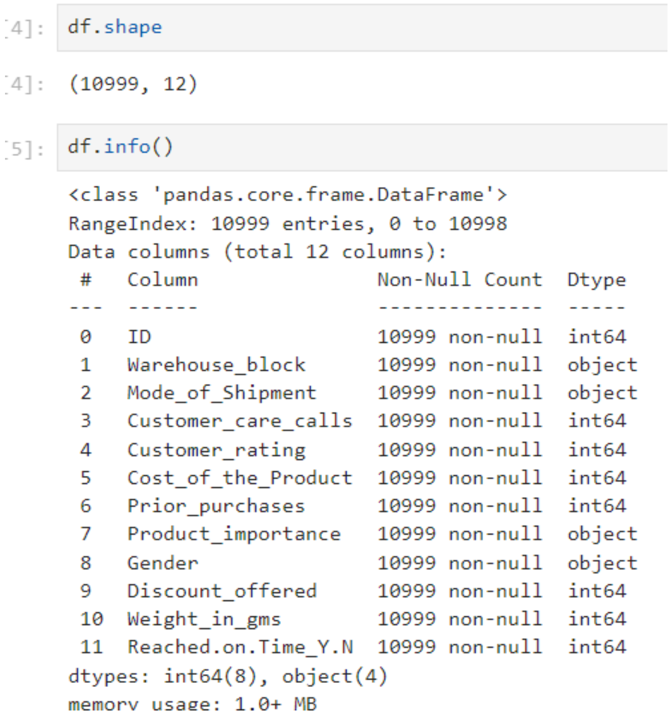
Data Collection and Preprocessing Phase

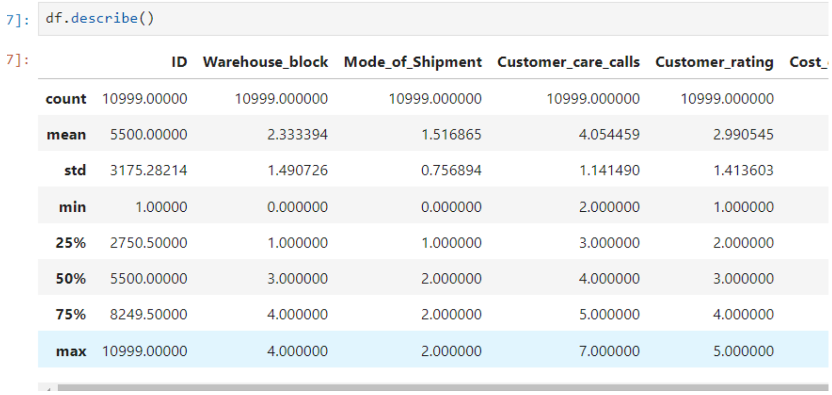
Date	8 JULY 2024
Team ID	SWTID1720108776
Project Title	Ecommerce Shipping Prediction Using Machine Learning
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	Description
Data Overview	Looked at the dataset for its shape, info and description of basic statistics of the features. 

```
[4]: df.shape

[4]: (10999, 12)

[5]: df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 10999 entries, 0 to 10998
     Data columns (total 12 columns):
      #   Column               Non-Null Count  Dtype
     ---  ------               --------------  -----
      0   ID                   10999 non-null  int64
      1   Warehouse_block      10999 non-null  object
      2   Mode_of_Shipment     10999 non-null  object
      3   Customer_care_calls  10999 non-null  int64
      4   Customer_rating      10999 non-null  int64
      5   Cost_of_the_Product  10999 non-null  int64
      6   Prior_purchases      10999 non-null  int64
      7   Product_importance   10999 non-null  object
      8   Gender               10999 non-null  object
      9   Discount_offered     10999 non-null  int64
      10  Weight_in_gms        10999 non-null  int64
      11  Reached.on.Time_Y.N  10999 non-null  int64
     dtypes: int64(8), object(4)
     memory usage: 1.0+ MB
```



