# Introduction

In our data analysis project, the target variable we are focusing on is 'balance.' The 'balance' column represents the financial standing of individuals in the dataset. By designating 'balance' as our target variable, we are likely interested in understanding the factors that influence or correlate with individuals' financial balances. This choice directs our analysis towards exploring patterns, relationships, and trends related to demographic features such as 'age,' 'marital status,' 'education,' and 'job' in the context of financial well-being.

Our initial code includes the importation of essential Python libraries—pandas, matplotlib.pyplot, and seaborn—to efficiently handle, analyze, and visualize the dataset. The dataset, as revealed by the code snippet df.shape = (45215, 17), comprises 45,215 rows and 17 columns. As we progress with the analysis, we will delve deeper into the relationships between the selected features ('age,' 'marital,' 'education,' 'job,' and 'balance') and the target variable 'balance.' The insights gained from this exploration, combined with the comprehensive dataset dimensions, can provide valuable information for understanding the dynamics of financial balances within the dataset and may contribute to making informed decisions or predictions based on the available data.

Dataset link= <https://www.kaggle.com/datasets/thedevastator/bank-term-deposit-predictions?select=train.csv>

# Data Cleaning

During the data cleaning phase, we systematically addressed missing values in our dataset. An initial inspection using df.isnull().sum() revealed that the 'day,' 'campaign,' and 'pdays' columns had 3, 4, and 6 missing values, respectively. To handle this, we filled the missing values in the 'pdays' column with the mean of the existing values. Subsequently, polynomial interpolation of order 2 was employed to estimate missing values in the 'campaign' column, ensuring a smooth continuity based on the existing data trends. For the 'day' column, missing values were replaced with the value 6. After implementing these strategies, a final check using df.isnull().sum() indicated that the dataset is now free of null values. we systematically addressed missing values in our dataset and further managed duplicate entries. An initial check using df.duplicated().sum() revealed that there were 4 duplicated rows. To ensure data integrity, these duplicate entries were removed using the df=df.drop\_duplicates() statement. Subsequently, we verified that there were no remaining duplicates by re-evaluating df.duplicated().sum(). With duplicates successfully eliminated, we reset the index of the DataFrame using df=df.reset\_index(). This step ensures a consistent and ordered representation of our data, and it is particularly relevant after removing duplicate rows. The resulting dataset is now free of both missing values and duplicate entries, making it well-prepared for subsequent analysis. This comprehensive cleaning process ensures that our dataset, comprising features such as 'age,' 'job,' 'marital,' 'education,' 'default,' 'balance,' 'housing,' 'loan,' 'contact,' 'day,' 'month,' 'duration,' 'campaign,' 'pdays,' 'previous,' 'poutcome,' and 'balance,' is ready for further analysis.

## Data cleaning of age

In addition to dropping the 'index' column for a streamlined representation of our dataset using df=df.drop('index', axis=1), we introduced a new categorical variable named 'age\_group' to capture age ranges. Employing the pd.cut() function, we categorized individuals into ten age bins, each spanning a decade from 0 to 100, with corresponding labels such as '0-10,' '10-20,' '20-30,' and so on. This categorization offers a nuanced perspective on age-related patterns in our analysis. To maintain clarity and avoid redundancy, we then dropped the original 'age' column using df=df.drop('age', axis=1). The resulting dataset now includes the 'age group' variable, allowing for a more granular exploration of age-specific trends while maintaining a concise and relevant set of features for our analysis.

## Splitting into numerical and categorical

In the given code snippet, we efficiently categorize columns in our DataFrame into two distinct types: numerical and categorical. The line numerical\_columns = df.select\_dtypes(include=['int64', 'float64']) identifies and extracts columns with numerical data types ('int64' and 'float64'), creating a DataFrame specifically tailored for numerical analyses. On the other hand, categorical\_columns = df.select\_dtypes(include=['object']) selects columns with 'object' data types, indicating categorical or text-based information. This separation facilitates a focused exploration of both numerical and categorical aspects of the dataset, enabling targeted analysis and visualization approaches for each data type.

## Summary statistic

The code generates a comprehensive statistical summary for the numerical columns in our dataset. The resulting DataFrame, named 'statistics,' includes key statistical measures for each numerical column. The 'Min' and 'Max' columns represent the minimum and maximum values, respectively, showcasing the range of each feature. The 'Range' column quantifies the spread between the maximum and minimum values. Additional measures include the 'Mean' (average), 'Median' (middle value), 'Variance' (spread of values around the mean), and 'Standard Deviation' (a measure of the dispersion of values).

