Credit Card Clients in Taiwan in 2005-Box Plots, Count Plots & Histograms REPORT

SAMUEL

SKILL LYNC TRAINEE EMPLOYEE

06.05.202

Index:

Introduction: **Based on the financial capability of a client, they get a credit limit, i.e., the maximum amount they can spend in a month through a credit card.**

Credit card companies maintain comprehensive data about each of their clients. By analysing the data, they can know what would be the maximum amount they won't be able to recover from their clients yet able to make a significant profit in a financial year to run a sustainable business.

Problem statement: **The credit card clients dataset is full of irregularities and incorrect values. You need to replace them with the right values. Additionally, you have to create box plots, count plots and histograms to find a specific trend (if there exists) in the dataset.**

Solution Approach: **Here we use python and machine learning tools to make the values right from given irregularities values in the given dataset of credit cards, Additionally we create box plots, count plots and histograms to find the specific trend in the dataset, from the plots we analyse the client's credit card data and make models for better credit card service**

Data overview

Data:

Here we are abstracting the required data from credit card resources by using **Python Libraries.**

Here are the screenshots of data abstraction from the source :

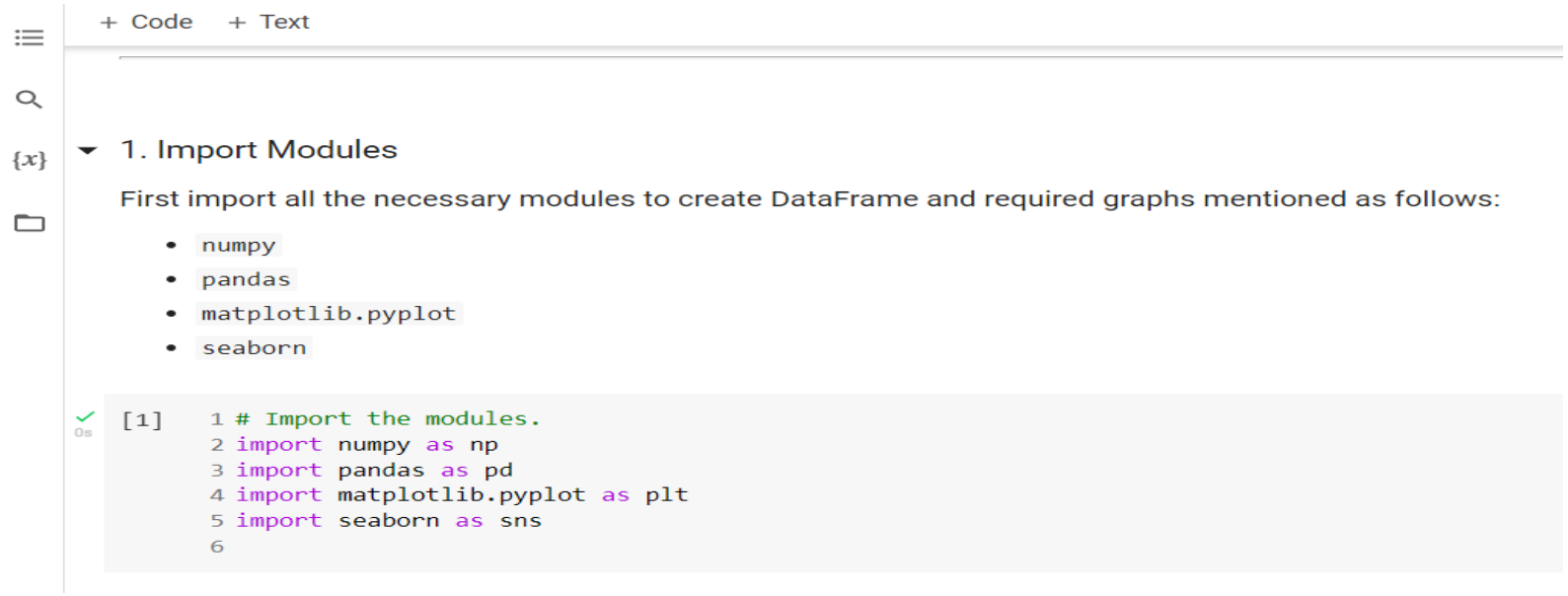


Figure 1

Source link: "https://raw.githubusercontent.com/m-narayanan22/datasets/main/UCI_Credit_Card.csv"

From the source link, we got the dataset values of credit card client details like their Age, Education, Limit balance and etc.,

```
# Load the dataset.
# Dataset link: 'https://raw.githubusercontent.com/m-narayanan22/datasets/main/UCI_Credit_Card.csv'
credit_cd_df = pd.read_csv("https://raw.githubusercontent.com/m-narayanan22/datasets/main/UCI_Credit_Card.csv")
print(credit_cd_df)
credit_cd_df.head()
credit_cd_df.tail()

```

```
           ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  \
0           1    20000.0    2          2         1   24      2      2     -1
1           2   120000.0    2          2         2   26     -1      2      0
2           3    90000.0    2          2         2   34      0      0      0
3           4    50000.0    2          2         1   37      0      0      0
4           5    50000.0    1          2         1   57     -1      0     -1
...       ...        ...  ...        ...       ...  ...    ...    ...    ...
29995   29996   220000.0    1          3         1   39      0      0      0
29996   29997   150000.0    1          3         2   43     -1     -1     -1
29997   29998    30000.0    1          2         2   37      4      3      2
29998   29999    80000.0    1          3         1   41      1     -1      0
29999   30000    50000.0    1          2         1   46      0      0      0

       PAY_4  ...  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  \
0         -1  ...        0.0     9000        0.0       0.0     689.0
1          0  ...     3272.0        . ...  3261.0       0.0    1000.0
2          0  ...    14331.0    14948.0    15549.0    1518.0    1500.0
3          0  ...    28314.0    28959.0    29547.0    2000.0    2019.0
4          0  ...    20940.0    19146.0    19131.0    2000.0   36681.0
...      ...  ...        ...        ...        ...       ...       ...
29995      0  ...    88004.0    31237.0    15980.0    8500.0   20000.0
29996     -1  ...     8979.0     5190.0        0.0    1837.0    3526.0
29997     -1  ...    20878.0    20582.0    19357.0       0.0       0.0
29998      0  ...    52774.0    11855.0    48944.0   85900.0    3409.0
29999      0  ...    36535.0    32428.0    15313.0    2078.0    1800.0

       PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  default.payment.next.month
0           0.0       0.0       0.0       0.0                          1
1        1000.0    1000.0       0.0    2000.0                          1
2        1000.0    1000.0    1000.0    5000.0                          0
3        1200.0    1100.0    1069.0    1000.0                          0
4       10000.0    9000.0     689.0     679.0                          0
...         ...       ...       ...       ...                        ...
29995    5003.0    3047.0    5000.0    1000.0                          0
29996    8998.0     129.0       0.0       0.0                          0
29997   22000.0    4200.0    2000.0    3100.0                          1
29998    1178.0    1926.0   52964.0    1804.0                          1
29999    1430.0    1000.0    1000.0    1000.0                          1
```

