```
/*******************************************************
Developer name: Mr.Mohammad Adnan
Job position: Data Scientist, APU SDN BHD
Program name: mydap_project_TP077702.sas
Description: Loan application status prediction - To perform EDA on loan application status and to assess the application
of logistic regression in predicting loan approval outcomes based on various applicant attributes
Date first written: Wed,27-Mar-2024
Date last updated: Fri,10-June-2024
Folder name:  MY_DAP_FT_MAR_2024_TP077702
Library name:  LIB77702

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

/* SAS Code to display the Data dictionary of LIB77702.TRAINING_DS */

PROC SQL;

DESCRIBE TABLE LIB77702.TRAINING_DS;

RUN;

PROC CONTENTS DATA= LIB77702.TRAINING_DS;

RUN;

TITLE 'Figure no - Univariate Analysis of the categoriical variable: MARITAL_STATUS';

PROC FREQ DATA = LIB77702.TRAINING_DS;

TABLE marital_status;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

VBAR marital_status;

TITLE 'Univariate Analysis of the categorical';

RUN;

TITLE 'Figure no - Univariate Analysis of the categoriical variable: FAMILY_MEMBERS';

PROC FREQ DATA = LIB77702.TRAINING_DS;

TABLE family_members;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

VBAR family_members;

TITLE 'Univariate Analysis of the categorical';

RUN;

TITLE 'Figure no - Univariate Analysis of the categoriical variable: LOAN_LOCATION';

PROC FREQ DATA = LIB77702.TRAINING_DS;

TABLE loan_location;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

VBAR loan_location;

TITLE 'Univariate Analysis of the categorical';
```

```
RUN;

TITLE 'Figure no - Univariate Analysis of the categoriical variable: GENDER';

PROC FREQ DATA = LIB77702.TRAINING_DS;

TABLE gender;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

VBAR gender;

TITLE 'Univariate Analysis of the categorical';

RUN;

TITLE 'Figure no - Univariate Analysis of the categorical variable: Qualification';

PROC FREQ DATA = LIB77702.TRAINING_DS;

TABLE qualification;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

VBAR qualification;

TITLE 'Univariate Analysis of the categorical';

RUN;

TITLE 'Figure no - Univariate Analysis of the continous/numeric variable: LOAN_AMOUNT';

PROC MEANS DATA = LIB77702.TRAINING_DS N NMISS MIN MAX MEAN MEDIAN STD;

VAR loan_amount;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

HISTOGRAM loan_amount;

TITLE 'Univariate Analysis of the continous variable';

RUN;

TITLE 'Figure no - Univariate Analysis of the continous/numeric variable: LOAN_DURATION';

PROC MEANS DATA = LIB77702.TRAINING_DS N NMISS MIN MAX MEAN MEDIAN STD;

VAR loan_duration;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

HISTOGRAM loan_duration;

TITLE 'Univariate Analysis of the continous variable';

RUN;

TITLE 'Figure no - Univariate Analysis of the continous/numeric variable: CANDIDATE_INCOME';
```

```
PROC MEANS DATA = LIB77702.TRAINING_DS N NMISS MIN MAX MEAN MEDIAN STD;

VAR candidate_income;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

HISTOGRAM candidate_income;

TITLE 'Univariate Analysis of the continous variable';

RUN;

TITLE 'Figure no - Univariate Analysis of the continous/numeric variable: GUARANTEE_INCOME';

PROC MEANS DATA = LIB77702.TRAINING_DS N NMISS MIN MAX MEAN MEDIAN STD;

VAR guarantee_income;

RUN;

ODS GRAPHICS / RESET WIDTH = 3.0 IN HEIGHT = 4.0 IN IMAGEMAP;

PROC SGPLOT DATA = LIB77702.TRAINING_DS;

HISTOGRAM guarantee_income;

TITLE 'Univariate Analysis of the continous variable';

RUN;

/* SAS Codes to display the bivariate analysis between variables (CATEGORICAL VS CATEGORICAL) */

TITLE1 'Bivariate analysis of the variables: ';
TITLE2 'Categorical variable[Gender] vs Categorical variable[LOAN_APPROVAL_STATUS]';
FOOTNOTE '--------END--------';

PROC FREQ DATA = LIB77702.TRAINING_DS;

TABLE gender * loan_approval_status/
PLOTS = FREQPLOT( TWOWAY = STACKED SCALE = GROUPPCT );

RUN;

TITLE1 'Bivariate analysis of the variables: ';
TITLE2 'Categorical variable[Gender] vs Categorical variable[MARITAL_STATUS]';
FOOTNOTE '--------END--------';

PROC FREQ DATA = LIB77702.TRAINING_DS;

TABLE gender * marital_status/
PLOTS = FREQPLOT( TWOWAY = STACKED SCALE = GROUPPCT );

RUN;

TITLE1 'Bivariate analysis of the variables: ';
TITLE2 'Categorical variable[Qualification] vs Categorical variable[Employment]';
FOOTNOTE '--------END--------';

PROC FREQ DATA = LIB77702.TRAINING_DS;

TABLE qualification * employment/
PLOTS = FREQPLOT( TWOWAY = STACKED SCALE = GROUPPCT );

RUN;

/* SAS Codes to display the bivariate analysis between variables (CATEGORICAL VS CONTINOUS) */

TITLE1 'Bivariate analysis of the variables: ';
TITLE2 'Categorical variable[GENDER] vs Continuous variable[LOAN_AMOUNT]';
FOOTNOTE '--------END--------';

PROC MEANS DATA = LIB77702.TRAINING_DS;
```

```
CLASS gender; /* It is a categorical variable */
VAR loan_amount; /* It is a numerical variable */

RUN;

TITLE1 'Bivariate analysis of the variables: ';
TITLE2 'Categorical variable[GENDER] vs Continuous variable[CANDIDATE_INCOME]';
FOOTNOTE '--------END--------';

PROC MEANS DATA = LIB77702.TRAINING_DS;

CLASS gender; /* It is a categorical variable */
VAR candidate_income; /* It is a numerical variable */

RUN;

TITLE1 'Bivariate analysis of the variables: ';
TITLE2 'Categorical variable[GENDER] vs Continuous variable[GUARANTEE_INCOME]';
FOOTNOTE '--------END--------';

PROC MEANS DATA = LIB77702.TRAINING_DS;

CLASS gender; /* It is a categorical variable */
VAR guarantee_income; /* It is a numerical variable */

RUN;

TITLE1 'Bivariate analysis of the variables: ';
