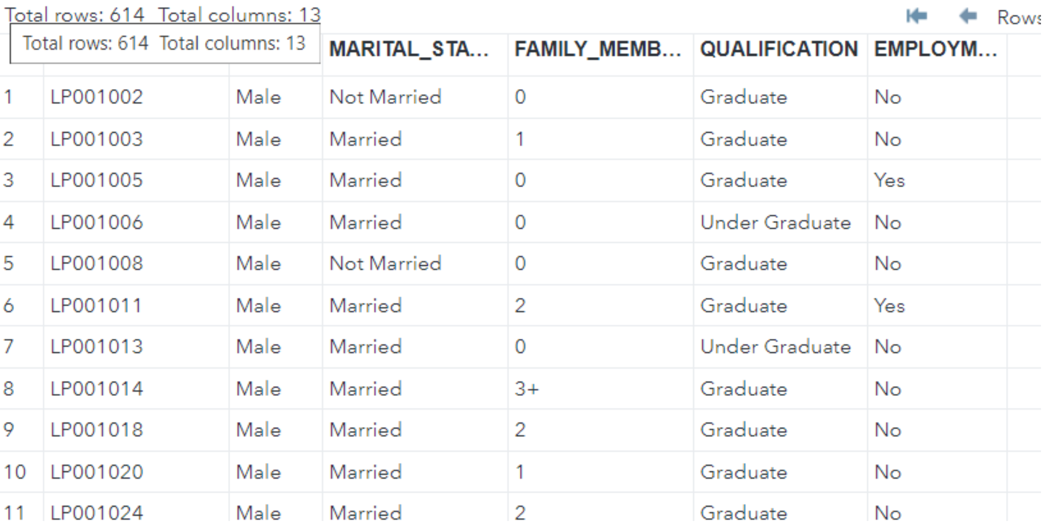
Step4: Backup the existing dataset.

SAS Code



```
/* Step4: Create a backup dataset */
PROC SQL;
CREATE TABLE LIB78601.TRAINING_BK_DS AS
SELECT *
FROM LIB78601.TRAINING_DS;
QUIT;
```

Screenshot(s)



Total rows: 614 Total columns: 13

Rows

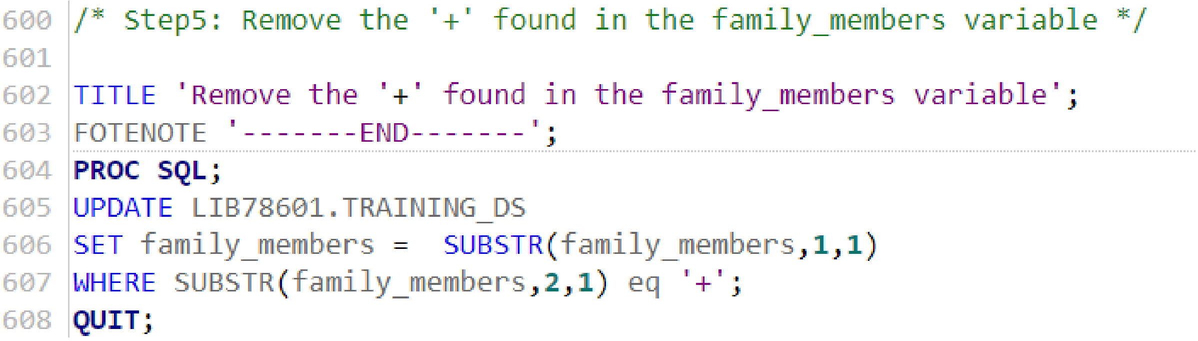
Total rows: 614 Total columns: 13			MARITAL_STA...	FAMILY_MEMB...	QUALIFICATION	EMPLOYM...
1	LP001002	Male	Not Married	0	Graduate	No
2	LP001003	Male	Married	1	Graduate	No
3	LP001005	Male	Married	0	Graduate	Yes
4	LP001006	Male	Married	0	Under Graduate	No
5	LP001008	Male	Not Married	0	Graduate	No
6	LP001011	Male	Married	2	Graduate	Yes
7	LP001013	Male	Married	0	Under Graduate	No
8	LP001014	Male	Married	3+	Graduate	No
9	LP001018	Male	Married	2	Graduate	No
10	LP001020	Male	Married	1	Graduate	No
11	LP001024	Male	Married	2	Graduate	No

Description

Create a table for duplicate the table before imputing the value for missing values cell. This action will guarantee that we will not corrupt the existing table.

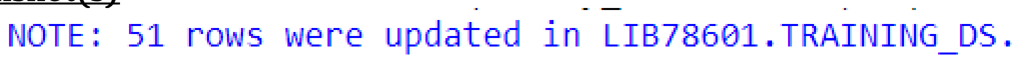
Step5: Remove the '+' found in the 'family_members' variable.

SAS Code



```
/* Step5: Remove the '+' found in the family_members variable */

TITLE 'Remove the '+' found in the family_members variable';
FOTENOTE '-------END-------';
PROC SQL;
UPDATE LIB78601.TRAINING_DS
SET family_members =  SUBSTR(family_members,1,1)
WHERE SUBSTR(family_members,2,1) eq '+';
QUIT;
```

Screenshot(s)



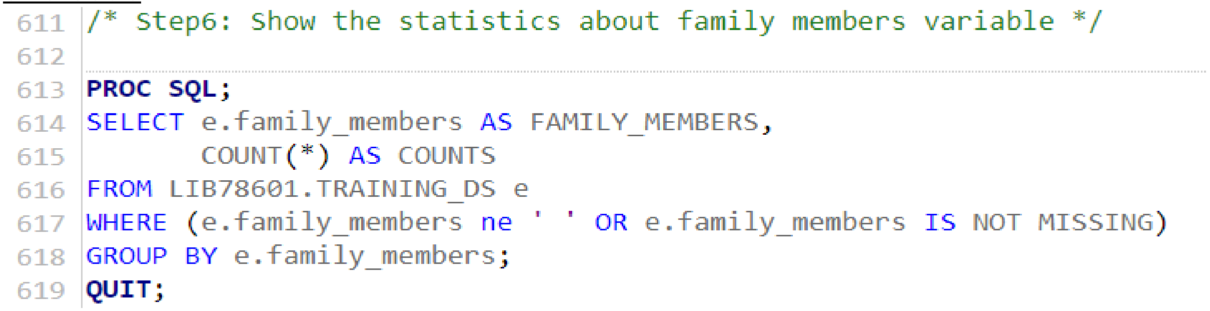
```
NOTE: 51 rows were updated in LIB78601.TRAINING_DS.
```

Description

In step, we remove the symbol ‘+’ from the dataset. By doing this step, the dataset will not lose any significant data since we used 3 to replace ‘3+’.

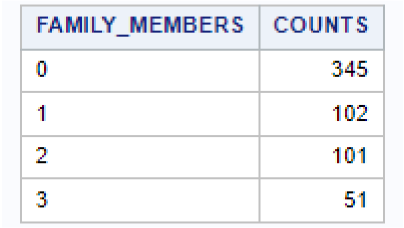
Step6: Show the statistics about family members variable.

SAS Code



```
/* Step6: Show the statistics about family members variable */

PROC SQL;
SELECT e.family_members AS FAMILY_MEMBERS,
       COUNT(*) AS COUNTS
FROM LIB78601.TRAINING_DS e
WHERE (e.family_members ne ' ' OR e.family_members IS NOT MISSING)
GROUP BY e.family_members;
QUIT;
```

Screenshot(s)



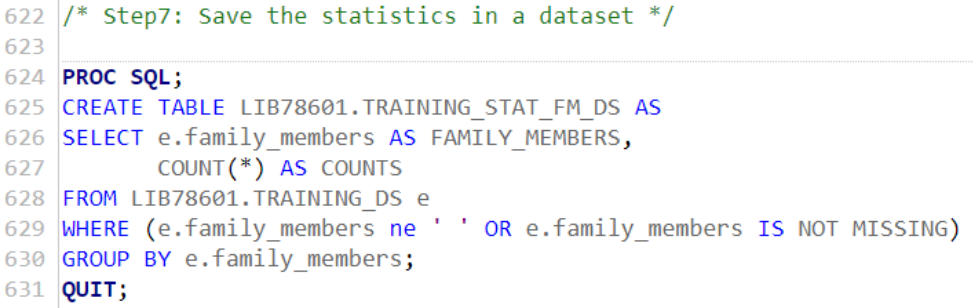
FAMILY_MEMBERS	COUNTS
0	345
1	102
2	101
3	51

Description

We can observe that the number of each category for variable family members, this information is useful for choosing treatment method for impute missing values.

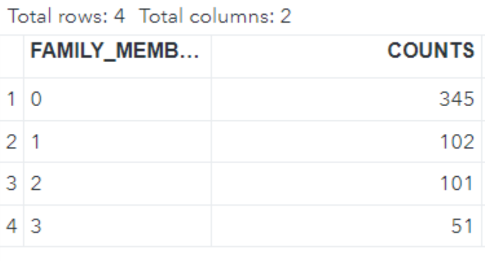
Step7: Save the statistics in a dataset.

SAS Code



```
/* Step7: Save the statistics in a dataset */

PROC SQL;
CREATE TABLE LIB78601.TRAINING_STAT_FM_DS AS
SELECT e.family_members AS FAMILY_MEMBERS,
       COUNT(*) AS COUNTS
FROM LIB78601.TRAINING_DS e
WHERE (e.family_members ne ' ' OR e.family_members IS NOT MISSING)
GROUP BY e.family_members;
QUIT;
```

Screenshot(s)



Total rows: 4 Total columns: 2

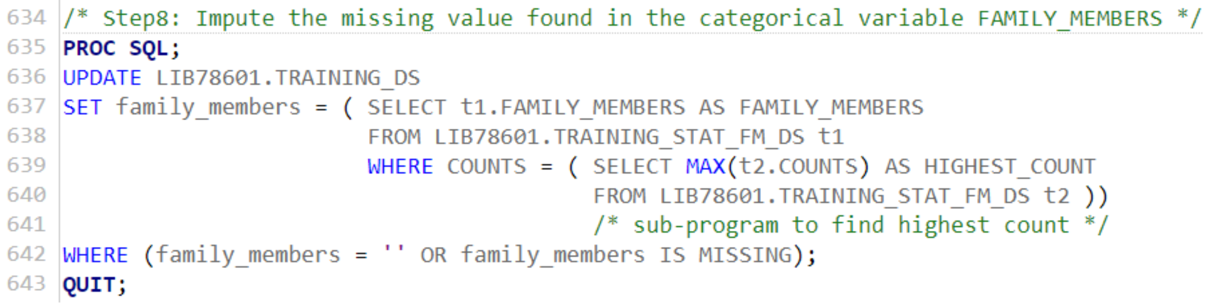
	FAMILY_MEMB...	COUNTS
1	0	345
2	1	102
3	2	101
4	3	51

Description

Save the output from last output as a table format, this will help for selecting specific value for impute missing value for specific variable in general.

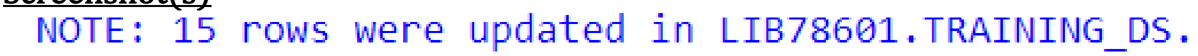
Step8: Impute the missing value found in the categorical variable FAMILY_MEMBERS.

SAS Code



```
/* Step8: Impute the missing value found in the categorical variable FAMILY_MEMBERS */
PROC SQL;
UPDATE LIB78601.TRAINING_DS
SET family_members = ( SELECT t1.FAMILY_MEMBERS AS FAMILY_MEMBERS
                       FROM LIB78601.TRAINING_STAT_FM_DS t1
                       WHERE COUNTS = ( SELECT MAX(t2.COUNTS) AS HIGHEST_COUNT
                                        FROM LIB78601.TRAINING_STAT_FM_DS t2 ))
                                        /* sub-program to find highest count */
WHERE (family_members = '' OR family_members IS MISSING);
QUIT;
```

Screenshot(s)



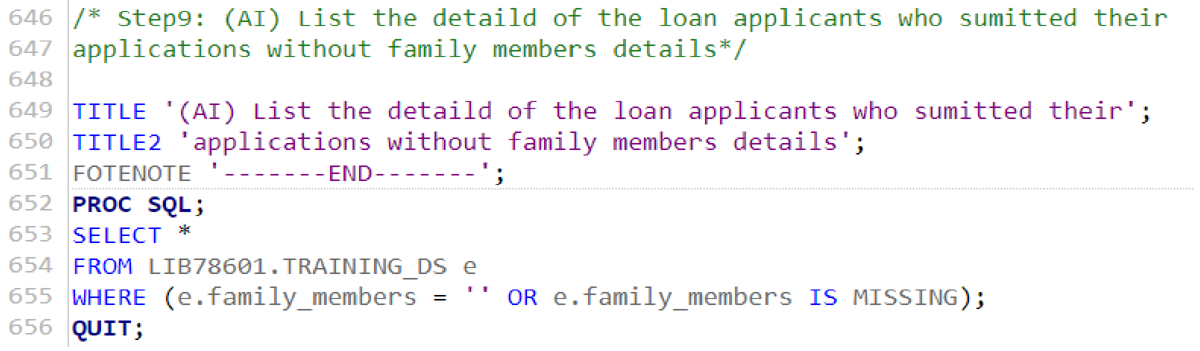
```
NOTE: 15 rows were updated in LIB78601.TRAINING_DS.
```

Description

Impute the missing values by the mode of the observation from its variable. From the screenshot, we can see that 15 rows were updated.

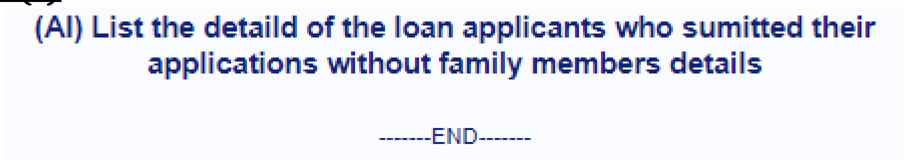
Step9: (AI) List the details of the loan applicants who submitted their applications without family members details.

SAS Code



```
/* Step9: (AI) List the detaild of the loan applicants who sumitted their
applications without family members details*/

TITLE '(AI) List the detaild of the loan applicants who sumitted their';
TITLE2 'applications without family members details';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.family_members = '' OR e.family_members IS MISSING);
QUIT;
```

Screenshot(s)



(AI) List the detaild of the loan applicants who sumitted their applications without family members details

-------END-------

Description

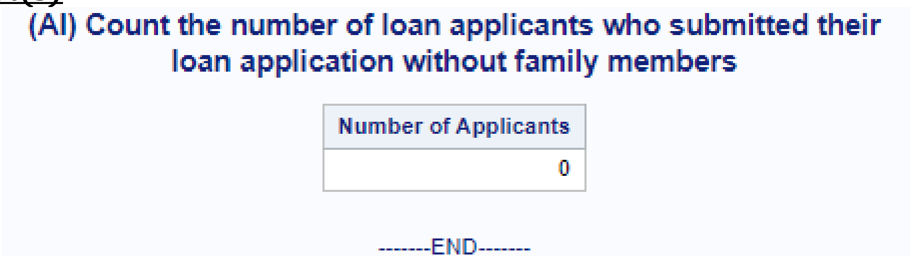
From the screenshot, we can easily to see that there is not any missing value after imputing. This step is to guarantee that the last step has impute all the missing values.

Step10: Step10: (AI) Count the number of loan applicants who submitted their loan application without family members.

SAS Code

```
/* Step10: (AI) Count the number of loan applicants who submitted their
loan application without family members */

TITLE '(AI) Count the number of loan applicants who submitted their';
TITLE2 'loan application without family members';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.family_members = '' OR e.family_members IS MISSING);
QUIT;
```

Screenshot(s)



(AI) Count the number of loan applicants who submitted their loan application without family members

Number of Applicants
0

-------END-------

Description

Clearly, we can see that there is total 0 missing value after imputing, this step is needed to double check that we have no making any mistake when imputing missing values.

6.5 Impute the CONTINUOUS missing values found in the dataset – LIB78601.TRAINING_DS.

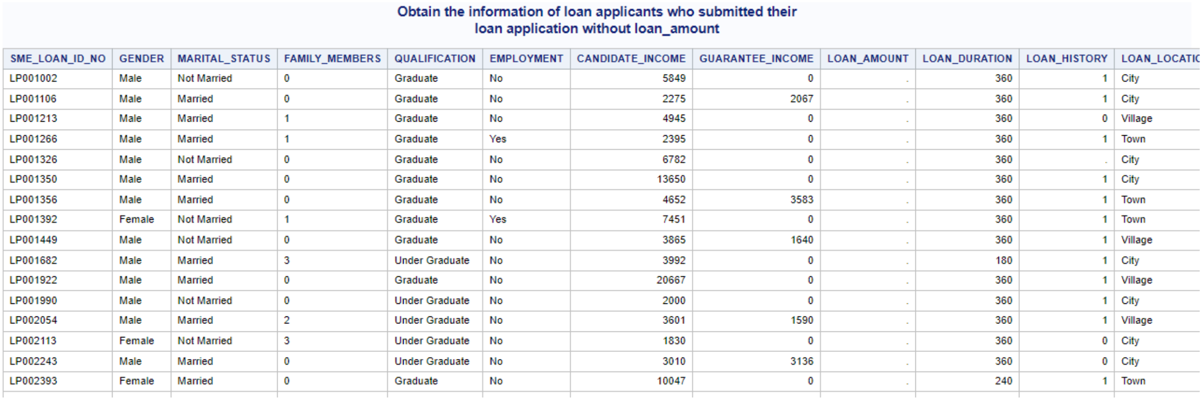
6.5.1 Impute the CONTINUOUS missing values found in the dataset – LIB78601.TRAINING_DS – [LOAN_AMOUNT]

Step1: Obtain the information.

SAS Code

```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without loan_amount */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_amount';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_amount EQ . OR e.loan_amount IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without loan_amount

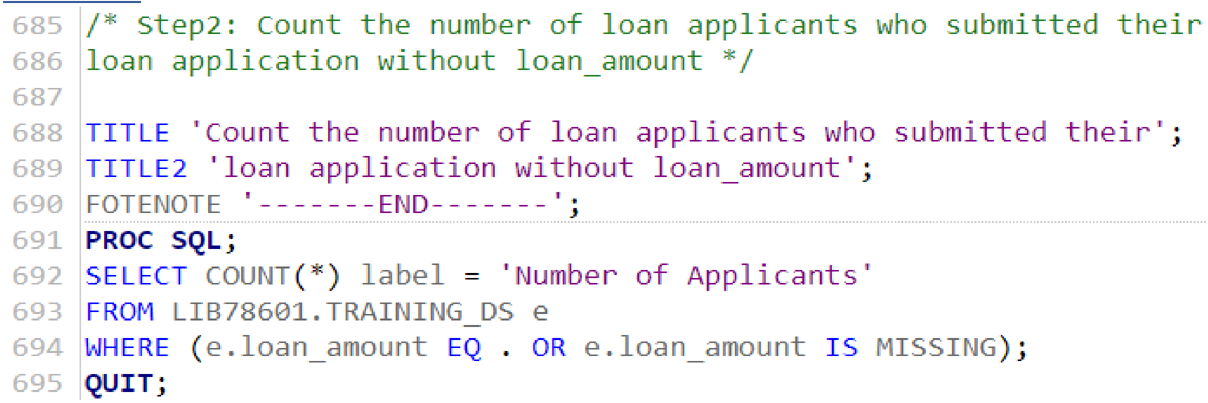
SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATIC
LP001002	Male	Not Married	0	Graduate	No	5849	0	.	360	1	City
LP001106	Male	Married	0	Graduate	No	2275	2067	.	360	1	City
LP001213	Male	Married	1	Graduate	No	4945	0	.	360	0	Village
LP001266	Male	Married	1	Graduate	Yes	2395	0	.	360	1	Town
LP001326	Male	Not Married	0	Graduate	No	6782	0	.	360	.	City
LP001350	Male	Married	0	Graduate	No	13650	0	.	360	1	City
LP001356	Male	Married	0	Graduate	No	4652	3583	.	360	1	Town
LP001392	Female	Not Married	1	Graduate	Yes	7451	0	.	360	1	Town
LP001449	Male	Not Married	0	Graduate	No	3865	1640	.	360	1	Village
LP001682	Male	Married	3	Under Graduate	No	3992	0	.	180	1	City
LP001922	Male	Married	0	Graduate	No	20667	0	.	360	1	Village
LP001990	Male	Not Married	0	Under Graduate	No	2000	0	.	360	1	City
LP002054	Male	Married	2	Under Graduate	No	3601	1590	.	360	1	Village
LP002113	Female	Not Married	3	Under Graduate	No	1830	0	.	360	0	City
LP002243	Male	Married	0	Under Graduate	No	3010	3136	.	360	0	City
LP002393	Female	Married	0	Graduate	No	10047	0	.	240	1	Town

Description

From the screenshot, we can easily to see the full details of applicants that have missing values for loan amount.

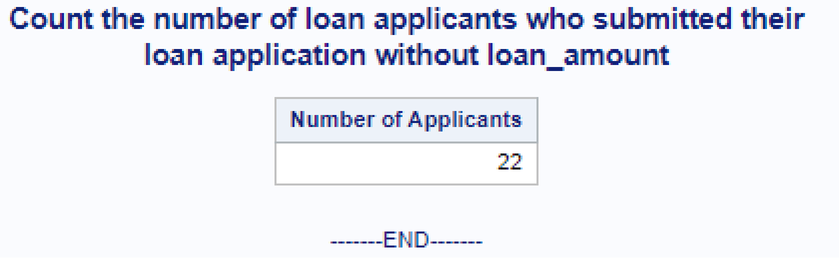
Step2: Count the number of missing values.

SAS Code



```
/* Step2: Count the number of loan applicants who submitted their
loan application without loan_amount */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_amount';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_amount EQ . OR e.loan_amount IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their
loan application without loan_amount

Number of Applicants
22

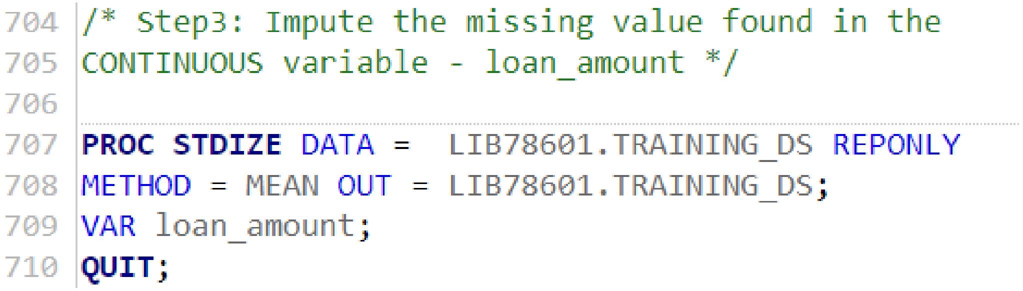
-------END-------

Description

Clearly, we can see that there is total 22 missing values for this variable, we will treat this issue in the coming steps.