Figure 2

From above FIgure 2, we got know that the clients' credit dataset is huge, so for to analyse the given data use

python libraries, and we will plot the data, and we will see the plot in the upcoming Figures

Here we are finding the rows and columns are there in the dataset.

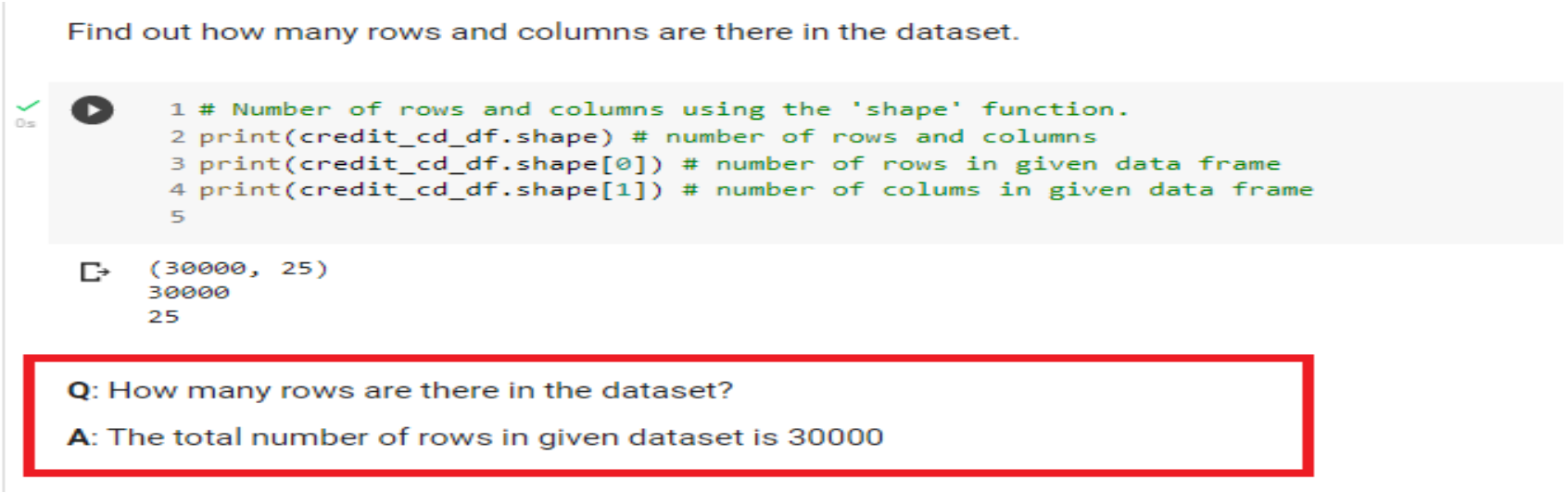


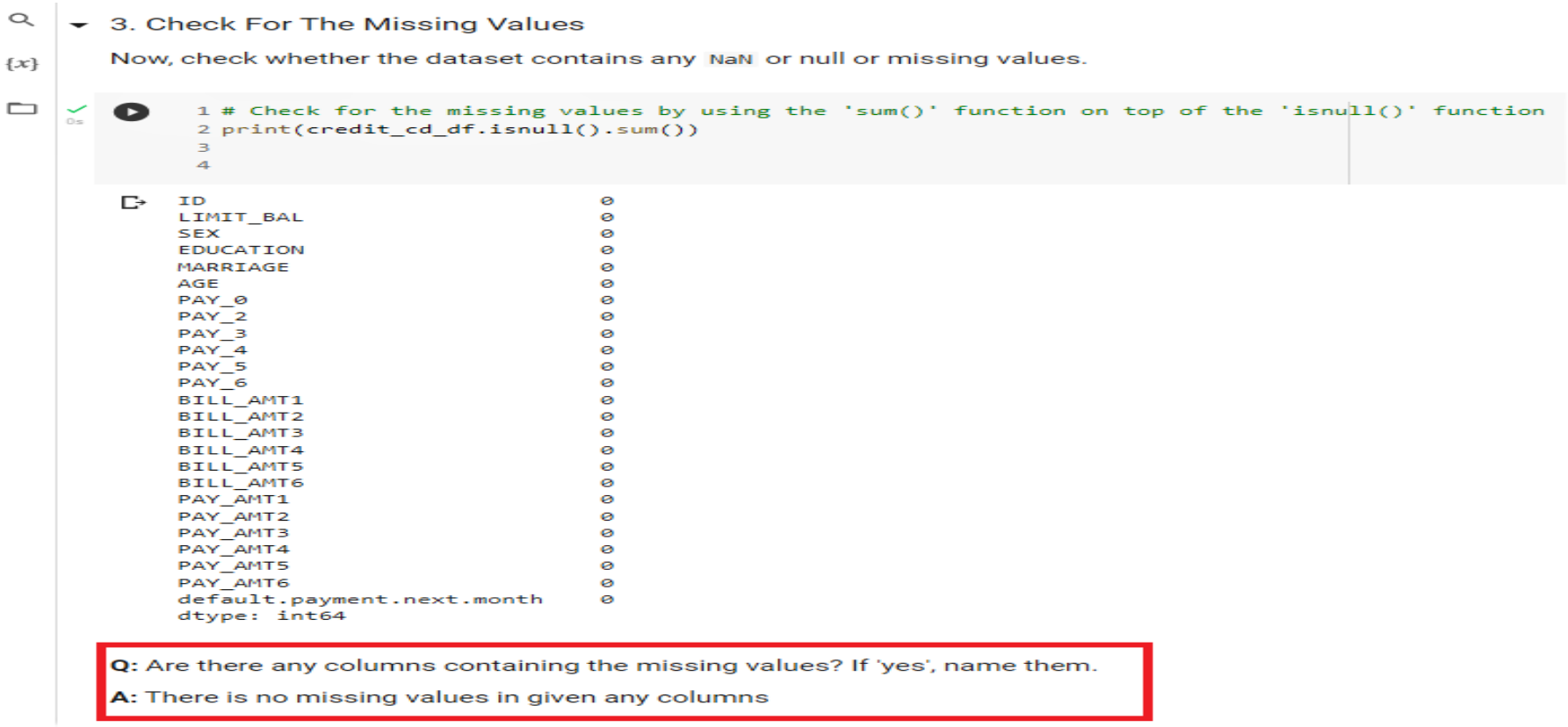
Figure 3

From the above Figure 3, the number of rows and columns is (3000, 25).

The red marked box represents the Question and answers for the specified task.

Checking For The Missing Values

Now, check whether the dataset contains any NaN or null or missing values



3. Check For The Missing Values

Now, check whether the dataset contains any `NaN` or null or missing values.

```
# Check for the missing values by using the 'sum()' function on top of the 'isnull()' function
print(credit_cd_df.isnull().sum())
```

```
ID                          0
LIMIT_BAL                   0
SEX                         0
EDUCATION                   0
MARRIAGE                    0
AGE                         0
PAY_0                       0
PAY_2                       0
PAY_3                       0
PAY_4                       0
PAY_5                       0
PAY_6                       0
BILL_AMT1                   0
BILL_AMT2                   0
BILL_AMT3                   0
BILL_AMT4                   0
BILL_AMT5                   0
BILL_AMT6                   0
PAY_AMT1                    0
PAY_AMT2                    0
PAY_AMT3                    0
PAY_AMT4                    0
PAY_AMT5                    0
PAY_AMT6                    0
default.payment.next.month  0
dtype: int64
```

Q: Are there any columns containing the missing values? If 'yes', name them.

A: There is no missing values in given any columns

Figure 4

From the above Figure 4, we can observe that there are no missing values in the given dataset.

Now, We are going to find the education column's value, count values and plotting of the education columns.

Q: What value(s) is/are contained in the `EDUCATION` column apart from the values `1` to `5`? And what will you do with them?