TITLE2 'Categorical variable[LOAN_APPROVAL_STATUS] vs Continuous variable[LOAN_AMOUNT]';
FOOTNOTE '--------END--------';

PROC MEANS DATA = LIB77702.TRAINING_DS;

CLASS loan_approval_status; /* It is a categorical variable */
VAR loan_amount; /* It is a numerical variable */

RUN;

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

%MACRO UVA_CAT_VAR(ptitle,pdataset,pcat_var);

TITLE &ptitle;

PROC FREQ DATA = &pdataset;

TABLE &pcat_var;

RUN;

%MEND UVA_CAT_VAR;
/* Macro ends here */

/* (TESTING SET) SAS Codes to display the univariate analysis of categorical variables using MACRO */

/* Call the SAS Macro */
%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - MARITAL_STATUS', LIB77702.TESTING_DS, MARITAL_STATUS);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - QUALIFICATION', LIB77702.TESTING_DS, QUALIFICATION);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - LOAN_HISTORY', LIB77702.TESTING_DS, LOAN_HISTORY);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - GENDER', LIB77702.TESTING_DS, GENDER);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - FAMILY_MEMBERS', LIB77702.TESTING_DS, FAMILY_MEMBERS);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - EMPLOYMENT', LIB77702.TESTING_DS, EMPLOYMENT);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - LOAN_LOCATION', LIB77702.TESTING_DS, LOAN_LOCATION);

/* (TRAINING SET) SAS Codes to display the univariate analysis of categorical variables using MACRO */

/* Call the SAS Macro */
%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - MARITAL_STATUS', LIB77702.TRAINING_DS, MARITAL_STATUS);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - QUALIFICATION', LIB77702.TRAINING_DS, QUALIFICATION);
```

```
%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - LOAN_HISTORY', LIB77702.TRAINING_DS, LOAN_HISTORY);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - GENDER', LIB77702.TRAINING_DS, GENDER);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - FAMILY_MEMBERS', LIB77702.TRAINING_DS, FAMILY_MEMBERS);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - EMPLOYMENT', LIB77702.TRAINING_DS, EMPLOYMENT);

%UVA_CAT_VAR('Univariate Analysis of the Categorical Variable - LOAN_LOCATION', LIB77702.TRAINING_DS, LOAN_LOCATION);

/*Continous Variable*/

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

%MACRO UVA_NUM_VAR(ptitle,pdataset,pnum_var);

TITLE &ptitle;

PROC MEANS DATA = &pdataset N NMISS MIN MAX MEAN MEDIAN STD;

VAR &pnum_var;

RUN;

%MEND UVA_NUM_VAR;
/* Macro ends here */

/* (TESTING SET) SAS Codes to display the univariate analysis of numerical variables using MACRO */

/* Call the SAS Macro */
%UVA_NUM_VAR('Univariate Analysis of the Numerical Variable - CANDIDATE_INCOME', LIB77702.TESTING_DS, CANDIDATE_INCOME);
%UVA_NUM_VAR('Univariate Analysis of the Numerical Variable - GUARANTEE_INCOME', LIB77702.TESTING_DS, GUARANTEE_INCOME);
%UVA_NUM_VAR('Univariate Analysis of the Numerical Variable - LOAN_AMOUNT', LIB77702.TESTING_DS, LOAN_AMOUNT);
%UVA_NUM_VAR('Univariate Analysis of the Numerical Variable - LOAN_DURATION', LIB77702.TESTING_DS, LOAN_DURATION);

/* (TRAINING SET) SAS Codes to display the univariate analysis of numerical variables using MACRO */

/* Call the SAS Macro */
%UVA_NUM_VAR('Univariate Analysis of the Numerical Variable - CANDIDATE_INCOME', LIB77702.TRAINING_DS, CANDIDATE_INCOME);
%UVA_NUM_VAR('Univariate Analysis of the Numerical Variable - GUARANTEE_INCOME', LIB77702.TRAINING_DS, GUARANTEE_INCOME);
%UVA_NUM_VAR('Univariate Analysis of the Numerical Variable - LOAN_AMOUNT', LIB77702.TRAINING_DS, LOAN_AMOUNT);
%UVA_NUM_VAR('Univariate Analysis of the Numerical Variable - LOAN_DURATION', LIB77702.TRAINING_DS, LOAN_DURATION);

/* SAS Codes to display the bivariate analysis of categorical variables using MACRO */

/* Macro begins here */
OPTIONS MCOMPILENOTE=ALL;
%MACRO BVA_CAT_CAT(ptitle1,ptitle2,pdataset,pcat_var1,pcat_var2);

TITLE1 &ptitle1;
TITLE2 &ptitle2;

PROC FREQ DATA = &pdataset;

TABLE &pcat_var1 * &pcat_var2/
PLOTS = FREQPLOT( TWOWAY = STACKED SCALE = GROUPPCT );

RUN;
%MEND BVA_CAT_CAT;
/* Macro ends here */

/* Call the SAS Macro */
%BVA_CAT_CAT('Bivariate Analysis of the Variables:',
'MARITAL_STATUS vs EMPLOYMENT',
LIB77702.TESTING_DS,
MARITAL_STATUS,EMPLOYMENT);

/* Call the SAS Macro */
%BVA_CAT_CAT('Bivariate Analysis of the Variables:',
'GENDER vs EMPLOYMENT',
LIB77702.TESTING_DS,
GENDER,EMPLOYMENT);

/* Call the SAS Macro */
%BVA_CAT_CAT('Bivariate Analysis of the Variables:',
'QUALIFICATION vs EMPLOYMENT',
LIB77702.TESTING_DS,
QUALIFICATION,EMPLOYMENT);

RUN;

/* SAS Codes to display the bivariate analysis of variables (CATEGORICAL VS CONTINOUS) using MACRO */

/* Macro begins here */
OPTIONS MCOMPILENOTE=ALL;
%MACRO BVA_CAT_NUM(ptitle1,ptitle2,pdataset,pcat_var,pnum_var);

TITLE1 &ptitle1;
TITLE2 &ptitle2;

PROC MEANS DATA = &pdataset;

CLASS &pcat_var; /* It is a categorical variable */
VAR &pnum_var/; /* It is a continous variable */

RUN;

%MEND BVA_CAT_NUM;
/* Macro ends here */

/* Call the SAS Macro */
%BVA_CAT_NUM('Bivariate Analysis of the variables(Categorical vs Continous)',
'MARITAL_STATUS VS LOAN_AMOUNT',
LIB77702.TESTING_DS,
MARITAL_STATUS,
LOAN_AMOUNT);

/* Call the SAS Macro */
%BVA_CAT_NUM('Bivariate Analysis of the variables(Categorical vs Continous)',
'GENDER VS CANDIDATE_INCOME',
LIB77702.TESTING_DS,
GENDER,
CANDIDATE_INCOME);

/* Call the SAS Macro */
%BVA_CAT_NUM('Bivariate Analysis of the variables(Categorical vs Continous)',
'GENDER VS LOAN_AMOUNT',
LIB77702.TESTING_DS,
GENDER,
LOAN_AMOUNT);