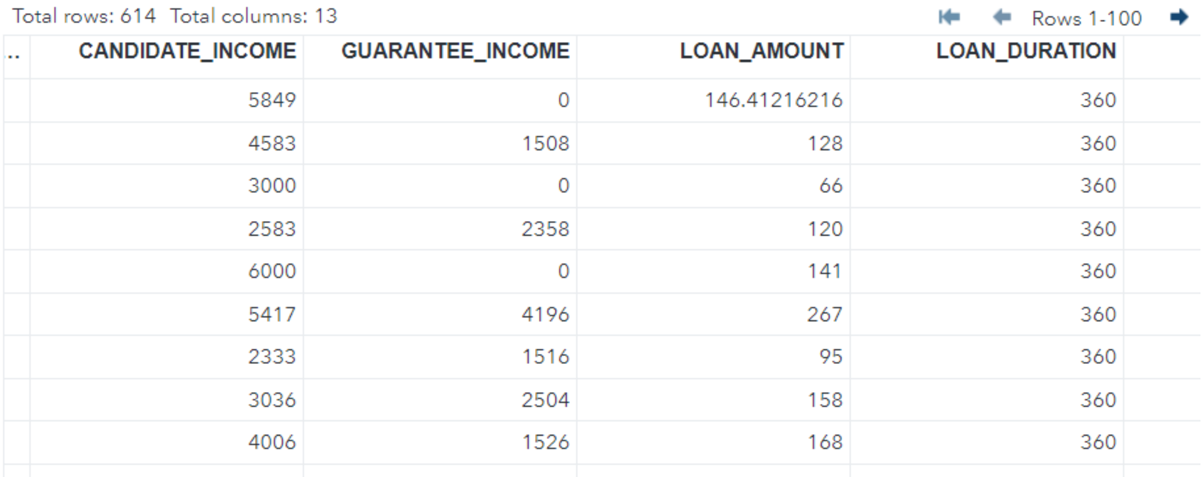
Step3: Impute the missing value found in the CONTINUOUS variable - loan_amount.

SAS Code



```
/* Step3: Impute the missing value found in the
CONTINUOUS variable - loan_amount */

PROC STDIZE DATA =  LIB78601.TRAINING_DS REPONLY
METHOD = MEAN OUT = LIB78601.TRAINING_DS;
VAR loan_amount;
QUIT;
```

Screenshot(s)



Total rows: 614 Total columns: 13

Rows 1-100

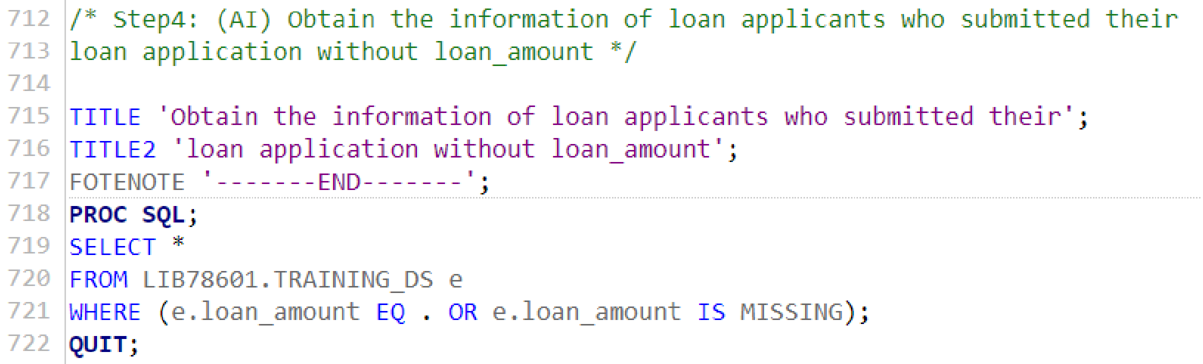
..	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION
	5849	0	146.41216216	360
	4583	1508	128	360
	3000	0	66	360
	2583	2358	120	360
	6000	0	141	360
	5417	4196	267	360
	2333	1516	95	360
	3036	2504	158	360
	4006	1526	168	360

Description

In this step, we impute the missing value of variable loan amount by its mean value, this can guarantee that the missing data didn’t affect the mean value of existing data.

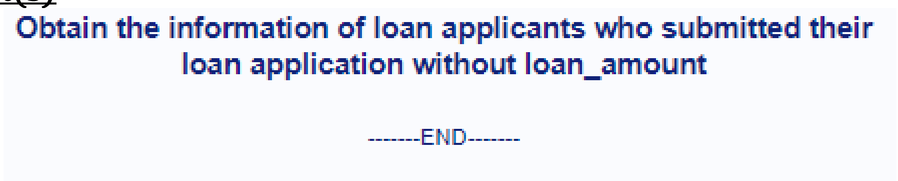
Step4: (AI) Obtain the information.

SAS Code



```
/* Step4: (AI) Obtain the information of loan applicants who submitted their
loan application without loan_amount */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_amount';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_amount EQ . OR e.loan_amount IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without loan_amount

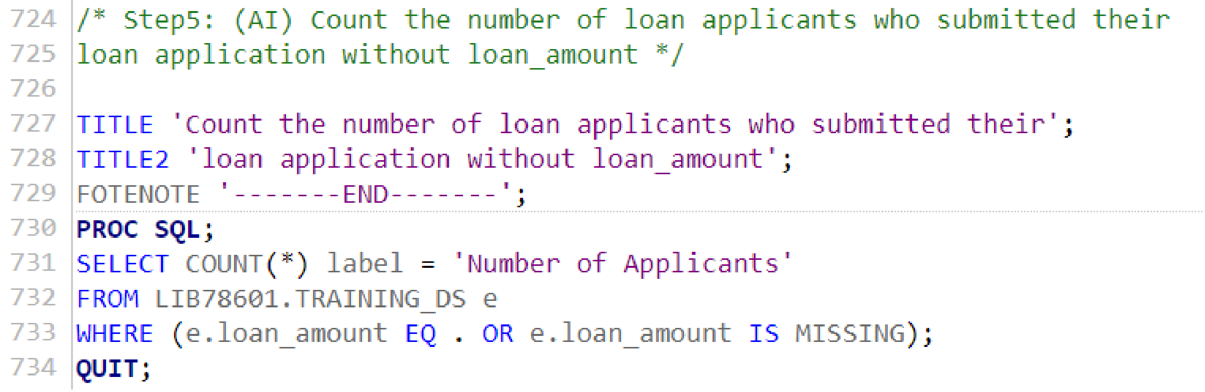
-------END-------

Description

From the screenshot, we can easily to see that there is not any missing value after imputing. This step is to guarantee that the last step has impute all the missing values.

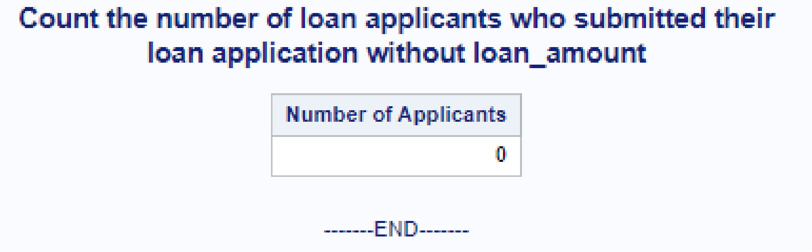
Step5: (AI) Count the number of missing values.

SAS Code



```
/* Step5: (AI) Count the number of loan applicants who submitted their
loan application without loan_amount */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_amount';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_amount EQ . OR e.loan_amount IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their loan application without loan_amount

Number of Applicants
0

-------END-------

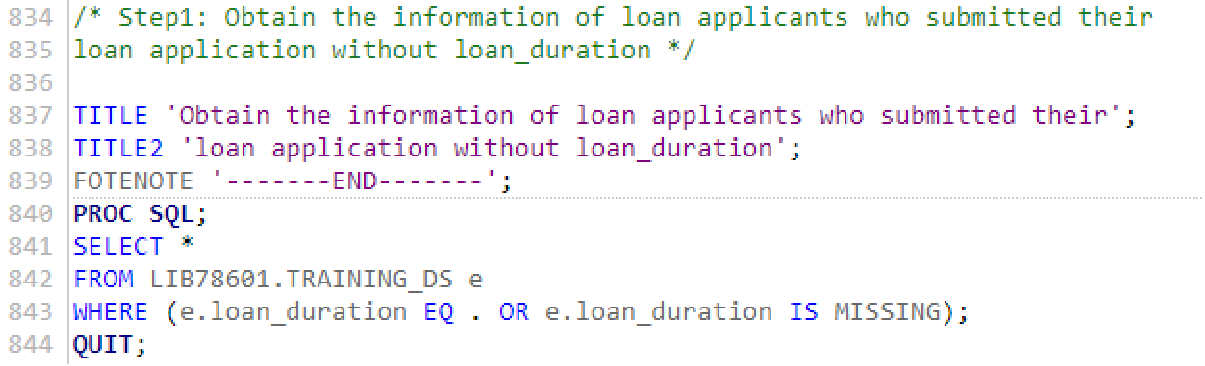
Description

Clearly, we can see that there is total 0 missing value after imputing, this step is needed to double check that we have no making any mistake when imputing missing values.

6.5.2 Impute the CONTINUOUS missing values found in the dataset – LIB78601.TRAINING_DS – [LOAN_DURATION]

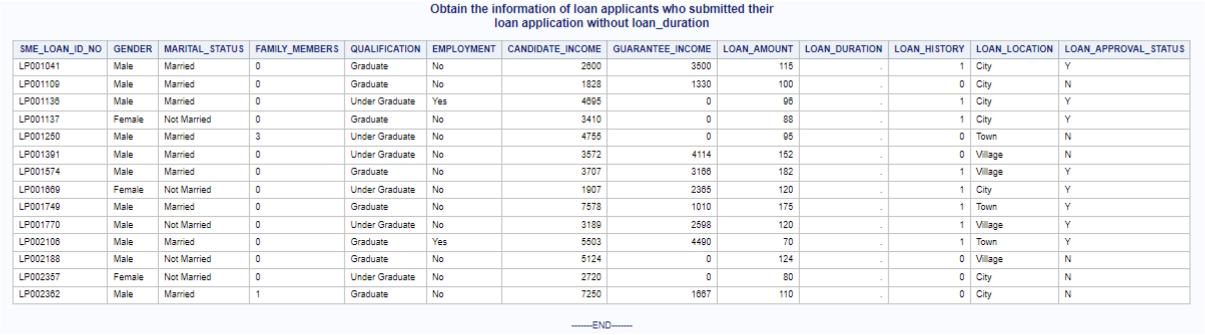
Step1: Obtain the information.

SAS Code



```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without loan_duration */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_duration';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_duration EQ . OR e.loan_duration IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without loan_duration

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001041	Male	Married	0	Graduate	No	2600	3500	115	.	1	City	Y
LP001109	Male	Married	0	Graduate	No	1828	1330	100	.	0	City	N
LP001136	Male	Married	0	Under Graduate	Yes	4695	0	96	.	1	City	Y
LP001137	Female	Not Married	0	Graduate	No	3410	0	88	.	1	City	Y
LP001250	Male	Married	3	Under Graduate	No	4755	0	95	.	0	Town	N
LP001391	Male	Married	0	Under Graduate	No	3572	4114	152	.	0	Village	N
LP001574	Male	Married	0	Graduate	No	3707	3166	182	.	1	Village	Y
LP001669	Female	Not Married	0	Under Graduate	No	1907	2365	120	.	1	City	Y
LP001749	Male	Married	0	Graduate	No	7578	1010	175	.	1	Town	Y
LP001770	Male	Not Married	0	Under Graduate	No	3189	2598	120	.	1	Village	Y
LP002106	Male	Married	0	Graduate	Yes	5503	4490	70	.	1	Town	Y
LP002188	Male	Not Married	0	Graduate	No	5124	0	124	.	0	Village	N
LP002357	Female	Not Married	0	Under Graduate	No	2720	0	80	.	0	City	N
LP002362	Male	Married	1	Graduate	No	7250	1667	110	.	0	City	N

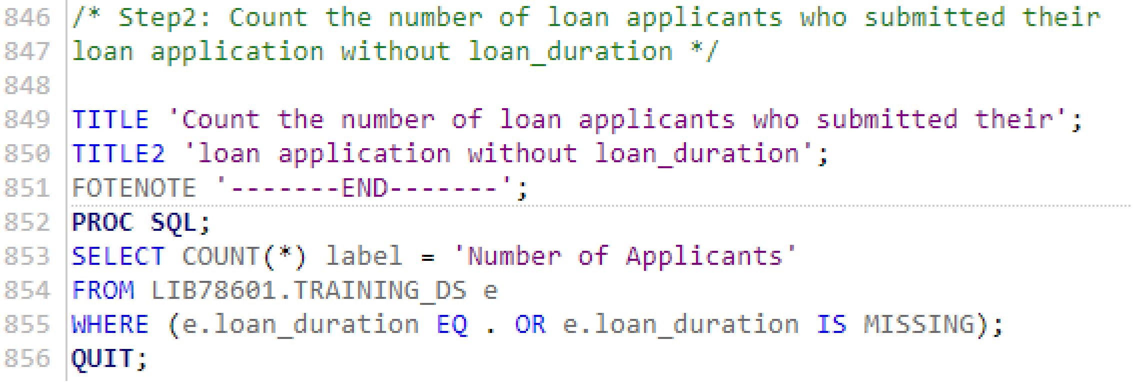
--------END--------

Description

From the screenshot, we can easily to see the full details of applicants that have missing values for loan duration.

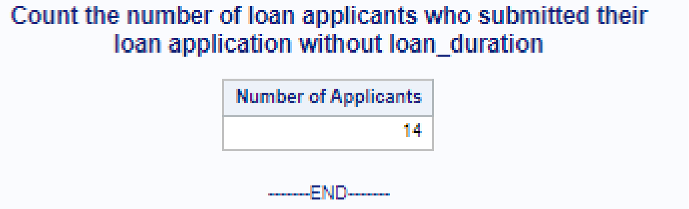
Step2: Count the number of missing values.

SAS Code



```
/* Step2: Count the number of loan applicants who submitted their
loan application without loan_duration */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_duration';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_duration EQ . OR e.loan_duration IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their
loan application without loan_duration

Number of Applicants
14

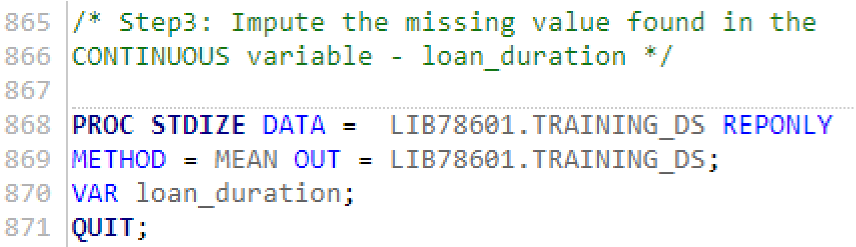
--------END--------

Description

Clearly, we can see that there is total 14 missing values for this variable, we will treat this issue in the coming steps.

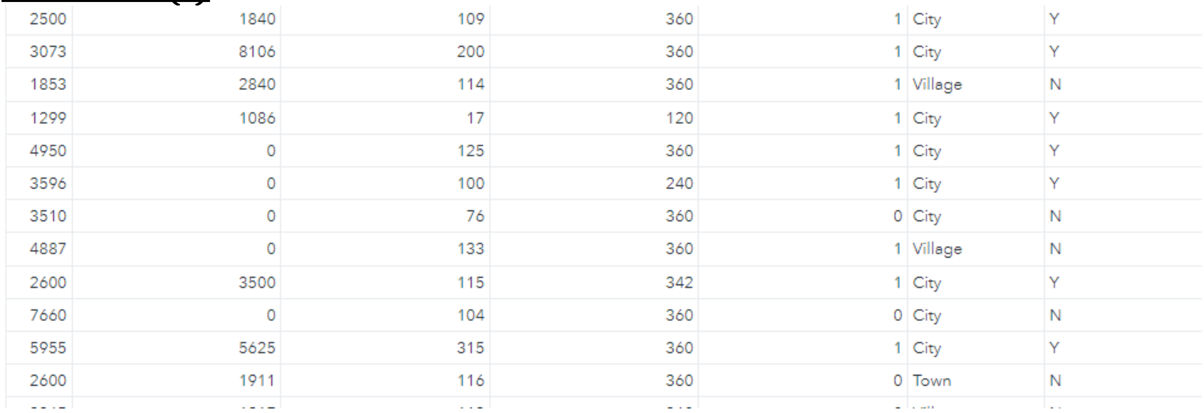
Step3: Impute the missing value found in the CONTINUOUS variable - loan_duration.

SAS Code



```
/* Step3: Impute the missing value found in the
CONTINUOUS variable - loan_duration */

PROC STDIZE DATA =  LIB78601.TRAINING_DS REPONLY
METHOD = MEAN OUT = LIB78601.TRAINING_DS;
VAR loan_duration;
QUIT;
```

Screenshot(s)



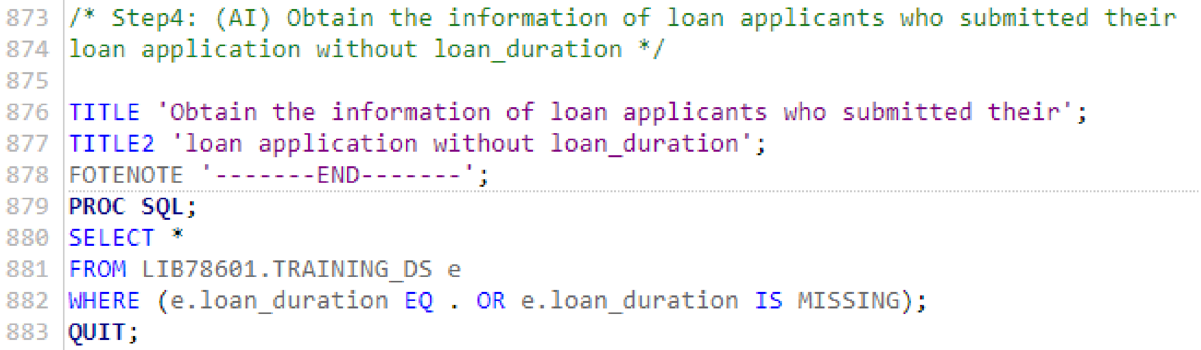
2500	1840	109	360	1	City	Y
3073	8106	200	360	1	City	Y
1853	2840	114	360	1	Village	N
1299	1086	17	120	1	City	Y
4950	0	125	360	1	City	Y
3596	0	100	240	1	City	Y
3510	0	76	360	0	City	N
4887	0	133	360	1	Village	N
2600	3500	115	342	1	City	Y
7660	0	104	360	0	City	N
5955	5625	315	360	1	City	Y
2600	1911	116	360	0	Town	N

Description

In this step, we impute the missing value of variable loan duration by its mean value, this can guarantee that the missing data didn't affect the mean value of existing data.

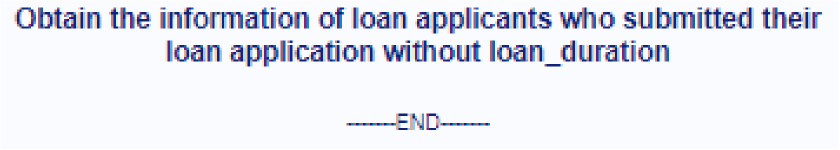
Step4: (AI) Obtain the information.