A: 0 and 6 is the values apart from the 1 to 5 and We have to replace 0 and 6 value with 5

```
[9]  1 # Replace the unwanted values in the 'EDUCATION' column. Ignore if there are none.
     2 replace_indices1 = credit_cd_df[credit_cd_df['EDUCATION'] == 6].index
     3 credit_cd_df.loc[replace_indices1, ['EDUCATION']] = 5
     4
     5 replace_indices2 = credit_cd_df[credit_cd_df['EDUCATION'] == 0].index
     6 credit_cd_df.loc[replace_indices2, ['EDUCATION']] = 5
     7 replace_indices2
     8
     9
```

```
Int64Index([ 3769,  5945,  6876, 14631, 15107, 16881, 16896, 17414, 19920,
             20030, 23234, 24137, 27155, 27270],
            dtype='int64')
```

Hint: You can replace the rows with `0` and `6` in the `Education` column by using the `loc[]` function.

Syntax: `df.loc[df['column_name'] == old_value, 'EDUCATION'] = new_value`

Calculate the percentage of each value in the `EDUCATION` column.

```
     1 # Percentage of each value in the 'EDUCATION' column.
     2 per_1 =credit_cd_df["EDUCATION"].value_counts() * 100 / credit_cd_df.shape[0]
     3 per_1
     4
     5
```

```
2    46.766667
1    35.283333
3    16.390000
5     1.150000
4     0.410000
Name: EDUCATION, dtype: float64
```

Hint: You can get the total number of counts of each value in the column by using the `value_counts()` function. Then you can calculate the percentage of each value by multiplying the total number of counts of each value with 100 and dividing the resultant value by the total number of rows in the DataFrame (`df.shape[0]`).

Q: What percent of clients were university graduates?

A: From above data we got..46.7% percent of cilents were university graduate (round off for 46.7666 is 47)

Figure 5

From above Figure 5, we can understand that the percentage of clients who were university graduates is 47%

So we can observe that half of the clients are Educated.

Creating a count plot for the `EDUCATION` column:

Create a count plot for the EDUCATION column.

```
# Count plot for the 'EDUCATION' column using the 'countplot()' function of 'seaborn' module.
plt.figure(figsize=(24,4))
sns.countplot(x="EDUCATION", data =credit_cd_df)
plt.grid()
```

