```python
from IPython.display import Image

Image(filename="M:\OLA CASE STUDY\ola-electric-scooter-
1536x858.jpg",width=500)
```

Important Libraries For Analysis

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import zscore
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Problem Statement:

Recruiting and retaining drivers is seen by industry watchers as a tough battle for Ola. Churn among drivers is high and it's very easy for drivers to stop working for the service on the fly or jump to Uber depending on the rates. As the companies get bigger, the high churn could become a bigger problem. To find new drivers, Ola is casting a wide net, including people who don't have cars for jobs. But this acquisition is really costly. Losing drivers frequently impacts the morale of the organization and acquiring new drivers is more expensive than retaining existing ones. You are working as a data scientist with the Analytics Department of Ola, focused on driver team attrition. You are provided with the monthly information for a segment of drivers for 2019 and 2020 and tasked to predict whether a driver will be leaving the company or not based on their attributes like:

```python
df=pd.read_csv("M:\OLA CASE STUDY\ola_driver.csv")
```

Data Exploration and Imputation

```
df.head()

   Unnamed: 0    MMM-YY  Driver_ID   Age  Gender City  Education_Level
\
0           0  01/01/19          1  28.0     0.0  C23                2

1           1  02/01/19          1  28.0     0.0  C23                2

2           2  03/01/19          1  28.0     0.0  C23                2

3           3  11/01/20          2  31.0     0.0   C7                2

4           4  12/01/20          2  31.0     0.0   C7                2


   Income Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0   57387      24/12/18             NaN                    1      1
1   57387      24/12/18             NaN                    1      1
2   57387      24/12/18        03/11/19                    1      1
3   67016      11/06/20             NaN                    2      2
4   67016      11/06/20             NaN                    2      2

   Total Business Value  Quarterly Rating
0               2381060                 2
1               -665480                 2
2                     0                 2
3                     0                 1
4                     0                 1

df.tail()

       Unnamed: 0    MMM-YY  Driver_ID   Age  Gender City
Education_Level  \
19099       19099  08/01/20       2788  30.0     0.0  C27
2
19100       19100  09/01/20       2788  30.0     0.0  C27
2
19101       19101  10/01/20       2788  30.0     0.0  C27
2
19102       19102  11/01/20       2788  30.0     0.0  C27
2
19103       19103  12/01/20       2788  30.0     0.0  C27
2

       Income Dateofjoining LastWorkingDate  Joining Designation
Grade  \
19099   70254      06/08/20             NaN                    2
2
19100   70254      06/08/20             NaN                    2
2
```

```
19101   70254      06/08/20             NaN                      2
2
19102   70254      06/08/20             NaN                      2
2
19103   70254      06/08/20             NaN                      2
2

       Total Business Value  Quarterly Rating
19099                740280                 3
19100                448370                 3
19101                     0                 2
19102                200420                 2
19103                411480                 2
```

```
df.shape
```

```
(19104, 14)
```

```
df.dtypes
```

```
Unnamed: 0                int64
MMM-YY                   object
Driver_ID                 int64
Age                     float64
Gender                  float64
City                     object
Education_Level           int64
Income                    int64
Dateofjoining            object
LastWorkingDate          object
Joining Designation       int64
Grade                     int64
Total Business Value      int64
Quarterly Rating          int64
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            19104 non-null  int64
 1   MMM-YY                19104 non-null  object
 2   Driver_ID             19104 non-null  int64
 3   Age                   19043 non-null  float64
 4   Gender                19052 non-null  float64
 5   City                  19104 non-null  object
 6   Education_Level       19104 non-null  int64
 7   Income                19104 non-null  int64
```

```
 8   Dateofjoining         19104 non-null  object
 9   LastWorkingDate       1616 non-null   object
 10  Joining Designation   19104 non-null  int64
 11  Grade                 19104 non-null  int64
 12  Total Business Value  19104 non-null  int64
 13  Quarterly Rating      19104 non-null  int64
dtypes: float64(2), int64(8), object(4)
memory usage: 2.0+ MB
```

```
df.describe()
```

```
        Unnamed: 0     Driver_ID           Age        Gender  \
count  19104.000000  19104.000000  19043.000000  19052.000000
mean    9551.500000   1415.591133     34.668435      0.418749
std     5514.994107    810.705321      6.257912      0.493367
min        0.000000      1.000000     21.000000      0.000000
25%     4775.750000    710.000000     30.000000      0.000000
50%     9551.500000   1417.000000     34.000000      0.000000
75%    14327.250000   2137.000000     39.000000      1.000000
max    19103.000000   2788.000000     58.000000      1.000000

       Education_Level         Income  Joining Designation
Grade  \
count     19104.000000   19104.000000         19104.000000
19104.000000
mean          1.021671   65652.025126             1.690536
2.252670
std           0.800167   30914.515344             0.836984
1.026512
min           0.000000   10747.000000             1.000000
1.000000
25%           0.000000   42383.000000             1.000000
1.000000
50%           1.000000   60087.000000             1.000000
2.000000
75%           2.000000   83969.000000             2.000000
3.000000
max           2.000000  188418.000000             5.000000
5.000000

       Total Business Value  Quarterly Rating
count          1.910400e+04      19104.000000
mean           5.716621e+05          2.008899
std            1.128312e+06          1.009832
min           -6.000000e+06          1.000000
25%            0.000000e+00          1.000000
50%            2.500000e+05          2.000000
75%            6.997000e+05          3.000000
max            3.374772e+07          4.000000
```

```
df.describe(include='object')
```