/*-----------TRAINING SET------------------------*/

/*MARITAL_STATUS*/

/* STEP 1: Find the details of the loan applicant who submitted their loan
application without marital status */

TITLE1 'Find the details of the loan applicant who submitted their';
TITLE2 'Loan application without marital status';
PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.marital_Status eq '' OR e.marital_status IS MISSING );

QUIT;

/* STEP 2: Count the number of the loan applicant who submitted their loan
application without marital status */

TITLE1 'Count the no.of loan applicant who submitted thier';
TITLE2 'Loan application without marital status';
PROC SQL;

SELECT COUNT(*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS e
WHERE ( e.marital_Status eq '' OR e.marital_status IS MISSING );

QUIT;

/* STEP 3: Find the statistics for marital status */
```

```
TITLE1 'Find the statistics for marital status';

PROC SQL;

SELECT e.marital_status AS MARITAL_STATUS,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.marital_status NE '' OR e.marital_status IS NOT MISSING )
GROUP BY e.marital_status;

QUIT;

/* STEP 4: Save the statistics for marital status in a dataset */

TITLE1 'Save the statistics for marital status in a dataset';

PROC SQL;

CREATE TABLE LIB77702.TRAINING_STATS_DS AS
SELECT e.marital_status AS MARITAL_STATUS,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.marital_status NE '' OR e.marital_status IS NOT MISSING )
GROUP BY e.marital_status;

QUIT;

/* STEP 4.1: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TRAINING_BACKUP_DS AS
SELECT *
FROM LIB77702.TRAINING_DS;

QUIT;

/* STEP 5: Impute the missing values in the categorical variable (MARITAL_STATUS) */

PROC SQL;

UPDATE LIB77702.TRAINING_DS
SET marital_status = ( SELECT marital_status AS marital_status
                        FROM LIB77702.TRAINING_STATS_DS t2
                        WHERE t2.counts eq ( SELECT MAX(t1.counts) AS HIGHEST_COUNT
                                             FROM LIB77702.TRAINING_STATS_DS t1 ) )
                                             /* Above is sub-program to find the highest count */
WHERE ( marital_status eq '' OR marital_status IS MISSING );

QUIT;

/* STEP 6: (AFTER IMPUTATION) Find missing values for marital status */

TITLE1 '(AFTER IMPUTATION) Find missing values for marital status';
TITLE2 'Loan application without marital status';
FOOTNOTE '------End------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.marital_Status eq '' OR e.marital_status IS MISSING );

QUIT;

/*FAMILY_MEMBERS*/

/* STEP 1: List the details of the loan applicants who submitted the applications
without family member details */
PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.family_members eq '' OR e.family_members IS NULL );

QUIT;
```

```
/* STEP 2: Count the number of the loan applicants who submitted the applications
without family member details */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS e
WHERE ( e.family_members eq '' OR e.family_members IS NULL );

QUIT;

/* STEP 3: List the details of the loan applicants with '3+' family members who submitted the applications
without family member details */
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 LIB77702.TRAINING_DS e
WHERE ( e.family_members ne '' OR e.family_members IS NOT MISSING );

QUIT;

/* STEP 4: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TRAINING_BACKUP_DS AS
SELECT *
FROM LIB77702.TRAINING_DS;

QUIT;

/* STEP 5: Remove the '+' found in the family_members variable */
PROC SQL;

UPDATE LIB77702.TRAINING_DS
SET family_members = SUBSTR(family_members,1,1)
WHERE SUBSTR(family_members,2,1) eq '+';

QUIT;

/* STEP 6: Display the statistics in a dataset */

PROC SQL;

SELECT e.family_members AS family_members,
COUNT (*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.family_members ne '' or e.family_members IS NOT MISSING )
GROUP BY e.family_members;

QUIT;

/* STEP 7: Save the statistics in a dataset */

PROC SQL;

CREATE TABLE LIB77702.TRAINING_STAT_FM AS
SELECT e.family_members AS family_members,
COUNT (*) AS counts
FROM LIB77702.TRAINING_DS e
WHERE ( e.family_members ne '' or e.family_members IS NOT MISSING )
GROUP BY e.family_members;

QUIT;

/* STEP 8: Impute the missing values found in the Categorical Variable - FAMILY_MEMBERS */

PROC SQL;

UPDATE LIB77702.TRAINING_DS
SET family_members = (  SELECT to.family_members AS family_members
                        FROM LIB77702.TRAINING_STAT_FM to
                        WHERE to.counts eq ( SELECT MAX (ti.counts) AS highest_count
                                            FROM LIB77702.TRAINING_STAT_FM ti ) )
                                            /* Above is a sub-program to find the highest count */
```

```
WHERE ( family_members eq '' OR family_members IS NULL );

QUIT;

/* STEP 9: (After Imputation) List the details of the loan applicants who submitted the
applications without family member details  */
PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.family_members eq '' OR e.family_members IS NULL );

QUIT;

/* STEP 10: (After Imputation) Count the number of the loan applicants who submitted the
applications without family member details */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS e
WHERE ( e.family_members eq '' OR e.family_members IS NULL );

QUIT;


/*EMPLOYMENT*/

/* STEP 1: Find missing values for employment */

TITLE1 'Find missing values for employment';
TITLE2 'Loan application without employment';
PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.employment eq '' OR e.employment IS MISSING );

QUIT;

