# **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.

# **Solutions:**

1. **Code:**

**import numpy as np** # numeric calculations

**import pandas as pd** # statistical analysis

**import seaborn as sns** # correlation, heat map

**import matplotlib.pyplot as plt** # graphs

**%matplotlib inline**

**sns.set(color\_codes =True)** # sets the default color palette for seaborn

**from warnings import filterwarnings**

**filterwarnings('ignore')**

**import plotly.express as px**

**from sklearn.preprocessing import StandardScaler**

Here we have imported the essential libraries to access and extract the values in the data set.

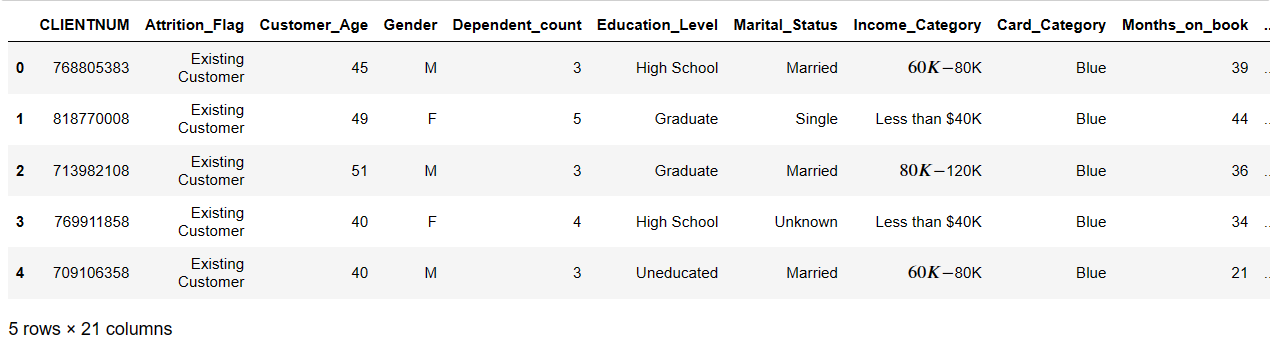
1. **Code:**

**df = pd.read\_csv('BankChurners.csv')**

Here, we are accessing the values of a data set of the given problem.

1. **Code:**

**df.head(5)**

**Output:**

Here, we are displaying the 5 rows from the beginning of the data set.

1. **Code:**

# defines how many rows and columns exists in data

**df.shape**

**Output:**



Here, the given data set contains 10127 rows and 21 columns.

1. **Code:**

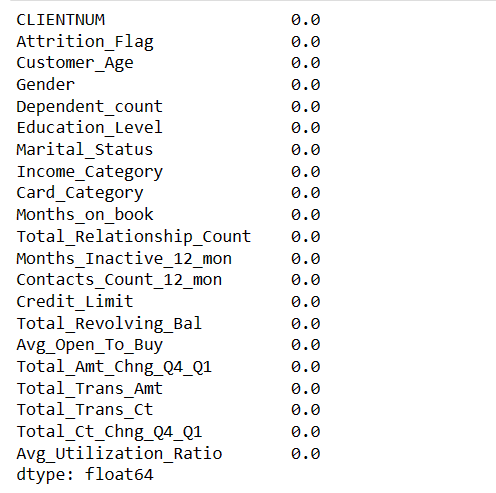
**df.isnull().any(axis=1).value\_counts()**

**missing\_Values\_percentage = df.isnull().mean() \* 100**

**missing\_Values\_percentage**

**Output:**

****

****

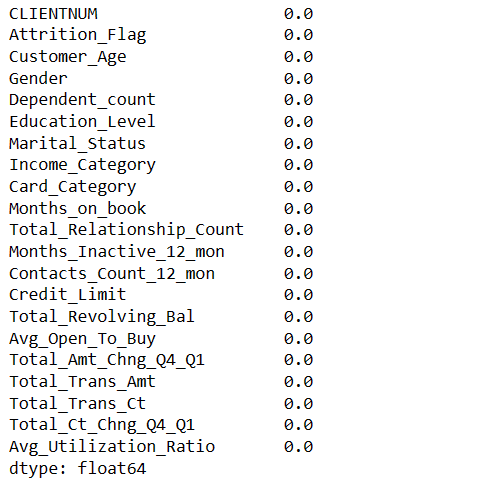
Firstly, we have checked if there exist any missing values in the data set. The output shows that there are no missing values. The False 10127 indicates a Boolean value that has no missing values.