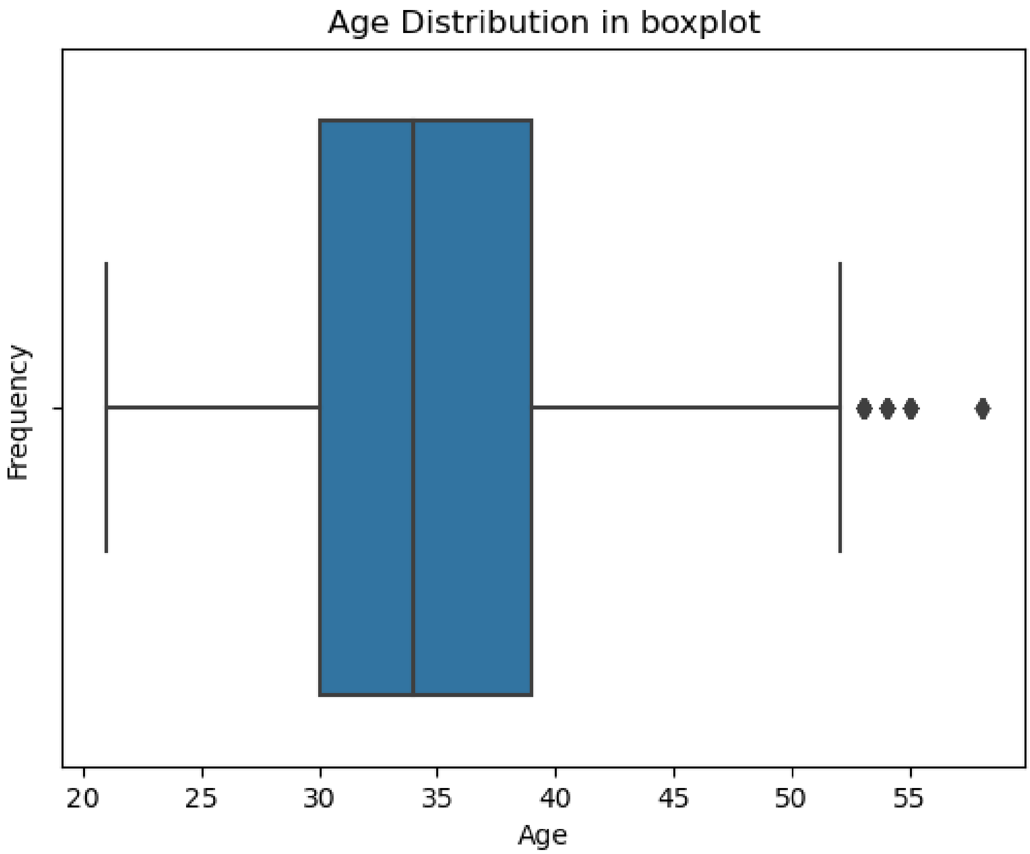
```
          MMM-YY   City Dateofjoining LastWorkingDate
count      19104  19104         19104            1616
unique        24     29           869             493
top     01/01/19    C20      23/07/15        29/07/20
freq        1022   1008           192              70
```

```
df.isnull().sum()
```

```
Unnamed: 0                  0
MMM-YY                      0
Driver_ID                   0
Age                        61
Gender                     52
City                        0
Education_Level             0
Income                      0
Dateofjoining               0
LastWorkingDate         17488
Joining Designation         0
Grade                       0
Total Business Value        0
Quarterly Rating            0
dtype: int64
```

```
sns.boxplot(x='Age',data=df)
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.title("Age Distribution in boxplot")
```

```
Text(0.5, 1.0, 'Age Distribution in boxplot')
```




```
df['Age']=df['Age'].fillna(df['Age'].mean())

df['Age'].isnull().sum()

0

df.isnull().sum()

Unnamed: 0                    0
MMM-YY                        0
Driver_ID                     0
Age                           0
Gender                       52
City                          0
Education_Level               0
Income                        0
Dateofjoining                 0
LastWorkingDate           17488
Joining Designation           0
Grade                         0
Total Business Value          0
Quarterly Rating              0
dtype: int64
```

```
non_null_genders=df.Gender.dropna().values
non_null_genders
```

```
array([0., 0., 0., ..., 0., 0., 0.])
```

```
df.Gender.value_counts()
```

```
Gender
0.0    11074
1.0     7978
Name: count, dtype: int64
```

```
df['Gender']=df['Gender'].apply(lambda
x:np.random.choice(non_null_genders) if pd.isnull(x) else x)
```

```
df['Gender'].isnull().sum()
```

```
0
```

```
df.Gender.value_counts()
```

```
Gender
0.0    11103
1.0     8001
Name: count, dtype: int64
```

