A picture containing logo

Description automatically generated

## ANL252

|  |  |
| --- | --- |
| **STUDENT NAME:** | KYNAN TAN |
| **P.I. NO.:** | Z2181984 |
| **TITLE:** | ECA |
| **COURSE CODE:** | ANL252 |
| **T:** | 09 |
| **SUBMISSION DATE:** | 5 September 2022 |

**Plagiarism and Collusion**

**Plagiarism** is the act of using or passing as one’s own, the ideas or writings of another without acknowledging or crediting the source from which the ideas are taken from.

**Collusion** is the act of submitting any academic work (including assignment, project or report) that was completed by another person and pass these work off as one’s own.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Declaration** | | | | |
|  | | | | |
| I declare that this assignment is my own work, unless otherwise acknowledge or credited by appropriate referencing. I have read and abide by the SUSS Honour Code and I am aware of the penalties associated with plagiarism and collusion listed in the Student Handbook. | | | | |
|  | | | | |
| Initial: | K |  | Date: | 5 Sept 2022 |
|  | | | | |

Question 1

The categorical variables in the dataset are GENDER, EDUCATION, MARITAL, RATING. The numeric variables in the dataset are LIMIT, BALANCE, INCOME, AGE, S(n), B(n), R(n).

Question 2

The results obtained from the four pre-processing tasks would be the (1) removal of null values within the dataset, (2) converting R3 column from object to integer and removing both dollar and comma signs, (3) removing duplicated rows from the dataframe, and (4) removing illogical outliers from AGE column. The results obtained will be explained in detail through the comments.

# import pandas

import pandas as pd

# import numpy

import numpy as np

# import matplotlib

import matplotlib.pyplot as plt

# import warnings

import warnings

warnings.filterwarnings('ignore')

### First data preprocessing task: Detecting and replacing null values within the dataset

# finding missing values of all columns in the data set

raw\_df.isnull().sum()

# finding the mode of education and marital to replace the null values with mode

print(raw\_df['EDUCATION'].mode())

print(raw\_df['MARITAL'].mode())

# replacing columns with null values with its mode

for column in ['EDUCATION', 'MARITAL']:

raw\_df[column].fillna(raw\_df[column].mode()[0], inplace=True)

raw\_df

# checking if all null values have been replaced

raw\_df.isnull().sum()

# using the shape to check if rows tally to the output

raw\_df.shape

### Second data preprocessing task: Converting R3 column from object to int64, and remove [$,]

# checking datatypes of all columns

raw\_df.dtypes

# trying to convert raw\_df['R3] to np.int64 but it gives ValueError: invalid literal for int() with base 10: '$0'

# the error causing R3 column to be an object datatype is $

try:

raw\_df['R3'].astype(np.int64)

except ValueError as value\_error:

print(value\_error)

print("There are dollar signs in the dataset, causing R3 to be an object datatype.")

# when replacing $ with "", there is another error which gives ValueError: invalid literal for int() with base 10: '2,620', proving that comma is the issue now

# Therefore replacing [$,] with "" since they were the errors preventing the datatype to be converted to np int64

raw\_df["R3"] = raw\_df["R3"].replace("[$,]", "", regex=True).astype(np.int64)

# checking if R3 column has been changed from object type to dtype

raw\_df.dtypes

# converting EDUCATION and MARITAL datatype to int64

raw\_df[['EDUCATION', 'MARITAL']] = raw\_df[['EDUCATION', 'MARITAL']].astype(np.int64)

# checking if EDUCATION and MARITAL column has been changed from float64 to int64

raw\_df.dtypes

### Third data pre-processing task: Removing duplicated rows from the dataframe

raw\_df

# suspecting that there are duplicates because there are 18769 rows but there are only 18766 IDs

# finding sum of duplicates within the dataframe

count\_duplicates = raw\_df.duplicated().sum()

print(count\_duplicates)

# locating the rows of duplicates within the dataframe

display(raw\_df.loc[raw\_df.duplicated(), :])

print("ID 132, 378 and 420 are duplicated rows")

# dropping duplicate rows and rows that contains at least one missing values

raw\_df = raw\_df.drop\_duplicates()

# checking if duplicates have been removed

raw\_df.duplicated().sum()

### Fourth data pre-processing task: Removing illogical outliers from AGE column

# descriptive statistics summary of raw\_df

raw\_df.describe() # found min and max value of age to be -1 and 199 respectively.

print("Dropping rows that contains values of -1 and 199 from AGE column because it is illogical”)

# find all unique values of AGE column

unique\_age = raw\_df['AGE'].unique()

# sorting values in ascending order

np.sort(unique\_age)

# Get names of indexes for which column AGE has value -1 and 199

indexNames = raw\_df[ (raw\_df['AGE'] == -1) | (raw\_df['AGE'] == 199)].index

print(indexNames)

print("There are 10 rows in the dataset with values -1 and 199 under the AGE column.")