We have calculated the percentage of each column of the data frame by an average of non-missing values and missing values and multiplied by 100.

1. **Code:**

**df[df.duplicated()].sum()**

**Output:**

****

Therefore, there are no identical rows within the data set. Here we used df[df. duplicated()].sum() to find and sum up the values int the columns for rows that are duplicated within a data frame(df).

Therefore, it returns the Boolean value, 1 (duplicate rows exists) and 0 (no duplicate rows).

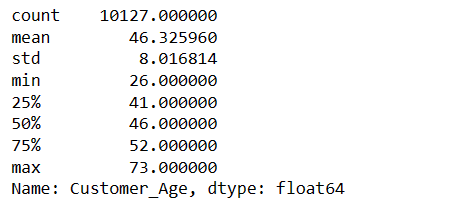
1. **Code:**

**# df.describe() -**

**# used to check distribution of all cloumns in the data**

**df['Customer\_Age'].describe()**

**Output:**

****

We need to find the statistical measures of the column ‘Customer\_ Age’, so we have used ‘describe’ keyword. Therefore, the distribution is

* Mean = 46.32
* Median = 46.00
* Standard deviation = 8.01

Hence, the arithmetic average of all values in the column is 46.32, it describes that the ages of the customers in data set are approximately 46.32 years. The 50 percentile is also known as median; this indicates that half of customers are older than 46 years and half are younger.

The standard deviation describes the spread or dispersion of the ages in the ‘Customer\_Age’ column. A higher deviation means the ages are more spread out from the mean. Here, standard deviation of 8.01 suggests that the ages vary from the mean by about 8.01 years on average. Therefore, there is a wide range of ages in dataset, with some customers being significantly older or younger than the average.

1. **Code:**

**(a)** # Pie charts

**plt.figure(figsize = (15, 6)) # setting the figure size**

**plt.subplot(1, 2, 1) # grid size 1x2**

**df['Income\_Category'].value\_counts().plot.pie(autopct='%1.1f%%') # for percentage labels to pie charts**

**plt.xlabel('Income Category') # labelling the x-axis**

**plt.title('Income Category Distribution Overview') #tilte for the pie chart 1**

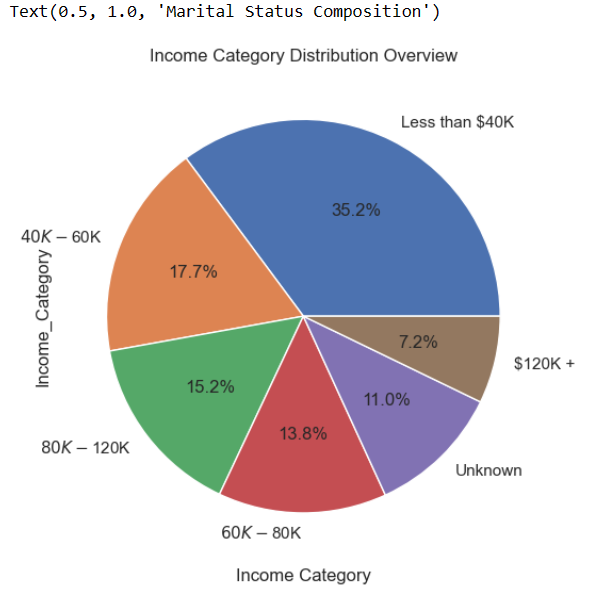
**plt.subplot(1, 2, 2) # grid size 1x2**

**df['Marital\_Status'].value\_counts().plot.pie(autopct='%1.1f%%')**

**plt.xlabel('Marital\_Status')**

**plt.title('Marital Status Composition') # tilte for the pie chart 2**

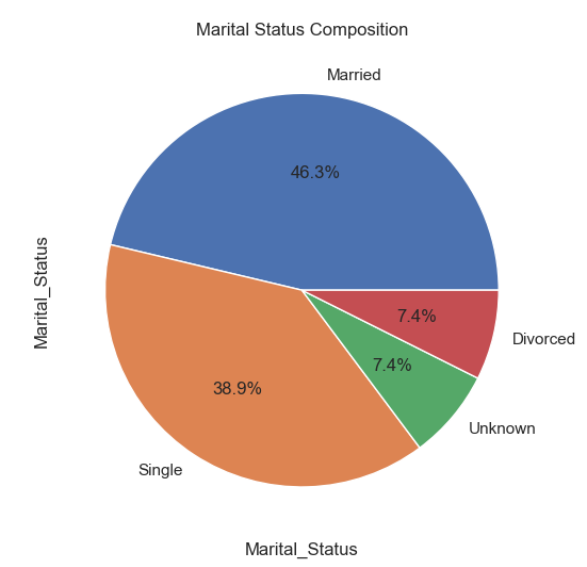
**Output:**

****

Here, we can see the pie chart of the Income Category column of the dataset. In the pie chart, there are different segments of categorical data of Income Category are shown. There are 6 slice points data of the Income Category. They are