Feature Engineering

```
df['Hasleftcompany']=df['LastWorkingDate'].apply(lambda x:0 if
pd.isnull(x) else 1)
```

```
df['Hasleftcompany'].value_counts()
```

```
Hasleftcompany
0    17488
1     1616
Name: count, dtype: int64
```

```
df['Dateofjoining']=pd.to_datetime(df['Dateofjoining'],errors='coerce'
)
current_year=pd.Timestamp.today().year
df['Birthyear']=current_year-df['Age']
df['AgeAtJoining']=df['Dateofjoining'].dt.year-df['Birthyear']
```

```
df['Birthyear']=df['Birthyear'].astype('int64')
```

```
df['AgeAtJoining']=df['AgeAtJoining'].astype('int64')
```

```
df.head()
```

```
   Unnamed: 0    MMM-YY  Driver_ID   Age  Gender City  Education_Level
\
0           0  01/01/19          1  28.0     0.0  C23                2
```

```
1           1  02/01/19          1  28.0     0.0  C23                2

2           2  03/01/19          1  28.0     0.0  C23                2

3           3  11/01/20          2  31.0     0.0   C7                2

4           4  12/01/20          2  31.0     0.0   C7                2


   Income Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0   57387    2018-12-24             NaN                    1      1
1   57387    2018-12-24             NaN                    1      1
2   57387    2018-12-24        03/11/19                    1      1
3   67016    2020-11-06             NaN                    2      2
4   67016    2020-11-06             NaN                    2      2

   Total Business Value  Quarterly Rating  Hasleftcompany
Birthyear  \
0               2381060                 2               0       1997

1               -665480                 2               0       1997

2                     0                 2               1       1997

3                     0                 1               0       1994

4                     0                 1               0       1994


   AgeAtJoining
0            21
1            21
2            21
3            26
4            26
```

```
df['Dateofjoining']=pd.to_datetime(df['Dateofjoining'],errors='coerce'
)
df['LastWorkingDate']=pd.to_datetime(df['LastWorkingDate'],errors='coe
rce')
df['Tenure']=df.apply(lambda row:(pd.Timestamp.today()-
row['Dateofjoining']).days
                      if pd.isnull(row['LastWorkingDate']) else
(row['LastWorkingDate']-row['Dateofjoining']).days, axis=1)

max_income=df['Income'].max()
print(f"Maximum income of any Employee is :{max_income}")
min_income=df['Income'].min()
print(f"Minimum income of any Employee is :{min_income}")
```

```
Maximum income of any Employee is :188418
Minimum income of any Employee is :10747
```

```python
df['Salary_Range']=pd.cut(df['Income'],bins=[10000,50000,100000,150000
,200000],labels=['Low Salary','Medium Salary','High Salary','Upper
High Salary'])
```

```python
df['Salary_Range']
```

```
0        Medium Salary
1        Medium Salary
2        Medium Salary
3        Medium Salary
4        Medium Salary
             ...
19099    Medium Salary
19100    Medium Salary
19101    Medium Salary
19102    Medium Salary
19103    Medium Salary
Name: Salary_Range, Length: 19104, dtype: category
Categories (4, object): ['Low Salary' < 'Medium Salary' < 'High
Salary' < 'Upper High Salary']
```

```python
df['Salary_Range'].value_counts()
```

```
Salary_Range
Medium Salary        9696
Low Salary           6604
High Salary          2664
Upper High Salary     140
Name: count, dtype: int64
```

```python
df.head(1)
```

```
   Unnamed: 0    MMM-YY  Driver_ID   Age  Gender City  Education_Level
\
0           0  01/01/19          1  28.0     0.0  C23                2


   Income Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0   57387    2018-12-24             NaT                    1      1

   Total Business Value  Quarterly Rating  Hasleftcompany
Birthyear  \
0               2381060                 2               0       1997


   AgeAtJoining  Tenure   Salary_Range
0            21    2226  Medium Salary
```

```python
df['Quarterly Rating'].value_counts()
```

```
Quarterly Rating
1    7679
2    5553
3    3895
4    1977
Name: count, dtype: int64
```

```
df['Quarterly_Range']=pd.cut(df['Quarterly
Rating'],bins=[1,2,3,4],labels=['Low','Medium','High'])
```

```
df['Quarterly_Range'].value_counts()
```

```
Quarterly_Range
Low       5553
Medium    3895
High      1977
Name: count, dtype: int64
```

```
max_age=df['Age'].max()
min_age=df['Age'].min()
print(f"Maximum Age of any person in the DataSet is : {max_age}")
print(f"Minimum Age of any person in the DataSet is : {min_age}")
```

```
Maximum Age of any person in the DataSet is : 58.0
Minimum Age of any person in the DataSet is : 21.0
```

```
df['Riders_Age_Category']=pd.cut(df['Age'],bins=[21,25,30,58],labels=[
'Young Riders','Medium Riders','Old Riders'])
```

```
df['Riders_Age_Category'].value_counts()
```

```
Riders_Age_Category
Old Riders       13820
Medium Riders     4241
Young Riders      1008
Name: count, dtype: int64
```

```
df['MMM-YY']=pd.to_datetime(df['MMM-YY'],format="%d/%m/%y")
df=df.sort_values(by=['Driver_ID','MMM-YY'])
def Rating_increasing(rating):
    return (rating>rating.shift(1)).astype(int)
```

```
df['Rating_Increased']=df.groupby('Driver_ID')['Quarterly
Rating'].apply(Rating_increasing).reset_index(level = 0, drop = True)
```

```
df[['MMM-YY','Driver_ID','Quarterly Rating','Rating_Increased']]
```

	MMM-YY	Driver_ID	Quarterly Rating	Rating_Increased
0	2019-01-01	1	2	0
1	2019-01-02	1	2	0
2	2019-01-03	1	2	0
3	2020-01-11	2	1	0

```
4     2020-01-12         2                  1                 0
...          ...        ...                ...               ...
19099 2020-01-08       2788                 3                 0
19100 2020-01-09       2788                 3                 0
19101 2020-01-10       2788                 2                 0
19102 2020-01-11       2788                 2                 0
19103 2020-01-12       2788                 2                 0

[19104 rows x 4 columns]
```

```
df.Rating_Increased.value_counts()
```

```
Rating_Increased
0    17859
1     1245
Name: count, dtype: int64
```

```
df['MMM-YY']=pd.to_datetime(df['MMM-YY'],format='%d%m%y')
df=df.sort_values(by=['Driver_ID','Income'])
def Income_Increasing(Income):
    return (Income>Income.shift(1)).astype('int64')
df['Salary_Increased']=df.groupby('Driver_ID')
['Income'].apply(Income_Increasing).reset_index(level=0,drop=True)

print(df[["MMM-YY", "Driver_ID", "Income", "Salary_Increased"]])
```

```
          MMM-YY  Driver_ID  Income  Salary_Increased
0     2019-01-01          1   57387                 0
1     2019-01-02          1   57387                 0
2     2019-01-03          1   57387                 0
3     2020-01-11          2   67016                 0
4     2020-01-12          2   67016                 0
...          ...        ...     ...               ...
19099 2020-01-08       2788   70254                 0
19100 2020-01-09       2788   70254                 0
19101 2020-01-10       2788   70254                 0
19102 2020-01-11       2788   70254                 0
19103 2020-01-12       2788   70254                 0

[19104 rows x 4 columns]
```

```
df.Salary_Increased.value_counts()
```

```
Salary_Increased
0    19060
1       44
Name: count, dtype: int64
```