/* STEP 2: Count the number of missing values for employment */

TITLE1 'Count missing values for employment';
TITLE2 'Loan application without employment';
PROC SQL;

SELECT COUNT(*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS e
WHERE ( e.employment eq '' OR e.employment IS MISSING );

QUIT;

/* STEP 3: Find the statistics for employment */

TITLE1 'Find the statistics for employment';

PROC SQL;

SELECT e.employment AS EMPLOYMENT,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.employment NE '' OR e.employment IS NOT MISSING )
GROUP BY e.employment;

QUIT;

/* STEP 4: Save the statistics for employment in a dataset */

TITLE1 'Save the statistics for employment in a dataset';

PROC SQL;

CREATE TABLE LIB77702.TRAINING_STATS_DS AS
SELECT e.employment AS EMPLOYMENT,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.employment NE '' OR e.employment IS NOT MISSING )
GROUP BY e.employment;
```

```
QUIT;

/* STEP 4.1: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TRAINING_BACKUP_DS AS
SELECT *
FROM LIB77702.TRAINING_DS;

QUIT;

/* STEP 5: Impute the missing values in the categorical variable (EMPLOYMENT) */

PROC SQL;

UPDATE LIB77702.TRAINING_DS
SET employment = ( SELECT employment AS employment
                        FROM LIB77702.TRAINING_STATS_DS t2
                        WHERE t2.counts eq ( SELECT MAX(t1.counts) AS HIGHEST_COUNT
                                              FROM LIB77702.TRAINING_STATS_DS t1 ) )
                                              /* Above is sub-program to find the highest count */
WHERE ( employment eq '' OR employment IS MISSING );

QUIT;

/* STEP 6: (AFTER IMPUTATION) Find missing values for employment */

TITLE1 '(AFTER IMPUTATION) Find missing values for employment';
TITLE2 'Loan application without employment';
FOOTNOTE '------End------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.employment eq '' OR e.employment IS MISSING );

QUIT;

/*GENDER*/

/* STEP 1: Find missing values for gender */

TITLE1 'Find missing values for gender';
TITLE2 'Loan application without gender';
PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.gender eq '' OR e.gender IS MISSING );

QUIT;

/* STEP 2: Count the number of missing values for gender */

TITLE1 'Count missing values for gender';
TITLE2 'Loan application without gender';
PROC SQL;

SELECT COUNT(*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS e
WHERE ( e.gender eq '' OR e.gender IS MISSING );

QUIT;

/* STEP 3: Find the statistics for gender */

TITLE1 'Find the statistics for gender';

PROC SQL;

SELECT e.gender AS GENDER,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.gender NE '' OR e.gender IS NOT MISSING )
```

```
GROUP BY e.gender;

QUIT;

/* STEP 4: Save the statistics for employment in a dataset */

TITLE1 'Save the statistics for employment in a dataset';

PROC SQL;

CREATE TABLE LIB77702.TRAINING_STATS_DS AS
SELECT e.gender AS GENDER,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.gender NE '' OR e.gender IS NOT MISSING )
GROUP BY e.gender;

QUIT;

/* STEP 4.1: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TRAINING_BACKUP_DS AS
SELECT *
FROM LIB77702.TRAINING_DS;

QUIT;

/* STEP 5: Impute the missing values in the categorical variable (GENDER) */

PROC SQL;

UPDATE LIB77702.TRAINING_DS
SET gender = ( SELECT gender AS gender
                        FROM LIB77702.TRAINING_STATS_DS t2
                        WHERE t2.counts eq ( SELECT MAX(t1.counts) AS HIGHEST_COUNT
                                              FROM LIB77702.TRAINING_STATS_DS t1 ) )
                                              /* Above is sub-program to find the highest count */
WHERE ( gender eq '' OR gender IS MISSING );

QUIT;

/* STEP 6: (AFTER IMPUTATION) Find missing values for gender */

TITLE1 '(AFTER IMPUTATION) Find missing values for gender';
TITLE2 'Loan application without gender';
FOOTNOTE '------End------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.gender eq '' OR e.gender IS MISSING );

QUIT;

/*LOAN_AMOUNT*/

/* STEP 1: List the details of the loan applicants who submitted the applications without loan_amount */
TITLE 'STEP 1: List the details of the loan applicants who submitted the applications without loan_amount';
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/* STEP 2: Count the number of the loan applicants who submitted the applications without loan_amount */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );
```

```
QUIT;

/* Create a Back-Up */
PROC SQL;

CREATE TABLE LIB77702.TRAINING_BACKUP_DS AS
SELECT *
FROM LIB77702.TRAINING_DS;

QUIT;

/* STEP 3: Impute the missing values found in the Continous varibale - LOAN_AMOUNT */

PROC STDIZE DATA = LIB77702.TRAINING_DS REPONLY

METHOD = MEAN OUT = LIB77702.TRAINING_DS;
var loan_amount;

QUIT;

/* STEP 4: (After Imputation) List the details of the loan applicants who submitted the applications without loan_amount */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/* STEP 5: (After Imputation) Count the number of the loan applicants who submitted the applications without loan_amount */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/*Loan History*/

/* STEP 1:List the details of the loan applicants who submitted the applications without loan_history */
PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.loan_history eq . OR e.loan_history IS MISSING );

QUIT;

/* STEP 2:Count the number of the loan applicants who submitted the applications without loan_history */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS e
WHERE ( e.loan_history eq . OR e.loan_history IS MISSING );

QUIT;