For instance, the statistics table reveals insights for the 'balance' column, such as a minimum value of -8019, a maximum value of 102127, a range of 110146, a mean of 1362.27, a median of 448, a variance of approximately 9.27 million, and a standard deviation of 3044.77. Similar insights are provided for the 'day,' 'duration,' 'campaign,' 'pdays,' and 'previous' columns. This summary table serves as a valuable resource for understanding the distribution and central tendencies of numerical features in our dataset, aiding in the interpretation and exploration of the data.

## Displaying unique

The provided code snippet extracts unique values from specific categorical columns in our dataset, offering a glimpse into the diversity and categories within these features. For the 'job' column, the unique values include: 'management,' 'technician,' 'entrepreneur,' 'blue-collar,' 'unknown,' 'retired,' 'admin.,' 'services,' 'self-employed,' 'unemployed,' 'housemaid,' and 'student.' The 'marital' column comprises three unique values: 'married,' 'single,' and 'divorced.' In the 'education' column, the unique values are: 'tertiary,' 'secondary,' 'unknown,' and 'primary.' These unique value arrays provide a quick overview of the distinct categories present in the specified categorical columns. Such insights are crucial for understanding the diversity and distribution of categorical data, paving the way for effective data exploration and potential feature engineering in subsequent analyses.

## Displaying mode

The code calculates the mode for each categorical column in our dataset, yielding a DataFrame named 'mode\_results.' The output highlights the most frequently occurring values (modes) for columns such as 'job,' 'marital,' 'education,' 'default,' 'housing,' 'loan,' 'contact,' 'month,' 'poutcome,' and the target variable 'balance.' For instance, the mode for the 'job' column is 'blue-collar,' 'married' is the mode for 'marital,' and 'secondary' is the mode for 'education.' This information provides a concise summary of the predominant categories within each categorical variable, offering valuable insights into the dataset's characteristic features for subsequent analyses.