Class Imbalance Treatment

```
df.columns
```

```
Index(['Unnamed: 0', 'MMM-YY', 'Driver_ID', 'Age', 'Gender', 'City',
       'Education_Level', 'Income', 'Dateofjoining',
'LastWorkingDate',
       'Joining Designation', 'Grade', 'Total Business Value',
       'Quarterly Rating', 'Hasleftcompany', 'Birthyear',
'AgeAtJoining',
       'Tenure', 'Salary_Range', 'Quarterly_Range',
'Riders_Age_Category',
       'Rating_Increased', 'Salary_Increased'],
      dtype='object')
```

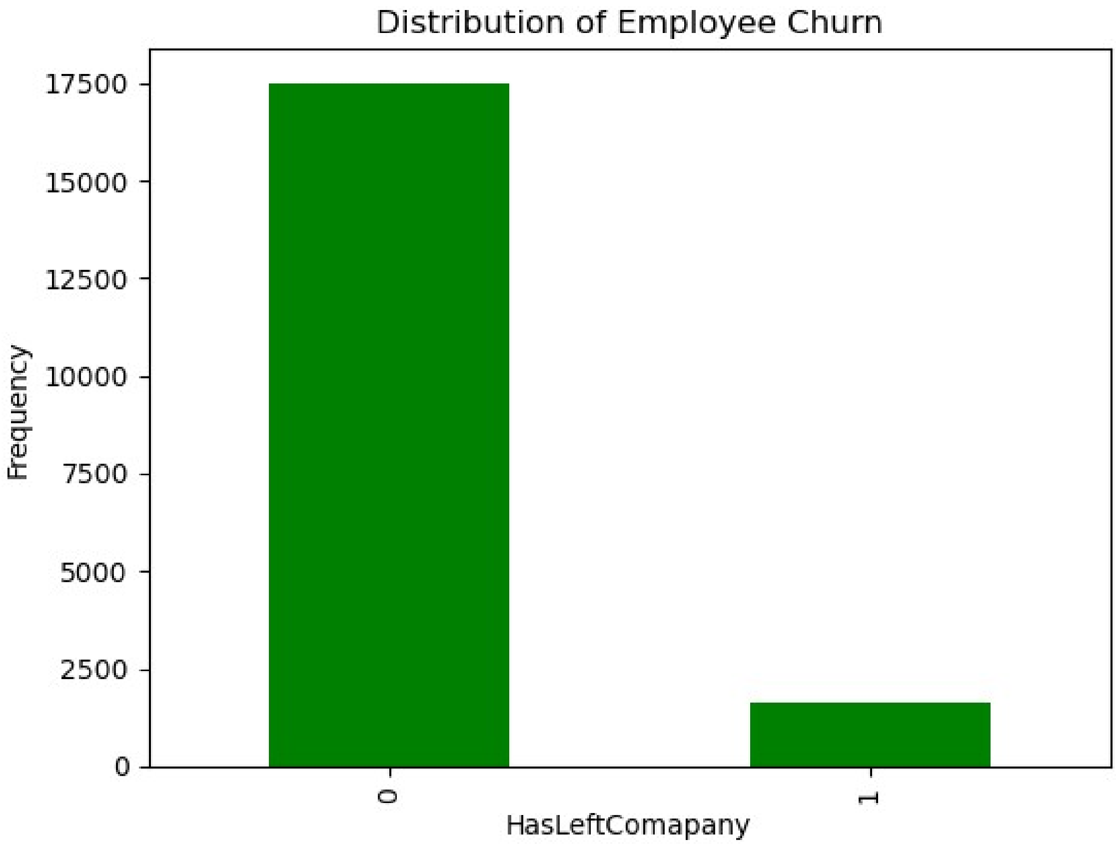
```
df['Hasleftcompany'].isnull().sum()
```

```
0
```