/* STEP 3: Find the statistics for loan_history */

TITLE1 'Find the statistics for loan_history';

PROC SQL;

SELECT e.loan_history AS LOAN_HISTORY,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.loan_history NE . OR e.loan_history IS NOT MISSING )
GROUP BY e.loan_history;

QUIT;
```

```
/* STEP 4: Save the statistics for loan_history in a dataset */

PROC SQL;

CREATE TABLE LIB77702.TRAINING_STATS_DS AS
SELECT e.loan_history AS LOAN_HISTORY,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.loan_history NE . OR e.loan_history IS NOT MISSING )
GROUP BY e.loan_history;

QUIT;

/* STEP 4.1: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TRAINING_BACKUP_DS AS
SELECT *
FROM LIB77702.TRAINING_DS;

QUIT;

/* STEP 5: Impute the missing values in the categorical variable (LOAN_HISTORY) */

PROC SQL;

UPDATE LIB77702.TRAINING_DS
SET loan_history = ( SELECT loan_history AS loan_history
                        FROM LIB77702.TRAINING_STATS_DS t2
                        WHERE t2.counts eq ( SELECT MAX(t1.counts) AS HIGHEST_COUNT
                                              FROM LIB77702.TRAINING_STATS_DS t1 ) )
                                              /* Above is sub-program to find the highest count */
WHERE ( loan_history eq . OR loan_history IS MISSING );

QUIT;

/* STEP 6: (AFTER IMPUTATION) Find missing values for loan_history */

TITLE1 '(AFTER IMPUTATION) Find missing values for loan_history';
TITLE2 'Loan application without loan_history';
FOOTNOTE '------End------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS e
WHERE ( e.loan_history eq . OR e.loan_history IS MISSING );

QUIT;

/*LOAN_AMOUNT*/

/* STEP 1: List the details of the loan applicants who submitted the applications without loan_amount */
TITLE 'STEP 1: List the details of the loan applicants who submitted the applications without loan_amount';
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/* STEP 2: Count the number of the loan applicants who submitted the applications without loan_amount */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/* Create a Back-Up */
PROC SQL;
```

```
CREATE TABLE LIB77702.TRAINING_BACKUP_DS AS
SELECT *
FROM LIB77702.TRAINING_DS;

QUIT;

/* STEP 3: Impute the missing values found in the Continous varibale - LOAN_AMOUNT */

PROC STDIZE DATA = LIB77702.TRAINING_DS REPONLY

METHOD = MEAN OUT = LIB77702.TRAINING_DS;
var loan_amount;

QUIT;

/* STEP 4: (After Imputation) List the details of the loan applicants who submitted the applications without loan_amount */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/* STEP 5: (After Imputation) Count the number of the loan applicants who submitted the applications without loan_amount */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/*LOAN_DURATION*/

/* STEP 1: List the details of the loan applicants who submitted the applications without loan_duration */
TITLE 'STEP 1: List the details of the loan applicants who submitted the applications without loan_duration';
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_duration eq . OR t.loan_duration IS MISSING );

QUIT;

/* STEP 2: Count the number of the loan applicants who submitted the applications without loan_duration */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_duration eq . OR t.loan_duration IS MISSING );

QUIT;

/* Create a Back-Up */
PROC SQL;

CREATE TABLE LIB77702.TRAINING_BACKUP_DS AS
SELECT *
FROM LIB77702.TRAINING_DS;

QUIT;

/* STEP 3: Impute the missing values found in the Continous varibale - loan_duration */

PROC STDIZE DATA = LIB77702.TRAINING_DS REPONLY

METHOD = MEAN OUT = LIB77702.TRAINING_DS;
var loan_duration;

QUIT;
```

```
/* STEP 4: (After Imputation) List the details of the loan applicants who submitted the applications without loan_duration */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_duration eq . OR t.loan_duration IS MISSING );

QUIT;

/* STEP 4.1: (After Imputation) Count the number of the loan applicants who submitted the applications without loan_duration
FOOTNOTE '----------END----------';

PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TRAINING_DS t
WHERE ( t.loan_duration eq . OR t.loan_duration IS MISSING );

QUIT;

/*--------------TESTING---------------*/

TITLE 'Before imputing the missing values, find the categorical and continous variable with missing values';
PROC FORMAT;

VALUE $missfmt '' = 'Missing' others = 'Not Missing';
VALUE $missfmt . = 'Missing' others = 'Not Missing';

RUN;

PROC FREQ DATA=LIB77702.TESTING_DS;

FORMAT _CHAR_ $missfmt.;
FORMAT _NUMERIC_ missfmt.;

TABLE _CHAR_ / missing nocum nopercent;
TABLE _NUMERIC_ / missing nocum nopercent;

RUN;

/*CATEGORICAL*/

/*MARITAL_STATUS*/

/* STEP 1: Find the details of the loan applicant who submitted their loan
application without marital status */

TITLE1 'Find the details of the loan applicant who submitted their';
TITLE2 'Loan application without marital status';
PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.marital_Status eq '' OR e.marital_status IS MISSING );

QUIT;

/* STEP 2: Count the number of the loan applicant who submitted their loan
application without marital status */

TITLE1 'Count the no.of loan applicant who submitted thier';
TITLE2 'Loan application without marital status';
PROC SQL;

SELECT COUNT(*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS e
WHERE ( e.marital_Status eq '' OR e.marital_status IS MISSING );

QUIT;

/* STEP 3: Find the statistics for marital status */

TITLE1 'Find the statistics for marital status';
```

```
PROC SQL;

SELECT e.marital_status AS MARITAL_STATUS,