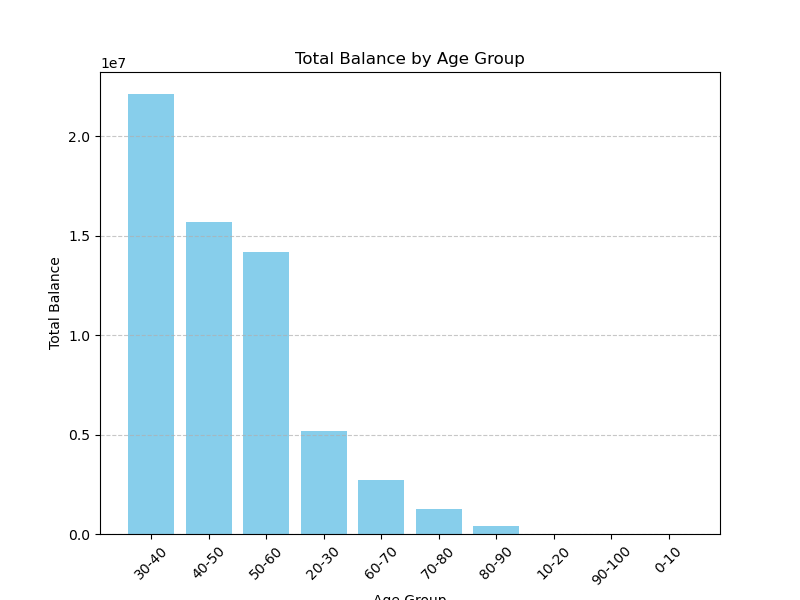
# Data Visualization and Explorations

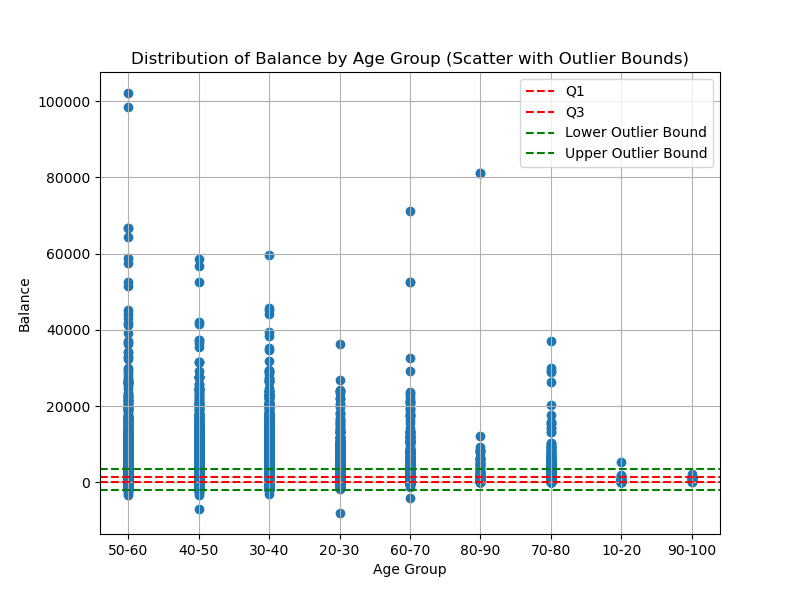
## By Age:-

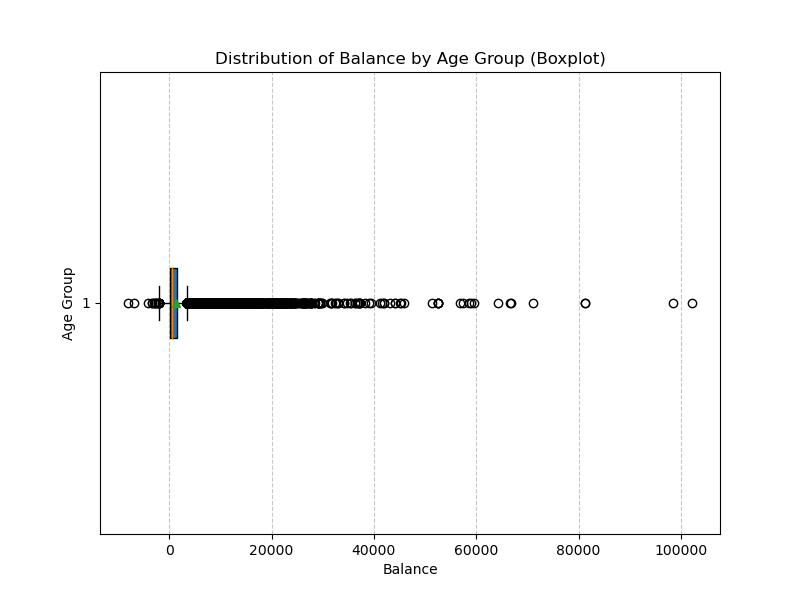
The provided code conducts a comprehensive exploration of the dataset, specifically focusing on the distribution of total balance across distinct age groups. The initial steps involve aggregating the total balance for each age group using the groupby function and subsequently sorting the resulting DataFrame, 'total\_balance\_by\_age\_group\_sorted,' in descending order. This DataFrame provides a clear representation of the distribution of total balances. To visually convey this distribution, a bar chart is generated using matplotlib, with age groups on the x-axis and total balance on the y-axis. The chart, created with labels, a title, and gridlines, offers a clear comparison of total balances across age groups. Additionally, to scrutinize the skewness and modality of the balance distribution, a boxplot is employed, providing insights into the presence of outliers. A scatter plot, incorporating horizontal lines representing quartiles and outlier bounds, aids in identifying and analyzing outliers within the balance distribution for each age group. Both the boxplot and scatter plot images are saved for future reference. These visualizations collectively offer a comprehensive view of the total balance distribution across various age groups, highlighting potential outliers and enabling a deeper understanding of the dataset's characteristics.

Additionally, short descriptions and interpretations are provided for each of the three charts. The bar chart illustrates that the age group '30-40' has the highest total balance, followed by '40-50' and '50-60.' The boxplot provides a clear view of the distribution of balance in each age group, with outliers marked for identification. The scatter plot highlights the distribution of balance in each age group, with horizontal lines representing quartiles and outlier bounds, aiding in the identification of potential outliers with extreme values.