SAS Code



```
/* Step4: (AI) Obtain the information of loan applicants who submitted their
loan application without loan_duration */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_duration';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_duration EQ . OR e.loan_duration IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without loan_duration

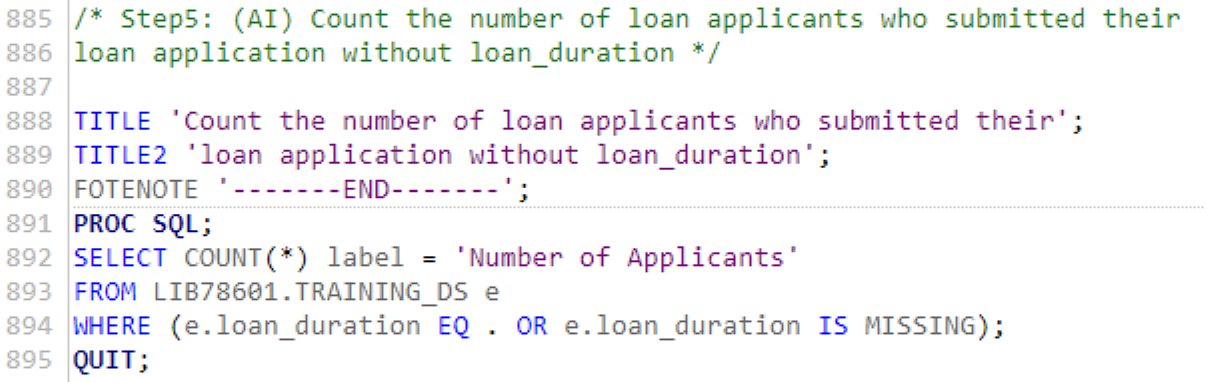
-------END-------

Description

From the screenshot, we can easily to see that there is not any missing value after imputing. This step is to guarantee that the last step has impute all the missing values.

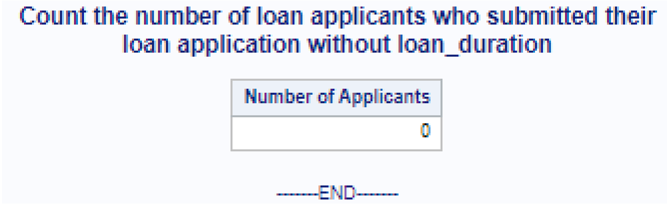
Step5: (AI) Count the number of missing values.

SAS Code



```
/* Step5: (AI) Count the number of loan applicants who submitted their
loan application without loan_duration */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_duration';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_duration EQ . OR e.loan_duration IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their
loan application without loan_duration

Number of Applicants
0

-------END-------

Description

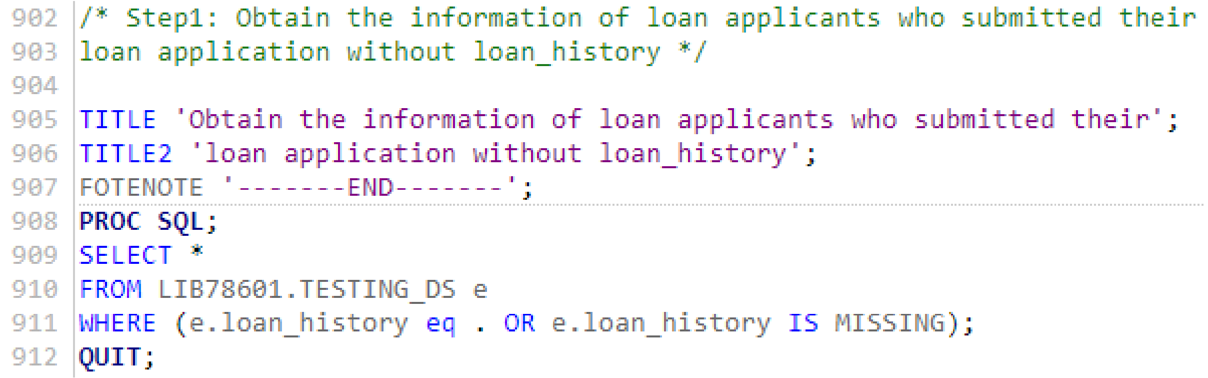
Clearly, we can see that there is total 0 missing value after imputing, this step is needed to double check that we have no making any mistake when imputing missing values.

6.6 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TESTING_DS.

6.6.1 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TRAINING_DS – [LOAN_HISTORY]

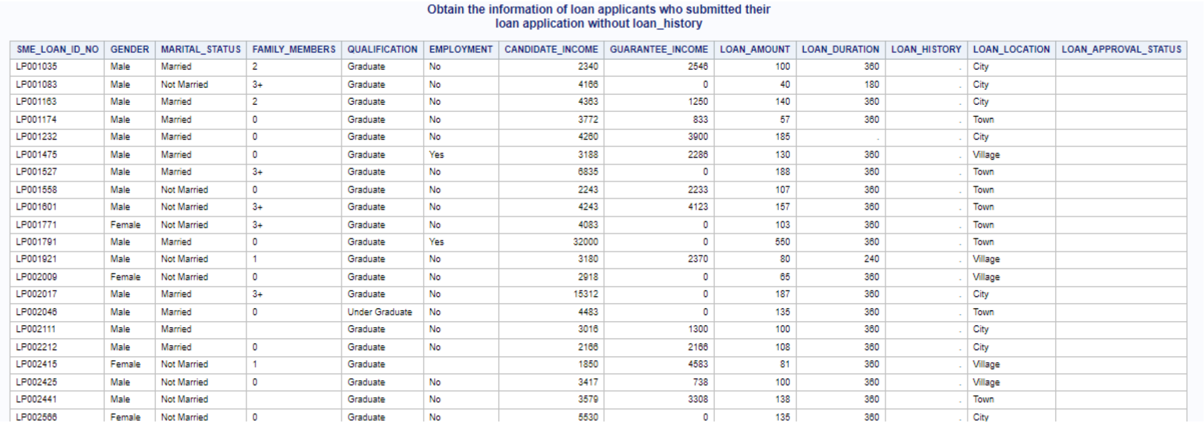
Step1: Obtain the information.

SAS Code



```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without loan_history */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_history';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.loan_history eq . OR e.loan_history IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without loan_history

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001035	Male	Married	2	Graduate	No	2340	2546	100	360	.	City	
LP001083	Male	Not Married	3+	Graduate	No	4166	0	40	180	.	City	
LP001163	Male	Married	2	Graduate	No	4363	1250	140	360	.	City	
LP001174	Male	Married	0	Graduate	No	3772	833	57	360	.	Town	
LP001232	Male	Married	0	Graduate	No	4260	3900	185	.	.	City	
LP001475	Male	Married	0	Graduate	Yes	3188	2286	130	360	.	Village	
LP001527	Male	Married	3+	Graduate	No	6835	0	188	360	.	Town	
LP001558	Male	Not Married	0	Graduate	No	2243	2233	107	360	.	Town	
LP001601	Male	Not Married	3+	Graduate	No	4243	4123	157	360	.	Town	
LP001771	Female	Not Married	3+	Graduate	No	4083	0	103	360	.	Town	
LP001791	Male	Married	0	Graduate	Yes	32000	0	550	360	.	Town	
LP001921	Male	Not Married	1	Graduate	No	3180	2370	80	240	.	Village	
LP002009	Female	Not Married	0	Graduate	No	2918	0	65	360	.	Village	
LP002017	Male	Married	3+	Graduate	No	15312	0	187	360	.	City	
LP002046	Male	Married	0	Under Graduate	No	4483	0	135	360	.	Town	
LP002111	Male	Married		Graduate	No	3016	1300	100	360	.	City	
LP002212	Male	Married	0	Graduate	No	2166	2166	108	360	.	City	
LP002415	Female	Not Married	1	Graduate		1850	4583	81	360	.	Village	
LP002425	Male	Not Married	0	Graduate	No	3417	738	100	360	.	Village	
LP002441	Male	Not Married		Graduate	No	3579	3308	138	360	.	Town	
LP002566	Female	Not Married	0	Graduate	No	5530	0	135	360	.	City	

Description

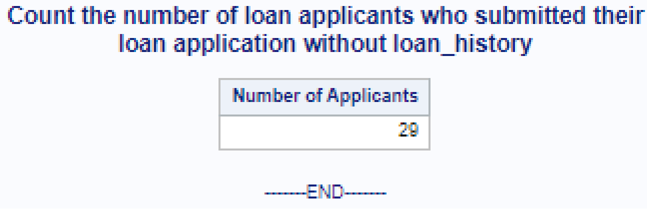
From the screenshot, we can easily to see the full details of applicants that have missing values for loan history.

Step2: Count the number of missing values.

SAS Code

```
/* Step2: Count the number of loan applicants who submitted their
loan application without loan_history */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_history';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TESTING_DS e
WHERE (e.loan_history eq . OR e.loan_history IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their
loan application without loan_history

Number of Applicants
29

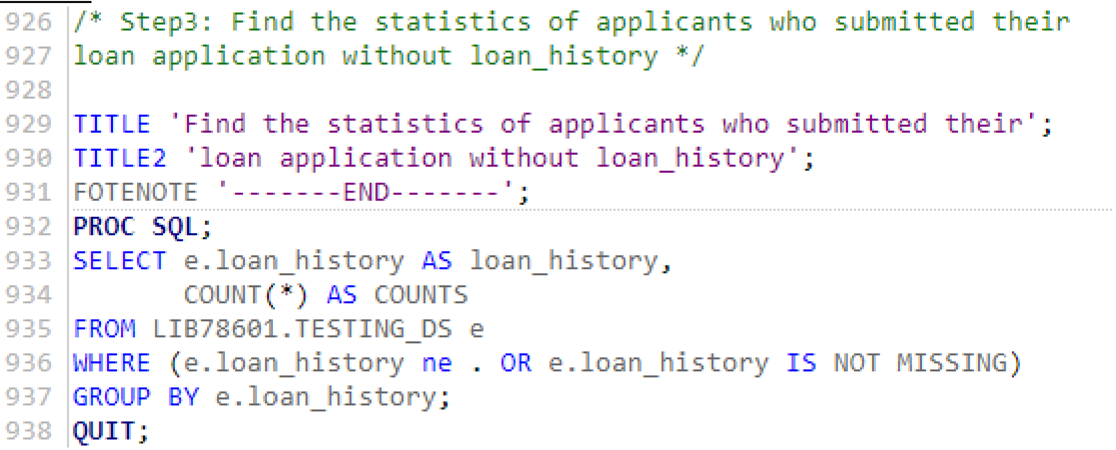
-------END-------

Description

Clearly, we can see that there is total 29 missing values for this variable, we will treat this issue in the coming steps.

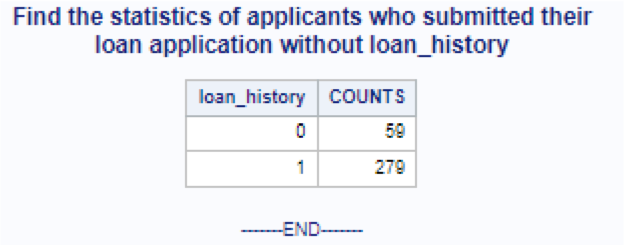
Step3: Statistics about missing values.

SAS Code



```
/* Step3: Find the statistics of applicants who submitted their
loan application without loan_history */

TITLE 'Find the statistics of applicants who submitted their';
TITLE2 'loan application without loan_history';
FOTENOTE '-------END-------';
PROC SQL;
SELECT e.loan_history AS loan_history,
       COUNT(*) AS COUNTS
FROM LIB78601.TESTING_DS e
WHERE (e.loan_history ne . OR e.loan_history IS NOT MISSING)
GROUP BY e.loan_history;
QUIT;
```

Screenshot(s)



Find the statistics of applicants who submitted their
loan application without loan_history

loan_history	COUNTS
0	59
1	279

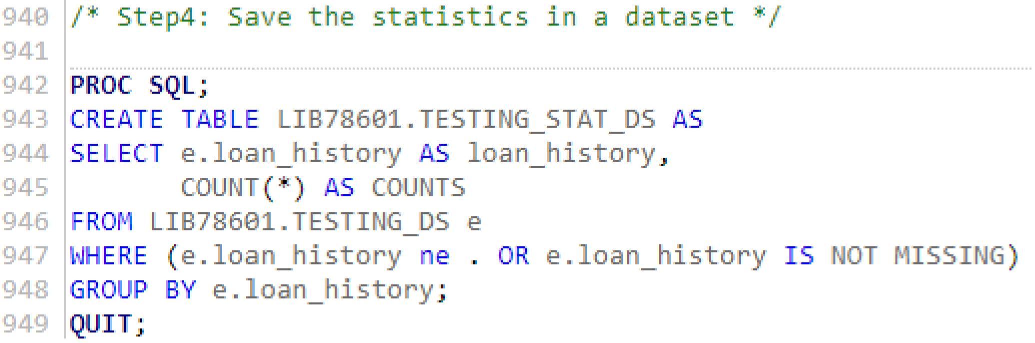
-------END-------

Description

We can observe that the number of each category for variable loan history, this information is useful for choosing treatment method for impute missing values.

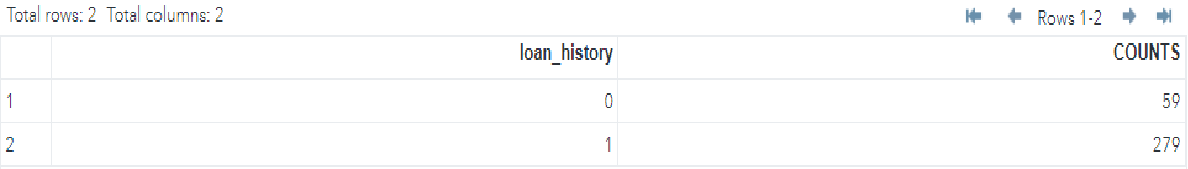
Step4: Save the statistics in a dataset.

SAS Code



```
/* Step4: Save the statistics in a dataset */

PROC SQL;
CREATE TABLE LIB78601.TESTING_STAT_DS AS
SELECT e.loan_history AS loan_history,
       COUNT(*) AS COUNTS
FROM LIB78601.TESTING_DS e
WHERE (e.loan_history ne . OR e.loan_history IS NOT MISSING)
GROUP BY e.loan_history;
QUIT;
```

Screenshot(s)



Total rows: 2 Total columns: 2

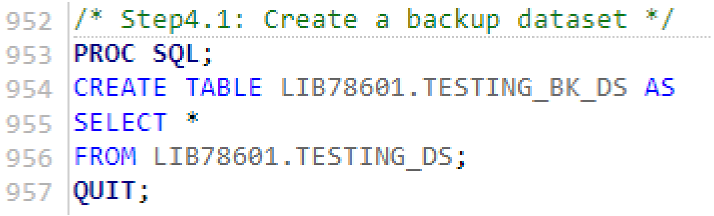
	loan_history	COUNTS
1	0	59
2	1	279

Description

Save the output from last output as a table format, this will help for selecting specific value for impute missing value for specific variable in general.

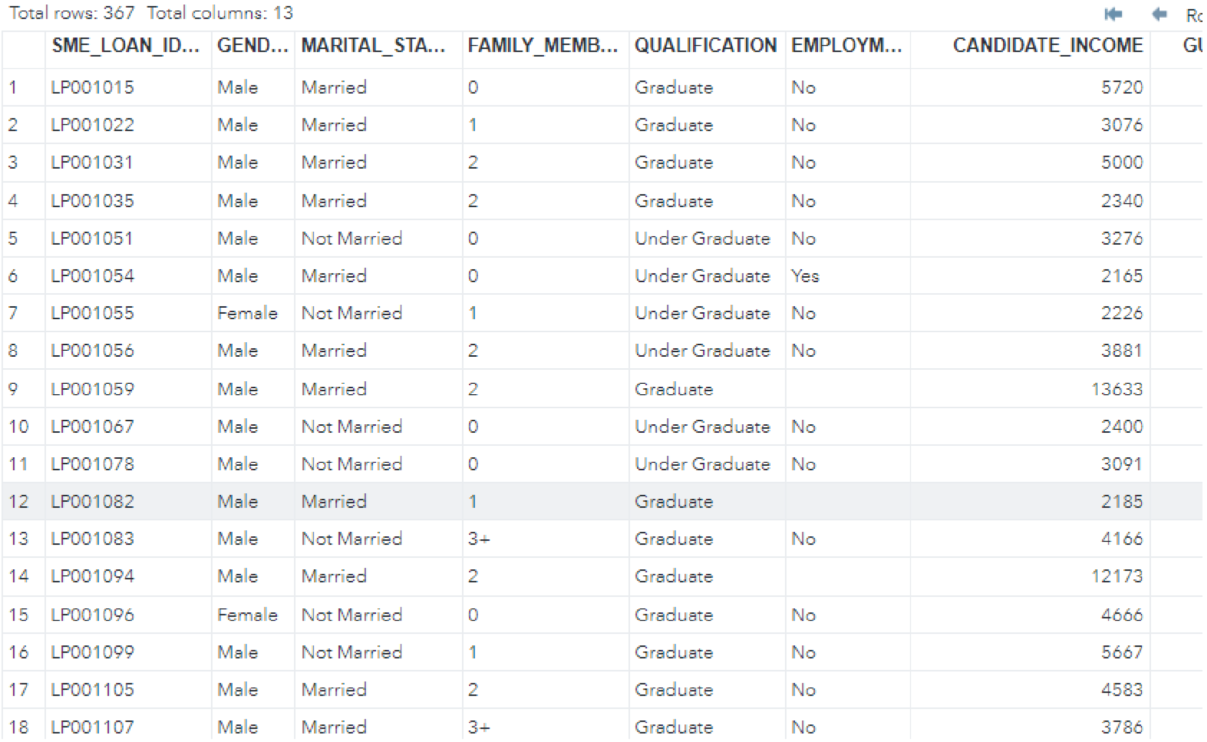
Step4.1: Backup the existing dataset.

SAS Code



```
/* Step4.1: Create a backup dataset */
PROC SQL;
CREATE TABLE LIB78601.TESTING_BK_DS AS
SELECT *
FROM LIB78601.TESTING_DS;
QUIT;
```

Screenshot(s)



Total rows: 367 Total columns: 13

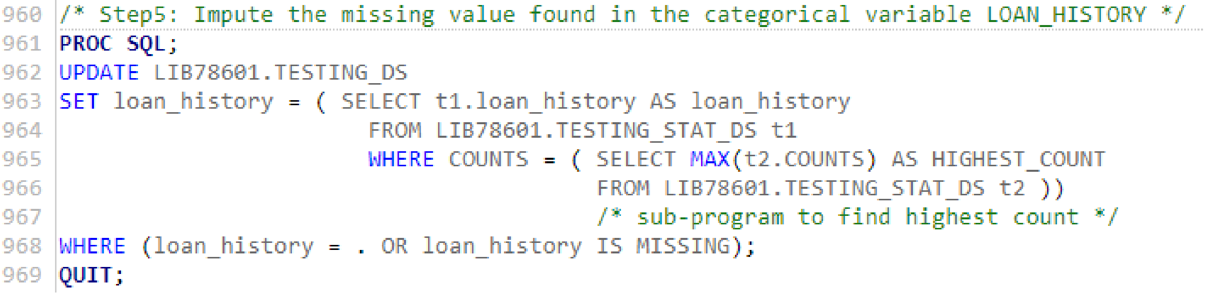
	SME_LOAN_ID...	GEND...	MARITAL_STA...	FAMILY_MEMB...	QUALIFICATION	EMPLOYM...	CANDIDATE_INCOME	GI
1	LP001015	Male	Married	0	Graduate	No	5720	
2	LP001022	Male	Married	1	Graduate	No	3076	
3	LP001031	Male	Married	2	Graduate	No	5000	
4	LP001035	Male	Married	2	Graduate	No	2340	
5	LP001051	Male	Not Married	0	Under Graduate	No	3276	
6	LP001054	Male	Married	0	Under Graduate	Yes	2165	
7	LP001055	Female	Not Married	1	Under Graduate	No	2226	
8	LP001056	Male	Married	2	Under Graduate	No	3881	
9	LP001059	Male	Married	2	Graduate		13633	
10	LP001067	Male	Not Married	0	Under Graduate	No	2400	
11	LP001078	Male	Not Married	0	Under Graduate	No	3091	
12	LP001082	Male	Married	1	Graduate		2185	
13	LP001083	Male	Not Married	3+	Graduate	No	4166	
14	LP001094	Male	Married	2	Graduate		12173	
15	LP001096	Female	Not Married	0	Graduate	No	4666	
16	LP001099	Male	Not Married	1	Graduate	No	5667	
17	LP001105	Male	Married	2	Graduate	No	4583	
18	LP001107	Male	Married	3+	Graduate	No	3786	

Description

Create a table for duplicate the table before imputing the value for missing values cell. This action will guarantee that we will not corrupt the existing table.

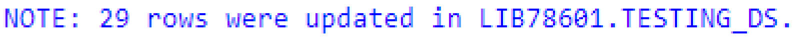
Step5: Impute the missing values.

SAS Code



```
/* Step5: Impute the missing value found in the categorical variable LOAN_HISTORY */
PROC SQL;
UPDATE LIB78601.TESTING_DS
SET loan_history = ( SELECT t1.loan_history AS loan_history
                     FROM LIB78601.TESTING_STAT_DS t1
                     WHERE COUNTS = ( SELECT MAX(t2.COUNTS) AS HIGHEST_COUNT
                                      FROM LIB78601.TESTING_STAT_DS t2 ))
                                      /* sub-program to find highest count */
WHERE (loan_history = . OR loan_history IS MISSING);
QUIT;
```

Screenshot(s)



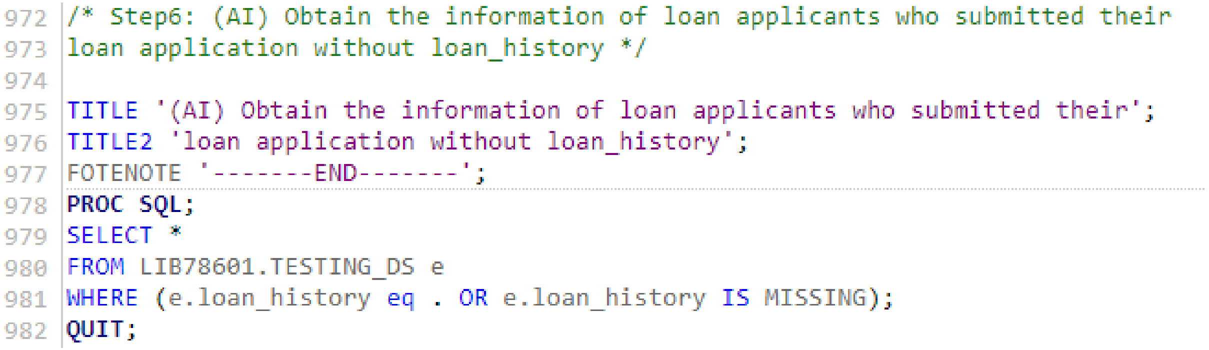
```
NOTE: 29 rows were updated in LIB78601.TESTING_DS.
```

Description

Impute the missing values by the mode of the observation from its variable. From the screenshot, we can see that 29 rows were updated.

