Project

EDA

Name: Sudarshan Suresh Disale

Employee ID: 26028

Dataset: BankChurners.csv

Problem Statement: Happy Bank provides various credit cards to customers. The manager of Happy Bank is disturbed by more and more customers leaving their credit card services. The team did a customer survey to check customer attrition. Various customer attributes like Customer_Age, Credit_Limit, Dependent_Count. The team would really appreciate it if one could predict for them who is gonna get churned so they can proactively go to the customer to provide them better services and turn customers' decisions in the opposite direction.

1. **Import Necessary Libraries.**

```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sb
        import matplotlib.pyplot as mp
        %matplotlib inline
        import warnings
        warnings.filterwarnings('ignore')
        df = pd.read_csv("BankChurners.csv")
```

- Imported all the necessary libraries to do exploratory data analysis.
- Pandas used to work with data, numpy for calculations, seaborn and matplotlib for plotting graphs, etc.

2. **Display a sample of five rows of the data frame.**

```
In [2]: df.sample(5)
```

Out[2]:

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_
6499	712305333	Existing Customer	59	M	0	Graduate	Single	$60K - 80K	Blue	
3609	720717108	Existing Customer	64	M	1	Uneducated	Single	Unknown	Blue	
3015	778447908	Existing Customer	51	M	2	College	Married	$80K - 120K	Blue	
3002	789125733	Existing Customer	38	F	4	Doctorate	Married	Less than $40K	Blue	
7232	710410158	Attrited Customer	49	M	2	Graduate	Unknown	$80K - 120K	Blue	

5 rows × 21 columns

- Display sample of 5 random rows using smaple() function of pandas

3. **Check the shape of the data (number of rows and columns).**

```
In [3]: df.shape

Out[3]: (10127, 21)
```

- Shape function of pandas gives size of table(rows*columns)

4. **Check the percentage of missing values in each column of the data frame.**

```
In [4]: (df.isnull().sum()/df.shape[0])*100

Out[4]: CLIENTNUM                 0.0
        Attrition_Flag            0.0
        Customer_Age              0.0
        Gender                    0.0
        Dependent_count           0.0
        Education_Level           0.0
        Marital_Status            0.0
        Income_Category           0.0
        Card_Category             0.0
        Months_on_book            0.0
        Total_Relationship_Count  0.0
        Months_Inactive_12_mon    0.0
        Contacts_Count_12_mon     0.0
        Credit_Limit              0.0
        Total_Revolving_Bal       0.0
        Avg_Open_To_Buy           0.0
        Total_Amt_Chng_Q4_Q1      0.0
        Total_Trans_Amt           0.0
        Total_Trans_Ct            0.0
        Total_Ct_Chng_Q4_Q1       0.0
        Avg_Utilization_Ratio     0.0
        dtype: float64
```

- Used isnull() function to find missing values
- There are no missing values present

5. **Check if there are any duplicate rows.**

```
In [5]: df.duplicated().sum()

Out[5]: 0
```

- Used duplicated() function to find any duplicate rows
- There are no duplicate rows present in database

6. **Check the distribution of the Customer_Age column. Check the basic statistics like mean, median, skewness, kurtosis, and standard deviation of the age column.**

```
In [6]: sb.distplot(df["Customer_Age"])
        mp.title("Distribution of the Customer_Age")
        mp.show()
```

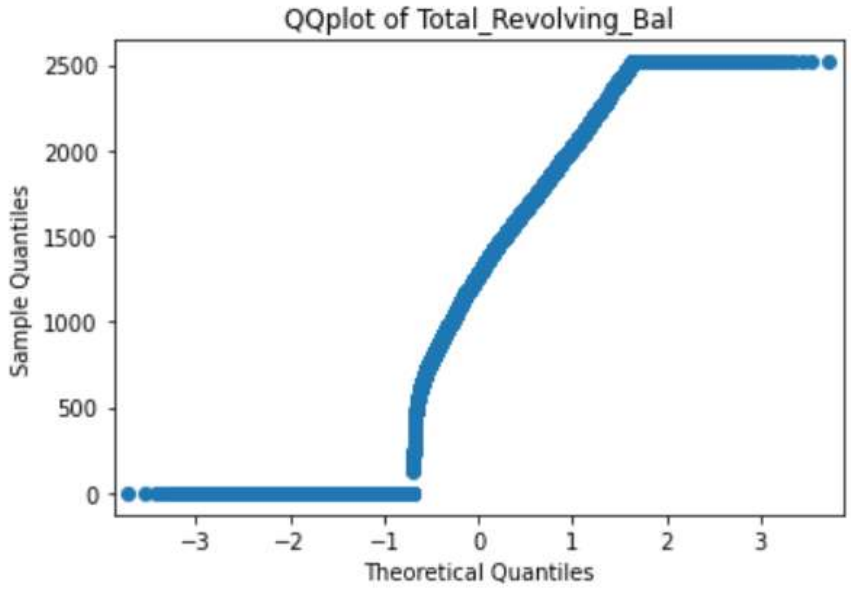

```
In [7]: df["Customer_Age"].mean()
Out[7]: 46.32596030413745

In [8]: df["Customer_Age"].median()
Out[8]: 46.0

In [9]: df["Customer_Age"].skew()
Out[9]: -0.033605016317173456

In [10]: df["Customer_Age"].kurtosis()
Out[10]: -0.2886199152745088

In [11]: df["Customer_Age"].std()
Out[11]: 8.016814032549084
```