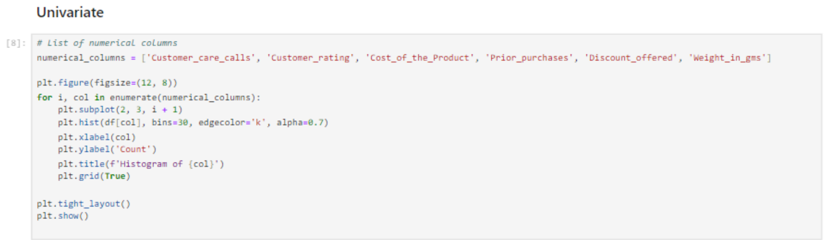
```
7]: df.describe()
```

7]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	
mean	5500.00000	2.333394	1.516865	4.054459	2.990545	
std	3175.28214	1.490726	0.756894	1.141490	1.413603	
min	1.00000	0.000000	0.000000	2.000000	1.000000	
25%	2750.50000	1.000000	1.000000	3.000000	2.000000	
50%	5500.00000	3.000000	2.000000	4.000000	3.000000	
75%	8249.50000	4.000000	2.000000	5.000000	4.000000	
max	10999.00000	4.000000	2.000000	7.000000	5.000000	

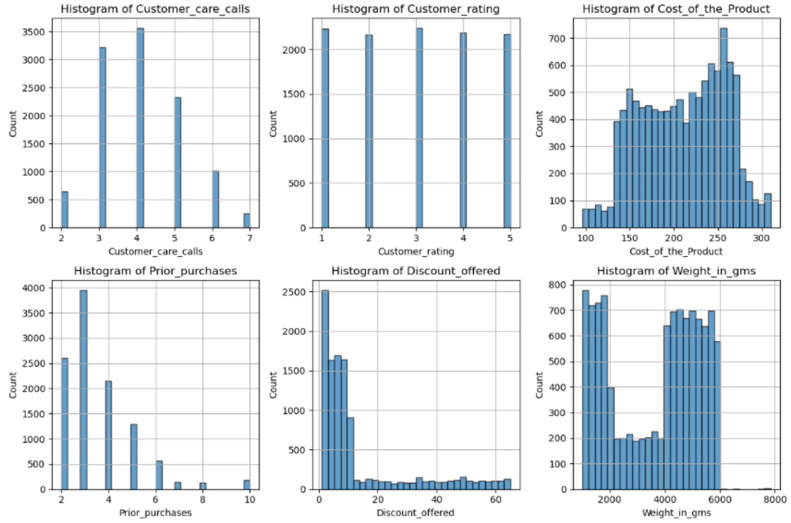
Univariate Analysis

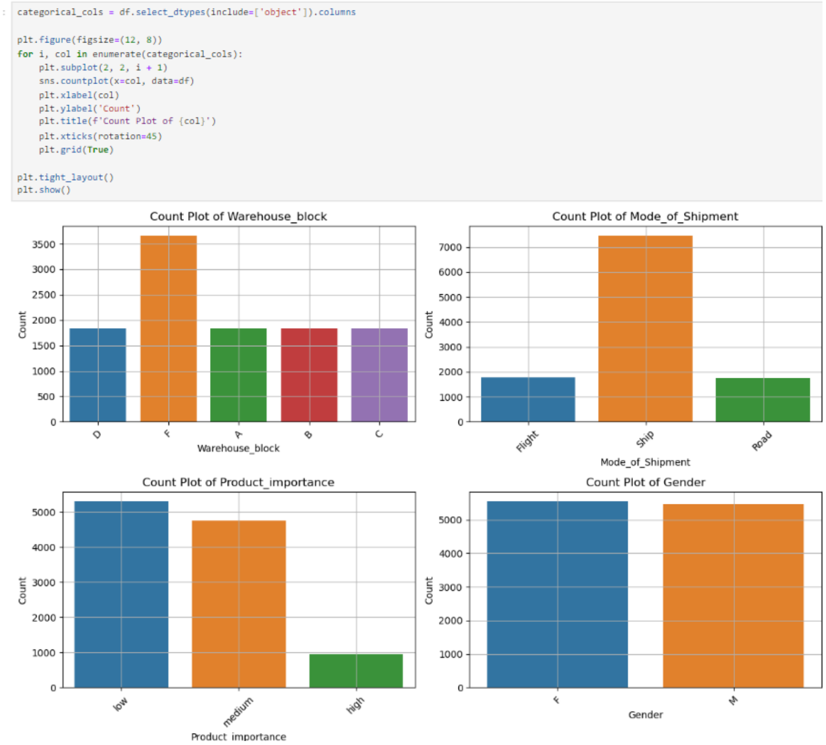
It is the single to single feature analysis, Used Histograms for Numerical Features and Count Plot for categorical Features with seaborn and matplotlyb libraries.