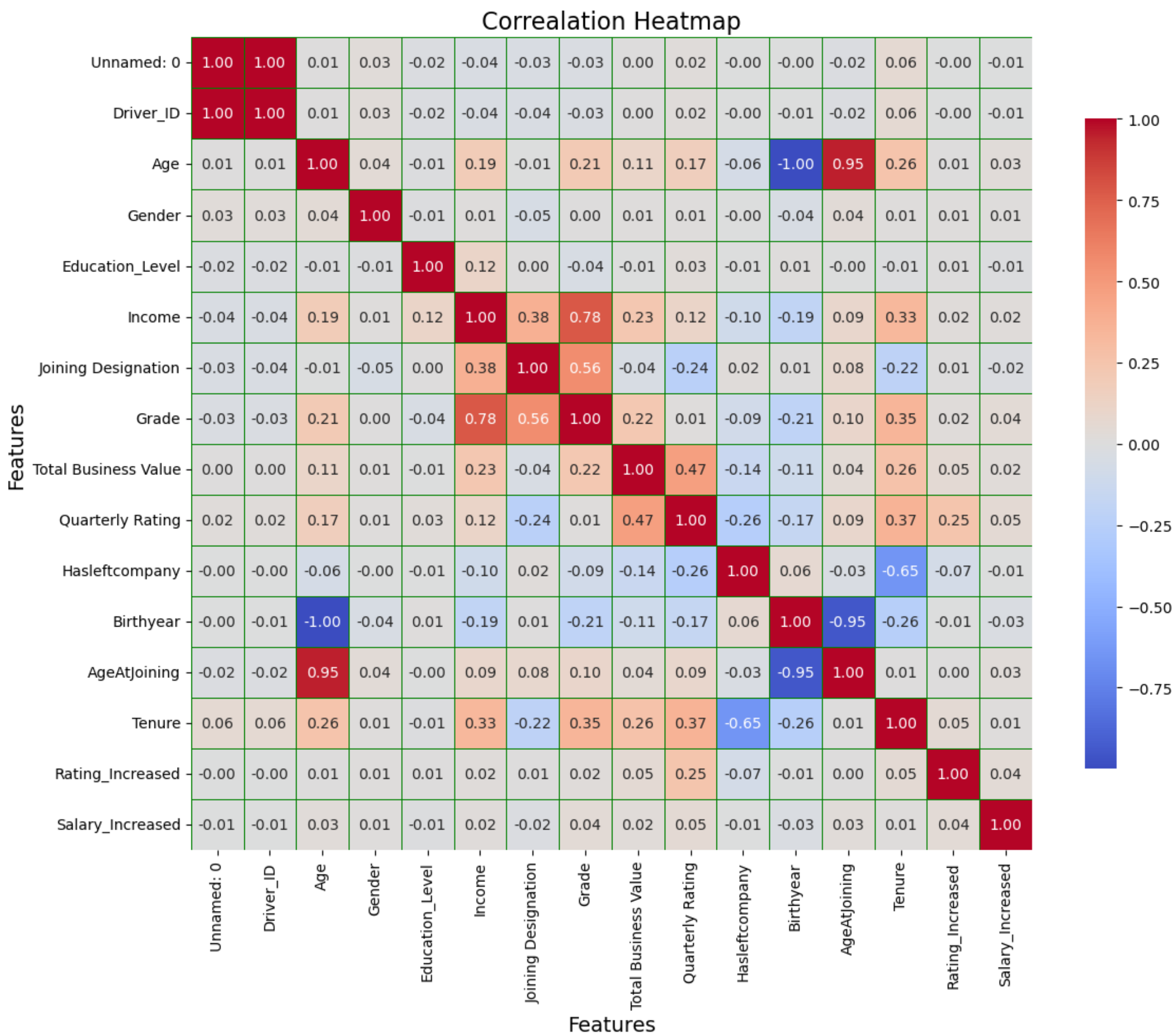
```
df['Hasleftcompany'].value_counts()
```

```
Hasleftcompany
0    17488
1     1616
Name: count, dtype: int64
```

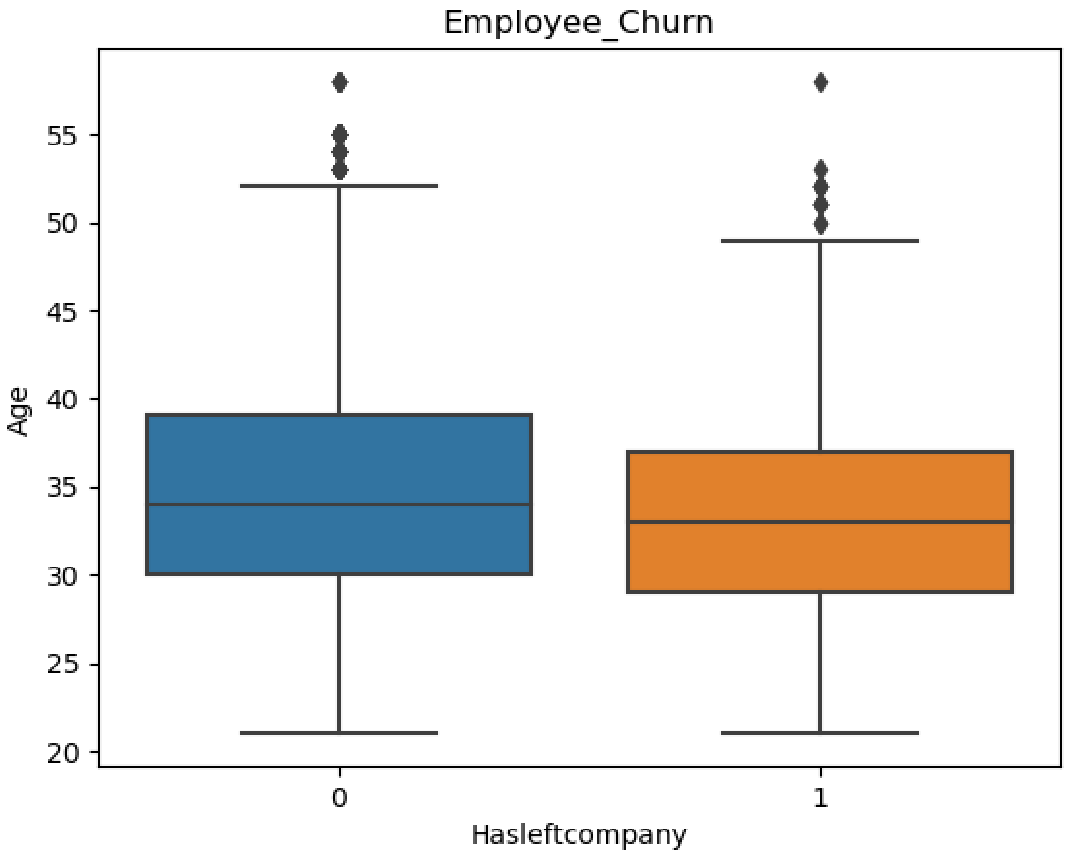
```
df['Hasleftcompany'].value_counts().plot(kind='bar',color='green')
plt.xlabel("HasLeftComapany")
plt.ylabel("Frequency")
plt.title("Distribution of Employee Churn")
plt.show()
```




```
numeric_dtypes_df=df.select_dtypes(include=[np.number])

plt.figure(figsize=(12,10))
corr=numeric_dtypes_df.corr()
sns.heatmap(corr,annot=True,cmap='coolwarm',fmt='0.2f',linewidths=0.5,
linecolor='green',cbar_kws={'shrink':0.8},annot_kws={'size':10})
plt.title("Correalation Heatmap", fontsize = 16)
plt.xlabel("Features", fontsize = 14)
plt.ylabel("Features", fontsize = 14)
plt.tight_layout()
plt.show()
```