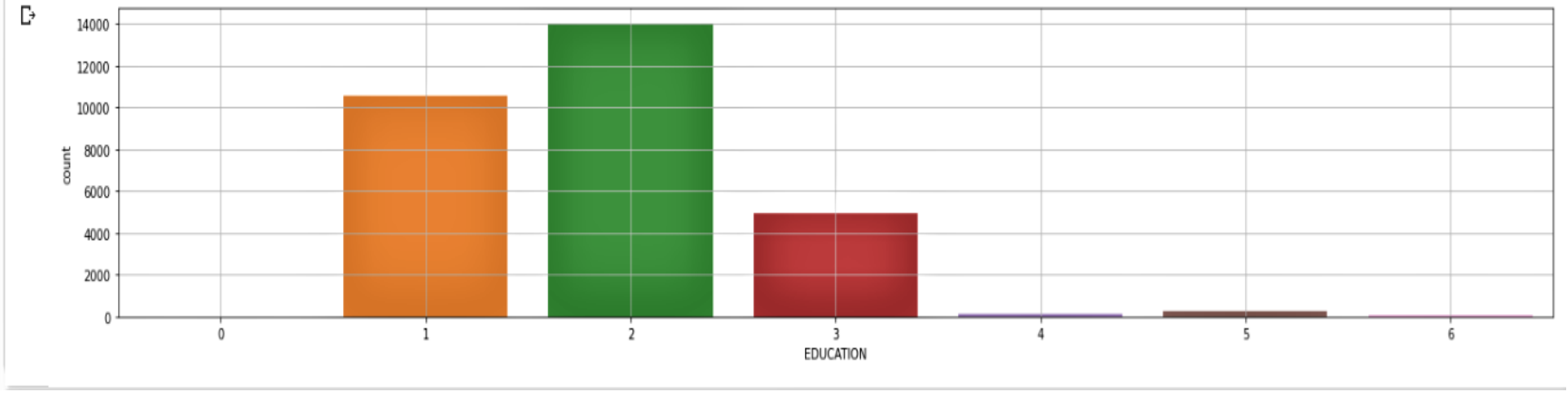


Figure 6

From above Figure 6, we can say that there are 6 bars with different count values, the green having the highest count of 14000.

Checking the data types of all the columns using the 'info()' function.

You may require to check the data-type of every column. So, instead of applying the `dtype` keyword one-by-one for each column, you can use the `info()` function to check the data-types of all the columns at once. It also tells you the total number of rows and columns in a DataFrame. Here's the syntax:

Syntax: `data_frame.info()`

where `data_frame` is a variable storing some Pandas DataFrame.

Note: This function is applicable only to Pandas DataFrame.

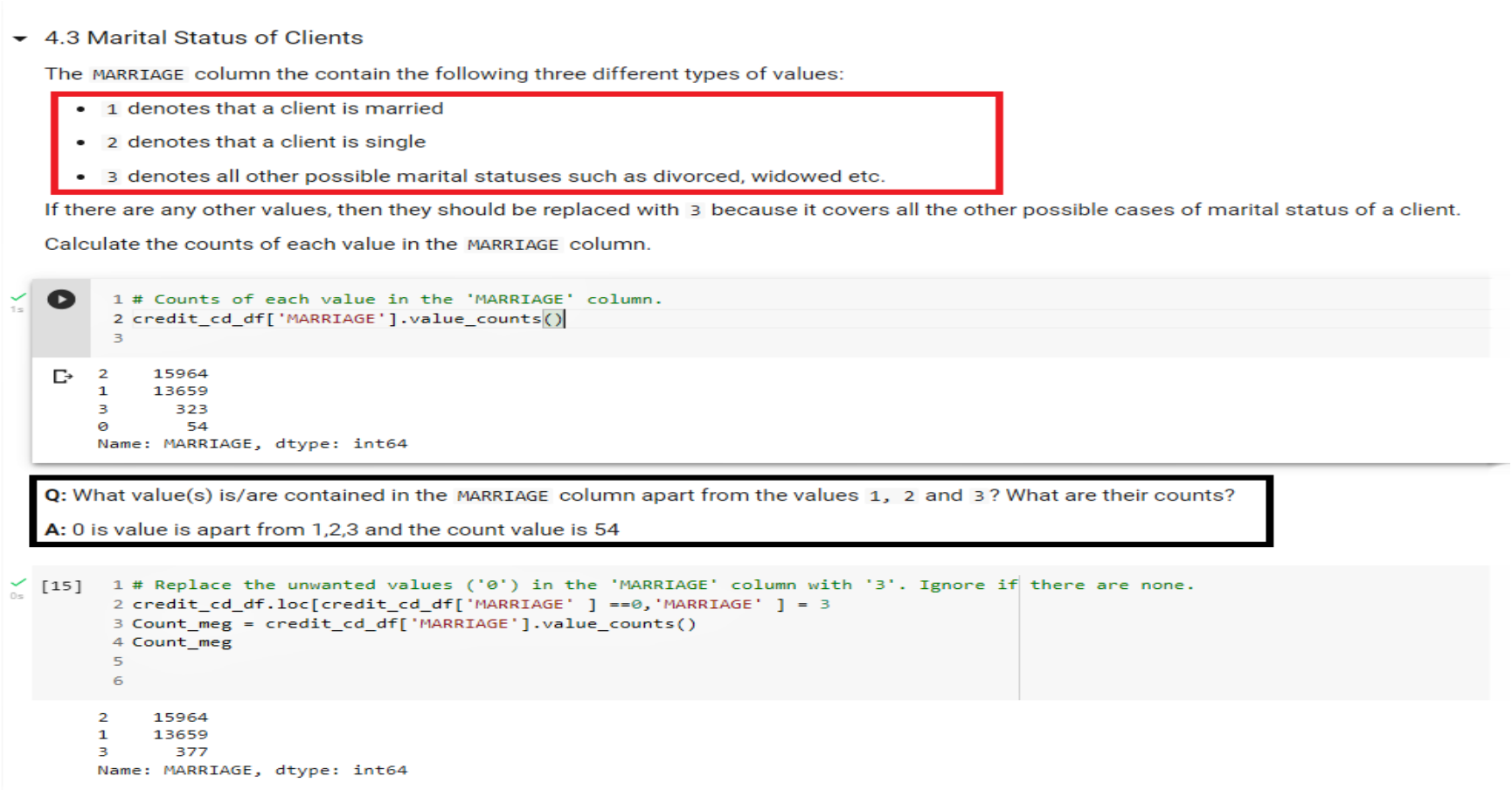
```
# Check the data-types of all the columns using the 'info()' function.
credit_cd_df.info()

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   ID                          30000 non-null  int64  
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64  
 3   EDUCATION                   30000 non-null  int64  
 4   MARRIAGE                    30000 non-null  int64  
 5   AGE                         30000 non-null  int64  
 6   PAY_0                       30000 non-null  int64  
 7   PAY_2                       30000 non-null  int64  
 8   PAY_3                       30000 non-null  int64  
 9   PAY_4                       30000 non-null  int64  
 10  PAY_5                       30000 non-null  int64  
 11  PAY_6                       30000 non-null  int64  
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64  
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

Figure 7

Marital Status of Clients:

The below figure represents the client's marital status which means who is married, single or divorced.



4.3 Marital Status of Clients

The `MARRIAGE` column the contain the following three different types of values:

- `1` denotes that a client is married
- `2` denotes that a client is single
- `3` denotes all other possible marital statuses such as divorced, widowed etc.

If there are any other values, then they should be replaced with `3` because it covers all the other possible cases of marital status of a client.

Calculate the counts of each value in the `MARRIAGE` column.