* Less than $40K
* 40K – 60K
* 60K – 80K
* 80K – 120K
* $120K +
* Unknown

The data reveals a significant majority of customers, 36.2%, have income below $40,000. And only a small minority of customers, 7.2%, report incomes exceeding $120,000. The majority of customers fall within the income range of $40,000 to $60,000. While targeting customers with lower income for better services and credit card offerings is a valid strategy. This suggests an opportunity to provide improved services and diverse credit card options to customers in various income brackets as part of a customer retention strategy.



Here, is a pie chart representing the marital status distribution of customers in the data set. A significant majority of customers, comprising 46.3% of the total, were married. In contrast, a smaller percentage, specifically 7.4% of customers reported being divorced. Additionally, 38.9% of customers were categorized as not married, while martial status of the remaining 7.4% remained unknown. Given the higher number of married customers, who often have more dependencies, there is an opportunity to enhance services for this segment to reduce the likelihood of churn.

**(b)** # Box Plots

**plt.figure(figsize = (15, 6))**

**plt.subplot(2, 2, 1)**

**sns.boxplot(data = df, x = 'Customer\_Age', y = 'Education\_Level')**

**plt.title('An Overview of Age Distribution Based on Education Levels Among Customers')**

**plt.subplot(2, 2, 2)**

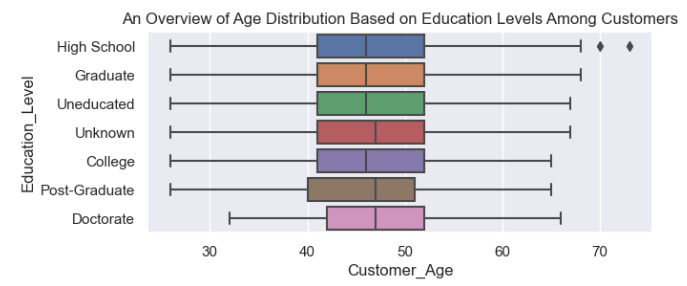
**sns.boxplot(data = df, x = 'Marital\_Status', y = 'Dependent\_count');**

**plt.title('Overview of Marital Status and the Count of Dependents')**

**plt.show();**

**Output:**

The box plot below shows the age distribution on education levels among customers. It also shows the variability or dispersion of data points present within a set. Here, the box plot typically includes a horizontal line within a box, which represents the median(middle) value of the given data.

****

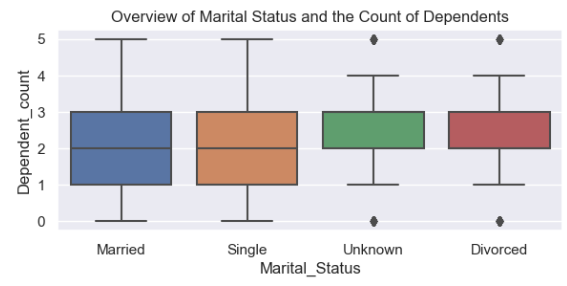
Each box plot represents the age distribution of customers within that specific education level.

The customer age distribution displays a broad range, with a significant number of customers both older and younger than the median age of 46. Moreover, a significant portion of customers have attained a high level of education. The median line within each box, and median age is almost the same for each education level among customers.

The box represents the interquartile range (IQR), which contains the middle 50% of the data. it extends from the first quartile (Q1) to the third quartile (Q3). The length of the box indicates the spread or variability of the data within each education level. And shows that it has a greater variability.

We can observe some lines extending from the box represent the range of data points that fall within a certain distance from the median. Outliers, they are present as individual data points beyond the lines.

There are some outliers that can be seen for education level ‘High School’. Therefore, in the given data set the ‘Education\_Level’, ‘Customer\_Age’ columns contain outliers.

****

The box plot above represents the relation between dependent count on marital status among customers.