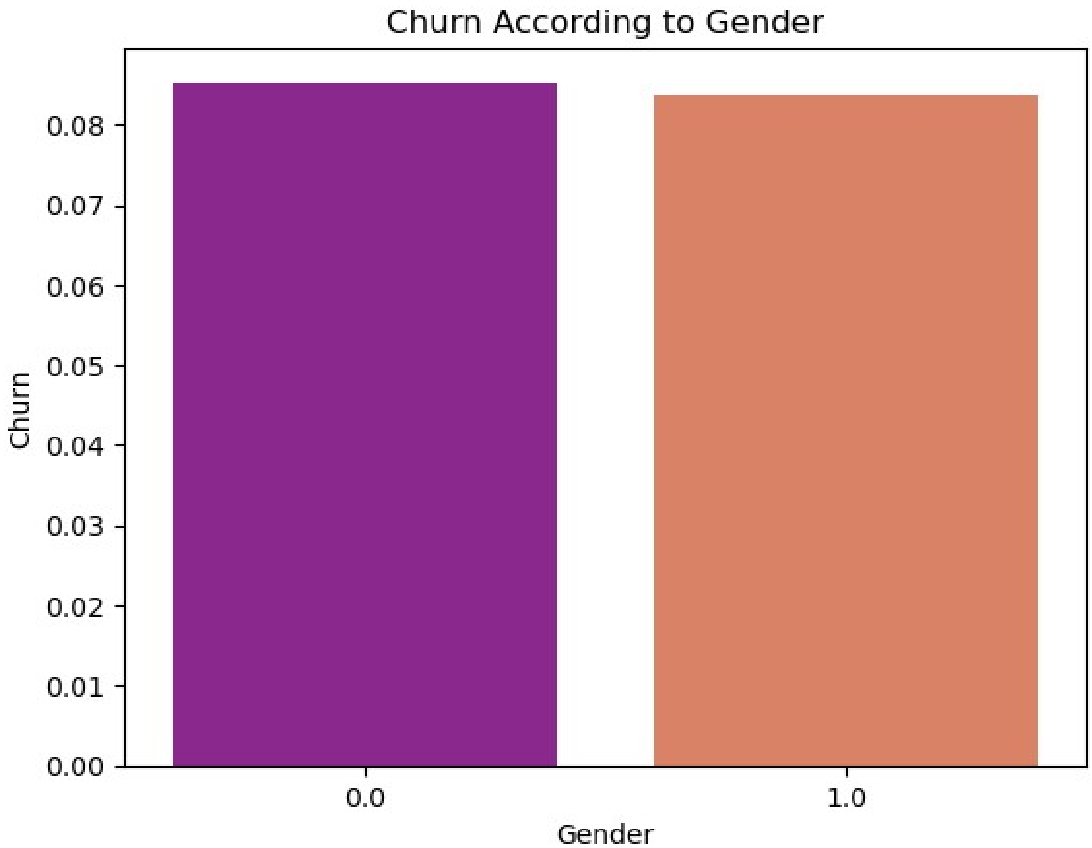



```python
sns.boxplot(x=df['Hasleftcompany'],y=df['Age'],data=df)
plt.title("Employee_Churn")
plt.show()
```