```
# Counts of each value in the 'MARRIAGE' column.
credit_cd_df['MARRIAGE'].value_counts()
```

```
2    15964
1    13659
3      323
0       54
Name: MARRIAGE, dtype: int64
```

Q: What value(s) is/are contained in the `MARRIAGE` column apart from the values `1, 2` and `3`? What are their counts?

A: 0 is value is apart from 1,2,3 and the count value is 54

```
[15]  # Replace the unwanted values ('0') in the 'MARRIAGE' column with '3'. Ignore if there are none.
      credit_cd_df.loc[credit_cd_df['MARRIAGE' ] ==0,'MARRIAGE' ] = 3
      Count_meg = credit_cd_df['MARRIAGE'].value_counts()
      Count_meg
```

```
2    15964
1    13659
3      377
Name: MARRIAGE, dtype: int64
```

Figure 8

From the above Figure 8, we can observe that the number of married, single or divorced and 1,2 and 3 represent the

marital status of clients.

Calculating the percentage of the values in the MARRIAGE column in Figure 9 below:

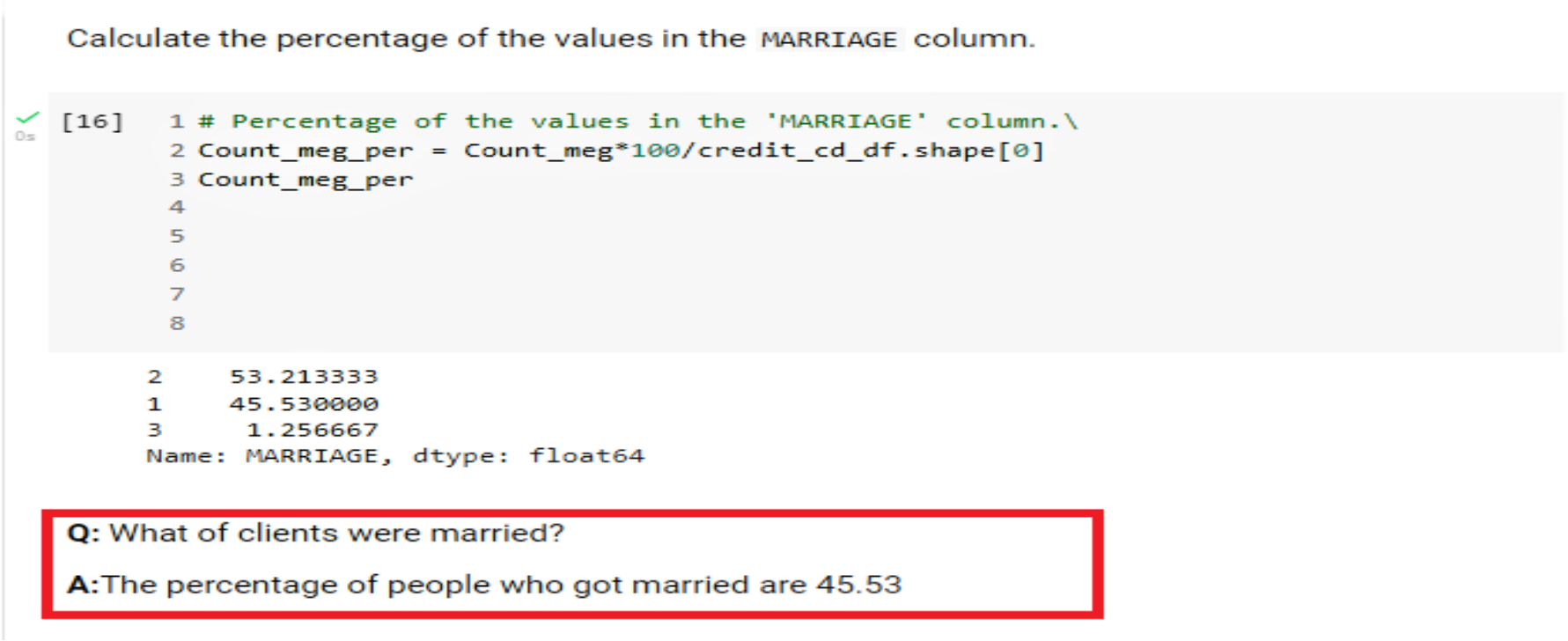


Figure 9

From above Figure 9, we can say that the percentage of married clients is 46 (Round off value).

Creating a count plot for the MARRIAGE column so that we can analyse the data:

```
# Count plot for the 'MARRIAGE' column.
plt.figure(figsize =(20,4))
sns.countplot(x='MARRIAGE', data =credit_cd_df, edgecolor ='Black')
plt.grid()
```

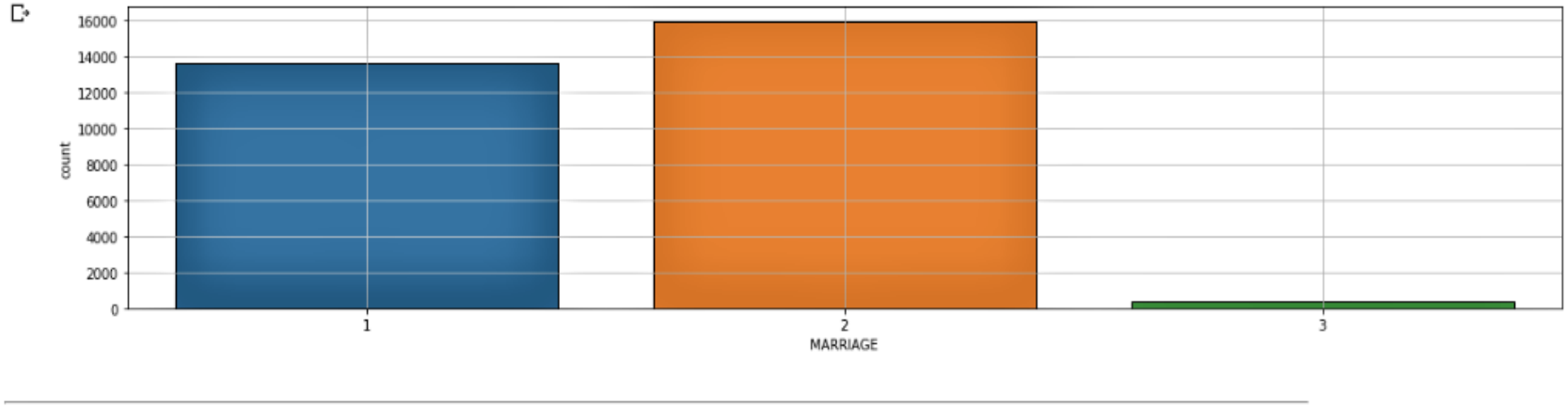


Figure 10

Creating box plots and histograms for the columns containing continuous numeric values.

Box Plot & Histogram For The AGE Column

All the histograms are in grids.

Creating a box plot for the AGE column by using the given dataset:

▾ 5. Box Plots & Histograms

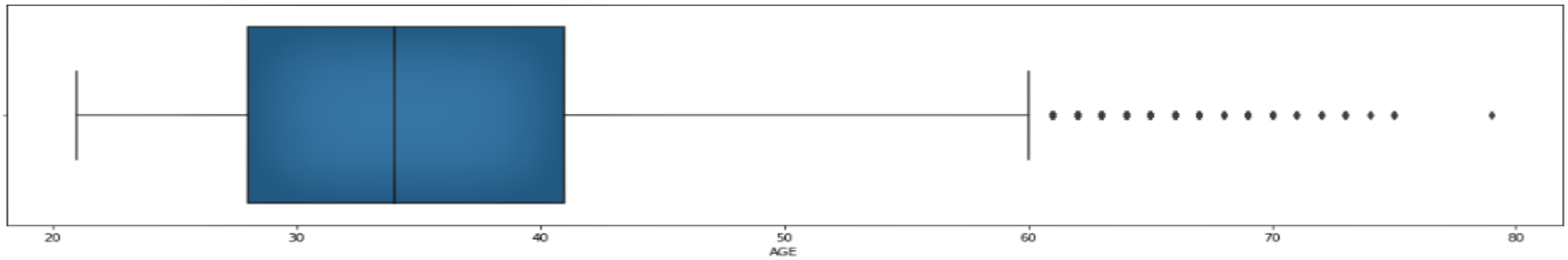
The final task is to create box plots and histograms for the columns containing continuous numeric values.

Note: All the histograms must have grids.

▾ 5.1 Box Plot & Histogram For The `AGE` Column

Create a box plot for the `AGE` column.

```
# Box plot for the 'AGE' column using the 'boxplot()' function.
plt.figure(figsize =(20,4))
sns.boxplot(x =credit_cd_df['AGE'])
plt.show()
```



Q: From the box plot for the `AGE` column, what is the approx median age of a credit card holder?

A: The approx. median age of a credit card holder is 35

Figure 11

Here, From the above FIgure,11 10we can observe the boxplot of age in different distributions at different intervals, also we can say that the Median age of the client is 35, and the points dots are very negligible values in the whole distribution in a boxplot**.**

Creating Histogram distplot for the column of Age:

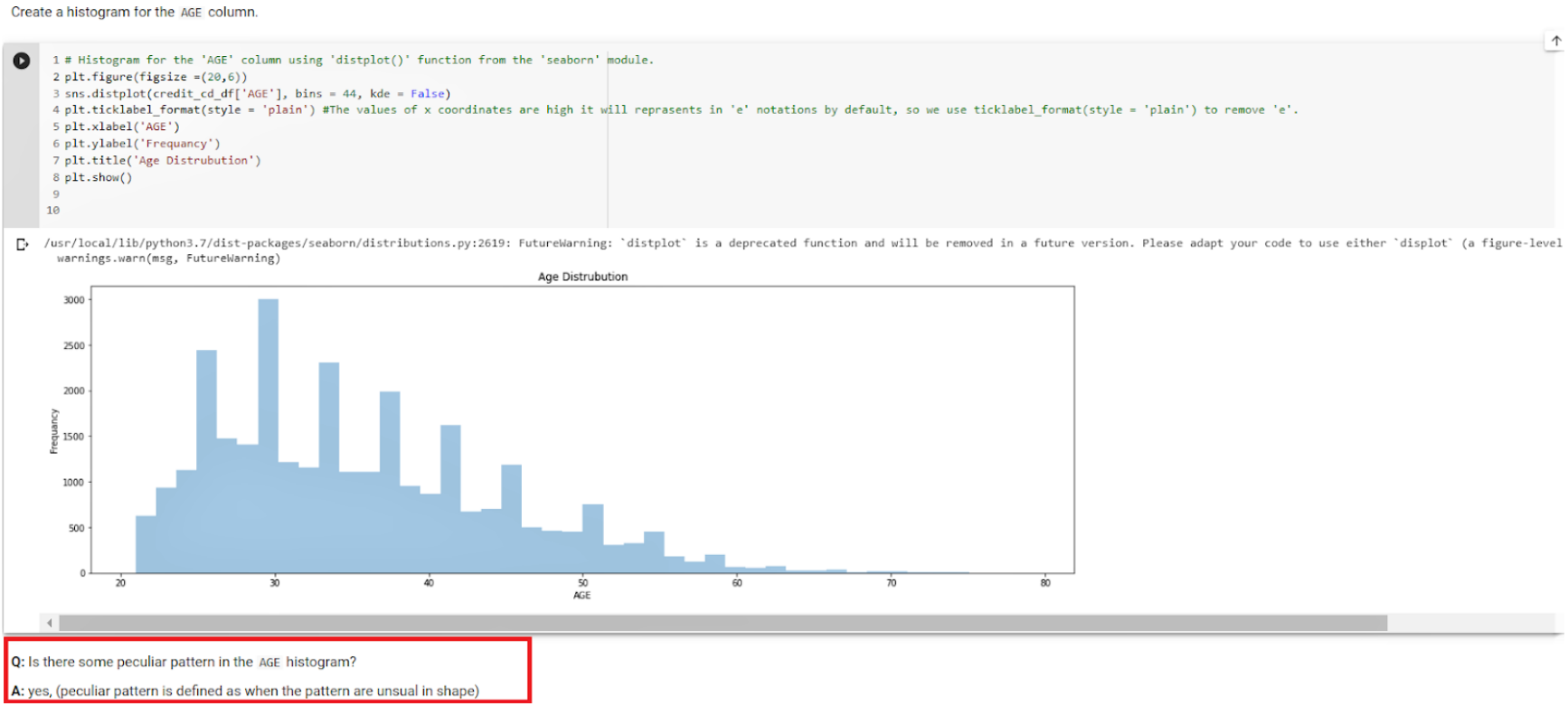


Figure 12

Here, From above 12 we can say that the histogram of the Age column is in a peculiar pattern, which means the Age distribution is not uniform.

Creating a box plot & histogram For The LIMIT_BAL column:

5.2 Box Plot & Histogram For The `LIMIT_BAL` Column

Create a box plot for the `LIMIT_BAL` column.