We can also observe the variability or dispersion of data points present within each category. The box plot typically includes a horizontal line within a box, which represents the median or middle value of the given data. However, we can notice that in some box plots, there is no horizontal line inside the box. This absence of a horizontal line indicates a specific characteristic of the data distribution. It suggests that the median and the data distribution itself may not have a clear central tendency.

The distribution of dependent counts varies significantly among different marital status categories. Through the box plot, some categories have symmetric distribution while others have dispersed or asymmetric distribution.

The dependent count on marital status reveals a notable range, with a significant number of customers of both married and single-status groups. The median value for dependency typically falls within the range of more than 2 and less than 2, indicating that on average customers tend to have between 2 to 3 dependents.

We can observe some lines extending from the box represent the range of data points that fall within a certain distance from the median. Outliers, they are present as individual data points beyond the lines.

There are some outliers that can be seen for ‘unknown’ and ‘divorced’ groups of marital status. Therefore, in the given data set the ‘Marital\_Status’, and ‘dependent\_count’ columns contain outliers.

1. **Code:**

**# Box Plots**

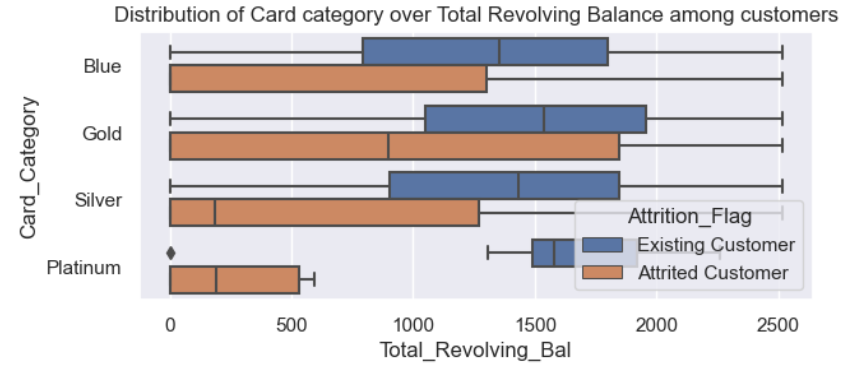
**plt.figure(figsize = (15, 6))**

**plt.subplot(2, 2, 1)**

**sns.boxplot(data = df, x = 'Total\_Revolving\_Bal', hue='Attrition\_Flag', y = 'Card\_Category')**

**plt.show()**

**Output:**



Here we have used a box plot to depict the distributions of Card\_Category and Total\_Revolving\_Bal characterizing with Attrition\_Flag groups of numeric data. In the Box, limits indicate the range of the central 50% of the data, with a central line marking the median value. Lines extend from each box to capture the range of the remaining data, with dots placed past the line edges to indicate outliers.

From the above plot, we can see that the total revolving balance of existing customers increased by 1200 of the card categories ‘Blue’ to about 1500 for the card category ‘Gold’. Whereas the total revolving balance for the attrited customers decreased by 900 for the card category ‘Gold’ to about 50 for the card category ‘Silver’.

1. **Code:**

**education\_vs\_attrition = pd.crosstab(df['Education\_Level'], df['Attrition\_Flag'], normalize='index')**

**education\_vs\_attrition.plot(kind='bar', stacked=True)**

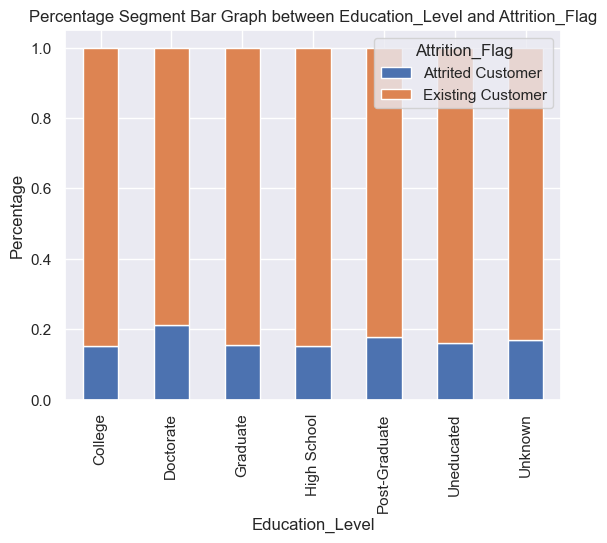
**plt.title('Percentage Segment Bar Graph between Education\_Level and Attrition\_Flag')**

**plt.xlabel('Education\_Level')**

**plt.ylabel('Percentage')**

**plt.show()**

**Output:**

****

Here, we can observe the percentage stacked bar graph between Education level and Attrition Flag. The x-axis represents the various categories or levels of the independent variable, ‘Education Level’. Whereas the y-axis represents the percentage composition of the dependent variable, ‘Attrition Flag’ within each category of the independent variable, ‘Education Level’.