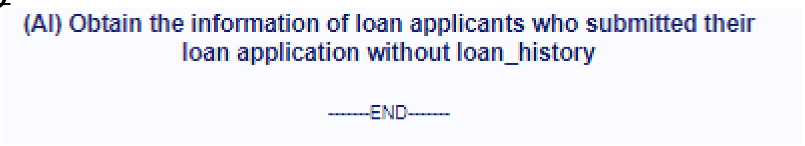
Step6: Ensure that the missing values was imputed.

SAS Code



```
/* Step6: (AI) Obtain the information of loan applicants who submitted their
loan application without loan_history */

TITLE '(AI) Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_history';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.loan_history eq . OR e.loan_history IS MISSING);
QUIT;
```

Screenshot(s)



(AI) Obtain the information of loan applicants who submitted their
loan application without loan_history

-------END-------

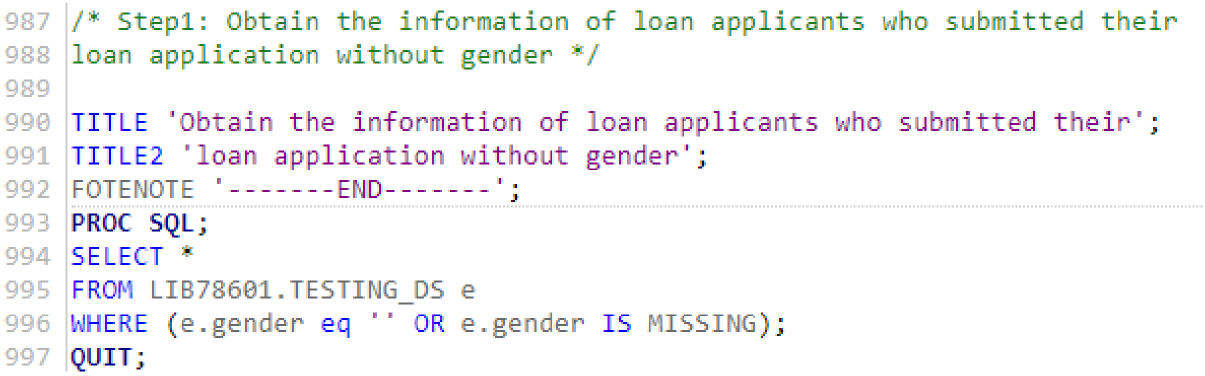
Description

We checked that there is not any missing value for this variable after imputing the existing missing values.

6.6.2 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TRAINING_DS – [GENDER]

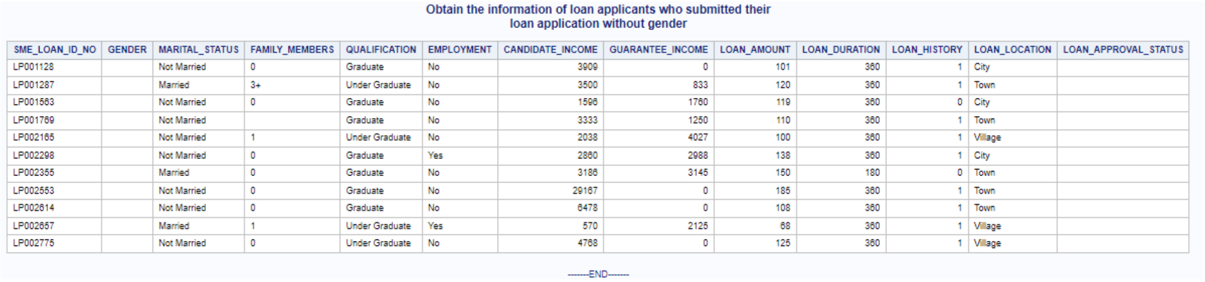
Step1: Obtain the information.

SAS Code



```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without gender */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without gender';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.gender eq '' OR e.gender IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without gender

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001128		Not Married	0	Graduate	No	3909	0	101	360	1	City	
LP001287		Married	3+	Under Graduate	No	3500	833	120	360	1	Town	
LP001563		Not Married	0	Graduate	No	1596	1760	119	360	0	City	
LP001769		Not Married		Graduate	No	3333	1250	110	360	1	Town	
LP002165		Not Married	1	Under Graduate	No	2038	4027	100	360	1	Village	
LP002298		Not Married	0	Graduate	Yes	2860	2988	138	360	1	City	
LP002355		Married	0	Graduate	No	3186	3145	150	180	0	Town	
LP002553		Not Married	0	Graduate	No	29167	0	185	360	1	Town	
LP002614		Not Married	0	Graduate	No	6478	0	108	360	1	Town	
LP002657		Married	1	Under Graduate	Yes	570	2125	68	360	1	Village	
LP002775		Not Married	0	Under Graduate	No	4768	0	125	360	1	Village	

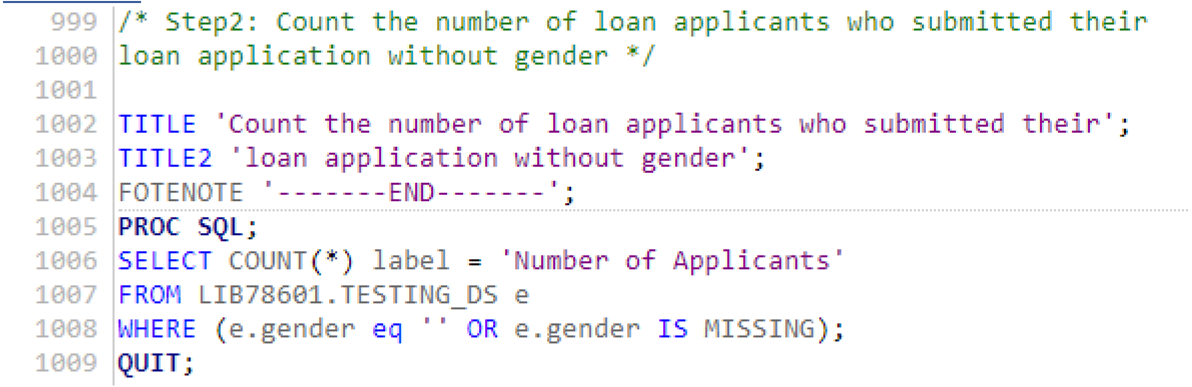
-------END-------

Description

From the screenshot, we can easily to see the full details of applicants that have missing values for gender.

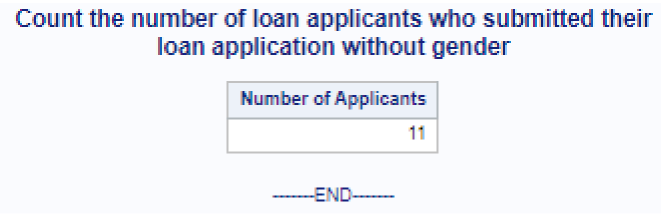
Step2: Count the number of missing values.

SAS Code



```
/* Step2: Count the number of loan applicants who submitted their
loan application without gender */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without gender';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TESTING_DS e
WHERE (e.gender eq '' OR e.gender IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their
loan application without gender

Number of Applicants
11

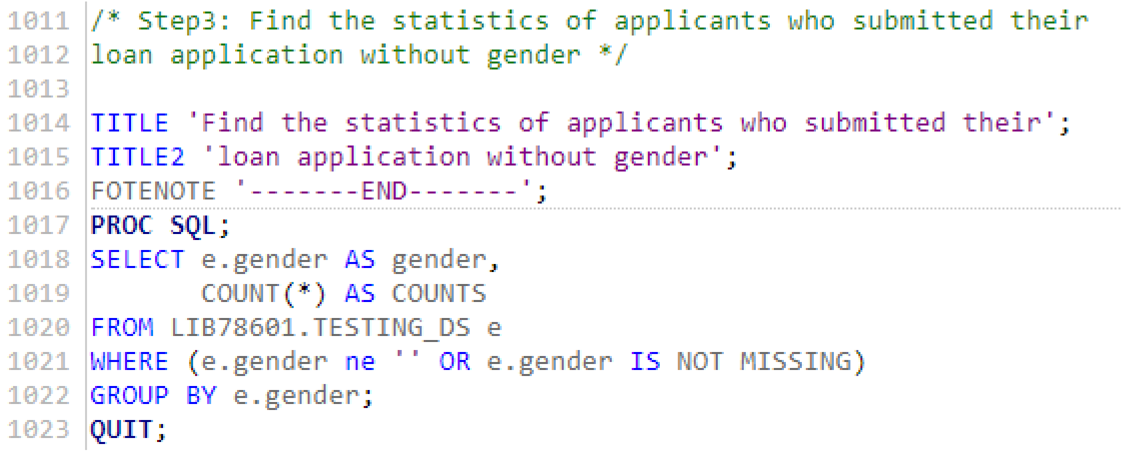
-------END-------

Description

Clearly, we can see that there is total 11 missing values for this variable, we will treat this issue in the coming steps.

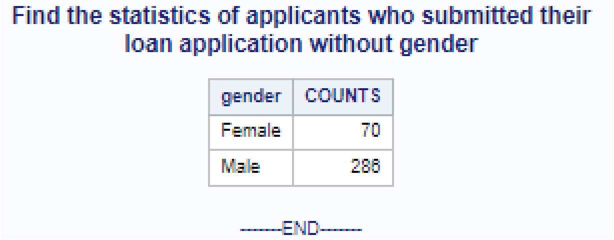
Step3: Statistics about missing values.

SAS Code



```
/* Step3: Find the statistics of applicants who submitted their
loan application without gender */

TITLE 'Find the statistics of applicants who submitted their';
TITLE2 'loan application without gender';
FOTENOTE '-------END-------';
PROC SQL;
SELECT e.gender AS gender,
       COUNT(*) AS COUNTS
FROM LIB78601.TESTING_DS e
WHERE (e.gender ne '' OR e.gender IS NOT MISSING)
GROUP BY e.gender;
QUIT;
```

Screenshot(s)



Find the statistics of applicants who submitted their
loan application without gender

gender	COUNTS
Female	70
Male	286

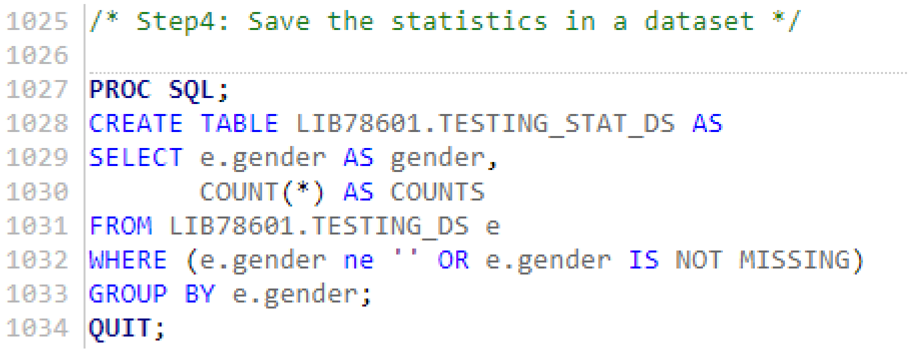
-------END-------

Description

We can observe that the number of each category for variable gender, this information is useful for choosing treatment method for impute missing values.

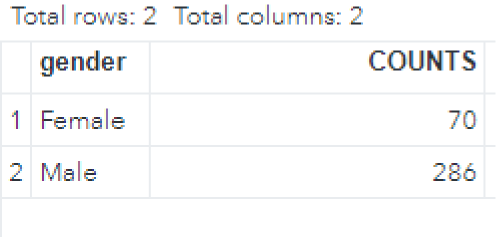
Step4: Save the statistics in a dataset.

SAS Code



```
/* Step4: Save the statistics in a dataset */

PROC SQL;
CREATE TABLE LIB78601.TESTING_STAT_DS AS
SELECT e.gender AS gender,
       COUNT(*) AS COUNTS
FROM LIB78601.TESTING_DS e
WHERE (e.gender ne '' OR e.gender IS NOT MISSING)
GROUP BY e.gender;
QUIT;
```

Screenshot(s)



Total rows: 2 Total columns: 2

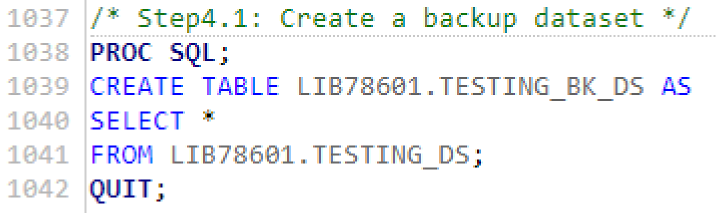
	gender	COUNTS
1	Female	70
2	Male	286

Description

Save the output from last output as a table format, this will help for selecting specific value for impute missing value for specific variable in general.

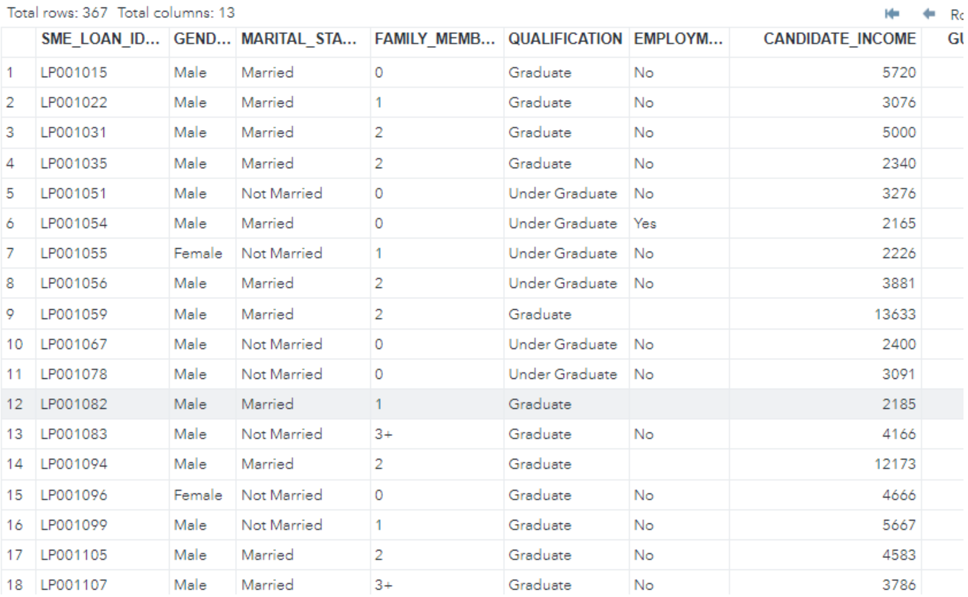
Step4.1: Backup the existing dataset.

SAS Code



```
/* Step4.1: Create a backup dataset */
PROC SQL;
CREATE TABLE LIB78601.TESTING_BK_DS AS
SELECT *
FROM LIB78601.TESTING_DS;
QUIT;
```

Screenshot(s)



Total rows: 367 Total columns: 13

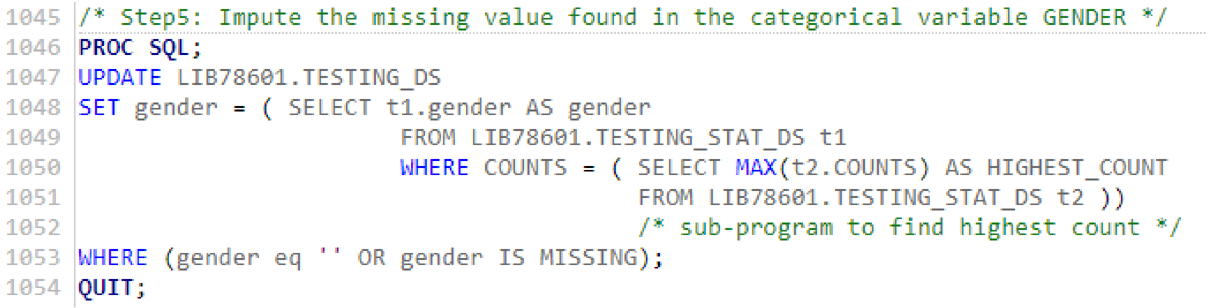
	SME_LOAN_ID...	GEND...	MARITAL_STA...	FAMILY_MEMB...	QUALIFICATION	EMPLOYM...	CANDIDATE_INCOME
1	LP001015	Male	Married	0	Graduate	No	5720
2	LP001022	Male	Married	1	Graduate	No	3076
3	LP001031	Male	Married	2	Graduate	No	5000
4	LP001035	Male	Married	2	Graduate	No	2340
5	LP001051	Male	Not Married	0	Under Graduate	No	3276
6	LP001054	Male	Married	0	Under Graduate	Yes	2165
7	LP001055	Female	Not Married	1	Under Graduate	No	2226
8	LP001056	Male	Married	2	Under Graduate	No	3881
9	LP001059	Male	Married	2	Graduate		13633
10	LP001067	Male	Not Married	0	Under Graduate	No	2400
11	LP001078	Male	Not Married	0	Under Graduate	No	3091
12	LP001082	Male	Married	1	Graduate		2185
13	LP001083	Male	Not Married	3+	Graduate	No	4166
14	LP001094	Male	Married	2	Graduate		12173
15	LP001096	Female	Not Married	0	Graduate	No	4666
16	LP001099	Male	Not Married	1	Graduate	No	5667
17	LP001105	Male	Married	2	Graduate	No	4583
18	LP001107	Male	Married	3+	Graduate	No	3786

Description

Create a table for duplicate the table before imputing the value for missing values cell. This action will guarantee that we will not corrupt the existing table.

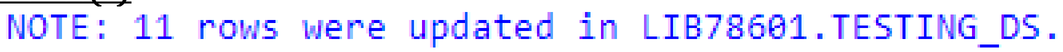
Step5: Impute the missing values.

SAS Code



```
/* Step5: Impute the missing value found in the categorical variable GENDER */
PROC SQL;
UPDATE LIB78601.TESTING_DS
SET gender = ( SELECT t1.gender AS gender
                       FROM LIB78601.TESTING_STAT_DS t1
                       WHERE COUNTS = ( SELECT MAX(t2.COUNTS) AS HIGHEST_COUNT
                                        FROM LIB78601.TESTING_STAT_DS t2 ))
                                        /* sub-program to find highest count */
WHERE (gender eq '' OR gender IS MISSING);
QUIT;
```

Screenshot(s)



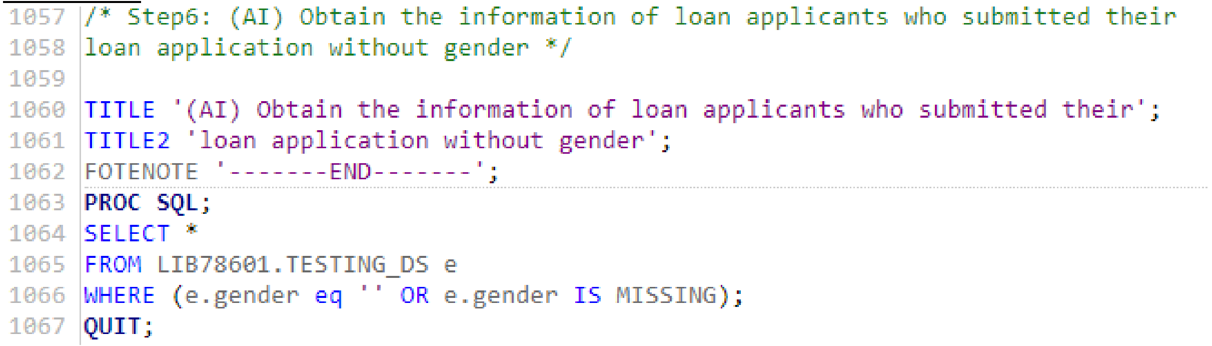
```
NOTE: 11 rows were updated in LIB78601.TESTING_DS.
```

Description

Impute the missing values by the mode of the observation from its variable. From the screenshot, we can see that 11 rows were updated.

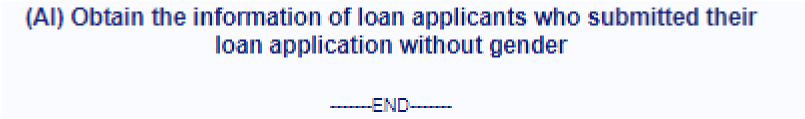
Step6: Ensure that the missing values was imputed.