- Used distplot() to plot distribution of Customer Age

- We can see that skewness is somewhat symmetric
- 46.33 is the mean customer age
- 46 is median customer age
- Skewness is -0.033 which is close to symmetry
- Kurtosis is -0.289 which means it is lightly tailed.
- Standard Deviation of 8.016 differ from mean by age of customer

7. Plot a QQ-plot of Total_Revolving_Bal

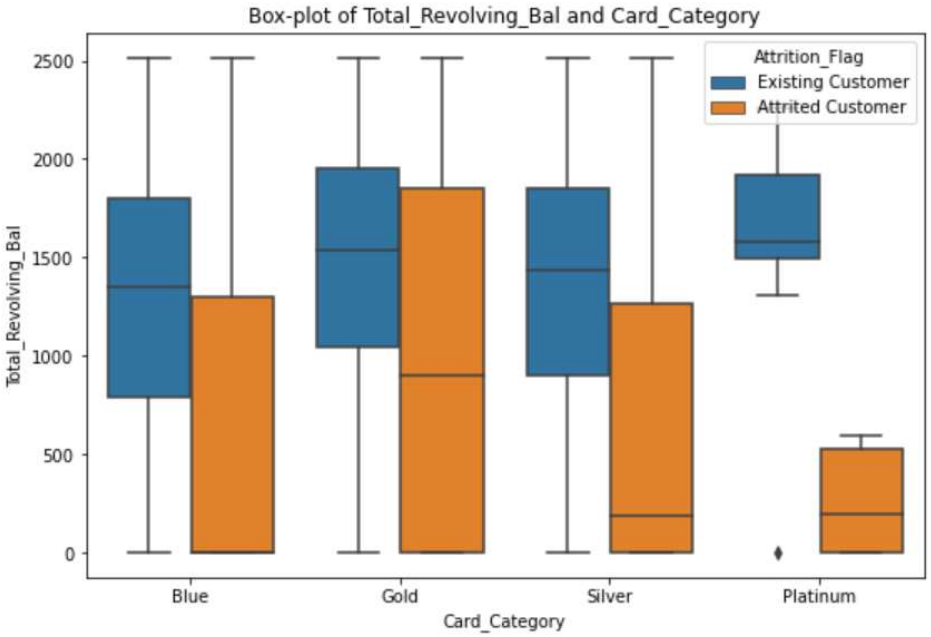
```
In [12]: import statsmodels.api as sm
         sm.qqplot(df["Total_Revolving_Bal"])
         mp.title("QQplot of Total_Revolving_Bal")
         mp.show()
```



- Plotted QQplot for Total_Revolving_bal using statsmodels.
- A QQplot is a plot of the two quantiles which ranges between (-3 to 3).