# drop these row indexes from raw\_df

raw\_df.drop(indexNames, inplace=True)

# checking if columns with AGE with values -1 and 199 has been successfully removed

new\_unique\_age = raw\_df['AGE'].unique()

np.sort(new\_unique\_age)

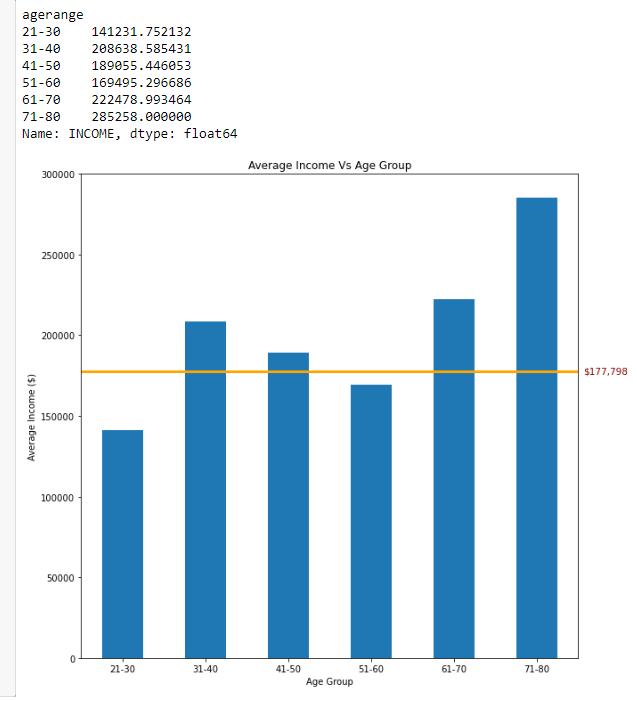
# the dataframe should have 10 less rows now

raw\_df.shape

Question 3

**Figure 1**

*Average Income Vs Age Group*

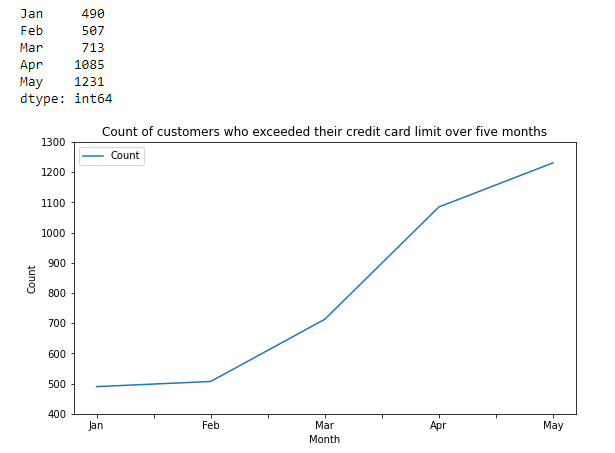


*Bar Chart.* This bar graph compares the mean income of all customers within their age group to the mean income of all customers.

The mean income of all customers is $177.798. From this bar graph and its corresponding table, we can deduce that people aged between 21-30 and 51-60 earn lesser than the mean income of all customers while the other 4 age groups earn more than the mean income of all customers. Additionally, the mean income of people aged 71-80 is two times that of people aged 21-30.

**Figure 2**

*Count of customers who exceeded their credit card limit over five months*

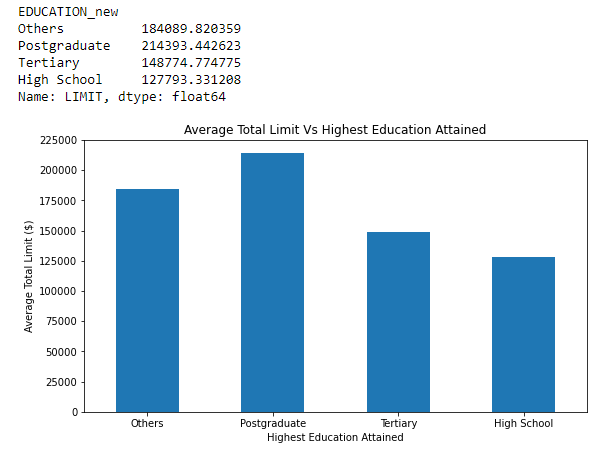


*Line Chart.* This line graph shows the count of customers who exceeded their credit card limit over five months.