SAS Code



```
/* Step6: (AI) Obtain the information of loan applicants who submitted their
loan application without gender */

TITLE '(AI) Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without gender';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.gender eq '' OR e.gender IS MISSING);
QUIT;
```

Screenshot(s)



(AI) Obtain the information of loan applicants who submitted their
loan application without gender

-------END-------

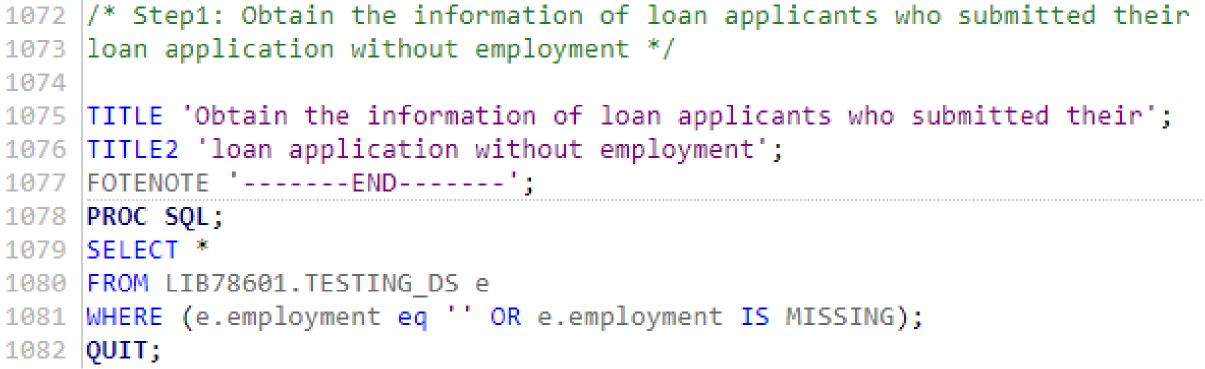
Description

We checked that there is not any missing value for this variable after imputing the existing missing values.

6.6.3 Impute the CATEGORICAL missing values found in the dataset – LIB78601.TRAINING_DS – [EMPLOYMENT]

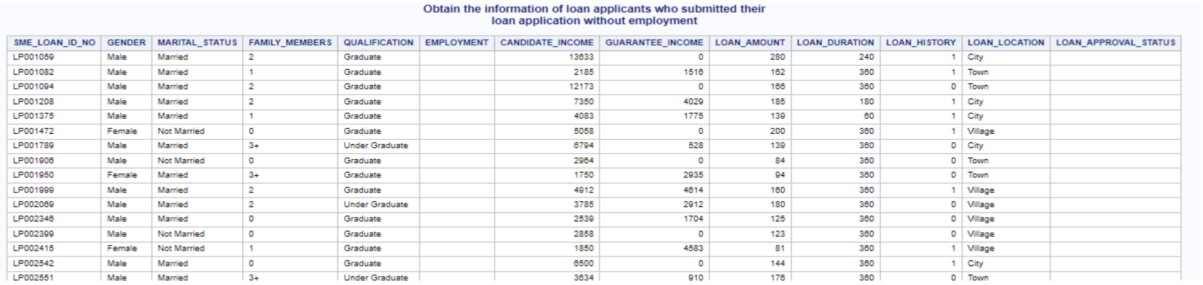
Step1: Obtain the information.

SAS Code



```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without employment */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without employment';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.employment eq '' OR e.employment IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without employment

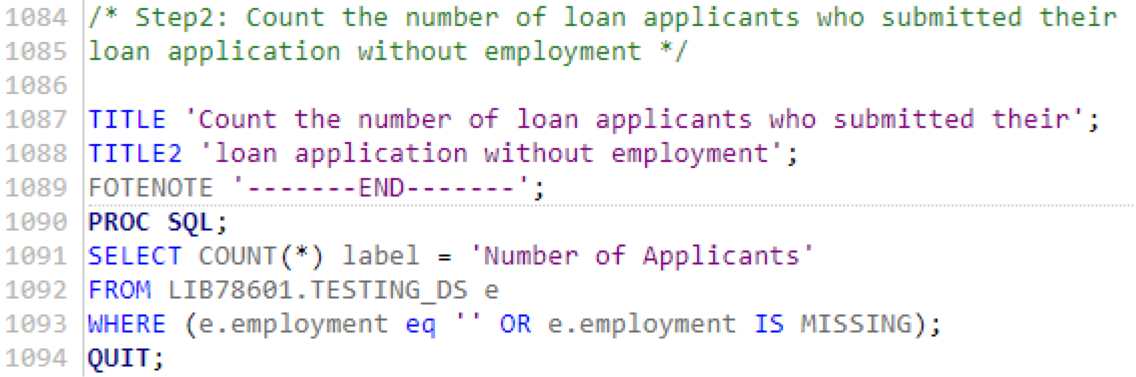
SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001059	Male	Married	2	Graduate		13633	0	280	240	1	City	
LP001082	Male	Married	1	Graduate		2185	1516	162	360	1	Town	
LP001094	Male	Married	2	Graduate		12173	0	166	360	0	Town	
LP001208	Male	Married	2	Graduate		7350	4029	185	180	1	City	
LP001375	Male	Married	1	Graduate		4083	1775	139	60	1	City	
LP001472	Female	Not Married	0	Graduate		5058	0	200	360	1	Village	
LP001789	Male	Married	3+	Under Graduate		6794	528	139	360	0	City	
LP001906	Male	Not Married	0	Graduate		2964	0	84	360	0	Town	
LP001950	Female	Married	3+	Graduate		1750	2935	94	360	0	Town	
LP001999	Male	Married	2	Graduate		4912	4614	160	360	1	Village	
LP002069	Male	Married	2	Under Graduate		3785	2912	180	360	0	Village	
LP002346	Male	Married	0	Graduate		2539	1704	125	360	0	Village	
LP002399	Male	Not Married	0	Graduate		2858	0	123	360	0	Village	
LP002415	Female	Not Married	1	Graduate		1850	4583	81	360	1	Village	
LP002542	Male	Married	0	Graduate		6500	0	144	360	1	City	
LP002551	Male	Married	3+	Under Graduate		3634	910	176	360	0	Town	

Description

From the screenshot, we can easily to see the full details of applicants that have missing values for employment.

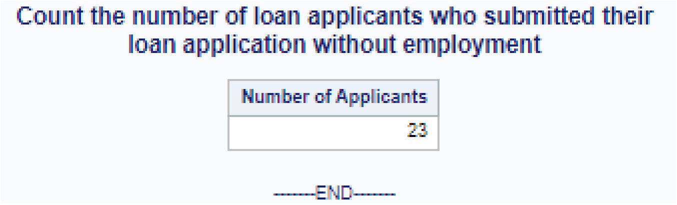
Step2: Count the number of missing values.

SAS Code



```
/* Step2: Count the number of loan applicants who submitted their
loan application without employment */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without employment';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TESTING_DS e
WHERE (e.employment eq '' OR e.employment IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their
loan application without employment

Number of Applicants
23

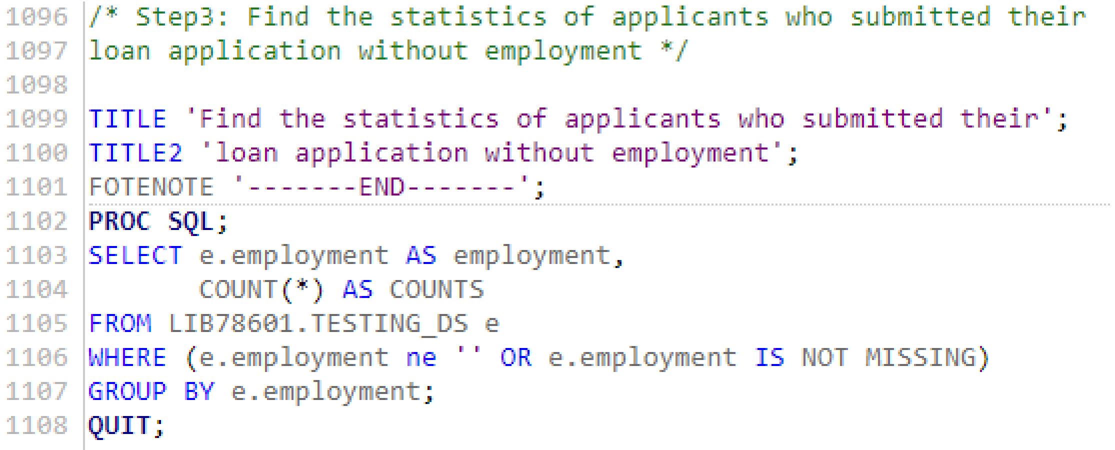
-------END-------

Description

Clearly, we can see that there is total 23 missing values for this variable, we will treat this issue in the coming steps.

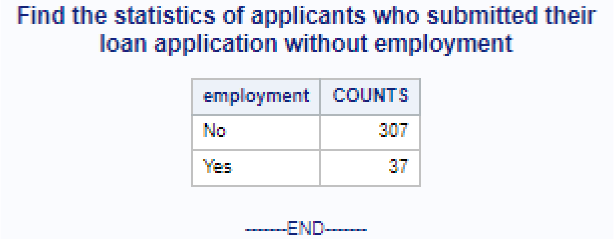
Step3: Statistics about missing values.

SAS Code



```
/* Step3: Find the statistics of applicants who submitted their
loan application without employment */

TITLE 'Find the statistics of applicants who submitted their';
TITLE2 'loan application without employment';
FOTENOTE '-------END-------';
PROC SQL;
SELECT e.employment AS employment,
       COUNT(*) AS COUNTS
FROM LIB78601.TESTING_DS e
WHERE (e.employment ne '' OR e.employment IS NOT MISSING)
GROUP BY e.employment;
QUIT;
```

Screenshot(s)



Find the statistics of applicants who submitted their
loan application without employment

employment	COUNTS
No	307
Yes	37

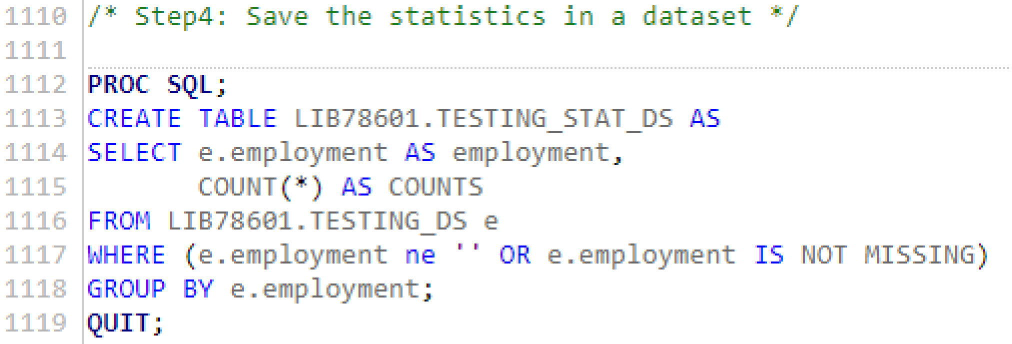
-------END-------

Description

We can observe that the number of each category for variable employment, this information is useful for choosing treatment method for impute missing values.

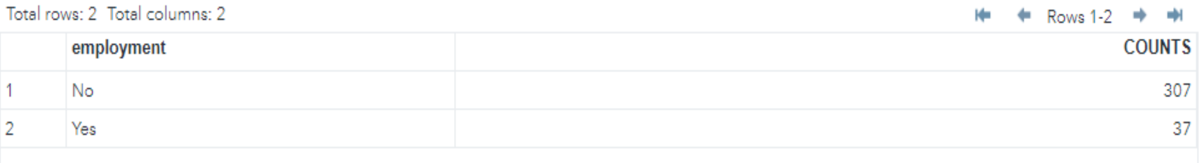
Step4: Save the statistics in a dataset.

SAS Code



```
/* Step4: Save the statistics in a dataset */

PROC SQL;
CREATE TABLE LIB78601.TESTING_STAT_DS AS
SELECT e.employment AS employment,
       COUNT(*) AS COUNTS
FROM LIB78601.TESTING_DS e
WHERE (e.employment ne '' OR e.employment IS NOT MISSING)
GROUP BY e.employment;
QUIT;
```

Screenshot(s)



Total rows: 2 Total columns: 2

Rows 1-2

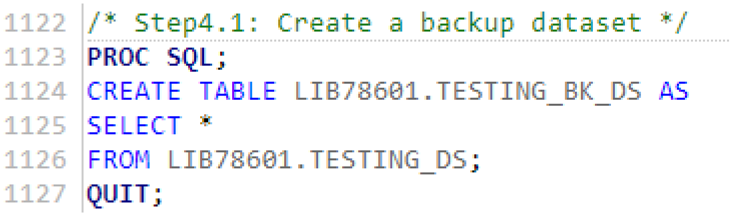
	employment	COUNTS
1	No	307
2	Yes	37

Description

Save the output from last output as a table format, this will help for selecting specific value for impute missing value for specific variable in general.

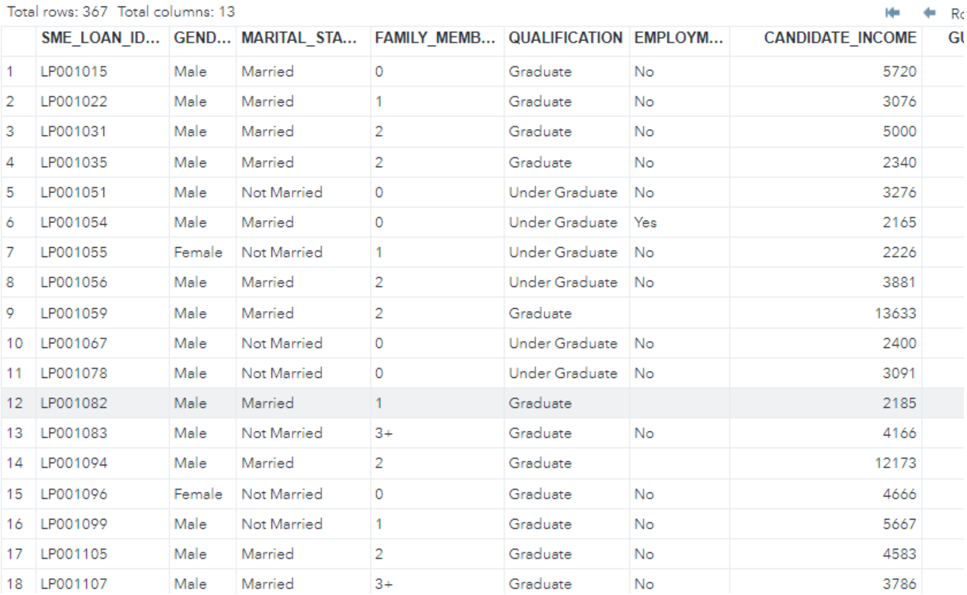
Step4.1: Backup the existing dataset.

SAS Code



```
/* Step4.1: Create a backup dataset */
PROC SQL;
CREATE TABLE LIB78601.TESTING_BK_DS AS
SELECT *
FROM LIB78601.TESTING_DS;
QUIT;
```

Screenshot(s)



Total rows: 367 Total columns: 13

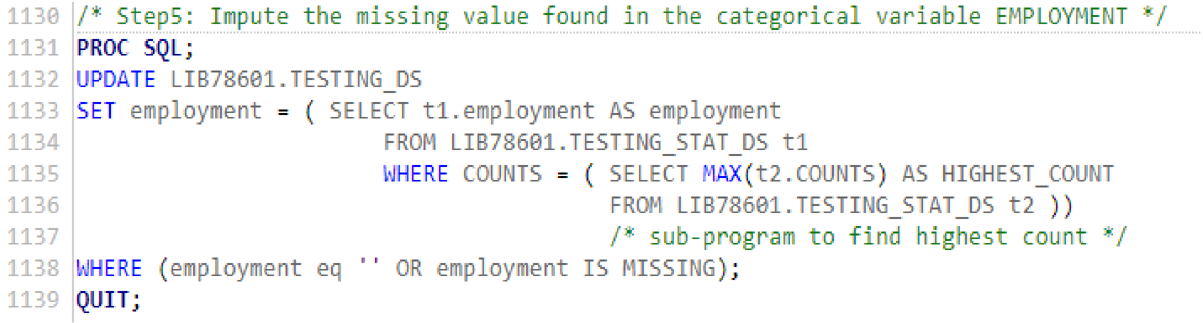
	SME_LOAN_ID...	GEND...	MARITAL_STA...	FAMILY_MEMB...	QUALIFICATION	EMPLOYM...	CANDIDATE_INCOME	Gl
1	LP001015	Male	Married	0	Graduate	No	5720	
2	LP001022	Male	Married	1	Graduate	No	3076	
3	LP001031	Male	Married	2	Graduate	No	5000	
4	LP001035	Male	Married	2	Graduate	No	2340	
5	LP001051	Male	Not Married	0	Under Graduate	No	3276	
6	LP001054	Male	Married	0	Under Graduate	Yes	2165	
7	LP001055	Female	Not Married	1	Under Graduate	No	2226	
8	LP001056	Male	Married	2	Under Graduate	No	3881	
9	LP001059	Male	Married	2	Graduate		13633	
10	LP001067	Male	Not Married	0	Under Graduate	No	2400	
11	LP001078	Male	Not Married	0	Under Graduate	No	3091	
12	LP001082	Male	Married	1	Graduate		2185	
13	LP001083	Male	Not Married	3+	Graduate	No	4166	
14	LP001094	Male	Married	2	Graduate		12173	
15	LP001096	Female	Not Married	0	Graduate	No	4666	
16	LP001099	Male	Not Married	1	Graduate	No	5667	
17	LP001105	Male	Married	2	Graduate	No	4583	
18	LP001107	Male	Married	3+	Graduate	No	3786	

Description

Create a table for duplicate the table before imputing the value for missing values cell. This action will guarantee that we will not corrupt the existing table.

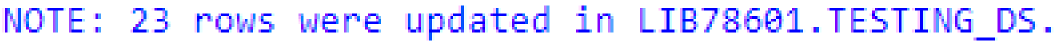
Step5: Impute the missing values.

SAS Code



```
/* Step5: Impute the missing value found in the categorical variable EMPLOYMENT */
PROC SQL;
UPDATE LIB78601.TESTING_DS
SET employment = ( SELECT t1.employment AS employment
                       FROM LIB78601.TESTING_STAT_DS t1
                       WHERE COUNTS = ( SELECT MAX(t2.COUNTS) AS HIGHEST_COUNT
                                        FROM LIB78601.TESTING_STAT_DS t2 ))
                                        /* sub-program to find highest count */
WHERE (employment eq '' OR employment IS MISSING);
QUIT;
```

Screenshot(s)



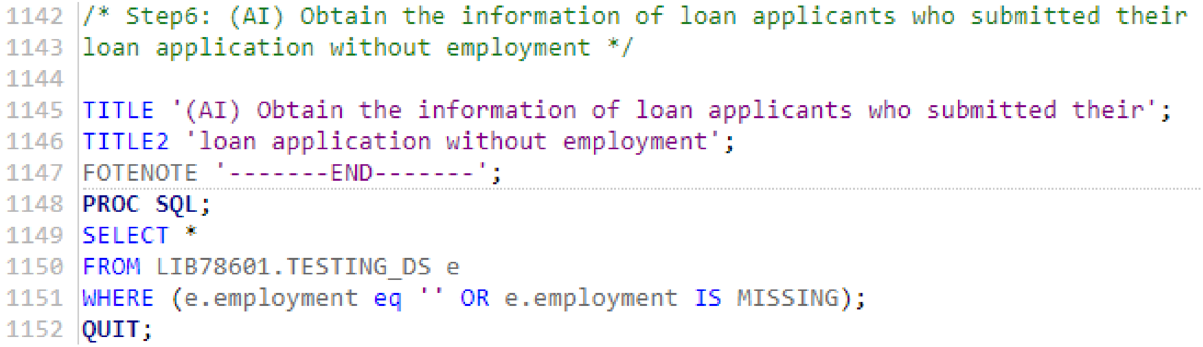
```
NOTE: 23 rows were updated in LIB78601.TESTING_DS.
```

Description

Impute the missing values by the mode of the observation from its variable. From the screenshot, we can see that 23 rows were updated.

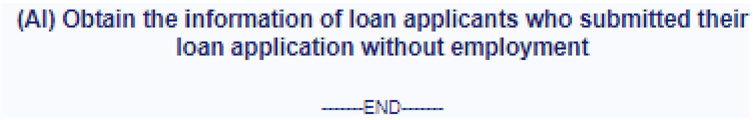
Step6: Ensure that the missing values was imputed.

SAS Code



```
/* Step6: (AI) Obtain the information of loan applicants who submitted their
loan application without employment */

TITLE '(AI) Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without employment';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.employment eq '' OR e.employment IS MISSING);
QUIT;
```

Screenshot(s)



(AI) Obtain the information of loan applicants who submitted their
loan application without employment

-------END-------

Description

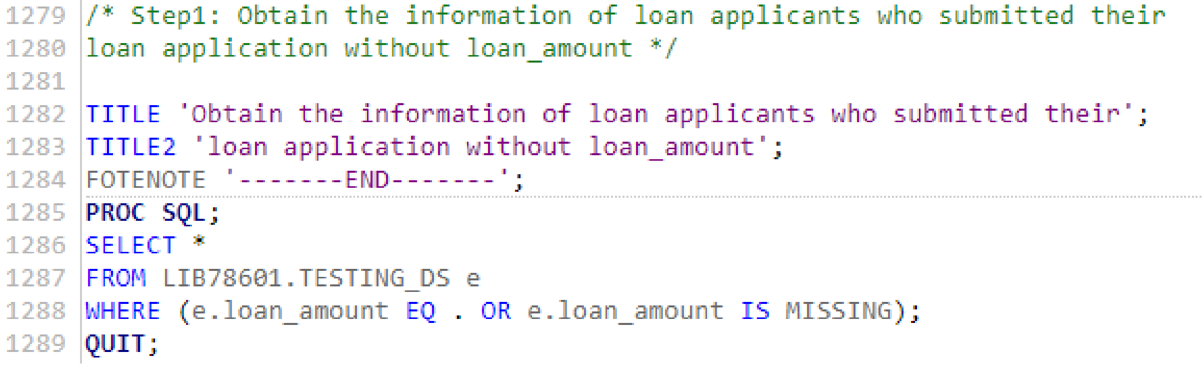
We checked that there is not any missing value for this variable after imputing the existing missing values.