Univariate

```
[8]: # List of numerical columns
numerical_columns = ['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product', 'Prior_purchases', 'Discount_offered', 'Weight_in_gms']

plt.figure(figsize=(12, 8))
for i, col in enumerate(numerical_columns):
    plt.subplot(2, 3, i + 1)
    plt.hist(df[col], bins=30, edgecolor='k', alpha=0.7)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.title(f'Histogram of {col}')
    plt.grid(True)

plt.tight_layout()
plt.show()
```

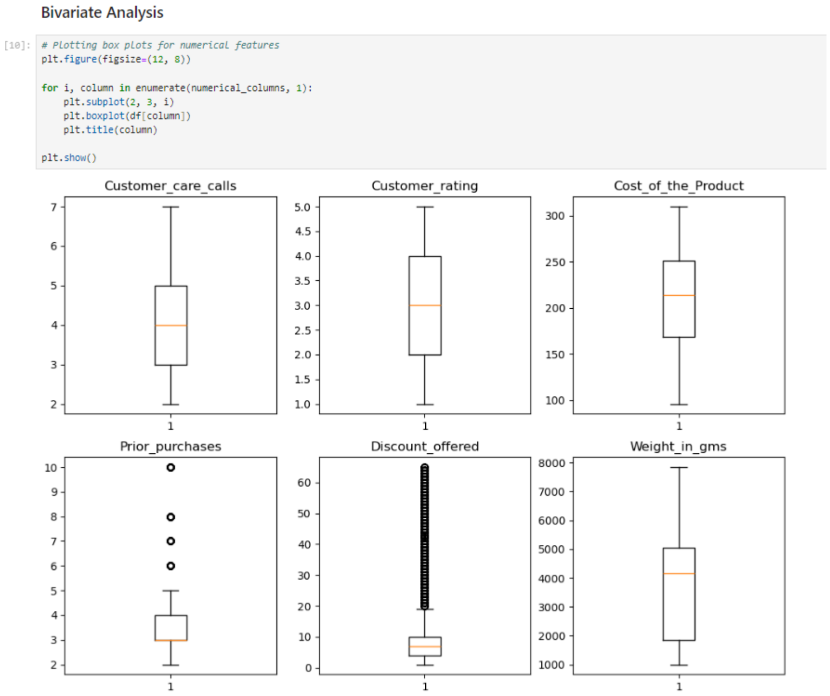




```
categorical_cols = df.select_dtypes(include=['object']).columns

plt.figure(figsize=(12, 8))
for i, col in enumerate(categorical_cols):
    plt.subplot(2, 2, i + 1)
    sns.countplot(x=col, data=df)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.title(f'Count Plot of {col}')
    plt.xticks(rotation=45)
    plt.grid(True)

plt.tight_layout()
plt.show()
```

Bivariate Analysis

Used Boxplots for Bivariate analysis, and also to check for outliers.