```
1 # Box plot for the LIMIT_BAL column.
2 plt.figure(figsize =(20,4))
3 sns.boxplot(x = credit_cd_df['LIMIT_BAL'])
4 plt.show()
5
6
```

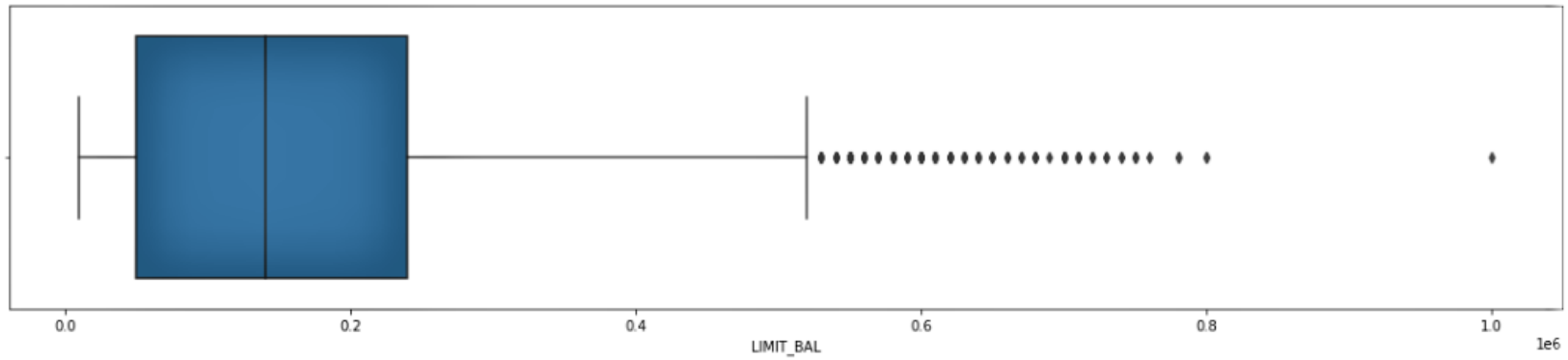


Figure 13

Here, in Figure 13 we can say that the Median of the LiMIT_BAl column is Median is 140000 and max value is 1000000, and the points which are dotted in shape are negligible values

.

Creating a dist plot & histogram For The LIMIT_BAL column:

```
# Histogram for the 'LIMIT_BAL' column using 'distplot()' function from the 'seaborn' module.
plt.figure(figsize =(25,6))
sns.distplot(credit_cd_df['LIMIT_BAL'], bins = 44, kde = False, color ='darkblue')
plt.ticklabel_format(style = 'plain')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to us
  warnings.warn(msg, FutureWarning)
```

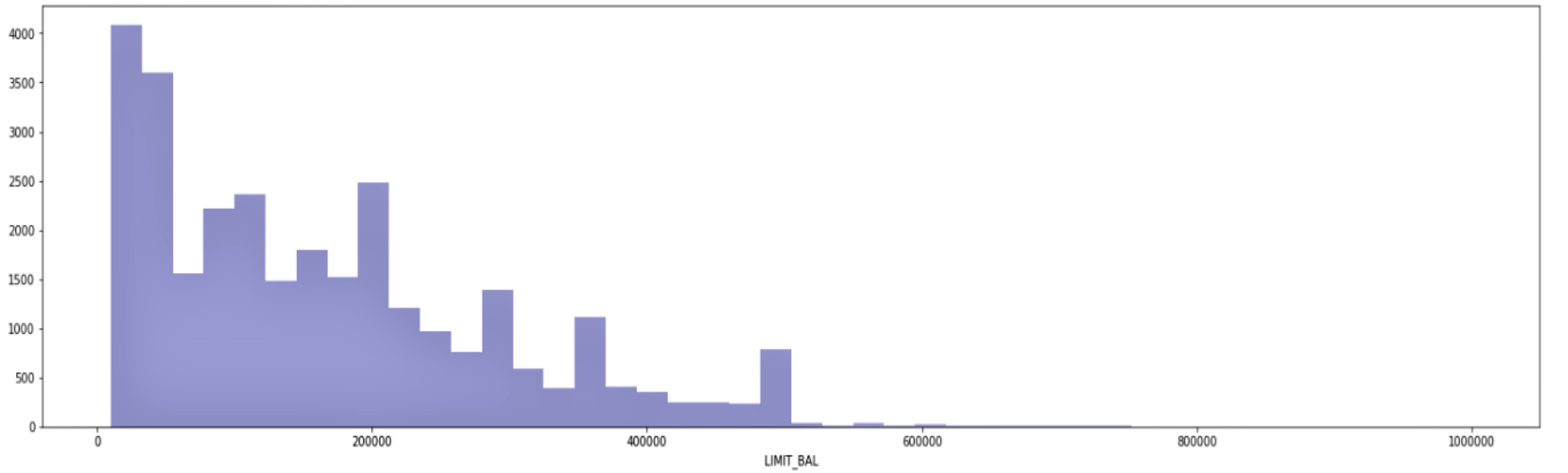


Figure 14

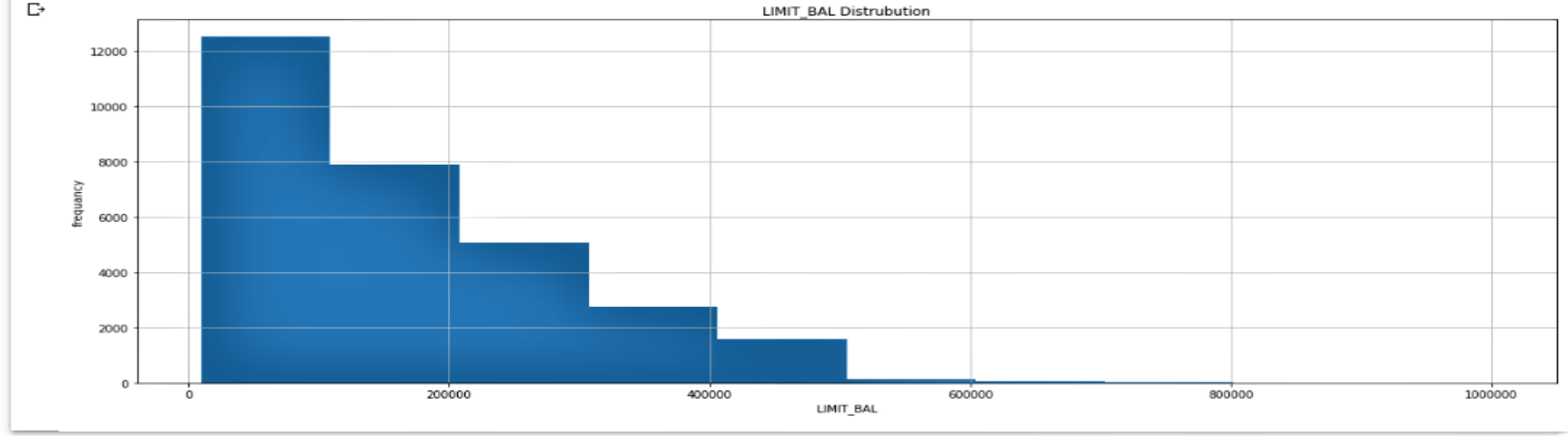
Here, in Figure 14 we observe that the LIMIT_BAL Column count values are in the high range, and the distribution is not in the distribution that's why the pattern of the histogram is a peculiar pattern.

Creating Histogram (having 50 bins) for the 'LIMIT_BAL' column :

Here is down below the histogram is

created with the required conditions :

```
# Histogram (having 50 bins) for the 'LIMIT_BAL' column using 'hist()' function from the 'matplotlib.pyplot' module.
plt.figure(figsize = (21, 7))
plt.hist(credit_cd_df["LIMIT_BAL"])
plt.grid()
plt.xlabel('LIMIT_BAL')
plt.ylabel('frequancy')
plt.ticklabel_format(style = 'plain') #this function is used to remove 'e' notation
plt.title('LIMIT_BAL Distrubution')
plt.show()
```



Q: What happens to the counts of `LIMIT_BAL` values as the `LIMIT_BAL` values increases? What does it signify?

A: if LIMIT_BAL is increases the histogram distrubation will change and it will effect the clients barrow limit from credit card aganics

Figure 15

Here, in Figure 15 we can say that the highest frequency is above 12000 and the distribution is decreasing with the value of LIMIT_BAL and the distribution is in decreasing pattern and if every client LIMIT_BAL is increasing the x coordinate only will affect. And frequency will not be

affected but the client will spend on things it will affect his lifestyle but the client spends more he/she will be debit in future.

Advantages of a Credit Card

1. One-Time Bonuses

There's nothing like an initial bonus opportunity when getting a new credit card. Oftentimes, applicants with good credit or excellent credit can get approved for credit cards that offer bonuses worth $150 or more (sometimes much more) in exchange for spending a certain amount (anywhere from $500 to several thousands of dollars) in the first several months the account is open.

2. Cash Back

The cash-back credit card was first popularized in the United States by Discover, and the idea was simple: Use the card and get 1% of your purchases rebated in the form of cashback. Today, the concept has grown and matured. Now, some cards now offer 2%, 3% or even as much as 6% cashback on selected purchases, though such lucrative offers involve quarterly or annual spending caps. The best cash-back cards are those that charge minimal fees and interest while offering a high rewards rate.

3. Rewards Points

Credit cards are set up to allow cardholders to earn one or more points per dollar in spending. Many reward credit cards provide bonus points for certain categories of spending like restaurants, groceries or gasoline. When certain earnings thresholds are reached, points can be redeemed for travel, gift cards from retailers

and restaurants, or merchandise items through the credit card company's online rewards portal.

5. Safety

Paying with a credit card makes it easier to avoid losses from fraud. When your debit card is used by a thief, the money is missing from your account instantly. Legitimate expenses for which you've scheduled online payments or mailed checks may bounce, triggering insufficient funds fees and affecting your credit. Even if not your fault, these late or missed payments can lower your credit score. It can take time for fraudulent transactions to be reversed and money restored to your account while the bank investigates.

Technologies used for the project:

We used Python tools to analyse the given data and make a plot of the different columns like Age, Education, Marriage etc., with different conditions for a better understanding of the dataset, and we find every possible condition in the dataset like Max, Median, Min etc., this makes to analyse given data and makes easy to credit card companies for clients requirements.

Conclusion:

From the given dataset of Credit Card Clients in Taiwan in 2005, we concluded that the age and sex is not a barrier to the user not having a credit card in the modern era, the credit card is becoming a part of human life and having a credit card makes users lead a better life and the user can make any translation by simpling using a credit card like online travel booking, making online purchase etc,

…..END…..