6.7 Impute the CONTINUOUS missing values found in the dataset – LIB78601.TESTING_DS.

6.7.1 Impute the CONTINUOUS missing values found in the dataset – LIB78601.TESTING_DS – [LOAN_AMOUNT]

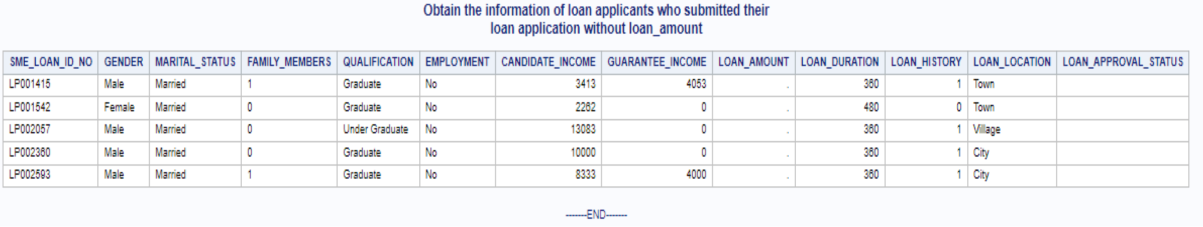
Step1: Obtain the information.

SAS Code



```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without loan_amount */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_amount';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.loan_amount EQ . OR e.loan_amount IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their
loan application without loan_amount

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001415	Male	Married	1	Graduate	No	3413	4053	.	360	1	Town	
LP001542	Female	Married	0	Graduate	No	2262	0	.	480	0	Town	
LP002057	Male	Married	0	Under Graduate	No	13083	0	.	360	1	Village	
LP002360	Male	Married	0	Graduate	No	10000	0	.	360	1	City	
LP002593	Male	Married	1	Graduate	No	8333	4000	.	360	1	City	

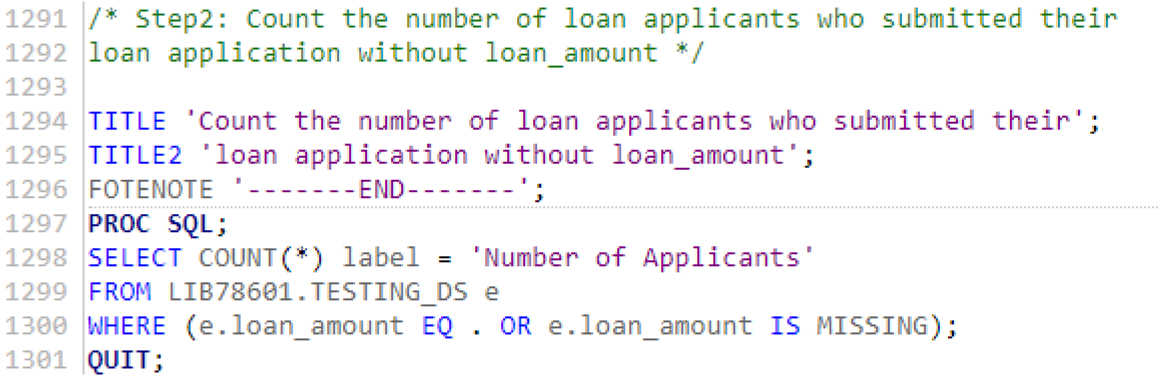
-------END-------

Description

From the screenshot, we can easily to see the full details of applicants that have missing values for loan amount.

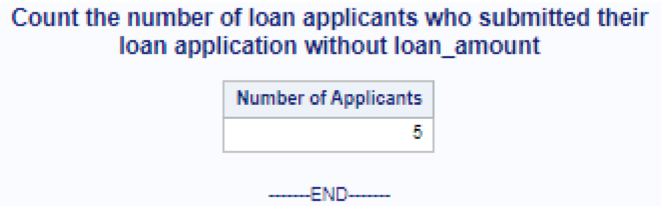
Step2: Count the number of missing values.

SAS Code



```
/* Step2: Count the number of loan applicants who submitted their
loan application without loan_amount */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_amount';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TESTING_DS e
WHERE (e.loan_amount EQ . OR e.loan_amount IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their loan application without loan_amount

Number of Applicants
5

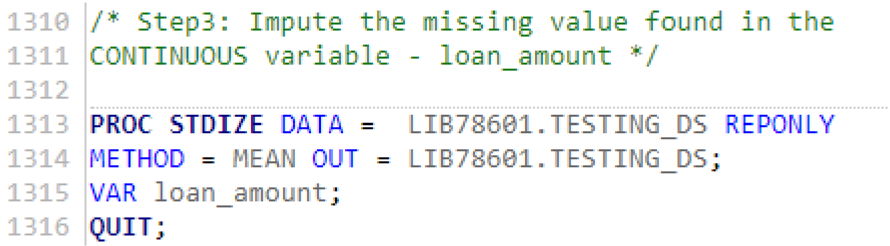
-------END-------

Description

Clearly, we can see that there is total 5 missing values for this variable, we will treat this issue in the coming steps.

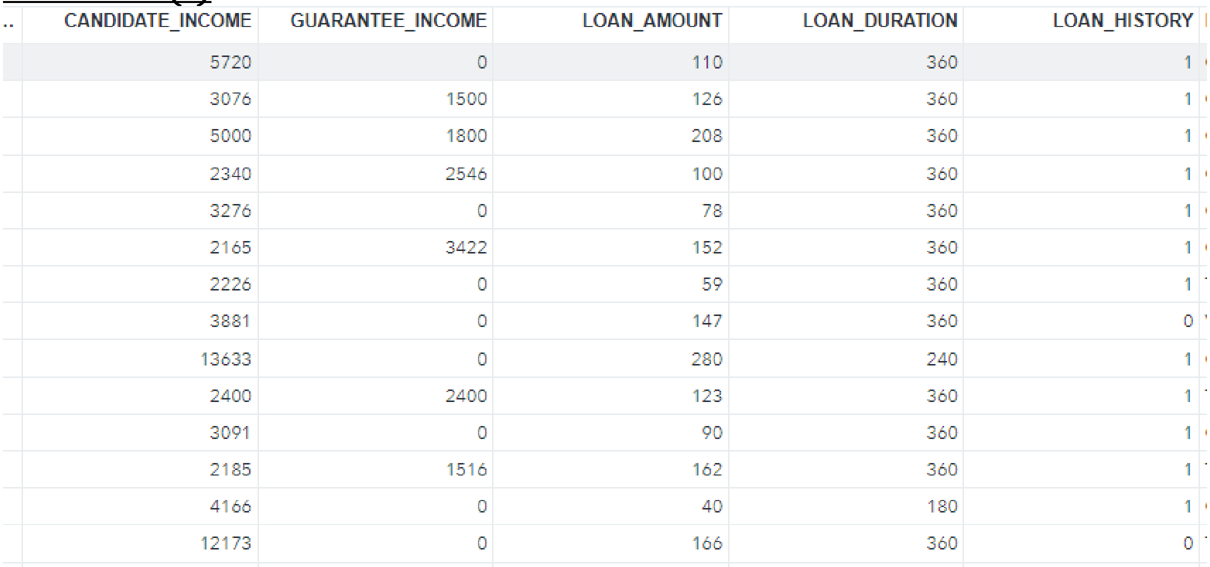
Step3: Impute the missing value found in the CONTINUOUS variable - loan_amount.

SAS Code



```
/* Step3: Impute the missing value found in the
CONTINUOUS variable - loan_amount */

PROC STDIZE DATA =  LIB78601.TESTING_DS REPONLY
METHOD = MEAN OUT = LIB78601.TESTING_DS;
VAR loan_amount;
QUIT;
```

Screenshot(s)



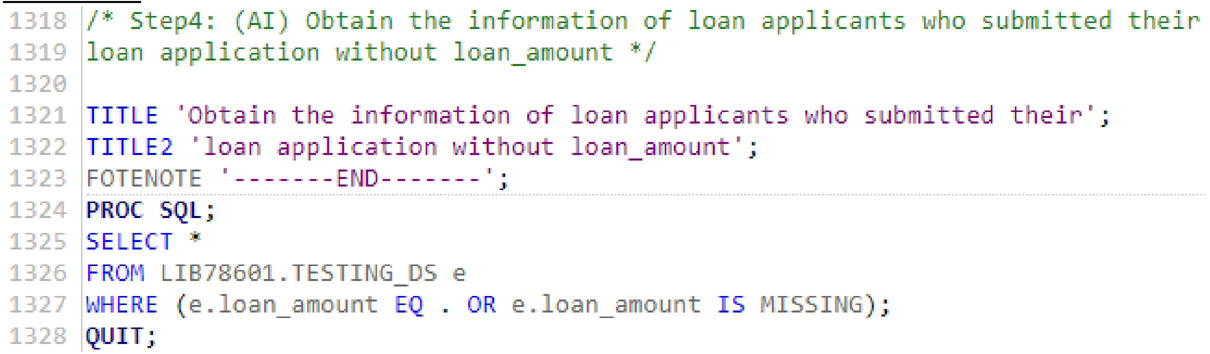
..	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY
	5720	0	110	360	1
	3076	1500	126	360	1
	5000	1800	208	360	1
	2340	2546	100	360	1
	3276	0	78	360	1
	2165	3422	152	360	1
	2226	0	59	360	1
	3881	0	147	360	0
	13633	0	280	240	1
	2400	2400	123	360	1
	3091	0	90	360	1
	2185	1516	162	360	1
	4166	0	40	180	1
	12173	0	166	360	0

Description

In this step, we impute the missing value of variable loan amount by its mean value, this can guarantee that the missing data didn't affect the mean value of existing data.

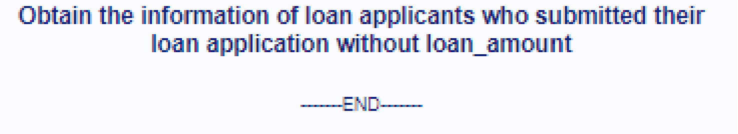
Step4: (AI) Obtain the information.

SAS Code



```
/* Step4: (AI) Obtain the information of loan applicants who submitted their
loan application without loan_amount */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_amount';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.loan_amount EQ . OR e.loan_amount IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their loan application without loan_amount

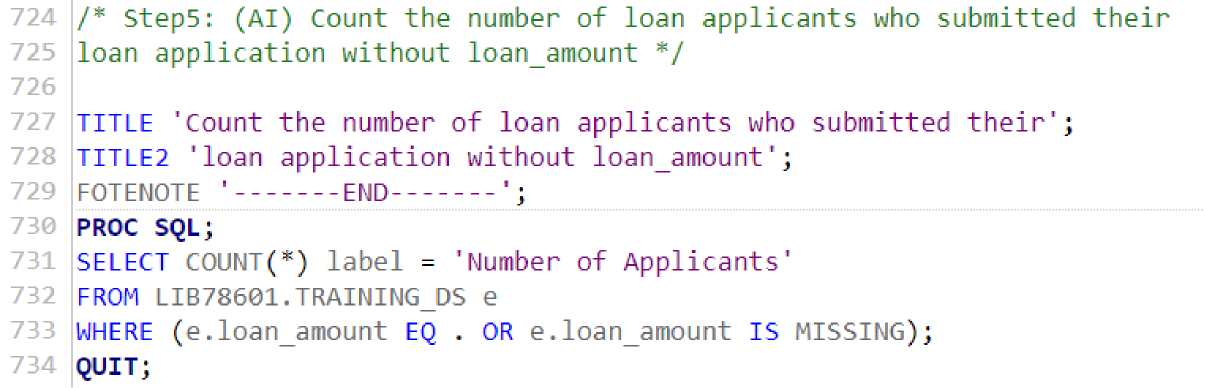
-------END-------

Description

From the screenshot, we can easily to see that there is not any missing value after imputing. This step is to guarantee that the last step has impute all the missing values.

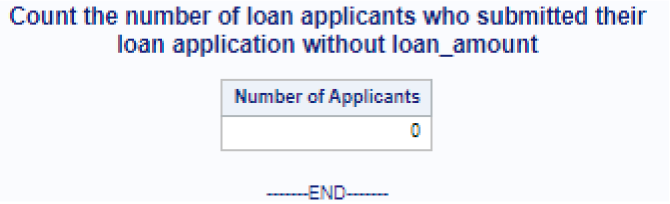
Step5: (AI) Count the number of missing values.

SAS Code



```
/* Step5: (AI) Count the number of loan applicants who submitted their
loan application without loan_amount */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_amount';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TRAINING_DS e
WHERE (e.loan_amount EQ . OR e.loan_amount IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their loan application without loan_amount

Number of Applicants
0

-------END-------

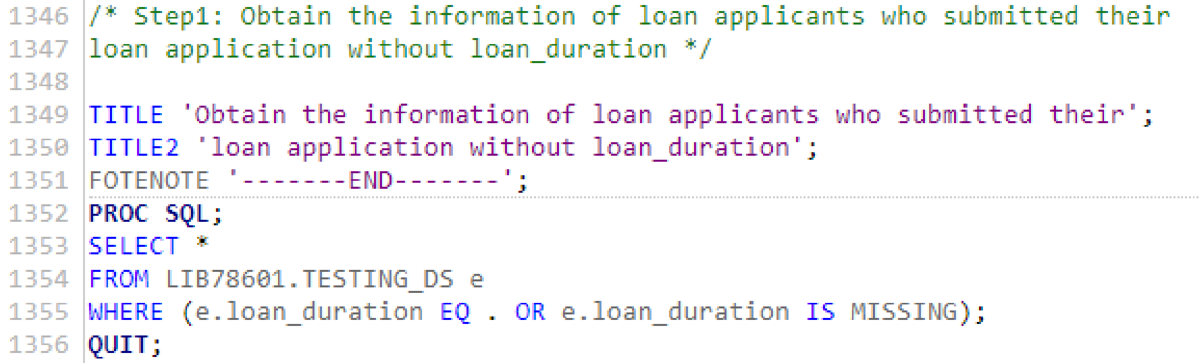
Description

Clearly, we can see that there is total 0 missing value after imputing, this step is needed to double check that we have no making any mistake when imputing missing values.

6.7.2 Impute the CONTINUOUS missing values found in the dataset – LIB78601.TRAINING_DS – [LOAN_DURATION]

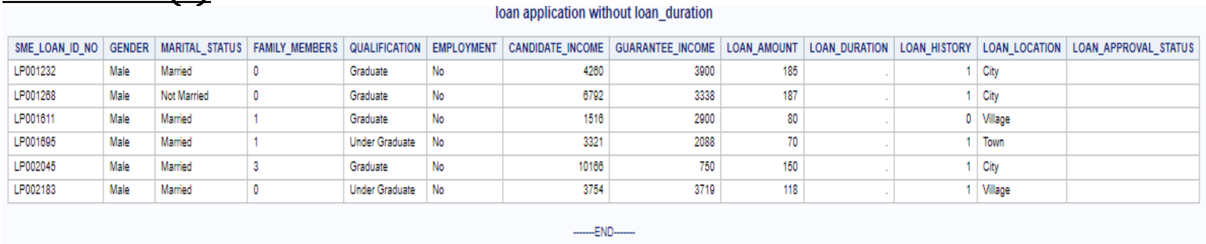
Step1: Obtain the information.

SAS Code



```
/* Step1: Obtain the information of loan applicants who submitted their
loan application without loan_duration */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_duration';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.loan_duration EQ . OR e.loan_duration IS MISSING);
QUIT;
```

Screenshot(s)



loan application without loan_duration

SME_LOAN_ID_NO	GENDER	MARITAL_STATUS	FAMILY_MEMBERS	QUALIFICATION	EMPLOYMENT	CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS
LP001232	Male	Married	0	Graduate	No	4260	3900	185	.	1	City	
LP001268	Male	Not Married	0	Graduate	No	6792	3338	187	.	1	City	
LP001611	Male	Married	1	Graduate	No	1516	2900	80	.	0	Village	
LP001695	Male	Married	1	Under Graduate	No	3321	2088	70	.	1	Town	
LP002045	Male	Married	3	Graduate	No	10166	750	150	.	1	City	
LP002183	Male	Married	0	Under Graduate	No	3754	3719	118	.	1	Village	

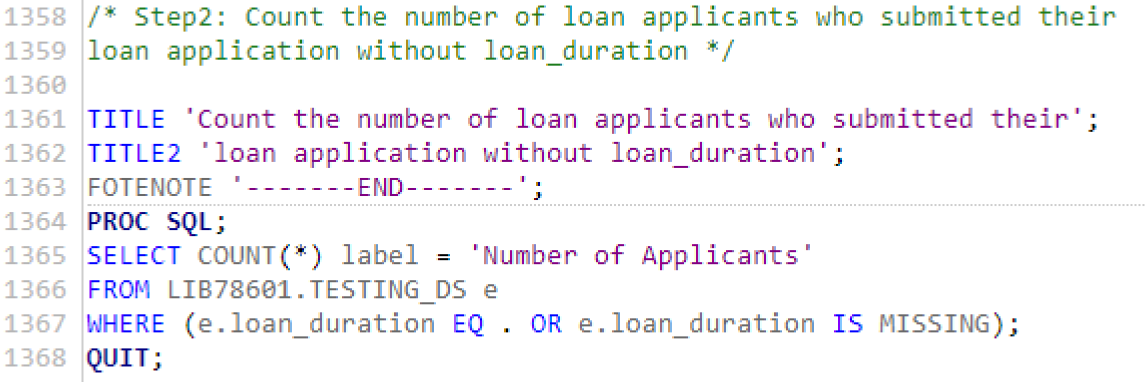
-------END-------

Description

From the screenshot, we can easily to see the full details of applicants that have missing values for loan duration.

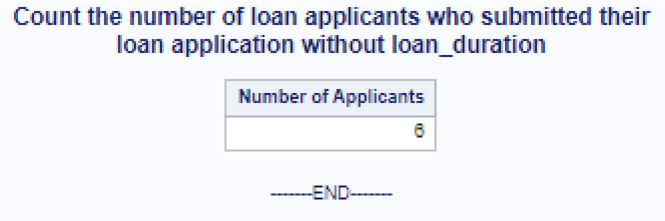
Step2: Count the number of missing values.