Each bar on the above graph corresponds to a category of the Education Level. The height of each section of the bar represents the proportion or percentage of the Attrition Flag associated with that category. Within each bar, there are two stacked sections, the blue color represents the attrited customer, and the orange color, the existing customer, subgroups of the Attrition Flag.

The total height of each bar represents 100% or the entirely of the attrition flag for that category of the education level. The existing customer subgroup has larger sections in all the bars compared to attrited customer subgroup. The patterns among the categories of education level of customer subgroups exhibit subtle variations, with only slight differences observed. Most existing customers have educational backgrounds across all categories, while there is also a notable number of existing customers lack formal education. As a result, the percentage among existing customers in education is nearly 80%, while attrited customers constitute a considerably lower percentage, approximately 20%.

1. **Code:**

**Income\_vs\_attrition = pd.crosstab(df['Income\_Category'], df['Attrition\_Flag'], normalize='index')**

**Income\_vs\_attrition.plot(kind='bar', stacked=True)**

**plt.title('Percentage Segment Bar Graph between Income\_Category and Attrition\_Flag')**

**plt.xlabel('Income\_Category')**

**plt.ylabel('Percentage')**

**plt.show()**

** Output:**

Here, we can observe the percentage stacked bar graph between Income Category and Attrition Flag. The x-axis represents the various categories or levels of the independent variable, ‘Income Category. Whereas the y-axis represents the percentage composition of the dependent variable, ‘Attrition Flag’ within each category of the independent variable, ‘Income Category’.

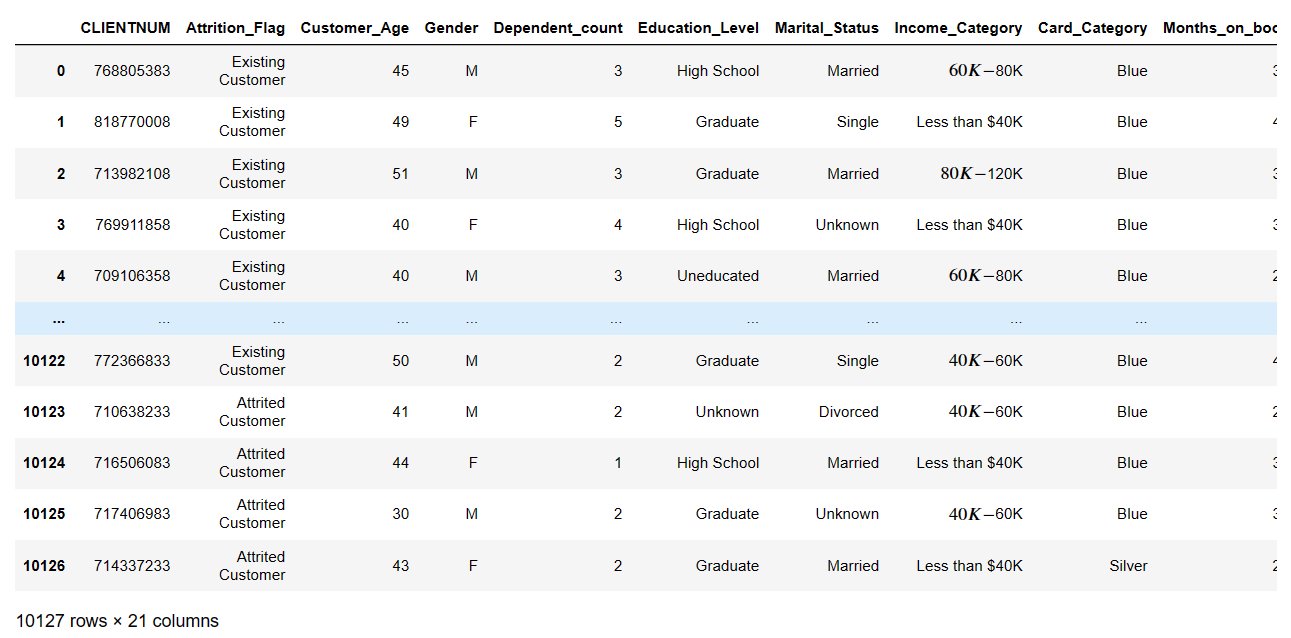
Each bar on the above graph corresponds to a category of the Income Category. The height of each section of the bar represents the proportion or percentage of the Attrition Flag associated with that category. Within each bar, there are two stacked sections, the blue color represents the attrited customer, and the orange color, the existing customer, subgroups of the Attrition Flag.