        COUNT(*) AS COUNTS
FROM LIB77702.TESTING_DS e
WHERE ( e.marital_status NE '' OR e.marital_status IS NOT MISSING )
GROUP BY e.marital_status;

QUIT;

/* STEP 4: Save the statistics for marital status in a dataset */

TITLE1 'Save the statistics for marital status in a dataset';

PROC SQL;

CREATE TABLE LIB77702.TESTING_STATS_DS AS
SELECT e.marital_status AS MARITAL_STATUS,
        COUNT(*) AS COUNTS
FROM LIB77702.TRAINING_DS e
WHERE ( e.marital_status NE '' OR e.marital_status IS NOT MISSING )
GROUP BY e.marital_status;

QUIT;

/* STEP 4.1: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TESTING_BACKUP_DS AS
SELECT *
FROM LIB77702.TESTING_DS;

QUIT;

/* STEP 5: Impute the missing values in the categorical variable (MARITAL_STATUS) */

PROC SQL;

UPDATE LIB77702.TESTING_DS
SET marital_status = ( SELECT marital_status AS marital_status
                        FROM LIB77702.TESTING_STATS_DS t2
                        WHERE t2.counts eq ( SELECT MAX(t1.counts) AS HIGHEST_COUNT
                                              FROM LIB77702.TESTING_STATS_DS t1 ) )
                                              /* Above is sub-program to find the highest count */
WHERE ( marital_status eq '' OR marital_status IS MISSING );

QUIT;

/* STEP 6: (AFTER IMPUTATION) Find missing values for marital status */

TITLE1 '(AFTER IMPUTATION) Find missing values for marital status';
TITLE2 'Loan application without marital status';
FOOTNOTE '------End------';

PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.marital_Status eq '' OR e.marital_status IS MISSING );

QUIT;

/*EMPLOYMENT*/

/* STEP 1: Find missing values for employment */

TITLE1 'Find missing values for employment';
TITLE2 'Loan application without employment';
PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.employment eq '' OR e.employment IS MISSING );

QUIT;
```

```
/* STEP 2: Count the number of missing values for employment */

TITLE1 'Count missing values for employment';
TITLE2 'Loan application without employment';
PROC SQL;

SELECT COUNT(*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS e
WHERE ( e.employment eq '' OR e.employment IS MISSING );

QUIT;

/* STEP 3: Find the statistics for employment */

TITLE1 'Find the statistics for employment';

PROC SQL;

SELECT e.employment AS EMPLOYMENT,
        COUNT(*) AS COUNTS
FROM LIB77702.TESTING_DS e
WHERE ( e.employment NE '' OR e.employment IS NOT MISSING )
GROUP BY e.employment;

QUIT;

/* STEP 4: Save the statistics for employment in a dataset */

TITLE1 'Save the statistics for employment in a dataset';

PROC SQL;

CREATE TABLE LIB77702.TESTING_STATS_DS AS
SELECT e.employment AS EMPLOYMENT,
        COUNT(*) AS COUNTS
FROM LIB77702.TESTING_DS e
WHERE ( e.employment NE '' OR e.employment IS NOT MISSING )
GROUP BY e.employment;

QUIT;

/* STEP 4.1: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TESTING_BACKUP_DS AS
SELECT *
FROM LIB77702.TESTING_DS;

QUIT;

/* STEP 5: Impute the missing values in the categorical variable (EMPLOYMENT) */

PROC SQL;

UPDATE LIB77702.TESTING_DS
SET employment = ( SELECT employment AS employment
                        FROM LIB77702.TESTING_STATS_DS t2
                        WHERE t2.counts eq ( SELECT MAX(t1.counts) AS HIGHEST_COUNT
                                              FROM LIB77702.TESTING_STATS_DS t1 ) )
                                              /* Above is sub-program to find the highest count */
WHERE ( employment eq '' OR employment IS MISSING );

QUIT;

/* STEP 6: (AFTER IMPUTATION) Find missing values for employment */

TITLE1 '(AFTER IMPUTATION) Find missing values for employment';
TITLE2 'Loan application without employment';
FOOTNOTE '------End------';

PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.employment eq '' OR e.employment IS MISSING );
```

```
QUIT;

/*Gender*/

/* STEP 1: Find missing values for gender */

TITLE1 'Find missing values for gender';
TITLE2 'Loan application without gender';
PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.gender eq '' OR e.gender IS MISSING );

QUIT;

/* STEP 2: Count the number of missing values for gender */

TITLE1 'Count missing values for gender';
TITLE2 'Loan application without gender';
PROC SQL;

SELECT COUNT(*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS e
WHERE ( e.gender eq '' OR e.gender IS MISSING );

QUIT;

/* STEP 3: Find the statistics for gender */

TITLE1 'Find the statistics for gender';

PROC SQL;

SELECT e.gender AS GENDER,
        COUNT(*) AS COUNTS
FROM LIB77702.TESTING_DS e
WHERE ( e.gender NE '' OR e.gender IS NOT MISSING )
GROUP BY e.gender;

QUIT;

/* STEP 4: Save the statistics for employment in a dataset */

TITLE1 'Save the statistics for employment in a dataset';

PROC SQL;

CREATE TABLE LIB77702.TESTING_STATS_DS AS
SELECT e.gender AS GENDER,
        COUNT(*) AS COUNTS
FROM LIB77702.TESTING_DS e
WHERE ( e.gender NE '' OR e.gender IS NOT MISSING )
GROUP BY e.gender;

QUIT;

/* STEP 4.1: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TESTING_BACKUP_DS AS
SELECT *
FROM LIB77702.TESTING_DS;

QUIT;

/* STEP 5: Impute the missing values in the categorical variable (GENDER) */

PROC SQL;

UPDATE LIB77702.TESTING_DS
SET gender = ( SELECT gender AS gender
                        FROM LIB77702.TESTING_STATS_DS t2
                        WHERE t2.counts eq ( SELECT MAX(t1.counts) AS HIGHEST_COUNT
                                              FROM LIB77702.TESTING_STATS_DS t1 ) )
                                              /* Above is sub-program to find the highest count */
```

```
WHERE ( gender eq '' OR gender IS MISSING );