SAS Code



```
/* Step2: Count the number of loan applicants who submitted their
loan application without loan_duration */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_duration';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TESTING_DS e
WHERE (e.loan_duration EQ . OR e.loan_duration IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their loan application without loan_duration

Number of Applicants
6

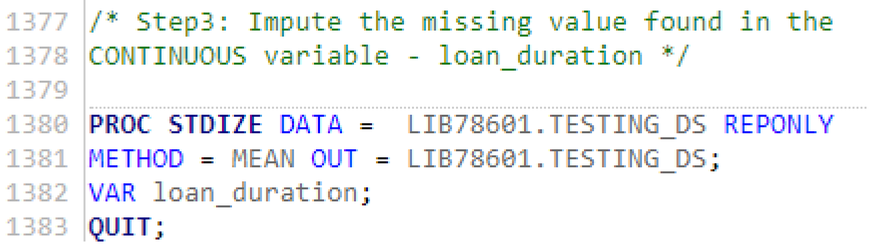
-------END-------

Description

Clearly, we can see that there is total 6 missing values for this variable, we will treat this issue in the coming steps.

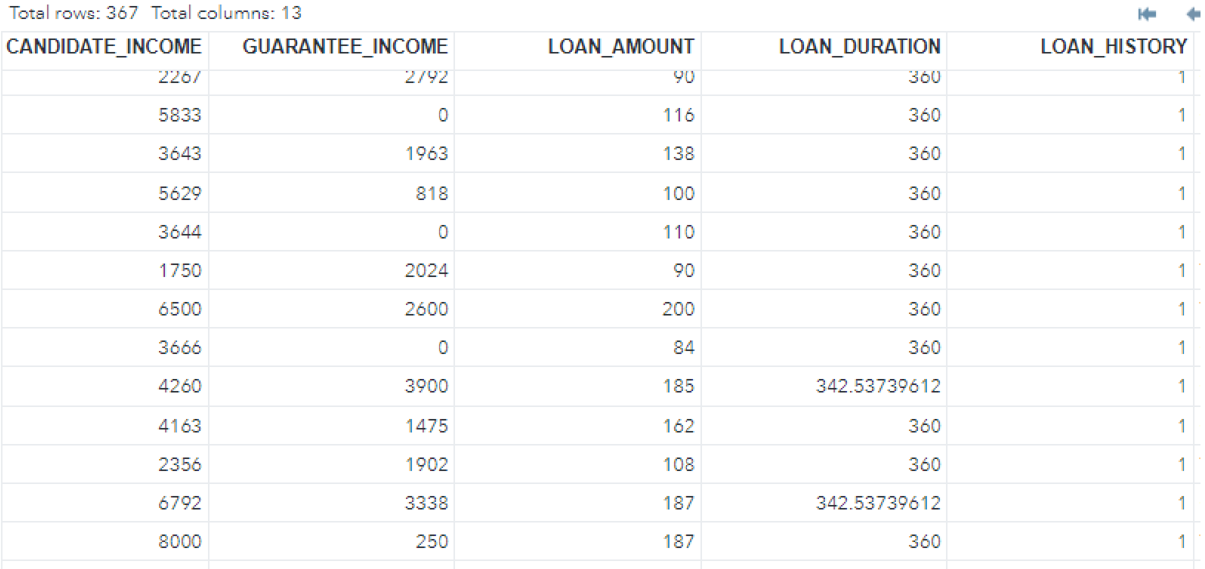
Step3: Impute the missing value found in the CONTINUOUS variable - loan_duration.

SAS Code



```
/* Step3: Impute the missing value found in the
CONTINUOUS variable - loan_duration */

PROC STDIZE DATA =  LIB78601.TESTING_DS REPONLY
METHOD = MEAN OUT = LIB78601.TESTING_DS;
VAR loan_duration;
QUIT;
```

Screenshot(s)



Total rows: 367 Total columns: 13

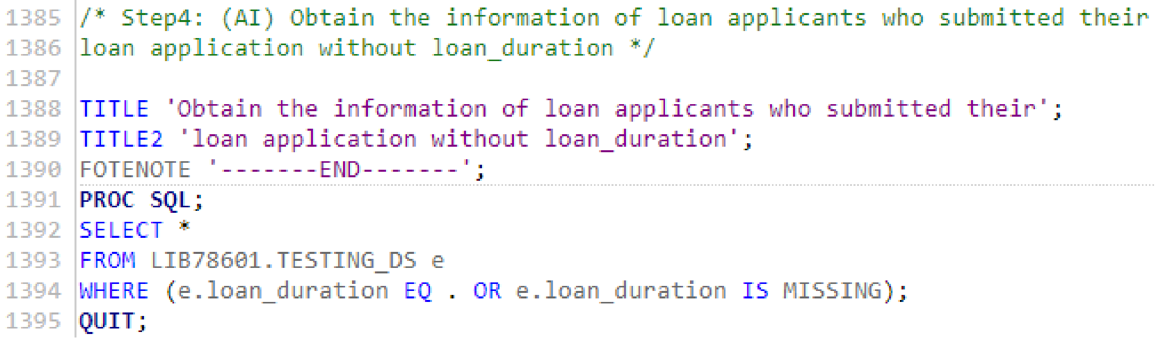
CANDIDATE_INCOME	GUARANTEE_INCOME	LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY
2267	2792	90	360	1
5833	0	116	360	1
3643	1963	138	360	1
5629	818	100	360	1
3644	0	110	360	1
1750	2024	90	360	1
6500	2600	200	360	1
3666	0	84	360	1
4260	3900	185	342.53739612	1
4163	1475	162	360	1
2356	1902	108	360	1
6792	3338	187	342.53739612	1
8000	250	187	360	1

Description

In this step, we impute the missing value of variable loan amount by its mean value, this can guarantee that the missing data didn't affect the mean value of existing data.

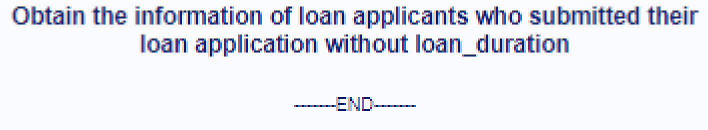
Step4: (AI) Obtain the information.

SAS Code



```
/* Step4: (AI) Obtain the information of loan applicants who submitted their
loan application without loan_duration */

TITLE 'Obtain the information of loan applicants who submitted their';
TITLE2 'loan application without loan_duration';
FOTENOTE '-------END-------';
PROC SQL;
SELECT *
FROM LIB78601.TESTING_DS e
WHERE (e.loan_duration EQ . OR e.loan_duration IS MISSING);
QUIT;
```

Screenshot(s)



Obtain the information of loan applicants who submitted their loan application without loan_duration

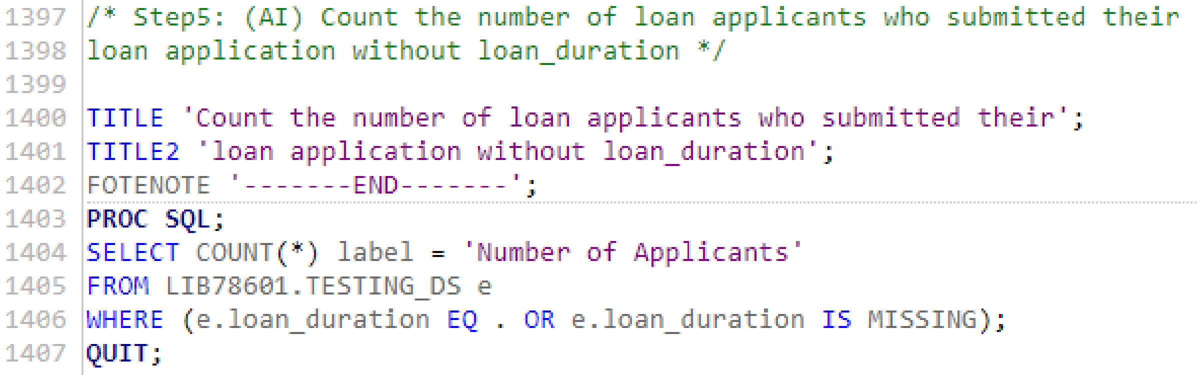
-------END-------

Description

From the screenshot, we can easily to see that there is not any missing value after imputing. This step is to guarantee that the last step has impute all the missing values.

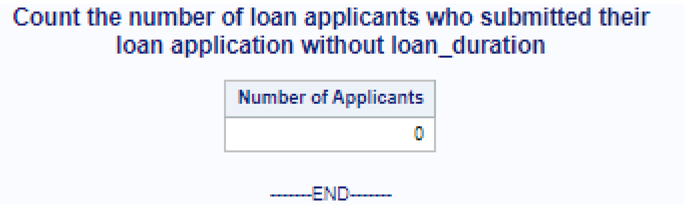
Step5: (AI) Count the number of missing values.

SAS Code



```
/* Step5: (AI) Count the number of loan applicants who submitted their
loan application without loan_duration */

TITLE 'Count the number of loan applicants who submitted their';
TITLE2 'loan application without loan_duration';
FOTENOTE '-------END-------';
PROC SQL;
SELECT COUNT(*) label = 'Number of Applicants'
FROM LIB78601.TESTING_DS e
WHERE (e.loan_duration EQ . OR e.loan_duration IS MISSING);
QUIT;
```

Screenshot(s)



Count the number of loan applicants who submitted their loan application without loan_duration

Number of Applicants
0

-------END-------

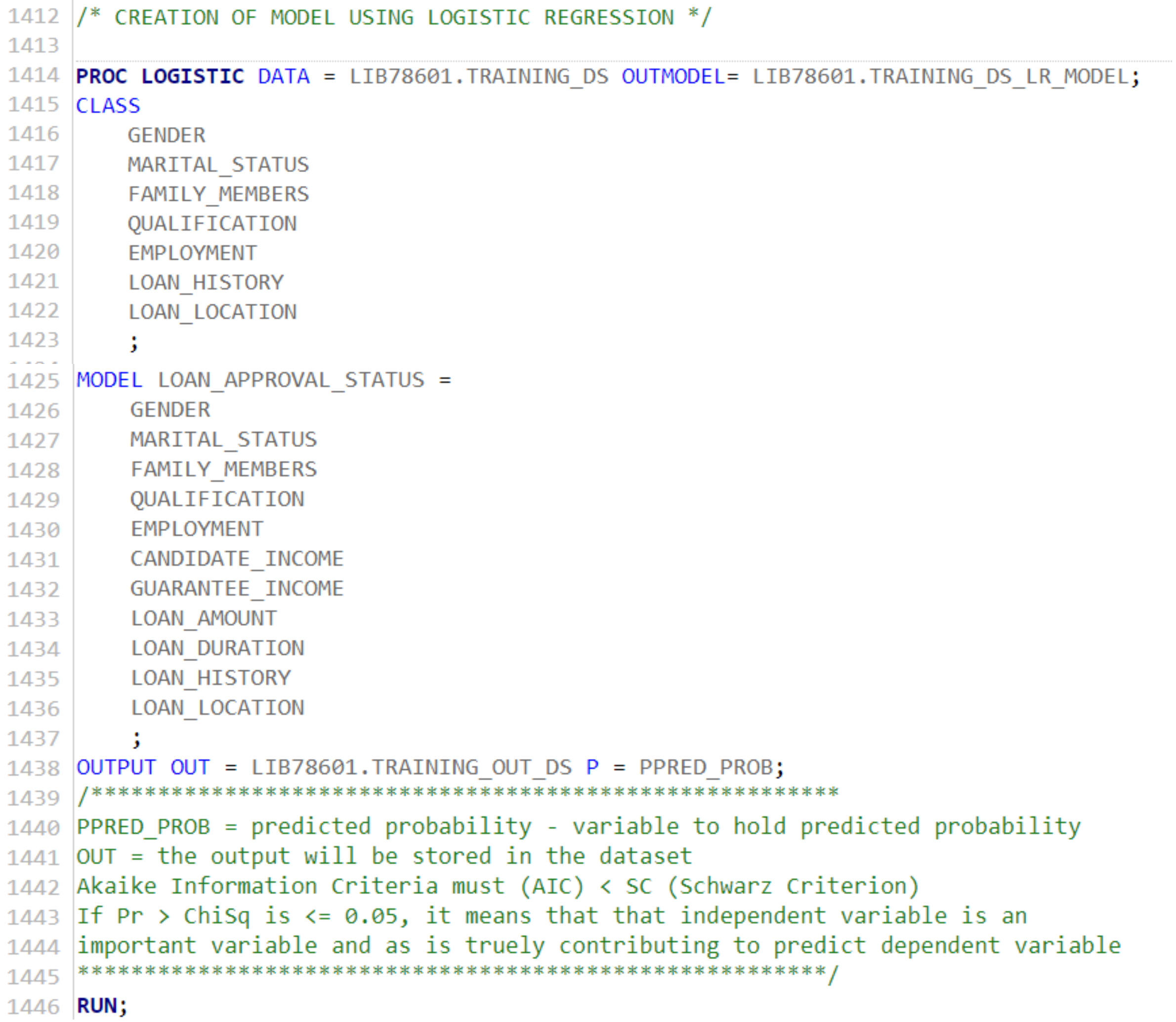
Description

Clearly, we can see that there is total 0 missing value after imputing, this step is needed to double check that we have no making any mistake when imputing missing values.

CHAPTER 7: CREATION OF MODEL USING LOGISTIC REGRESSION

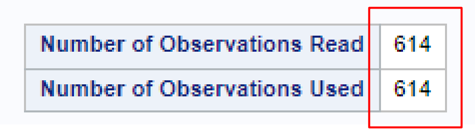
7.1 Create Logistic Regression Model

SAS Code



```
/* CREATION OF MODEL USING LOGISTIC REGRESSION */

PROC LOGISTIC DATA = LIB78601.TRAINING_DS OUTMODEL= LIB78601.TRAINING_DS_LR_MODEL;
CLASS
    GENDER
    MARITAL_STATUS
    FAMILY_MEMBERS
    QUALIFICATION
    EMPLOYMENT
    LOAN_HISTORY
    LOAN_LOCATION
    ;
MODEL LOAN_APPROVAL_STATUS =
    GENDER
    MARITAL_STATUS
    FAMILY_MEMBERS
    QUALIFICATION
    EMPLOYMENT
    CANDIDATE_INCOME
    GUARANTEE_INCOME
    LOAN_AMOUNT
    LOAN_DURATION
    LOAN_HISTORY
    LOAN_LOCATION
    ;
OUTPUT OUT = LIB78601.TRAINING_OUT_DS P = PPRED_PROB;
/*******************************************************
PPRED_PROB = predicted probability - variable to hold predicted probability
OUT = the output will be stored in the dataset
Akaike Information Criteria must (AIC) < SC (Schwarz Criterion)
If Pr > ChiSq is <= 0.05, it means that that independent variable is an
important variable and as is truely contributing to predict dependent variable
*******************************************************/
RUN;
```

Screenshot(s)

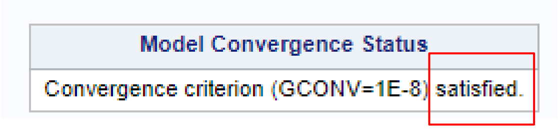


Number of Observations Read	614
Number of Observations Used	614

Description

Now, we can start to build the prediction model with logistic regression to predict the loan approval status by applicants' information. In the coding part, we have entered the variable used for building up the model. From the output, we can see that there are 614 observations in the training dataset and all of them are used in training the logistic regression model. The trained model will fit the observation through minimize the misclassification that suit the issue from loan approval.

Screenshot(s)

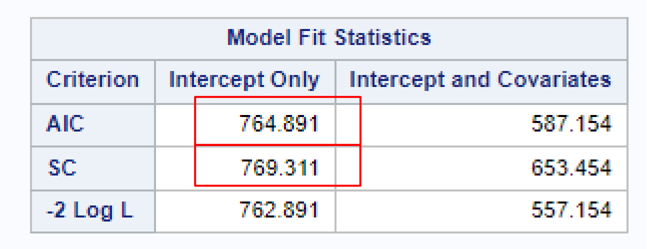


Model Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Description

From the output, we can observe that the convergence criterion is satisfied, this means that the model is trained well.

Screenshot(s)

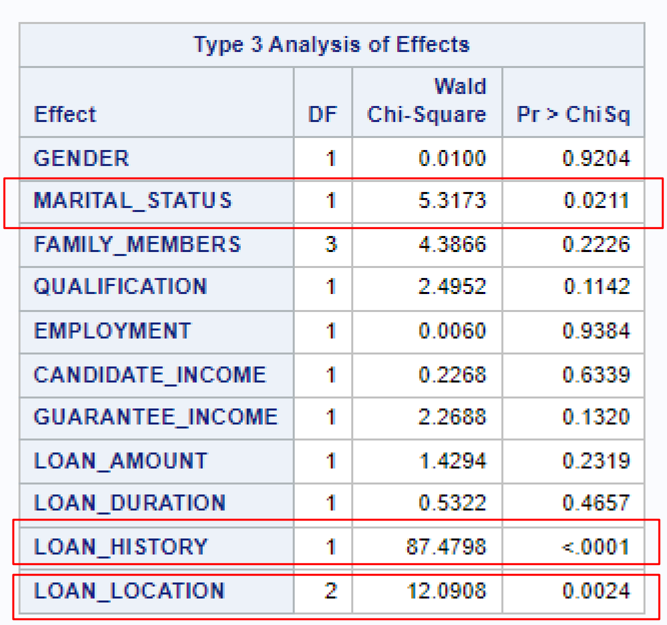


Model Fit Statistics		
Criterion	Intercept Only	Intercept and Covariates
AIC	764.891	587.154
SC	769.311	653.454
-2 Log L	762.891	557.154

Description

Clearly, we can see that the value for Akaike Information Criteria (AIC) was 764.891 and the value for Schwarz Criterion (SC) was 769.311. Since we have AIC < AC, this indicate that the model is good fit.

Screenshot(s)



Type 3 Analysis of Effects

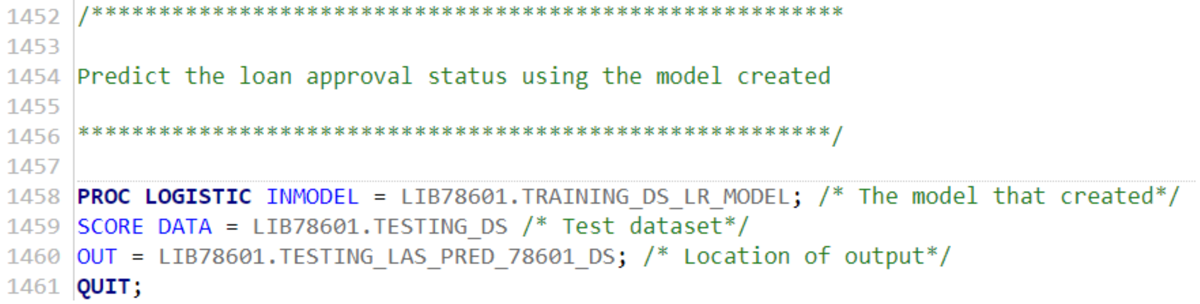
Effect	DF	Wald Chi-Square	Pr > ChiSq
GENDER	1	0.0100	0.9204
MARITAL_STATUS	1	5.3173	0.0211
FAMILY_MEMBERS	3	4.3866	0.2226
QUALIFICATION	1	2.4952	0.1142
EMPLOYMENT	1	0.0060	0.9384
CANDIDATE_INCOME	1	0.2268	0.6339
GUARANTEE_INCOME	1	2.2688	0.1320
LOAN_AMOUNT	1	1.4294	0.2319
LOAN_DURATION	1	0.5322	0.4657
LOAN_HISTORY	1	87.4798	<.0001
LOAN_LOCATION	2	12.0908	0.0024

Description

From the output, we focus on the right column. We can see that the probability of variables marital_status, loan_history, and loan_location was 0.0211, <0.0001, and 0.0024 respectively, this indicates that these three variables have truly contributing to the logistic regression model for loan approval prediction.

7.2 Predicting the loan approval status using the LR model created.

SAS Code



```
/***********************************************************

Predict the loan approval status using the model created

**********************************************************/

PROC LOGISTIC INMODEL = LIB78601.TRAINING_DS_LR_MODEL; /* The model that created*/
SCORE DATA = LIB78601.TESTING_DS /* Test dataset*/
OUT = LIB78601.TESTING_LAS_PRED_78601_DS; /* Location of output*/
QUIT;
```

Screenshot(s)

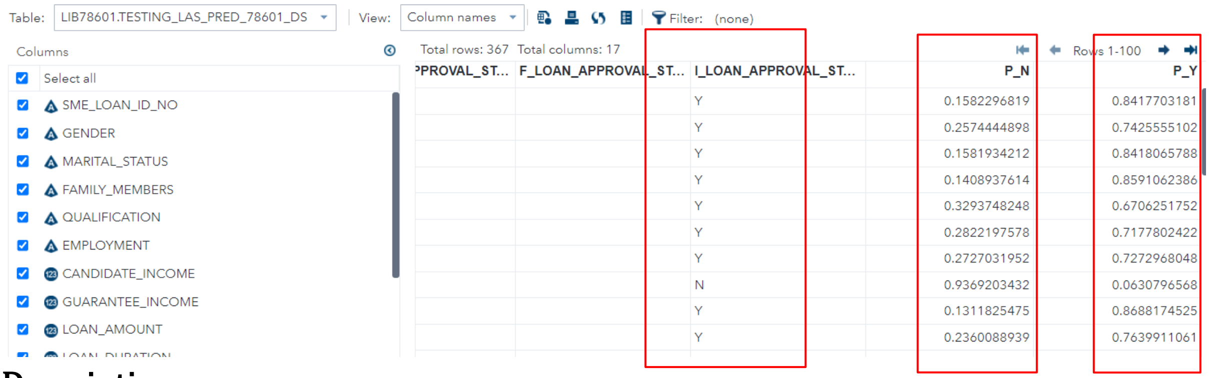


Table: LIB78601.TESTING_LAS_PRED_78601_DS View: Column names Filter: (none)

Columns

☑ Select all
☑ SME_LOAN_ID_NO
☑ GENDER
☑ MARITAL_STATUS
☑ FAMILY_MEMBERS
☑ QUALIFICATION
☑ EMPLOYMENT
☑ CANDIDATE_INCOME
☑ GUARANTEE_INCOME
☑ LOAN_AMOUNT

Total rows: 367 Total columns: 17 Rows 1-100

PPROVAL_ST...	F_LOAN_APPROVAL_ST...	I_LOAN_APPROVAL_ST...	P_N	P_Y
		Y	0.1582296819	0.8417703181
		Y	0.2574444898	0.7425555102
		Y	0.1581934212	0.8418065788
		Y	0.1408937614	0.8591062386
		Y	0.3293748248	0.6706251752
		Y	0.2822197578	0.7177802422
		Y	0.2727031952	0.7272968048
		N	0.9369203432	0.0630796568
		Y	0.1311825475	0.8688174525
		Y	0.2360088939	0.7639911061

Description

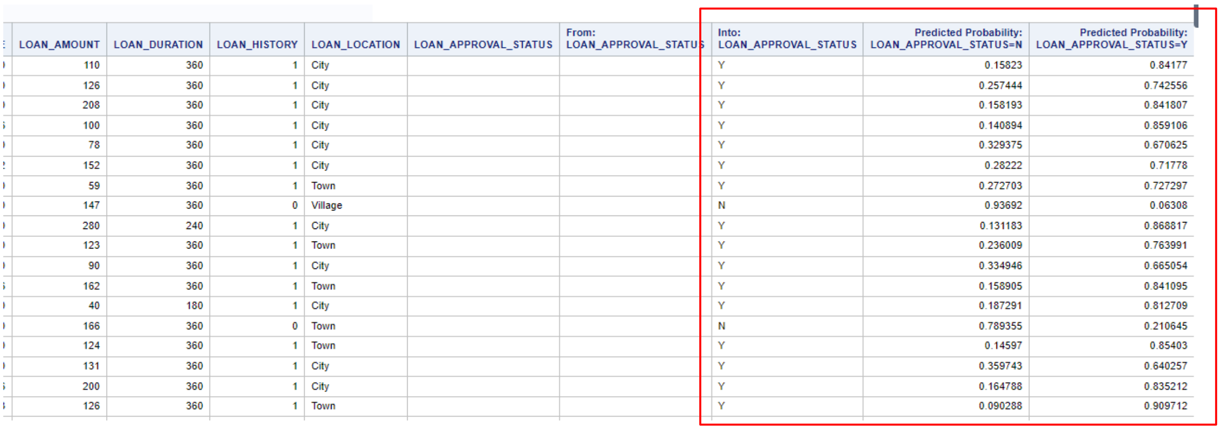
From the output, we can see that the logistic regression has given the probability of no and yes for each observation, the probability was listed in the right column respectively. Other than that, the predicted loan approval status also listed in the list. From the list, Y represents 'Yes' and N represents 'No' for the loan approval status.

7.3 Report generation using the SAS ODL – output delivery/display system.

SAS Code