Assuming B1 is the month of January, this line graph shows the increase in the number of customers who exceeded their credit card limit over the course of five months. From its corresponding table, the table shows that there is twice the amount of customers who exceed their credit card limit in April as compared to February.

**Figure 3**

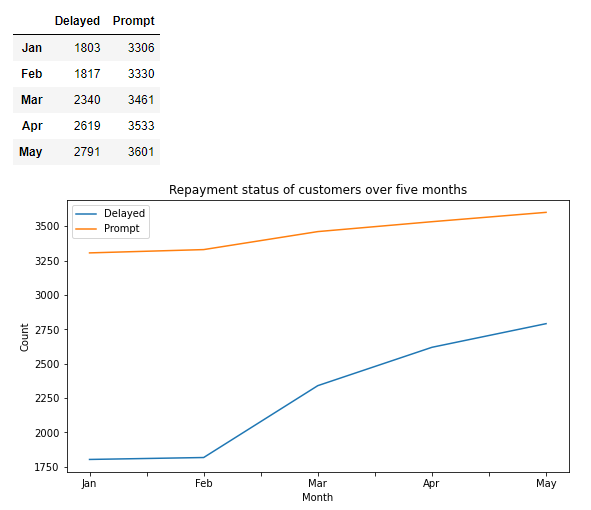
*Average Total Limit Vs Highest Education Attained*

*Bar Chart.* This bar graph compares the mean total limit across the four different education levels.

From the bar chart, it is deduced that the customers who attained “Postgraduate” status have a higher mean total limit as compared to customers with “Others”, “Tertiary” and “High School” education status. One assumption could be the increase in purchasing power for postgraduates as they are able to get a higher-paying job as compared to customers with "Others", "Tertiary" and "High School" education status, hence explaining the higher mean total limit.

**Figure 4**

*Repayment status of customers over five months*

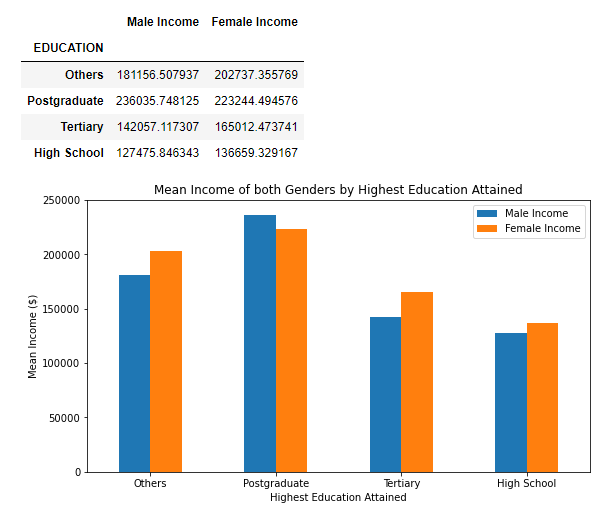


*Line Chart.* This line graph shows the count of customers who promptly paid, and those who delayed their payment over five months.

Assuming S1 is the month of January, the line graph shows an increase in customers who have been promptly paying their bills and those who delay their payment over five months. Compared to the gradual increase in customers who have been promptly paying their bills, there is a steep increase in the customers who have been delaying their payments. As seen from its corresponding table, the count of customers who has been delaying their payment has increased by at least 1.5 times from Jan., from 1803 to 2791.

**Figure 5**

*Mean Income of both Genders by Highest Education Attained*



*Bar Chart.* This bar graph compares the mean income of both genders across the four different education levels.

The bar graph shows that male postgraduates earn more than female postgraduates whereas female customers who attained the other three education levels have a higher income compared to their male counterparts. Additionally, it is observed that postgraduates have a higher income than customers who attained “Others”, “Tertiary” and “High School” education levels. This could be common in modern society because people who pursue higher education are often associated with individuals who earn a higher income.