Fig =





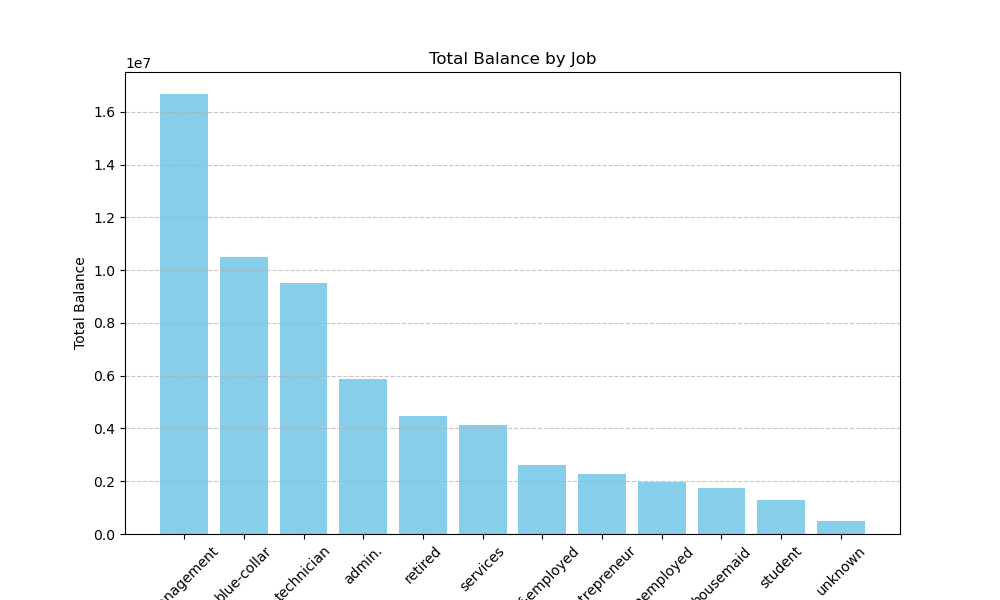


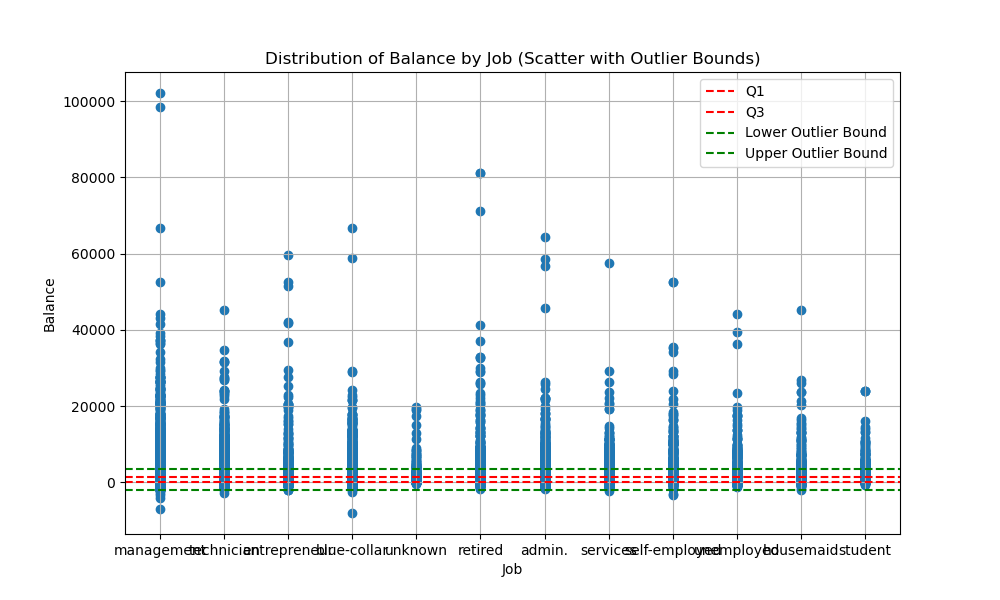
## By Job:-

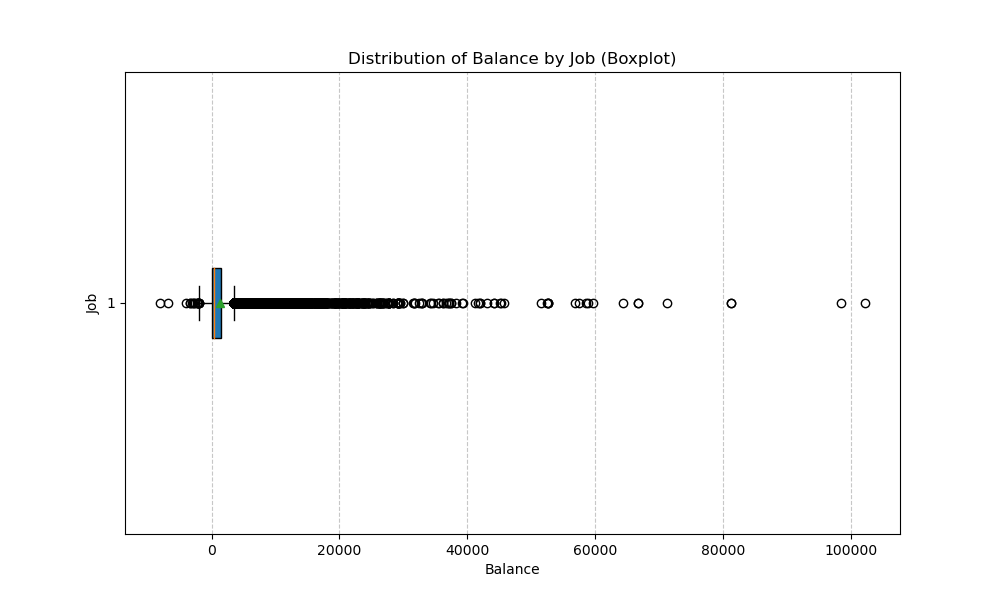
The provided code segments undertake a thorough examination of the total balance distribution across different job categories in the dataset. The initial steps involve aggregating the total balance for each job using the groupby function, resulting in the creation of the 'total\_balance\_by\_job\_sorted' DataFrame, which is then sorted in descending order based on total balances. To visually represent this distribution, a bar chart is generated using matplotlib, featuring job categories on the x-axis and total balance on the y-axis. The chart is created with labels, a title, and gridlines to enhance interpretability, offering a clear overview of total balances across various jobs. Subsequently, skewness and modality of the balance distribution are examined using a boxplot, providing insights into the presence of outliers within job categories. Additionally, a scatter plot with horizontal lines denoting quartiles and outlier bounds aids in the identification and analysis of outliers for each job category. These visualizations, saved as image files, collectively offer an in-depth understanding of how total balances are distributed across different job categories, helping to identify potential outliers and patterns within the dataset. Short descriptions and interpretations are provided for each of the three charts.

The bar chart illustrates that the job category 'management' has the highest total balance, followed by 'blue-collar' and 'technician.' The boxplot provides a clear view of the distribution of balance in each job category, with outliers marked for identification, and the means indicated by points in the boxes offering insights into central tendencies. The scatter plot highlights the distribution of balance in each job category, with horizontal lines representing quartiles and outlier bounds for the entire dataset, aiding in the identification of potential outliers with extreme values.