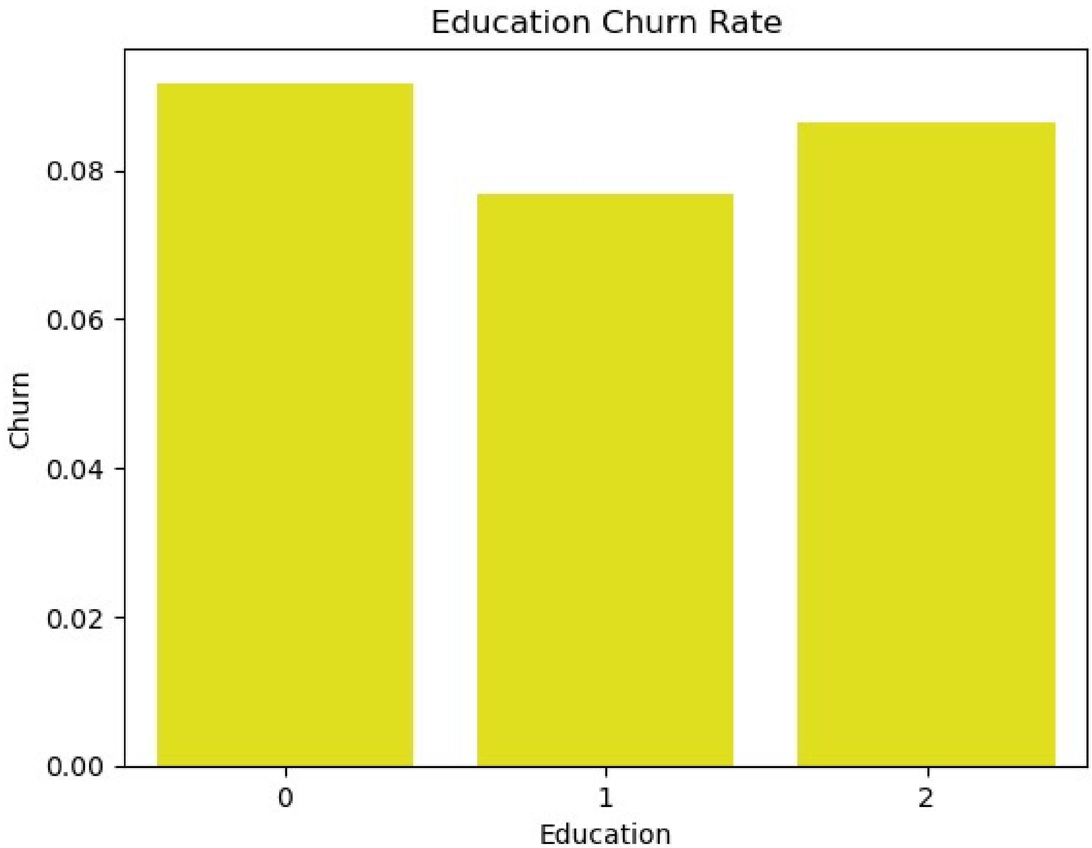



```
Gender_Churn=df.groupby('Gender')['Hasleftcompany'].mean()
print(Gender_Churn)

sns.barplot(x=Gender_Churn.index,y=Gender_Churn.values,color='green',p
alette='plasma')
plt.title("Churn According to Gender")
plt.xlabel("Gender")
plt.ylabel("Churn")
plt.show()
```

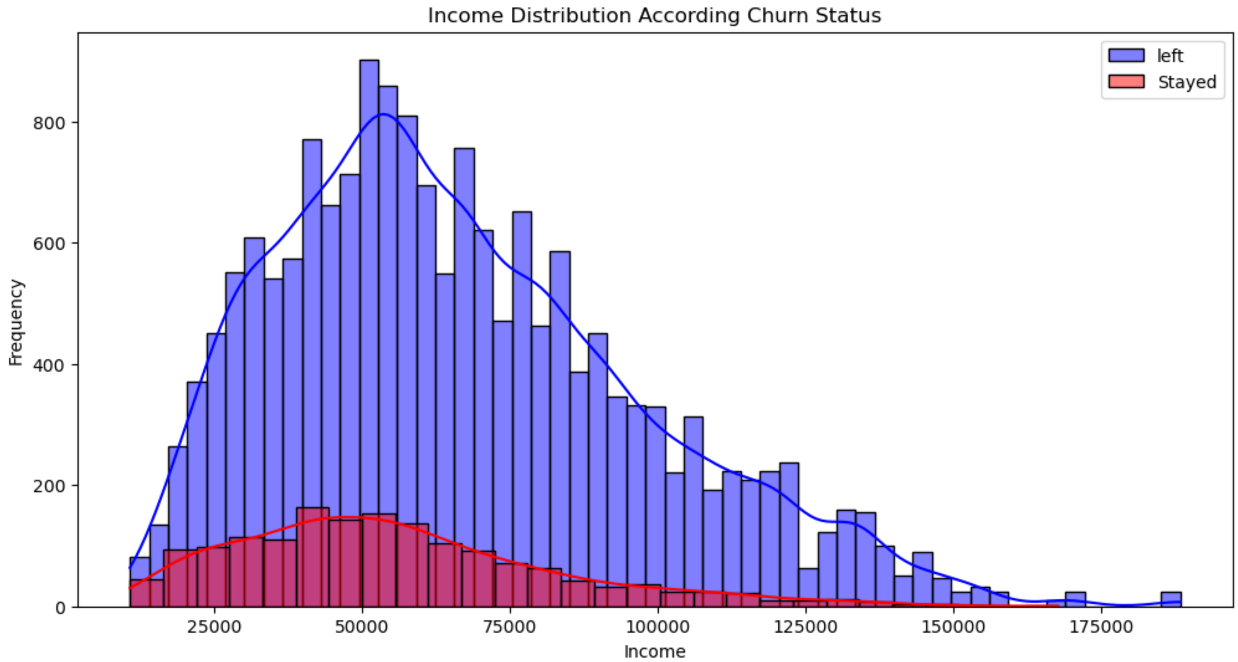
```
Gender
0.0    0.085202
1.0    0.083740
Name: Hasleftcompany, dtype: float64
```