QUIT;

/* STEP 6: (AFTER IMPUTATION) Find missing values for gender */

TITLE1 '(AFTER IMPUTATION) Find missing values for gender';
TITLE2 'Loan application without gender';
FOOTNOTE '------End------';

PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.gender eq '' OR e.gender IS MISSING );

QUIT;

/* STEP 1:List the details of the loan applicants who submitted the applications without loan_history */
PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.loan_history eq . OR e.loan_history IS MISSING );

QUIT;

/* STEP 2:Count the number of the loan applicants who submitted the applications without loan_history */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS e
WHERE ( e.loan_history eq . OR e.loan_history IS MISSING );

QUIT;

/* STEP 3: Find the statistics for loan_history */

TITLE1 'Find the statistics for loan_history';

PROC SQL;

SELECT e.loan_history AS LOAN_HISTORY,
        COUNT(*) AS COUNTS
FROM LIB77702.TESTING_DS e
WHERE ( e.loan_history NE . OR e.loan_history IS NOT MISSING )
GROUP BY e.loan_history;

QUIT;

/* STEP 4: Save the statistics for loan_history in a dataset */

PROC SQL;

CREATE TABLE LIB77702.TESTING_STATS_DS AS
SELECT e.loan_history AS LOAN_HISTORY,
        COUNT(*) AS COUNTS
FROM LIB77702.TESTING_DS e
WHERE ( e.loan_history NE . OR e.loan_history IS NOT MISSING )
GROUP BY e.loan_history;

QUIT;

/* STEP 4.1: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TESTING_BACKUP_DS AS
SELECT *
FROM LIB77702.TESTING_DS;

QUIT;

/* STEP 5: Impute the missing values in the categorical variable (LOAN_HISTORY) */

PROC SQL;
```

```
UPDATE LIB77702.TESTING_DS
SET loan_history = ( SELECT loan_history AS loan_history
                        FROM LIB77702.TESTING_STATS_DS t2
                        WHERE t2.counts eq ( SELECT MAX(t1.counts) AS HIGHEST_COUNT
                                              FROM LIB77702.TESTING_STATS_DS t1 ) )
                                              /* Above is sub-program to find the highest count */
WHERE ( loan_history eq . OR loan_history IS MISSING );

QUIT;

/* STEP 6: (AFTER IMPUTATION) Find missing values for loan_history */

TITLE1 '(AFTER IMPUTATION) Find missing values for loan_history';
TITLE2 'Loan application without loan_history';
FOOTNOTE '------End------';

PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.loan_history eq . OR e.loan_history IS MISSING );

QUIT;

/*FAMILY MEMBER*/

/* STEP 1: List the details of the loan applicants who submitted the applications without family member details */
PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.family_members eq '' OR e.family_members IS NULL );

QUIT;

/* STEP 2: Count the number of the loan applicants who submitted the applications without family member details */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS e
WHERE ( e.family_members eq '' OR e.family_members IS NULL );

QUIT;

/* STEP 3: List the details of the loan applicants with '3+' family members who submitted
the applications without family member details */
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 LIB77702.TESTING_DS e
WHERE ( e.family_members ne '' OR e.family_members IS NOT MISSING );

QUIT;

/* STEP 4: Make a backup of the dataset */

PROC SQL;

CREATE TABLE LIB77702.TESTING_BACKUP_DS AS
SELECT *
FROM LIB77702.TESTING_DS;

QUIT;

/* STEP 5: Remove the '+' found in the family_members variable */
PROC SQL;

UPDATE LIB77702.TESTING_DS
SET family_members = SUBSTR(family_members,1,1)
WHERE SUBSTR(family_members,2,1) eq '+';

QUIT;

/* STEP 6: Display the statistics in a dataset */
```

```
PROC SQL;

SELECT e.family_members AS family_members,
COUNT (*) AS COUNTS
FROM LIB77702.TESTING_DS e
WHERE ( e.family_members ne '' or e.family_members IS NOT MISSING )
GROUP BY e.family_members;

QUIT;

/* STEP 7: Save the statistics in a dataset */

PROC SQL;

CREATE TABLE LIB77702.TESTING_STAT_FM AS
SELECT e.family_members AS family_members,
COUNT (*) AS counts
FROM LIB77702.TESTING_DS e
WHERE ( e.family_members ne '' or e.family_members IS NOT MISSING )
GROUP BY e.family_members;

QUIT;

/* STEP 8: Impute the missing values found in the Categorical Variable - FAMILY_MEMBERS */

PROC SQL;

UPDATE LIB77702.TESTING_DS
SET family_members = (  SELECT to.family_members AS family_members
                        FROM LIB77702.TESTING_STAT_FM to
                        WHERE to.counts eq ( SELECT MAX (ti.counts) AS highest_count
                                             FROM LIB77702.TESTING_STAT_FM ti ) )
                                             /* Above is a sub-program to find the highest count */
WHERE ( family_members eq '' OR family_members IS NULL );

QUIT;

/* STEP 9: (After Imputation) List the details of the loan applicants who submitted the applications without family member de
PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS e
WHERE ( e.family_members eq '' OR e.family_members IS NULL );

QUIT;

/* STEP 9.1: (After Imputation) Count the number of the loan applicants who submitted
the applications without family member details */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS e
WHERE ( e.family_members eq '' OR e.family_members IS NULL );

QUIT;

/*LOAN AMOUNT*/

/* STEP 1: List the details of the loan applicants who submitted the applications without loan_amount */
TITLE 'STEP 1: List the details of the loan applicants who submitted the applications without loan_amount';
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/* STEP 2: Count the number of the loan applicants who submitted the applications without loan_amount */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );
```

```
QUIT;

/* Create a Back-Up */
PROC SQL;

CREATE TABLE LIB77702.TESTING_BACKUP_DS AS
SELECT *
FROM LIB77702.TESTING_DS;

QUIT;

/* STEP 3: Impute the missing values found in the Continous varibale - LOAN_AMOUNT */

PROC STDIZE DATA = LIB77702.TESTING_DS REPONLY

METHOD = MEAN OUT = LIB77702.TESTING_DS;
var loan_amount;

QUIT;

/* STEP 4: (After Imputation) List the details of the loan applicants who submitted the applications without loan_amount */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/* STEP 5: (After Imputation) Count the number of the loan applicants who submitted the applications without loan_amount */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS t
WHERE ( t.loan_amount eq . OR t.loan_amount IS MISSING );

QUIT;

/* STEP 1: List the details of the loan applicants who submitted the applications without loan_duration */
TITLE 'STEP 1: List the details of the loan applicants who submitted the applications without loan_duration';
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS t
WHERE ( t.loan_duration eq . OR t.loan_duration IS MISSING );

QUIT;

/* STEP 2: Count the number of the loan applicants who submitted the applications without loan_duration */
PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS t
WHERE ( t.loan_duration eq . OR t.loan_duration IS MISSING );

QUIT;

/* Create a Back-Up */
PROC SQL;

CREATE TABLE LIB77702.TESTING_BACKUP_DS AS
SELECT *
FROM LIB77702.TESTING_DS;

QUIT;

/* STEP 3: Impute the missing values found in the Continous varibale - loan_duration */

PROC STDIZE DATA = LIB77702.TESTING_DS REPONLY

METHOD = MEAN OUT = LIB77702.TESTING_DS;
```

```
var loan_duration;

QUIT;

/* STEP 4: (After Imputation) List the details of the loan applicants who submitted the applications without loan_duration */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT *
FROM LIB77702.TESTING_DS t
WHERE ( t.loan_duration eq . OR t.loan_duration IS MISSING );

QUIT;

/* STEP 5: (After Imputation) Count the number of the loan applicants who submitted the applications without loan_duration */
FOOTNOTE '----------END----------';

PROC SQL;

SELECT COUNT (*) Label = 'Number of Loan Applicants'
FROM LIB77702.TESTING_DS t
WHERE ( t.loan_duration eq . OR t.loan_duration IS MISSING );

QUIT;

/*---------LOGISTIC REGRESSION-------------------*/

/* Creation of model using logistic regression algorithm */

PROC LOGISTIC DATA=LIB77702.TRAINING_DS OUTMODEL = LIB77702.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
    LOAN_HISTORY
    LOAN_LOCATION
    CANDIDATE_INCOME
    GUARANTEE_INCOME
    LOAN_AMOUNT
    LOAN_HISTORY
    LOAN_DURATION
    ;

OUTPUT OUT = LIB77702.TRAINING_OUT_DS P = PRED_PROB;
/*PRED_PROB -> Predicted probability - variable to hold predicted probabilty
OUT -> the output will be sotred in the dataset
Akaike Information Criteria ( AIC ) < SC ( Schwarz Criterion ) */
/*If Pr > ChiSq is <= 0.05, it means that independent variable is an important variable and it is
truly contributing to predict the dependent variable */
RUN;

/**********************************************************************
Predict the loan approval status using the model created

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

PROC LOGISTIC INMODEL = LIB77702.TRAINING_DS_LR_MODEL; /* The model that was created */

SCORE DATA = LIB77702.TESTING_DS /*Test ds*/
OUT = LIB77702.TESTING_LAS_PREDICTED_77702_DS; /*Location of output */

QUIT;

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

```
/* Display the details of the predicted loan approval status */

PROC SQL;

SELECT *
FROM LIB77702.TESTING_LAS_PREDICTED_77702_DS;

QUIT;

/* Generate report using SAS ODS -Output Delivery/Display System */

ODS HTML CLOSE;
ODS PDF CLOSE;
/* Determine where the PDF is stored */
ODS PDF FILE = "/home/u63691876/DAP_FT_MAR_2024_TP077702/LAS_REPORT_077702.pdf";
OPTIONS NODATE;
TITLE1 'Predicted Bank Loan Approval Status';
TITLE2 'ASIA PACIFIC UNIVERSITY';
PROC REPORT DATA = LIB77702.TESTING_LAS_PREDICTED_77702_DS NOWINDOWS;
BY SME_LOAN_ID_NO;
DEFINE SME_LOAN_ID_NO / GROUP 'LOAN ID';
DEFINE GENDER / GROUP 'GENDER';
DEFINE MARITAL_STATUS / GROUP 'MARITAL STATUS';
DEFINE FAMILY_MEMBERS / GROUP 'FAMILY MEMBERS';
DEFINE CANDIDATE_INCOME / GROUP 'MONTHLY INCOME';
DEFINE GUARANTEE_INCOME / GROUP 'CO-APPLICANT INCOME';
DEFINE LOAN_AMOUNT / GROUP 'LOAN AMOUNT';
DEFINE LOAN_DURATION / GROUP 'LOAN DURATION';
DEFINE LOAN_LOCATION / GROUP 'LOAN LOCATION';
DEFINE LOAN_HISTORY / GROUP 'LOAN HISTORY';
FOOTNOTE '----------END----------';
RUN;


/* Data Visualization using SAS codes
Graphical representation of information and data */

/* SAS Simple Bar Chart */
PROC SGPLOT DATA= LIB77702.TESTING_LAS_PREDICTED_TP77702_DS;
VBAR loan_location;
TITLE 'Loan Applicants by Loan Location';
RUN;

/*Stacked Bar Chart
The groups were stacked one above the other*/
TITLE 'Number of family members by loan location';
PROC SGPLOT data= LIB77702.TESTING_LAS_PREDICTED_TP77702_DS;
vbar family_members /group = loan_location groupdisplay = cluster;
Label family_members = 'Number of family members';
RUN;

/*Pie Chart*/
TITLE 'Loan approval status';
PROC GCHART data= LIB77702.TESTING_LAS_PREDICTED_TP77702_DS;
pie3d I_LOAN_APPROVAL_STATUS;
RUN;
QUIT;

/*Pie Chart*/
TITLE 'Loan approval status';
PROC GCHART data= LIB77702.TESTING_LAS_PREDICTED_TP77702_DS;
pie3d I_LOAN_APPROVAL_STATUS;
RUN;
QUIT;

GOPTIONS RESET=ALL BORDER;
TITLE 'family members vs loan location';
PROC GCHART DATA=LIB77702.TESTING_LAS_PREDICTED_TP77702_DS;
pie family_members / detail=loan_location
```