The total height of each bar represents 100% or the entirely of the attrition flag for that category of the income category. The existing customer subgroup has larger sections in all the bars compared to attrited customer subgroup. The patterns among the categories of income level of customer subgroups exhibit subtle variations, with only slight differences observed. The percentage of all attrited customers in all categories of income has less 20%.

1. **Code:**

**df1 =df**

**df1**



Here we are copying the values of data Frame (df) into a new data Frame (df1), to find the correlation of the columns in the data Frame.

**df1.drop(df1.columns[0], axis = 1, inplace = True)**

**#df1**

**numerical\_df1 = df1.select\_dtypes(include = ['int64', 'float64'])**

**#numerical\_df1**

**numerical\_df1.insert(0, "Attrition\_Flag", df1['Attrition\_Flag'])**

**#numerical\_df1**

**numerical\_df2 = numerical\_df1.select\_dtypes(include = ['int64', 'float64','object'])**

**#numerical\_df2**

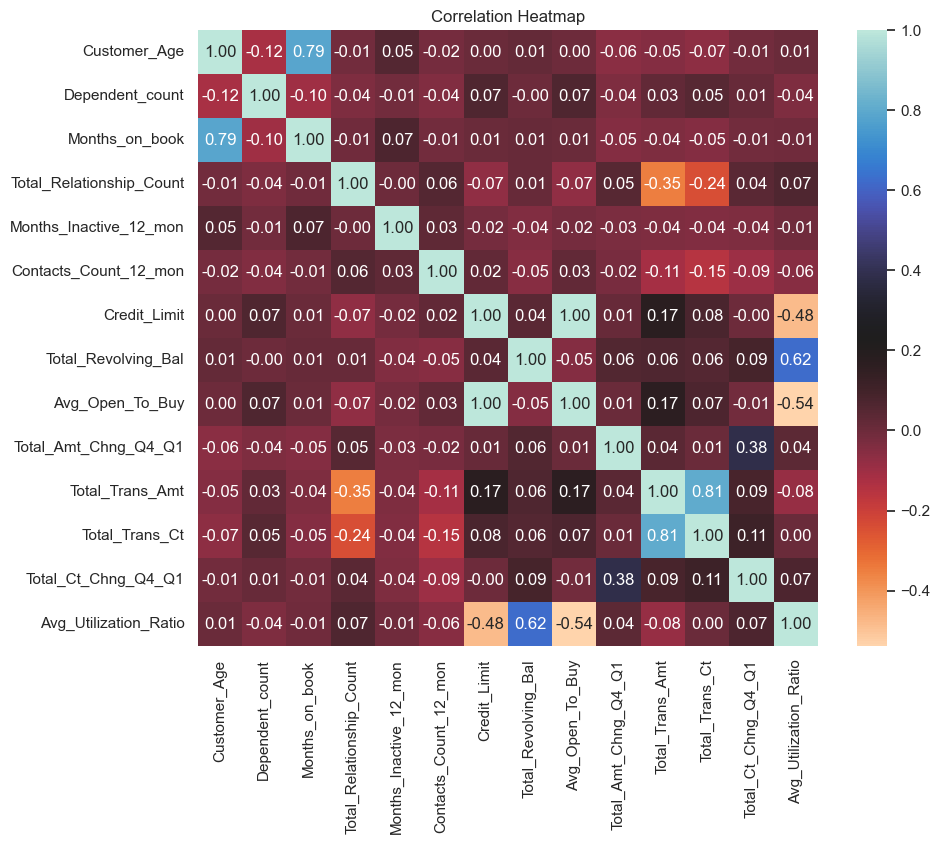
**correlation\_matrix = numerical\_df2.corr()**

**plt.figure(figsize=(10, 8))**

**sns.heatmap(correlation\_matrix, annot=True, cmap='icefire\_r', fmt=".2f")**

**plt.title('Correlation Heatmap')**

**plt.show()**

 **Output:**

For a visual representation of the correlation coefficients between pairs of variables in a given dataset, we generate a correlation heatmap. It typically uses color to indicate the strength and direction of correlations. These correlation maps help in deciding which things are important for predictions or analysis.