8. Plot a Boxplot of Total_Revolving_Bal and Card_Category by characterizing with Attrition_Flag.

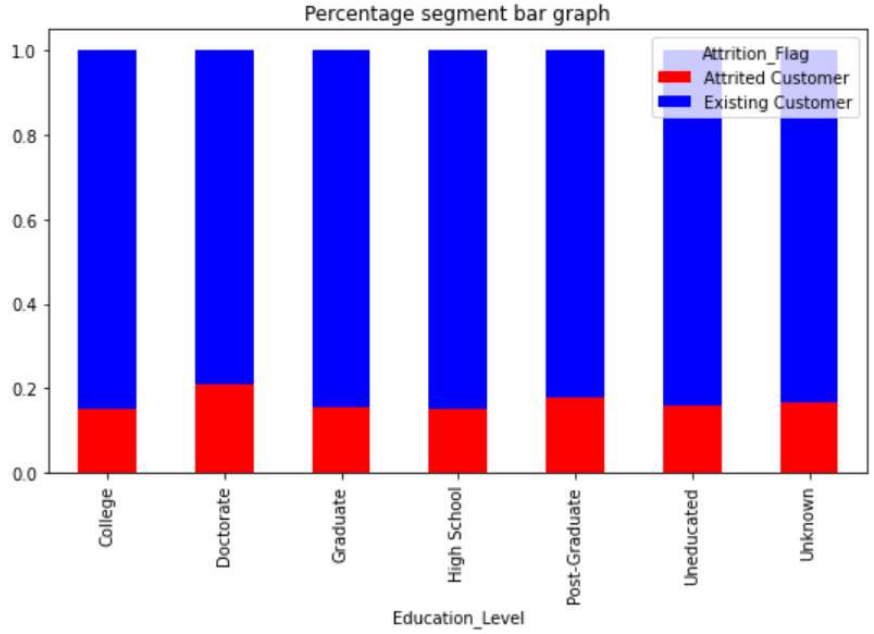
```
In [13]: fig = mp.figure(figsize=(9,6))
sb.boxplot(x=df.Card_Category, y=df.Total_Revolving_Bal,hue=df.Attrition_Flag)
mp.title("Box-plot of Total_Revolving_Bal and Card_Category")
mp.show()
```



- Plotted a Boxplot Card_Category vs Total_Revolvong_bal with Attrition as hue
- We can see boxplot means are different for Card_Category.
- Total_Revolving_Bal is more for Attrited Customers compare to Existing Customers.

9. **Plot a percentage segment bar graph between Education_Level and Attrition_Flag of the customers.**

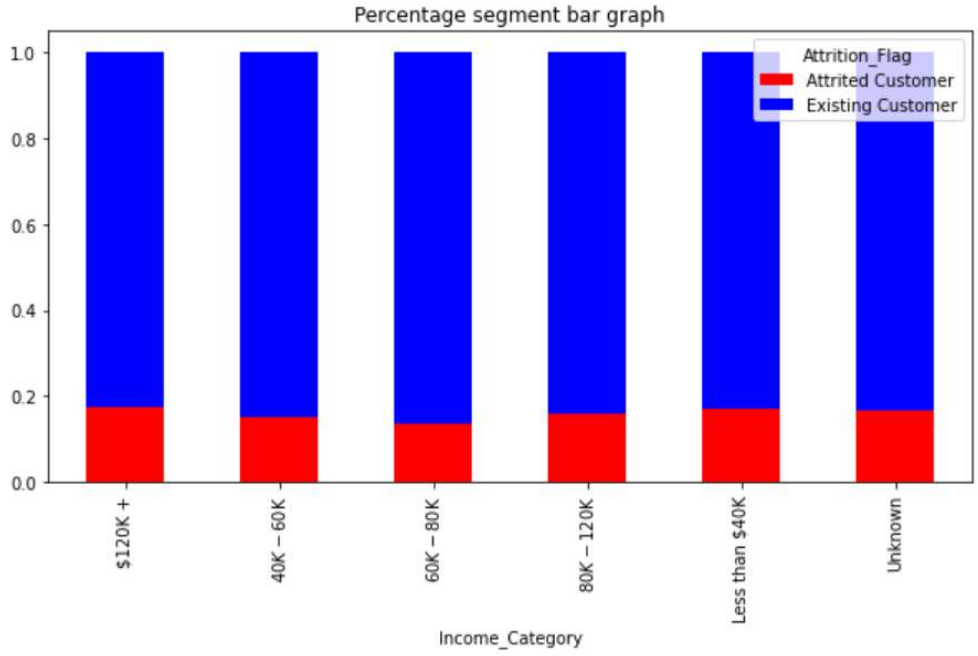
```
In [32]: data=df.groupby(by=df.Education_Level)["Attrition_Flag"].value_counts(1).unstack()
         #print(data)
         data.plot(kind='bar',stacked=True,figsize=(9,5),legend=True,color=['red','blue'])
         mp.title("Percentage segment bar graph")
         mp.show()
```



- Plotted percentage segment bar graph between Education_Level and Attrition_Flag
- We can Existing Customers has more percentage of Education in each education level as compare to Attrited Customer

10. Plot a percentage segment bar graph between Income_Category and Attrition_Flag of the customers.

```
In [34]: data=df.groupby(by=df.Income_Category)["Attrition_Flag"].value_counts(1).unstack()
         data.plot(kind='bar',stacked=True,figsize=(10,5),legend=True,color=['red','blue'])
         mp.title("Percentage segment bar graph")
         mp.show()
```



- Plotted percentage segment bar graph between Income_Category and Attrition_Flag
- We can Existing Customers has more percentage Income in each category as compare to Attrited Customer