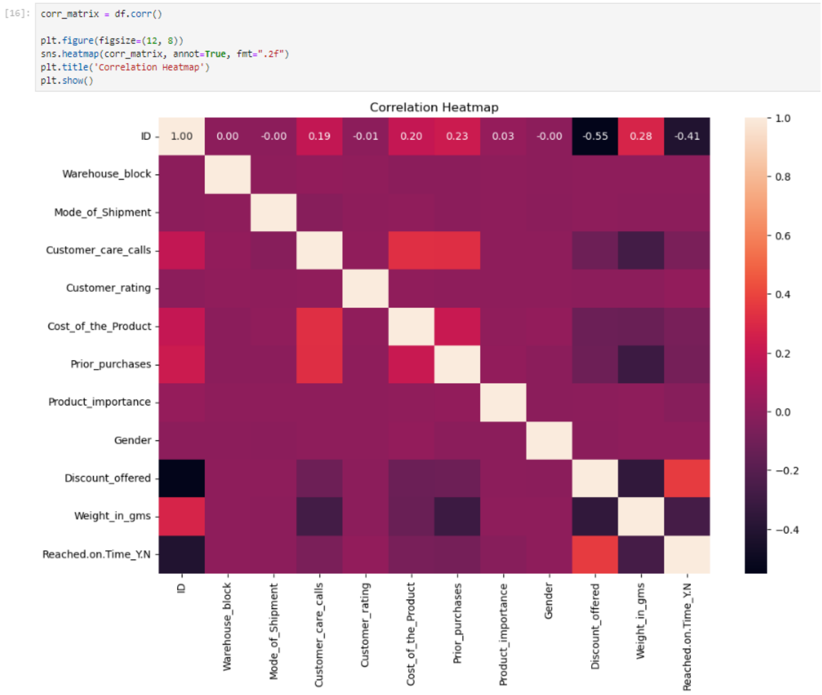


Bivariate Analysis

```
[10]: # Plotting box plots for numerical features
plt.figure(figsize=(12, 8))

for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    plt.boxplot(df[column])
    plt.title(column)

plt.show()
```

<table>
<tr>
<td>Multivariate Analysis</td>
<td>

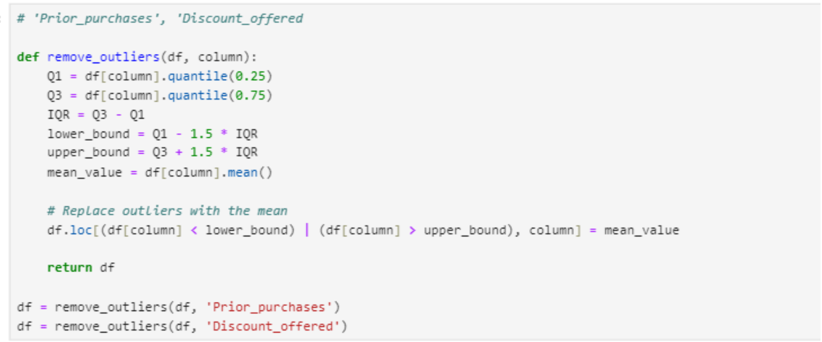
Used Heatmap which is the best way for multivariate analysis, it is plotted based on correlation values between each Feature. -Due to some version issues the numbers are not getting to every cell.



```
[16]: corr_matrix = df.corr()

      plt.figure(figsize=(12, 8))
      sns.heatmap(corr_matrix, annot=True, fmt=".2f")
      plt.title('Correlation Heatmap')
      plt.show()
```