### Discretising the numeric value of all categorical variables into bins in which they are going to be used for plotting the visualisation charts

#creating ranges for the ratings (0: Good, 1: Bad)

ratings\_bins = [-1, 0, 1]

ratings\_labels = ['Good', 'Bad']

# creating new column RATING\_new with pd.cut

raw\_df['RATING\_new'] = pd.cut(raw\_df.RATING, ratings\_bins, labels = ratings\_labels, include\_lowest = True)

display(raw\_df)

#creating ranges for the gender (0: Male, 1: Female)

gender\_bins = [-1, 0, 1]

gender\_labels = ['Male', 'Female']

# creating new column GENDER\_new with pd.cut

raw\_df['GENDER\_new'] = pd.cut(raw\_df.GENDER, gender\_bins, labels = gender\_labels, include\_lowest = True)

display(raw\_df)

#creating ranges for the education (0: Others, 1: Postgraduate, 2: Tertiary, 3: High School)

education\_bins = [-1, 0, 1, 2, 3]

education\_labels = ['Others', 'Postgraduate', 'Tertiary', 'High School']

# creating new column EDUCATION\_new with pd.cut

raw\_df['EDUCATION\_new'] = pd.cut(raw\_df.EDUCATION, education\_bins, labels = education\_labels, include\_lowest = True)

display(raw\_df)

#creating ranges for the marital status (0: Others, 1: Single, 2: Married)

marital\_bins = [-1, 0, 1, 2]

marital\_labels = ['Others', 'Single', 'Married']

# creating new column MARITAL\_new with pd.cut

raw\_df['MARITAL\_new'] = pd.cut(raw\_df.MARITAL, marital\_bins, labels = marital\_labels, include\_lowest = True)

display(raw\_df)

### First Visualisation Chart: Mean income of all customers within their age group vs mean income of all customers

#creating ranges for the ages

age\_bins = [21, 30, 40, 50, 60, 70, 80]

age\_labels = ['21-30', '31-40', '41-50', '51-60', '61-70', '71-80']

# creating new column agerange with pd.cut

raw\_df['agerange'] = pd.cut(raw\_df.AGE, age\_bins, labels = age\_labels, include\_lowest = True)

display(raw\_df)

# finding mean income of all customers in the dataset

mean\_income = raw\_df['INCOME'].mean()

# print(mean\_income)

# finding the mean income of the different age groups using .groupby

avg\_agegroup\_income = raw\_df.groupby(['agerange'])['INCOME'].mean()

display(avg\_agegroup\_income)

fig, ax = plt.subplots()

# plotting bar chart

avg\_agegroup\_income.plot(kind = 'bar', title = 'Average Income Vs Age Group',

ylabel = 'Average Income ($)', xlabel = 'Age Group', figsize = (10 , 10))

# plotting horizontal line using mean\_income

plt.axhline(mean\_income, color='orange', linewidth=3)

# horizontal line to represent mean income of all customers

ax.text(6.1,mean\_income,("${:,.0f}".format(mean\_income)),color="maroon",fontsize = 10.5, ha="right",va = 'center')

# plotting y limit

plt.ylim(0,300000)

# rotate x-axis labels vertically

plt.xticks(rotation = 0)

plt.show()

### Second Visualisation Chart: Increasing amount of customers who have exceeded their card limit over five months

# Creating new column, B1\_Limit, to identify customers who exceeded their credit card limit

raw\_df['B1\_Limit'] = raw\_df.apply(lambda x: 'ExceedB1'

if x['LIMIT'] <= x['B1'] else 'Under', axis=1)

# Finding the total count of people who exceeded their credit card limit

B1\_LIMIT = raw\_df.groupby(['B1\_Limit'])['B1\_Limit'].count()

# Creating new column, B2\_Limit, to identify customers who exceeded their credit card limit

raw\_df['B2\_Limit'] = raw\_df.apply(lambda x: 'ExceedB2'

if x['LIMIT'] <= x['B2'] else 'Under', axis=1)

# Finding the total count of people who exceeded their credit card limit

B2\_LIMIT = raw\_df.groupby(['B2\_Limit'])['B2\_Limit'].count()

# Creating new column, B3\_Limit, to identify customers who exceeded their credit card limit

raw\_df['B3\_Limit'] = raw\_df.apply(lambda x: 'ExceedB3'

if x['LIMIT'] <= x['B3'] else 'Under', axis=1)

# Finding the total count of people who exceeded their credit card limit

B3\_LIMIT = raw\_df.groupby(['B3\_Limit'])['B3\_Limit'].count()