11. Drop CLIENTNUM column.Make a sub data frame which consists of all the numerical columns(i.e.int64,float64) along with the Attrition_Flag column. Plot a clear heatmap to view the correlation using seaborn.

```
In [16]: df.drop("CLIENTNUM",axis=1,inplace=True)
```

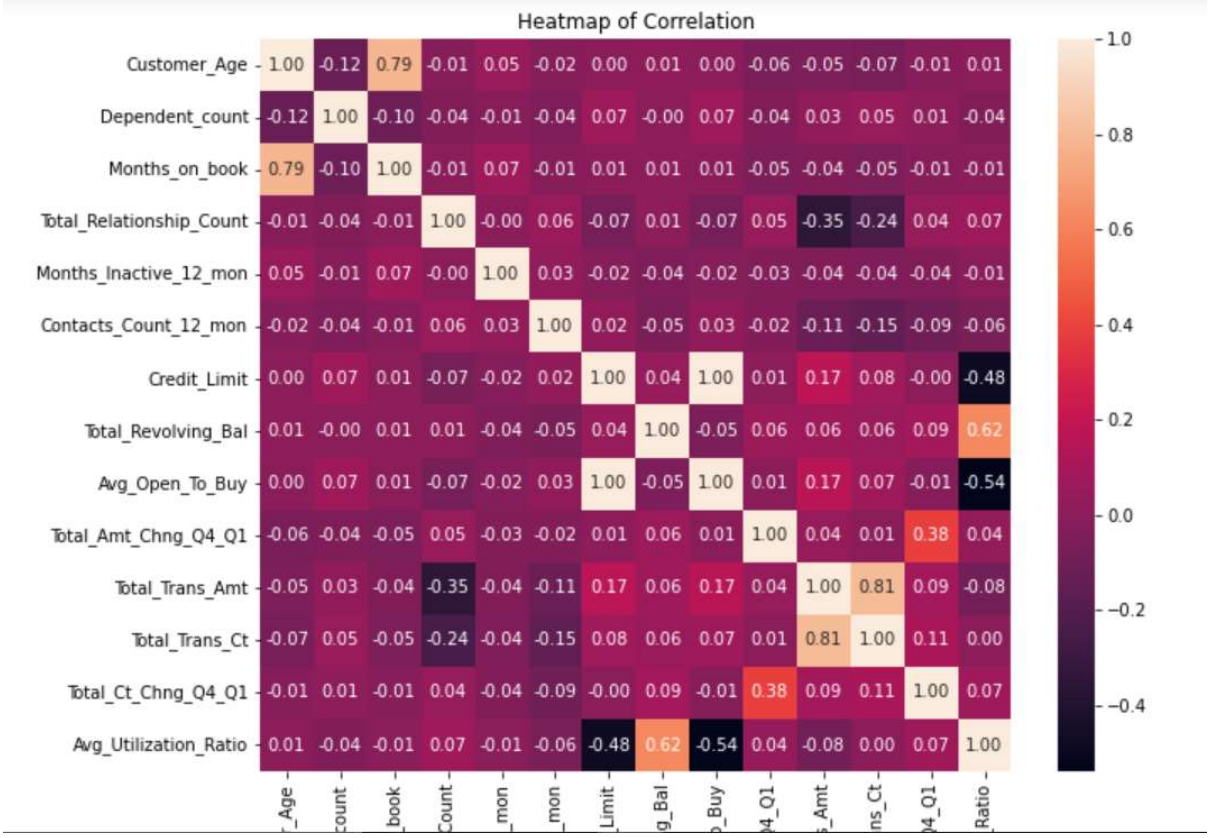
```
In [17]: df.shape
Out[17]: (10127, 20)
```

```
In [18]: sub_df = df.select_dtypes(include=[np.number])
```

```
In [19]: sub_df=sub_df.join(df["Attrition_Flag"])
         sub_df.info()
         sub_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Customer_Age              10127 non-null  int64
 1   Dependent_count           10127 non-null  int64
 2   Months_on_book            10127 non-null  int64
 3   Total_Relationship_Count  10127 non-null  int64
 4   Months_Inactive_12_mon    10127 non-null  int64
 5   Contacts_Count_12_mon     10127 non-null  int64
 6   Credit_Limit              10127 non-null  float64
 7   Total_Revolving_Bal       10127 non-null  int64
 8   Avg_Open_To_Buy           10127 non-null  float64
 9   Total_Amt_Chng_Q4_Q1      10127 non-null  float64
 10  Total_Trans_Amt           10127 non-null  int64
 11  Total_Trans_Ct            10127 non-null  int64
 12  Total_Ct_Chng_Q4_Q1       10127 non-null  float64
 13  Avg_Utilization_Ratio     10127 non-null  float64
 14  Attrition_Flag            10127 non-null  object
dtypes: float64(5), int64(9), object(1)
memory usage: 1.2+ MB
```