</td>
</tr>
<tr>
<td>Outliers and Anomalies</td>
<td>

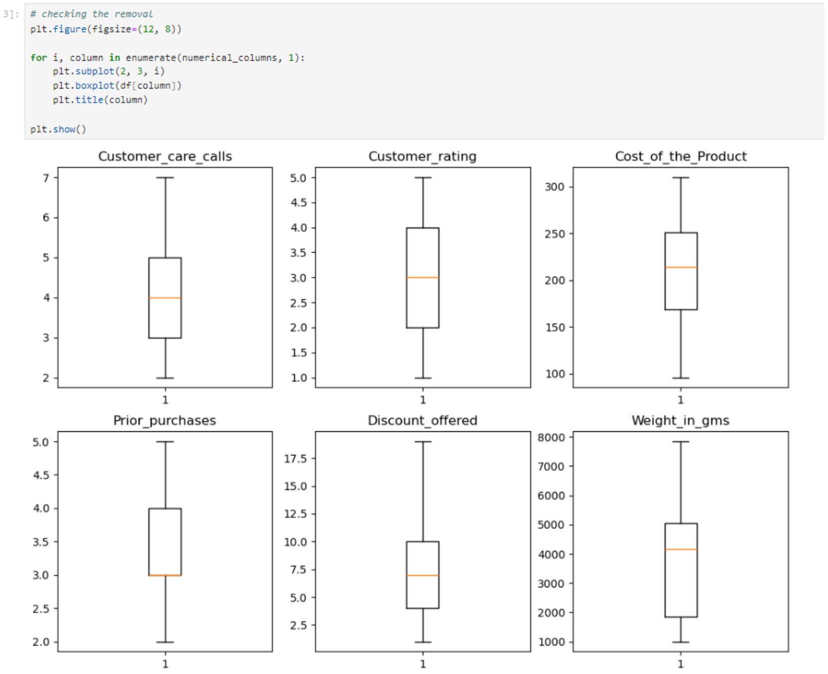
Found the outliers and replaced them with Mean value of the column, because removing them causing 2000 data points loss, and remaining 8 columns (Features) which are valuable for prediction are removing because of just 2 columns.



```
# 'Prior_purchases', 'Discount_offered

def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    mean_value = df[column].mean()

    # Replace outliers with the mean
    df.loc[(df[column] < lower_bound) | (df[column] > upper_bound), column] = mean_value

    return df

df = remove_outliers(df, 'Prior_purchases')
df = remove_outliers(df, 'Discount_offered')
```

</td>
</tr>
</table>

After removing

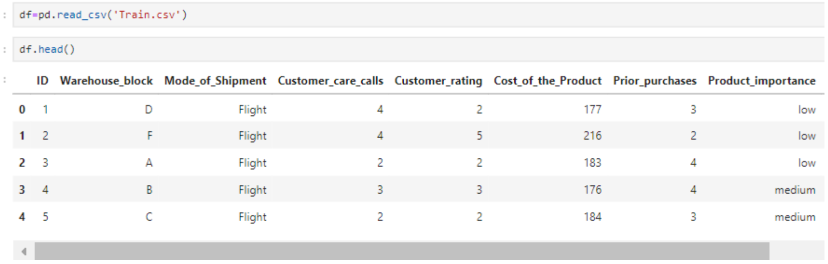


```
3]: # checking the removal
    plt.figure(figsize=(12, 8))

    for i, column in enumerate(numerical_columns, 1):
        plt.subplot(2, 3, i)
        plt.boxplot(df[column])
        plt.title(column)

    plt.show()
```

Data Preprocessing Code Screenshots

Loading Data

With pandas loaded the dataset downloaded from Kaggle.



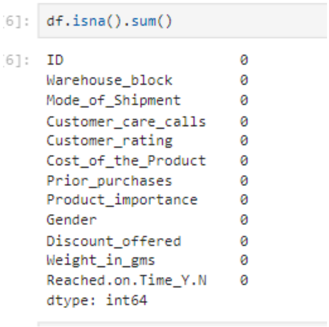
```
: df=pd.read_csv('Train.csv')
```

```
: df.head()
```

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance
0	1	D	Flight	4	2	177	3	low
1	2	F	Flight	4	5	216	2	low
2	3	A	Flight	2	2	183	4	low
3	4	B	Flight	3	3	176	4	medium
4	5	C	Flight	2	2	184	3	medium