# Creating new column, B4\_Limit, to identify customers who exceeded their credit card limit

raw\_df['B4\_Limit'] = raw\_df.apply(lambda x: 'ExceedB4'

if x['LIMIT'] <= x['B4'] else 'Under', axis=1)

# Finding the total count of people who exceeded their credit card limit

B4\_LIMIT = raw\_df.groupby(['B4\_Limit'])['B4\_Limit'].count()

# Creating new column, B5\_Limit, to identify customers who exceeded their credit card limit

raw\_df['B5\_Limit'] = raw\_df.apply(lambda x: 'ExceedB5'

if x['LIMIT'] <= x['B5'] else 'Under', axis=1)

# Finding the total count of people who exceeded their credit card limit

B5\_LIMIT = raw\_df.groupby(['B5\_Limit'])['B5\_Limit'].count()

# making a list for xlabels name

month\_list = ["Jan", "Feb", "Mar", "Apr", "May"]

# appending all the months of the count of people who exceeded their credit card limit

LIMIT\_table = B5\_LIMIT.append([B4\_LIMIT, B3\_LIMIT, B2\_LIMIT, B1\_LIMIT])

LIMIT\_table = LIMIT\_table.rename({"ExceedB5":"Jan", "ExceedB4":"Feb",

"ExceedB3":"Mar", "ExceedB2":"Apr",

"ExceedB1":"May"

})

# Dropping customers who did not exceed their credit card limit

LIMIT\_table = LIMIT\_table.drop('Under')

print(LIMIT\_table)

# creating a dictionary from the dataframe and setting LIMIT as index

linegraph\_table = pd.DataFrame({"Month":month\_list, "Count":LIMIT\_table

}).set\_index("Month")

# plotting line chart

linegraph\_table.plot(kind = 'line', title = 'Count of customers who exceeded their credit card limit over five months',

ylabel = 'Count', xlabel = 'Month', figsize = (9 , 5))

# setting y limit

plt.ylim(400,1300)

plt.show()

### Third Visualisation Chart: Distribution of average total limit vs highest education attained

# finding the mean total limit of the different highest education attained by customers using .groupby

avg\_education\_limit = raw\_df.groupby(['EDUCATION\_new'])['LIMIT'].mean()

# creating dataframe for better presentation

avg\_education\_limit\_df = pd.DataFrame(avg\_education\_limit)

display(avg\_education\_limit\_df)

# plotting bar chart

avg\_education\_limit.plot(kind = 'bar', title = 'Average Total Limit Vs Highest Education Attained',

ylabel = 'Average Total Limit ($)', xlabel = 'Highest Education Attained', figsize = (9 , 5))

# plotting y limit

plt.ylim(0,225000)

# rotate x-axis labels vertically

plt.xticks(rotation = 0)

plt.show()

### Fourth Visualisation Chart: Repayment status of customers who paid promptly and delayed payment over five months

raw\_df.describe()

# Creating new column, S1\_repayment\_status, to identify customers who delayed paying their bills, paid on time, and paid promptly.

raw\_df['S1\_repayment\_status'] = raw\_df.apply(lambda x: 'Prompt'

if x['S1'] == -1 else 'Minimum' if x['S1'] == 0 else 'Delayed', axis=1)

# Finding the total count of people for each category (Delayed, Minimum, Prompt)

S1\_repayment\_status = raw\_df.groupby(['S1\_repayment\_status'])['S1\_repayment\_status'].count()

# converting S1\_repayment\_status to a dataframe with to\_frame()

S1\_repayment\_status.to\_frame()

# Creating new column, S2\_repayment\_status, to identify customers who delayed paying their bills, paid on time, and paid promptly.

raw\_df['S2\_repayment\_status'] = raw\_df.apply(lambda x: 'Prompt'

if x['S2'] == -1 else 'Minimum' if x['S2'] == 0 else 'Delayed', axis=1)

# Finding the total count of people for each category (Delayed, Minimum, Prompt)

S2\_repayment\_status = raw\_df.groupby(['S2\_repayment\_status'])['S2\_repayment\_status'].count()

# converting S2\_repayment\_status to a dataframe with to\_frame()

S2\_repayment\_status.to\_frame()

# Creating new column, S3\_repayment\_status, to identify customers who delayed paying their bills, paid on time, and paid promptly.