```
TITLE 'Display the details of the loan approval status predicted';
FOTENOTE '-------END-------';

PROC SQL;
SELECT *
FROM LIB78601.TESTING_LAS_PRED_78601_DS;
QUIT;
```

Screenshot(s)



LOAN_AMOUNT	LOAN_DURATION	LOAN_HISTORY	LOAN_LOCATION	LOAN_APPROVAL_STATUS	From: LOAN_APPROVAL_STATUS	Into: LOAN_APPROVAL_STATUS	Predicted Probability: LOAN_APPROVAL_STATUS=N	Predicted Probability: LOAN_APPROVAL_STATUS=Y
110	360	1	City			Y	0.15823	0.84177
126	360	1	City			Y	0.257444	0.742556
208	360	1	City			Y	0.158193	0.841807
100	360	1	City			Y	0.140894	0.859106
78	360	1	City			Y	0.329375	0.670625
152	360	1	City			Y	0.28222	0.71778
59	360	1	Town			Y	0.272703	0.727297
147	360	0	Village			N	0.93692	0.06308
280	240	1	City			Y	0.131183	0.868817
123	360	1	Town			Y	0.236009	0.763991
90	360	1	City			Y	0.334946	0.665054
162	360	1	Town			Y	0.158905	0.841095
40	180	1	City			Y	0.187291	0.812709
166	360	0	Town			N	0.789355	0.210645
124	360	1	Town			Y	0.14597	0.85403
131	360	1	City			Y	0.359743	0.640257
200	360	1	City			Y	0.164788	0.835212
126	360	1	Town			Y	0.090288	0.909712

Description

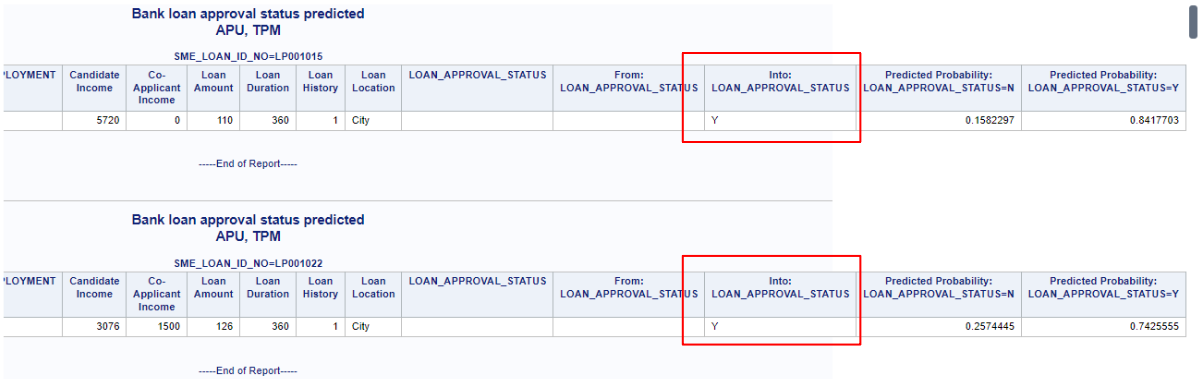
In this section, we want to finalize the predictions output. First, we see that all observations have their output. If there are some missing values in the dataset, the predictions output will also be a missing value. From the output, we can check that there is no missing value from the output.

SAS Code

```
/* Generate the report using SAS ODS - output delivery/display system
Display the details of the loan approval status predicted */


ODS HTML CLOSE;
ODS PDF CLOSE;
/* Determine the physical location of pdf*/
ODS PDF FILE = "/home/u63691871/DAP_FT_MAR_2024_TP078601/LAS78601_report.pdf";
OPTIONS NODATE;
TITLE1 'Bank loan approval status predicted';
TITLE2 'APU, TPM';
PROC REPORT DATA = LIB78601.TESTING_LAS_PRED_78601_DS NOWINDOWS;
BY SME_LOAN_ID_NO;
DEFINE SME_LOAN_ID_NO / GROUP 'Loan ID';
DEFINE GENDER / GROUP 'Gender Name';
DEFINE MARITAL_STATUS / GROUP 'Marital Status';
DEFINE FAMILY_MEMBERS / GROUP 'Family Members';
DEFINE CANDIDATE_INCOME / GROUP 'Candidate Income';
DEFINE GUARANTEE_INCOME / GROUP 'Co-Applicant Income';
DEFINE LOAN_AMOUNT / GROUP 'Loan Amount';
DEFINE LOAN_DURATION / GROUP 'Loan Duration';
DEFINE LOAN_HISTORY / GROUP 'Loan History';
DEFINE LOAN_LOCATION / GROUP 'Loan Location';
FOOTNOTE '-----End of Report-----';
RUN;
```

Screenshot(s)

v



Bank loan approval status predicted
APU, TPM

SME_LOAN_ID_NO=LP001015

'LOYMENT	Candidate Income	Co-Applicant Income	Loan Amount	Loan Duration	Loan History	Loan Location	LOAN_APPROVAL_STATUS	From: LOAN_APPROVAL_STATUS	Into: LOAN_APPROVAL_STATUS	Predicted Probability: LOAN_APPROVAL_STATUS=N	Predicted Probability: LOAN_APPROVAL_STATUS=Y
	5720	0	110	360	1	City			Y	0.1582297	0.8417703

-----End of Report-----

Bank loan approval status predicted
APU, TPM

SME_LOAN_ID_NO=LP001022

'LOYMENT	Candidate Income	Co-Applicant Income	Loan Amount	Loan Duration	Loan History	Loan Location	LOAN_APPROVAL_STATUS	From: LOAN_APPROVAL_STATUS	Into: LOAN_APPROVAL_STATUS	Predicted Probability: LOAN_APPROVAL_STATUS=N	Predicted Probability: LOAN_APPROVAL_STATUS=Y
	3076	1500	126	360	1	City			Y	0.2574445	0.7425555

-----End of Report-----

Description

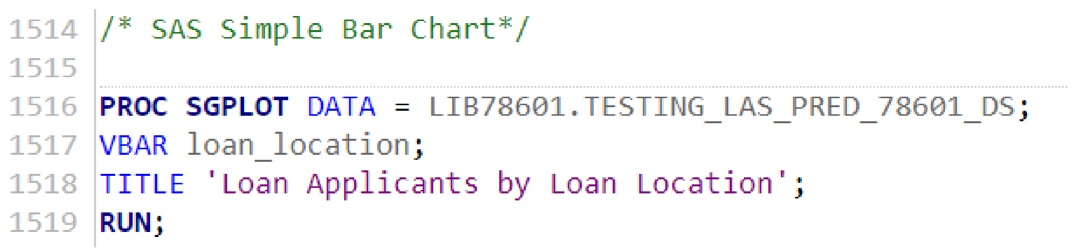
Now, we want to generate the report for predicted output. From the screenshot, we can see that every single prediction will be written in a single form. This is easy to distribute the result for each applicant. The report has recorded the loan approval status is approve or disapprove.

CHAPTER 8: DATA VISUALIZATION

8.1 Introduction

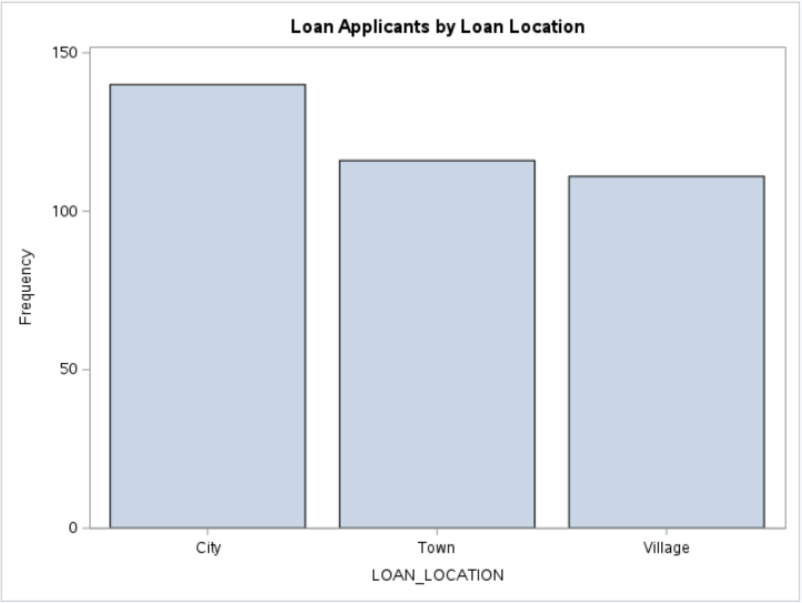
In this section, the predicted outcomes from the loan approval prediction model are shown graphically. These data visualizations show important trends and patterns in the data, providing clear and meaningful visualizations of the model's performance. We seek to provide more comprehension of the outcomes of the model and their implications for loan process approval by using several of charts and graphs.

SAS Code



```
/* SAS Simple Bar Chart*/

PROC SGPLOT DATA = LIB78601.TESTING_LAS_PRED_78601_DS;
VBAR loan_location;
TITLE 'Loan Applicants by Loan Location';
RUN;
```

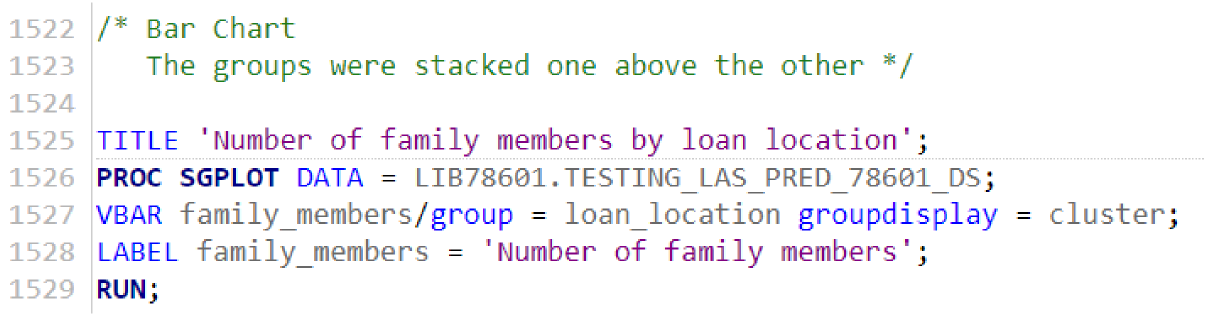
Screenshot(s)



Description

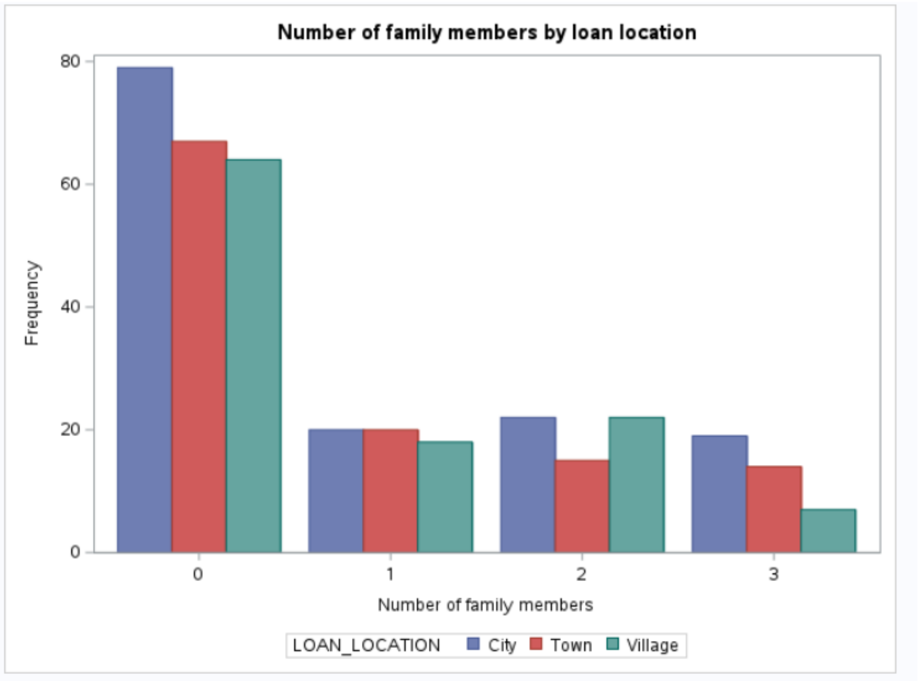
It is obvious to see that most of the applicants are coming from city. From the distribution, we can see that the number of applicants from town and village are almost same, which is less than city by approximately 40.

SAS Code



```
/* Bar Chart
   The groups were stacked one above the other */

TITLE 'Number of family members by loan location';
PROC SGPLOT DATA = LIB78601.TESTING_LAS_PRED_78601_DS;
VBAR family_members/group = loan_location groupdisplay = cluster;
LABEL family_members = 'Number of family members';
RUN;
```

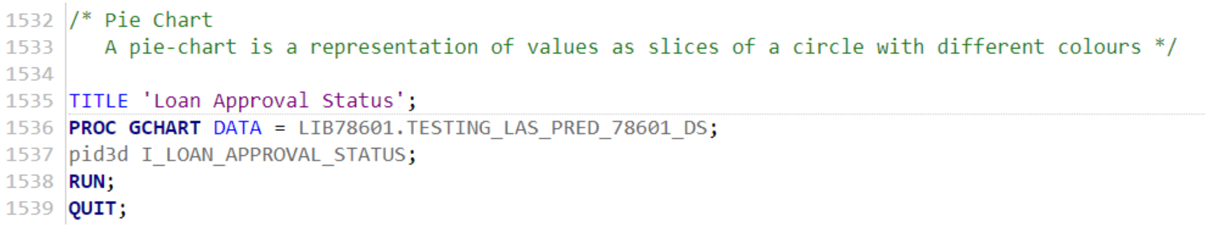
Screenshot(s)



Description

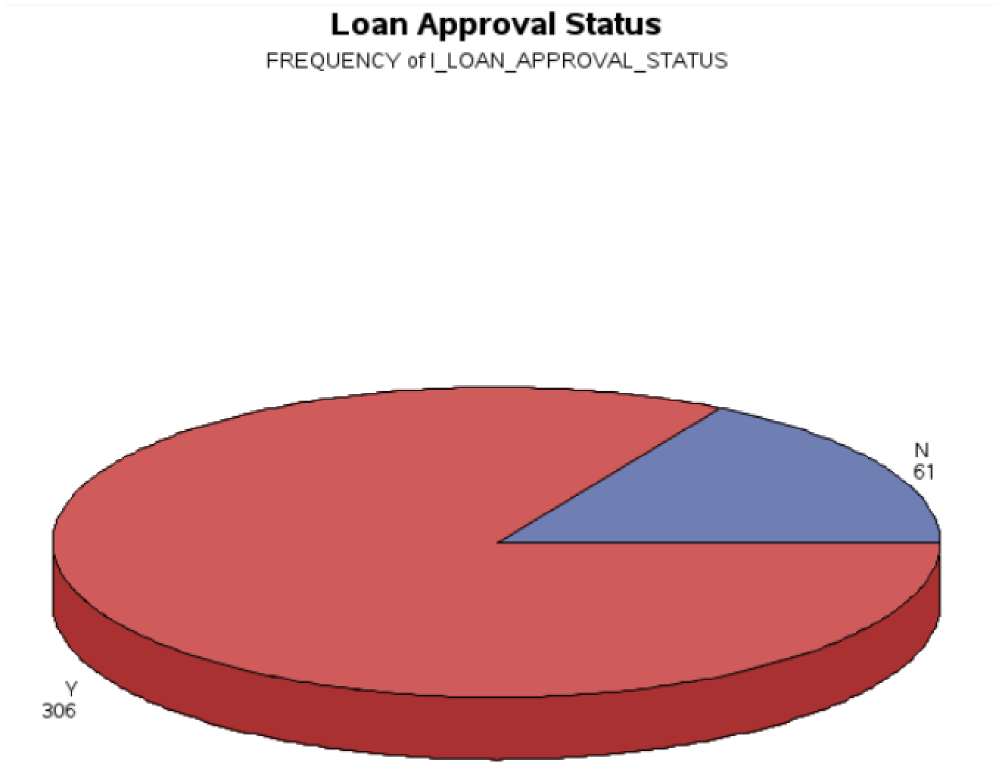
This diagram shows the relationship between loan location and number of family members. From the output, we can observe that majority of the applicants didn’t have any family members no matter where they come from.

SAS Code



```
/* Pie Chart
   A pie-chart is a representation of values as slices of a circle with different colours */

TITLE 'Loan Approval Status';
PROC GCHART DATA = LIB78601.TESTING_LAS_PRED_78601_DS;
pid3d I_LOAN_APPROVAL_STATUS;
RUN;
QUIT;
```

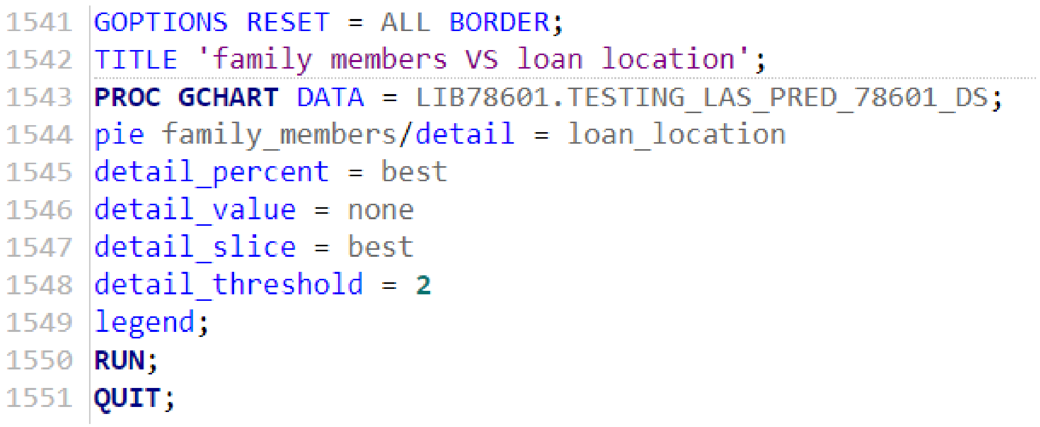
Screenshot(s)



Description

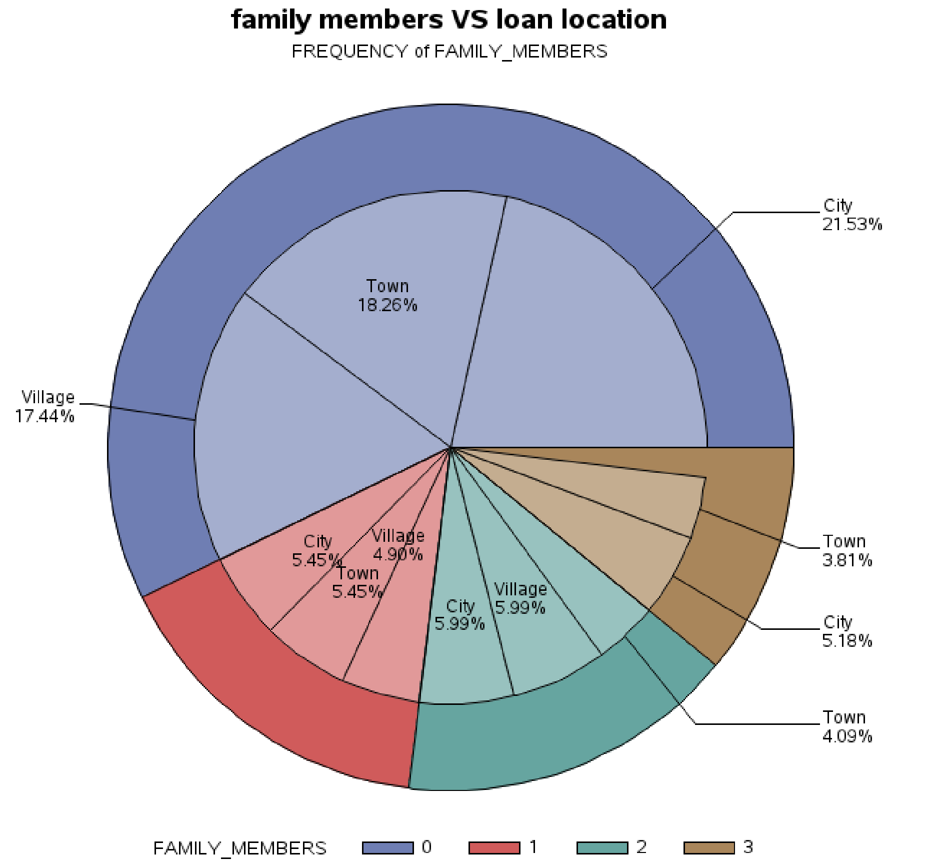
This diagram is a 3-dimensional graph, the pie chart above shows the distribution of loan approval status for the testing set. From the label, we can easily to see that there are 306 applicants get approved for their loan application.

SAS Code



```
GOPTIONS RESET = ALL BORDER;
TITLE 'family members VS loan location';
PROC GCHART DATA = LIB78601.TESTING_LAS_PRED_78601_DS;
pie family_members/detail = loan_location
detail_percent = best
detail_value = none
detail_slice = best
detail_threshold = 2
legend;
RUN;
QUIT;
```

Screenshot(s)



Description

This is a detailed pie chart for showing the relationship between family members and loan location. The colour represent the ratio of family members, and the inner pie chart shows the distribution of loan location for each family members category.