Fig:-





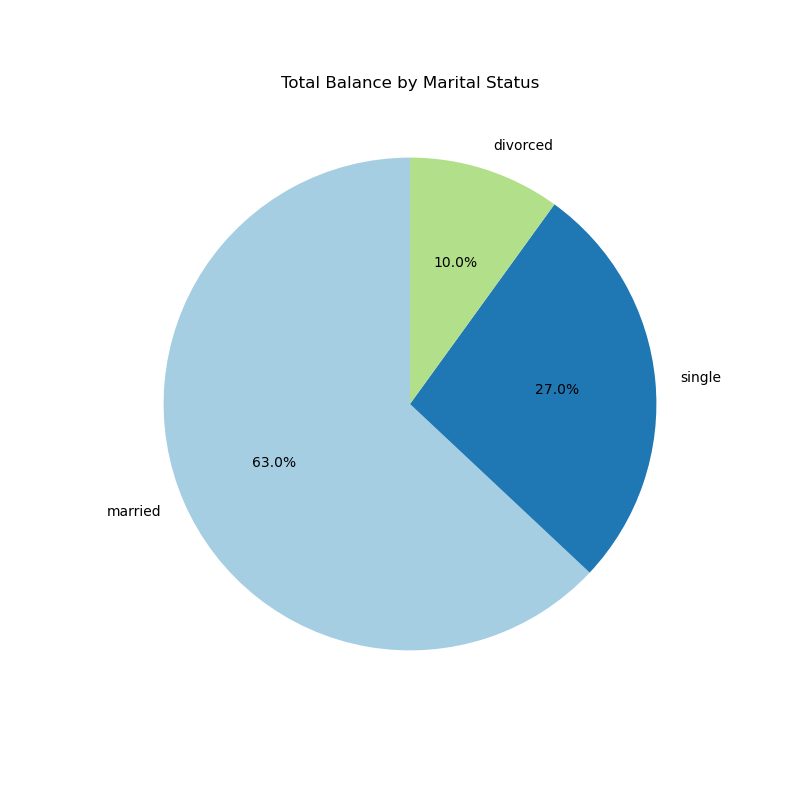


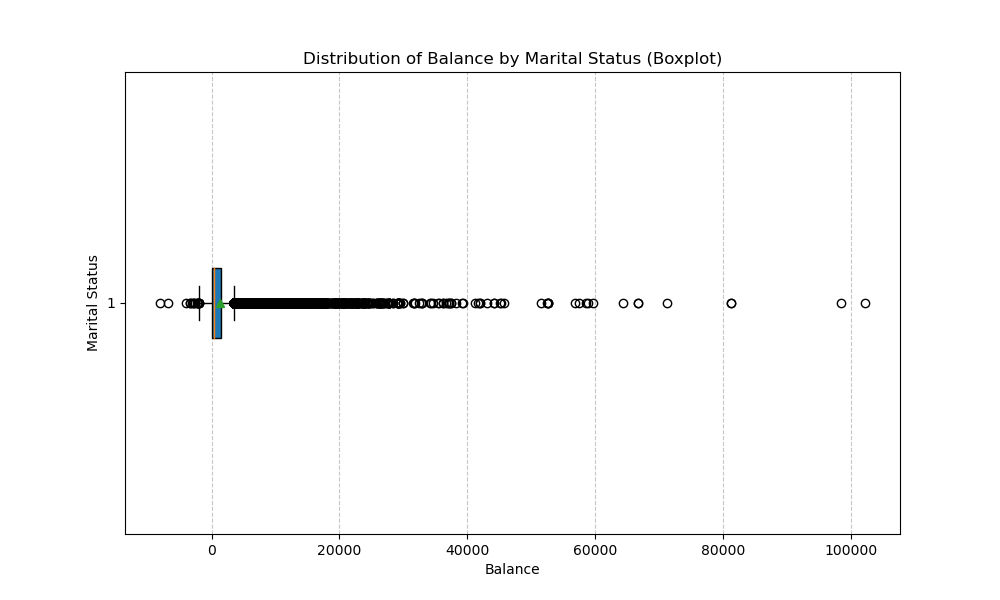
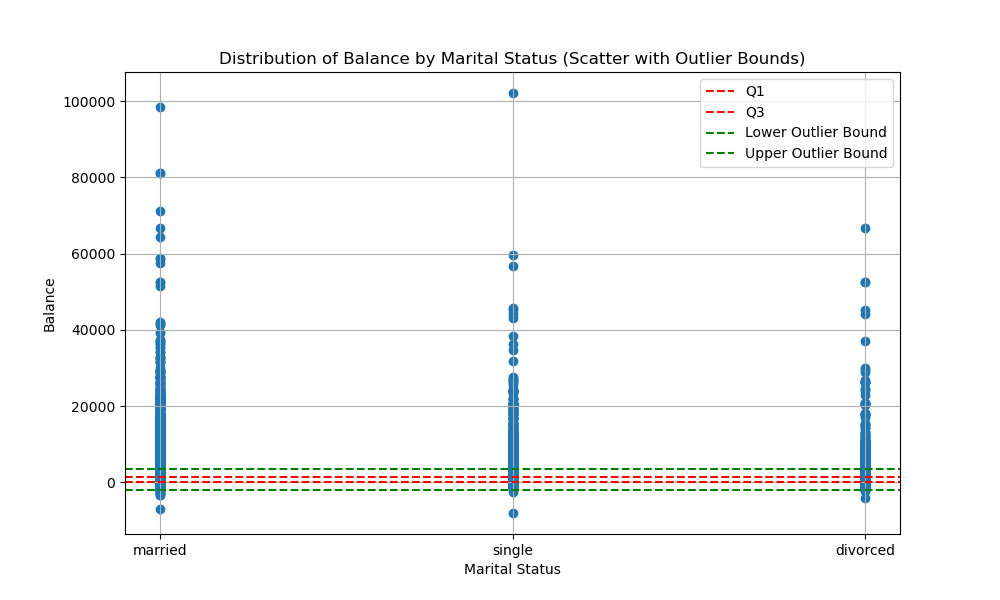
## By Marital:-

The presented code segments conduct a thorough examination of the total balance distribution across different marital statuses within the dataset. The initial steps involve aggregating the total balance for each marital status using the groupby function, resulting in the creation of the 'total\_balance\_by\_marital\_sorted' DataFrame, which is then sorted in descending order based on total balances. To visually represent this distribution, a pie chart is generated using matplotlib, featuring marital statuses on the x-axis and total balance on the y-axis. The chart is created with labels, a title, and gridlines to enhance interpretability, offering a clear overview of total balances across various marital statuses. Subsequently, skewness and modality of the balance distribution are examined using a boxplot, providing insights into the presence of outliers within marital statuses. Additionally, a scatter plot with horizontal lines denoting quartiles and outlier bounds aids in the identification and analysis of outliers for each marital status. These visualizations, saved as image files, collectively offer an in-depth understanding of how total balances are distributed across different marital statuses, helping to identify potential outliers and patterns within the dataset. Short descriptions and interpretations are provided for each of the three charts.