raw\_df['S3\_repayment\_status'] = raw\_df.apply(lambda x: 'Prompt'

if x['S3'] == -1 else 'Minimum' if x['S3'] == 0 else 'Delayed', axis=1)

# Finding the total count of people for each category (Delayed, Minimum, Prompt)

S3\_repayment\_status = raw\_df.groupby(['S3\_repayment\_status'])['S3\_repayment\_status'].count()

# converting S3\_repayment\_status to a dataframe with to\_frame()

S3\_repayment\_status.to\_frame()

# Creating new column, S4\_repayment\_status, to identify customers who delayed paying their bills, paid on time, and paid promptly.

raw\_df['S4\_repayment\_status'] = raw\_df.apply(lambda x: 'Prompt'

if x['S4'] == -1 else 'Minimum' if x['S4'] == 0 else 'Delayed', axis=1)

# Finding the total count of people for each category (Delayed, Minimum, Prompt)

S4\_repayment\_status = raw\_df.groupby(['S4\_repayment\_status'])['S4\_repayment\_status'].count()

# converting S4\_repayment\_status to a dataframe with to\_frame()

S4\_repayment\_status.to\_frame()

# Creating new column, S5\_repayment\_status, to identify customers who delayed paying their bills, paid on time, and paid promptly.

raw\_df['S5\_repayment\_status'] = raw\_df.apply(lambda x: 'Prompt'

if x['S5'] == -1 else 'Minimum' if x['S5'] == 0 else 'Delayed', axis=1)

# Finding the total count of people for each category (Delayed, Minimum, Prompt)

S5\_repayment\_status = raw\_df.groupby(['S5\_repayment\_status'])['S5\_repayment\_status'].count()

# converting S5\_repayment\_status to a dataframe with to\_frame()

S5\_repayment\_status.to\_frame()

# concatenating all 5 dataframes

repayment\_status\_table = pd.concat([S5\_repayment\_status, S4\_repayment\_status, S3\_repayment\_status, S2\_repayment\_status, S1\_repayment\_status], axis=1)

# dropping Minimum column because it will scale up the graph

repayment\_status\_table = repayment\_status\_table.drop(["Minimum"])

# swapping the column and row index

transpose\_repayment\_status\_table = repayment\_status\_table.transpose()

transpose\_repayment\_status\_table = transpose\_repayment\_status\_table.rename({

"S5\_repayment\_status":"Jan", "S4\_repayment\_status":"Feb",

"S3\_repayment\_status":"Mar", "S2\_repayment\_status":"Apr",

"S1\_repayment\_status":"May"

})

display(transpose\_repayment\_status\_table)

# plotting line chart

transpose\_repayment\_status\_table.plot(kind = 'line', title = 'Repayment status of customers over five months',

ylabel = 'Count', xlabel = 'Month', figsize = (9 , 5))

plt.show()

### Fifth Visualisation Chart: Mean income of males and females with different highest education attained

# groupby mean income by gender and highest education level attained

mean\_income = raw\_df.groupby(["GENDER\_new", "EDUCATION\_new"])['INCOME'].mean()

# extracting education levels from groupby

education\_cat = ["Others", "Postgraduate", "Tertiary", "High School"]

# extracting mean male income from groupby

mean\_male = mean\_income[:4].tolist()

# extracting mean female income from groupby

mean\_female = mean\_income[4:].tolist()

# creating a dictionary from the dataframe and setting EDUCATION as index

graph\_table = pd.DataFrame({"EDUCATION":education\_cat,"Male Income":mean\_male,"Female Income":mean\_female

}).set\_index("EDUCATION") #creating a new dataframe table

display(graph\_table)

# plotting the bar chart

graph\_table.plot(kind = 'bar', title = 'Mean Income of both Genders by Highest Education Attained',

ylabel = 'Mean Income ($)', xlabel = 'Highest Education Attained', figsize = (9 , 5))

# setting y limit

plt.ylim(0,250000)

# rotate x-axis labels vertically

plt.xticks(rotation = 0)

plt.show()

Question 4

Firstly, I created a dataframe for the independent variables that I wish to use to predict the dependent variable, B1. The independent variables are LIMIT, BALANCE, INCOME, RATING, GENDER, EDUCATION, MARITAL, and AGE. Thereafter, the categorical variables (GENDER, EDUCATION, MARITAL, RATING) are all converted to dummy variables using pd.get\_dummies() function before it can be used to perform linear regression. Additionally, a dataframe was created to store the dependent variable, B1.