Colors shifting from black to blue are interpreted as positive correlations. Here, blue represents strong positive correlations, while dark blue indicates moderately positive correlations.

Colors shifting from dark violet to light orange are associated with negative correlations. Here, dark orange represents strong negative correlations, while light orange indicates moderately negative correlations.

In the heatmap, the diagonal line from the top-left to bottom-right shows how each thing relates to itself. It is always a correlation of 1.

We can also figure out outliers in a heatmap if something unusual data points make correlations stronger or weaker than they actually are.

Let’s see the strength of correlations in the heatmap. The neutral or moderate blue color in the correlation heatmap indicates a lack of a strong positive correlation between variables. We can observe a neutral or moderate blue between the total revolving balance and average utilization ratio in the correlation heatmap. This implies that there is no strong positive correlation between these two variables.

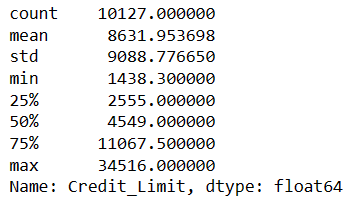
There is a strong positive correlation between credit limit and average open-to-buy variables. This indicates a consistent and significant relationship where customers with higher credit limits tend to have larger amounts of unused credit available, reflecting their financial capacity and credit utilization patterns.

In the heatmap, there is a notable perfect positive correlation between the “Age” and “Months on Book” variables. This correlation suggests that older customers might have longer credit histories with the bank.

1. **Code:**

**df['Credit\_Limit'].describe()**

Identifying and dealing with outliers is an essential part of the data analytics process. The outliers are the extreme values within the dataset. That means the outlier data points vary greatly from the expected values – either being much larger or significantly smaller.



Here we have used .describe() to generate some summary statistics which helps in determining whether or not the dataset has outliers. As we can see the ‘Credit\_Limit’ column has outliers. The max credit limit is 34,516 while its mean is 8,631. The mean is sensitive to outliers, but the fact the mean is small compared to the max value indicates the max value is an outlier.

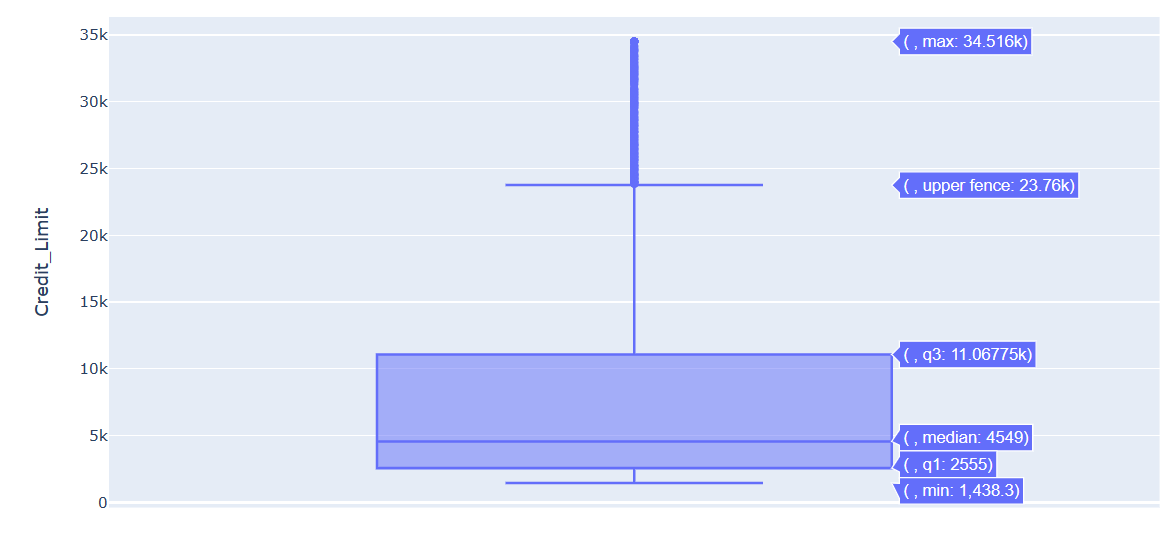
To confirm and analyze these outliers, we need to explore additional methods to identify and handle the outliers.

**#create a box plot**

**fig = px.box(df, y="Credit\_Limit")**

**fig.show()**

**Output:**

****

**outliers**

The above Box plot allows us to identify the univariate outliers or outliers for one variable. We can also observe minimum and maximum values, the median, and the interquartile range of the data above.