Handling Missing Data

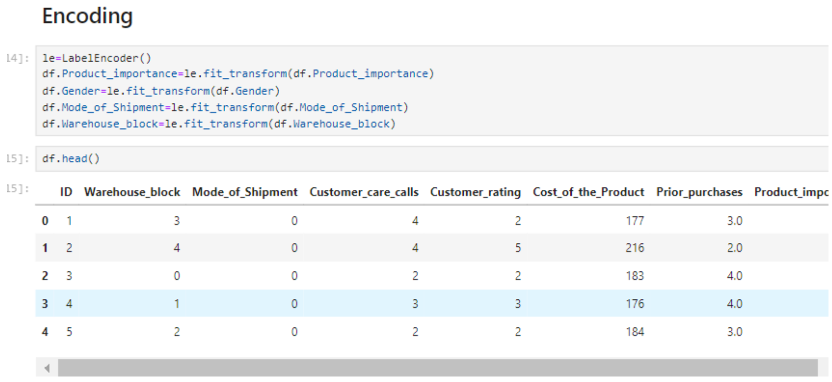
There are no Missing Values in the dataset.



```
6]: df.isna().sum()

6]: ID                     0
    Warehouse_block        0
    Mode_of_Shipment       0
    Customer_care_calls    0
    Customer_rating        0
    Cost_of_the_Product    0
    Prior_purchases        0
    Product_importance     0
    Gender                 0
    Discount_offered       0
    Weight_in_gms          0
    Reached.on.Time_Y.N    0
    dtype: int64
```

<table>
<tr>
<td>Data Transformation</td>
<td>

Used Label Encoding to transform Categorical features, and Standard Scaler is used to scale the values.



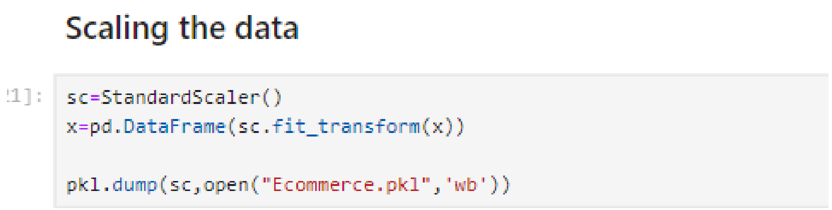
Encoding

```
14]: le=LabelEncoder()
     df.Product_importance=le.fit_transform(df.Product_importance)
     df.Gender=le.fit_transform(df.Gender)
     df.Mode_of_Shipment=le.fit_transform(df.Mode_of_Shipment)
     df.Warehouse_block=le.fit_transform(df.Warehouse_block)
```

```
15]: df.head()
```

15]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_impc
0	1	3	0	4	2	177	3.0	
1	2	4	0	4	5	216	2.0	
2	3	0	0	2	2	183	4.0	
3	4	1	0	3	3	176	4.0	
4	5	2	0	2	2	184	3.0	

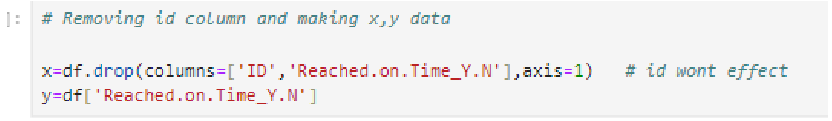


Scaling the data

```
1]: sc=StandardScaler()
    x=pd.DataFrame(sc.fit_transform(x))

    pkl.dump(sc,open("Ecommerce.pkl",'wb'))
```

</td>
</tr>
<tr>
<td>Feature Engineering</td>
<td>

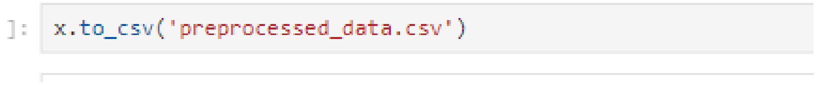
Just removed the ID column which has no use in predicting the target feature(‘Reached on time’)



```
]: # Removing id column and making x,y data

   x=df.drop(columns=['ID','Reached.on.Time_Y.N'],axis=1)    # id wont effect
   y=df['Reached.on.Time_Y.N']
```

</td>
</tr>
<tr>
<td>Save Processed Data</td>
<td>

We can save the processed data with the code,



```
]: x.to_csv('preprocessed_data.csv')
```

</td>
</tr>
</table>