Thereafter, I used the .fit() function to parse the independent variables and dependent variables to initialise the linear regression model. A for-loop was created to predict the values of the dependent variable, B1.

# creating a dataframe for the independent variables

x\_pred = raw\_df[['LIMIT', 'BALANCE', 'INCOME', 'RATING',

'GENDER', 'EDUCATION','MARITAL','AGE']]

# getting dummies for the categorical variables to be used for linear regression, not dropping first labels will be more representative (i.e. education level will be more represented)

x\_pred = pd.get\_dummies(x\_pred, columns =['RATING', 'GENDER', 'EDUCATION', 'MARITAL'])

display(x\_pred)

# creating a dataframe for the dependent variable, B1

y\_pred = raw\_df[['B1']]

y\_pred

# fitting independent variables and dependent variables to the model

model = LinearRegression().fit(x\_pred.values, y\_pred.values)

print(model.score(x\_pred.values, y\_pred.values))

# coefficients of the model

print(f"coefficients are {np.round(model.coef\_, 5)}")

# rounding coefficients to 5 decimal places

model\_coeff = np.round(model.coef\_, 5)

# creating dataframe for the coefficients

model\_coeff\_df = pd.DataFrame(model\_coeff.T, index=x\_pred.columns)

# renaming column name to coefficient

model\_coeff\_df.rename(columns = {0:"Coefficient"}, inplace=True)

model\_coeff\_df

# creating a loop to predict the values of B1

predicted\_b1 = []

for i in range(len(raw\_df[['B1']])):

predicted\_b1.append(model.predict(x\_pred.values[[i]])[0][0])

# creating a dataframe for predicted\_B1 values

predicted\_b1\_df = pd.DataFrame(predicted\_b1, columns=['predicted\_B1'])

# changing the datatype to integer

predicted\_b1\_df = predicted\_b1\_df.astype(int)

predicted\_b1\_df

Question 5

# intercept of the model

print(f"this is the intercept {model.intercept\_}")  
The intercept of the multiple linear regression model is 187.62986961.

# Multiple linear regression equation

Ŷ = 187.62987 + 0.00692(LIMIT) + 5.23285(BALANCE) + (-0.00295)(INCOME) + GENDER + EDUCATION + MARITAL + 37.09053(AGE) + RATING

|  |  |  |  |
| --- | --- | --- | --- |
| GENDER | EDUCATION | MARITAL | RATING |
| Where GENDER = 2.16708 when GENDER is Male | Where EDUCATION = -1427.82172 when EDUCATION is Others | Where MARITAL = 138.73086 when MARITAL is Others | Where RATING = -572 when RATING is Good |
| Where GENDER = -2.16708 when GENDER is Female | Where EDUCATION = 591.00530 when EDUCATION is Postgraduate | Where MARITAL = -555.61854 when MARITAL is Single | Where RATING = 572 when RATING is Bad |
|  | Where EDUCATION = 514.37115 when EDUCATION is Tertiary | Where MARITAL = 416.88767 when MARITAL is Married |  |
|  | Where EDUCATION = 322.44527 when EDUCATION is High School |  |  |

For every increase in $1 of LIMIT, BALANCE, and INCOME, the estimated billable amount increases by $0.01 and $5.23, and decreases by $0.003 respectively. For every increase in 1 year of AGE, the estimated billable amount increases by $37.09.

From the findings, the billable amount increases by $2.17 when the gender is male and the billable amount decreases by $2.17 when the gender is female. However, it does not make logical sense that either gender should be paying more or less than the other. Moreover, it can be concluded that gender is statistically insignificant.

When the highest education attained is Others, the individual’s billable amount decreases by $1427.82. Conversely, an individual’s billable amount increases by $322.45, $514.37, and $591.00 when the highest education attained are High School, Tertiary and Postgraduate respectively. The values may be assumed that the higher the education attained, an individual’s spending power increases and hence may spend more money, resulting in a higher billable amount.

When the marital status of an individual is single, their billable amount decreases by $555.62. Conversely, an individual’s billable amount increases by $138.73 and $416.89 when the marital status is Others and Married respectively. The values may be assumed that individuals who are single do not spend as much as compared to married individuals because married individuals may have more family members in the household and hence will generate a higher billable amount.

From the findings, the billable amount increases by $572.00 when the rating given is good and the billable amount decreases by $572.00 when the rating given is bad. However, it does not make logical sense that the individual should be paying more or less based on their rating.