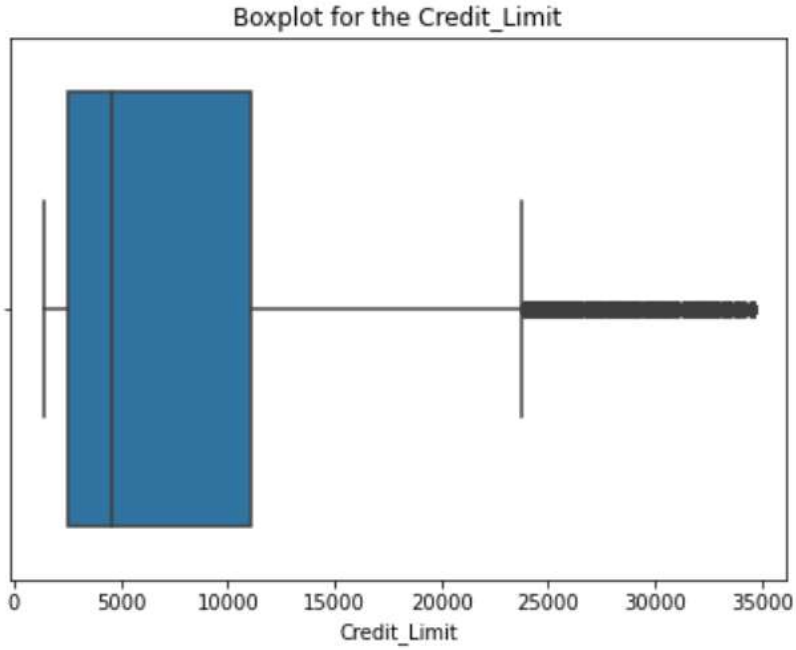
```
In [30]: figure=mp.figure(figsize=(10,8))
         sb.heatmap(sub_df.corr(),annot=True,fmt=".2f")
         mp.title("Heatmap of Correlation")
         mp.show()
```



- Dropped the CLIENTNUM column from dataframe permanently using drop and inplace=True function
- Created a sub dataframe which contains all numeric datatype columns
- Added column Attrition_Flag using join to table.
- Plotted heatmap plot using seaborn which gives correlation among columns.
- In heatmap we can see that Avg_open_to_buy is completely correlated with Credit_limit.
- Whereas Avg_Utilization_ratio is negatively correlated to Avg_open_to_buy the most in this heatmap.

12. Plot a boxplot for the Credit_Limit column and check if it contains any outlier or not.

In [21]:
```
figure=mp.figure(figsize=(7,5))
sb.boxplot(x=df.Credit_Limit)
mp.title("Boxplot for the Credit_Limit")
mp.show()
```



In [26]:
```
Q1 = df["Credit_Limit"].quantile(0.25)
Q3 =  df["Credit_Limit"].quantile(0.75)
IQR = Q3-Q1
s= df[(df.Credit_Limit<(Q1-1.5*IQR)) | (df.Credit_Limit> (Q3+1.5*IQR))]["Credit_Limit"]
print(s.count())
```

```
984
```

- Plotted a boxplot for Credit_limit using seaborn
- We can see that median is about 4500
- Highest and Lowest limits are about 24000 and 2000 resp.
- There are many outliers toward higher side which are 984 by count

13. Map the Attrition_Flag values to 0 and 1(i.e., Existing Customer=0 and Attrited Customer=1. Standardize the columns.

In [23]:
```
dummy=pd.get_dummies(df.Attrition_Flag)
dummy
```

Out[23]:

	Attrited Customer	Existing Customer
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1
...	...	...
10122	0	1
10123	1	0
10124	1	0
10125	1	0
10126	1	0

10127 rows × 2 columns

In [36]:
```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
standardized_dummy = scaler.fit_transform(dummy)
print(standardized_dummy)
```

```
[[-0.4375063   0.4375063 ]
 [-0.4375063   0.4375063 ]
 [-0.4375063   0.4375063 ]
 ...
 [ 2.28568136 -2.28568136]
 [ 2.28568136 -2.28568136]
 [ 2.28568136 -2.28568136]]
```

- Created Dummy variables for Attrition_Flag which as two distinct Categorical values.
- Attrited and Existing Customer are the two dummy variables created and assigned with 0s and 1s.
- Dummy variables can be standardize using StandardScaler from preprocessing library.

Thank You