The bar chart illustrates that individuals with a 'married' marital status have the highest total balance, followed by 'single' and 'divorced.' The boxplot provides a clear view of the distribution of balance in each marital status category, with outliers marked for identification, and the means indicated by points in the boxes offering insights into central tendencies. The scatter plot highlights the distribution of balance in each marital status category, with horizontal lines representing quartiles and outlier bounds for the entire dataset, aiding in the identification of potential outliers with extreme values.

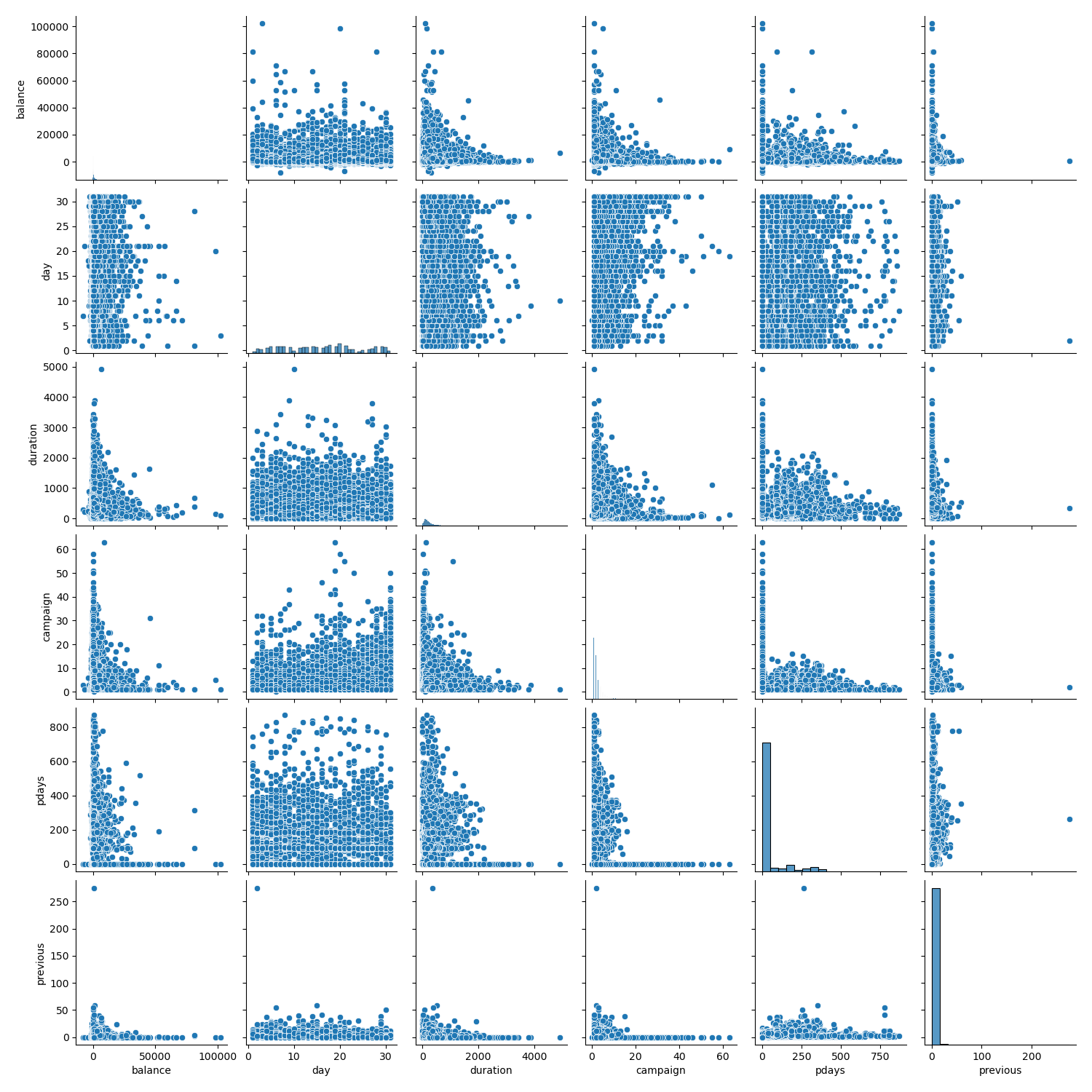
Fig:-

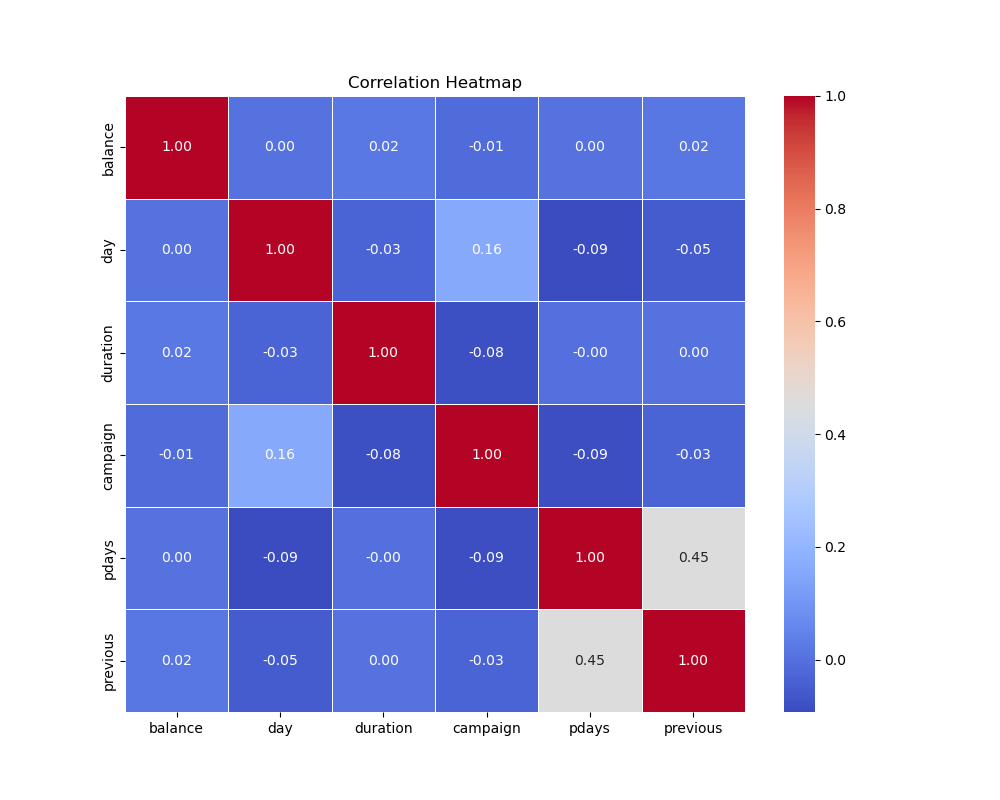




# Univarent

The provided code generates two insightful visualizations to explore numerical columns in the dataset. The first, a pair plot provides scatterplots for pairs of variables like 'balance,' 'day,' 'duration,' 'campaign,' 'pdays,' and 'previous,' revealing potential patterns and relationships. The second visualization is a correlation heatmap that highlights correlation coefficients between numerical columns. These visualizations offer a concise yet comprehensive understanding of relationships and correlations within the numerical features, aiding in pattern exploration and data analysis. The pair plot visually captures relationships between various numerical variables, while the correlation heatmap quantifies and highlights the strength and direction of these correlations.

Fig:-



# Conclusion

In conclusion, the comprehensive analysis of the dataset has provided valuable insights into its diverse aspects. From exploring the distribution of total balances across different demographic factors, such as age groups, jobs, and marital statuses, to uncovering intricate relationships and correlations within numerical features, the visualizations generated offer a nuanced understanding of the data. The thoughtful selection of visualization techniques, including bar charts, pie chart, boxplots, scatter plots, pair plots, and correlation heat maps, has facilitated a holistic exploration of patterns, potential outliers, and dependencies. These findings equip stakeholders with a robust foundation for decision-making and further exploration. The visualizations not only enhance interpretability but also serve as a visual narrative, fostering a deeper comprehension of the dataset's characteristics. This analytical journey underscores the power of visualizations in unraveling complex datasets, providing actionable insights, and paving the way for informed decision-making in data-driven endeavors.