The bottom of the box is the 25% percentile and the top is the 75% percentile value of the data. So, essentially the box represents the middle 50% of all the data points which represents the core region where the data is situated. The height of the boxplot is also called the Inter Quartile Range (IQR), which mathematically is the difference between the 75th and 25th percentile values of the data.

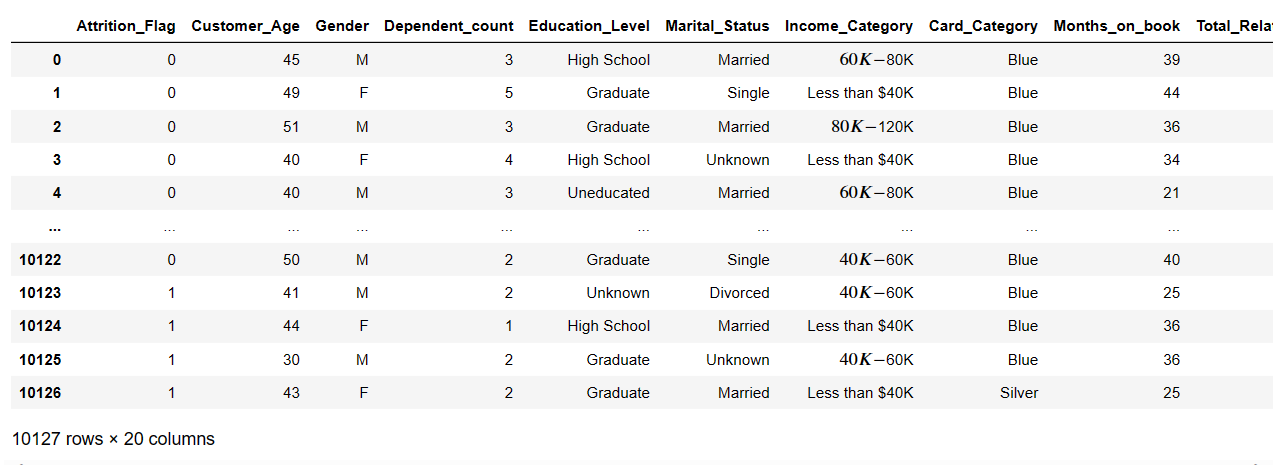
The points that lie outside the fences, that is (1.5 x IQR) in both directions are generally considered as outliers. There are lot of outliers in the column of the data set. Above the box and upper fence are some points showing outliers.

1. **Code:**

**df['Attrition\_Flag'] = df['Attrition\_Flag'].map({'Existing Customer': 0, 'Attrited Customer': 1})**

**df**

**Output:**



Here, we are modifying the ‘Attrition\_Flag’ column in the data frame (df). It is mapping the values in this column to new values (object into int64) i.e., categorical label into numerical values for analysis or modeling.

* Existing Customer is mapped to 0
* Attrited Customer is mapped to 1.

We are mapping the columns to find the standardization values of columns in the data Frame.

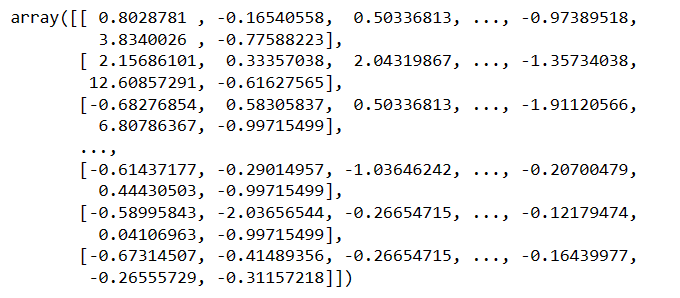
**scaler = StandardScaler()**

**numerical\_df4 = df.select\_dtypes(include = ['int64', 'float64'])**

**numerical\_df4 = scaler.fit\_transform(numerical\_df4)**

**numerical\_df4**

**output:**

****Here ‘StandardScaler’ object is used to standardize the numerical features in the data Frame. Standardization typically involves transforming data so that it has a mean of 0 and a standard deviation of 1.

Further, it calculates the mean and standard deviation of each numerical column in numerical\_df4 and then scales the data accordingly.

In normalization, the data values will shrink down to a specific range which is from 0 to 1. Whereas in standardization, there are no specific boundaries for the data to shrink down to. While seeing the data in an order being influenced by the outliers and putting the model at risk. We commonly normalise or standardize the data.