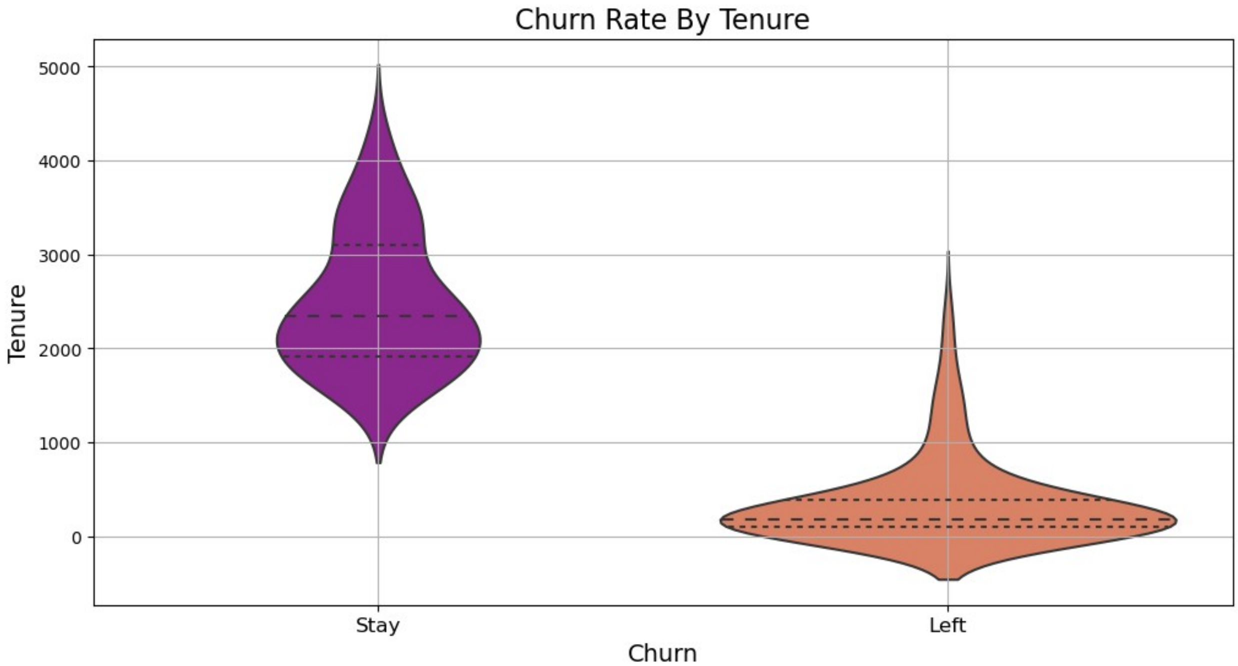

```
Education_Churn=df.groupby('Education_Level')['Hasleftcompany'].mean()
print(Education_Churn)

sns.barplot(x=Education_Churn.index,y=Education_Churn.values,color='ye
llow')
plt.title("Education Churn Rate")
plt.xlabel("Education")
plt.ylabel("Churn")
plt.show()

Education_Level
0    0.091662
1    0.076777
2    0.086455
Name: Hasleftcompany, dtype: float64
```



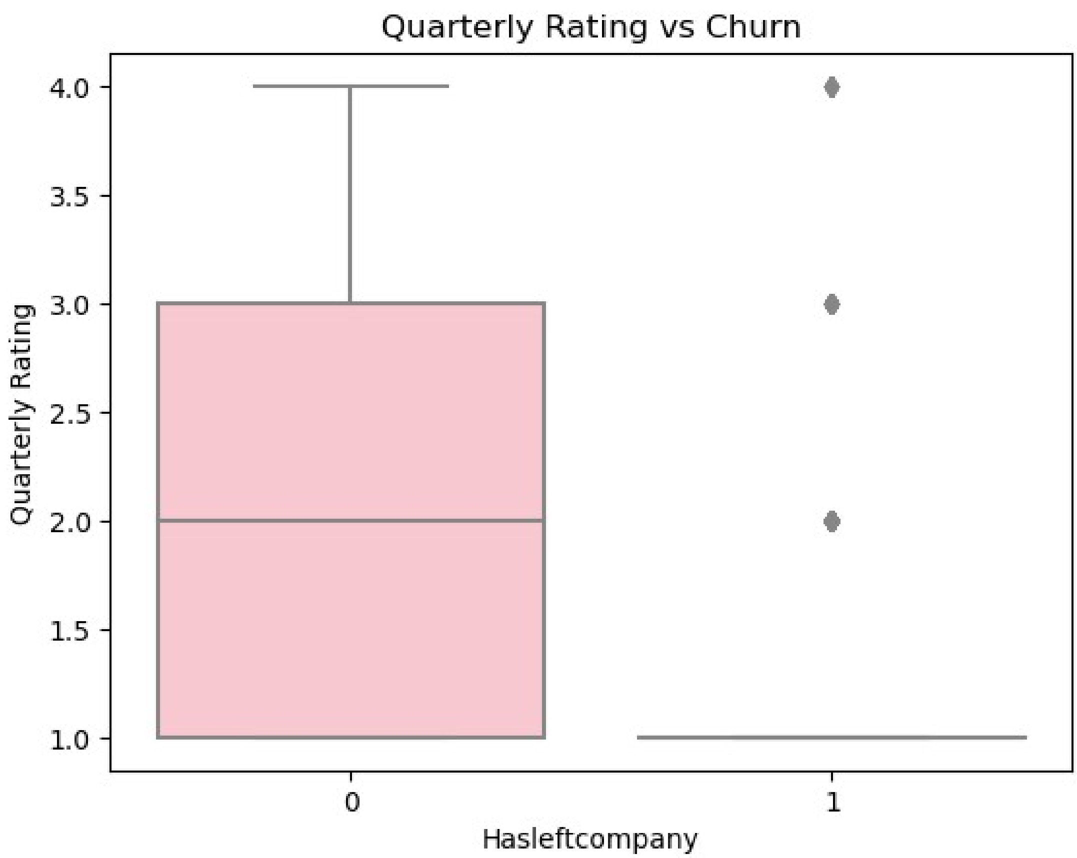

```
plt.figure(figsize=(12,6))
sns.histplot(df[df['Hasleftcompany']==0]
['Income'],kde=True,color='blue',label='left')
sns.histplot(df[df['Hasleftcompany']==1]
['Income'],kde=True,color='red',label='Stayed')
plt.xlabel("Income")
plt.ylabel("Frequency")
plt.title("Income Distribution According Churn Status")
plt.legend()
plt.show()
```




```
plt.figure(figsize=(12,6))

sns.violinplot(x='Hasleftcompany',y='Tenure',data=df,color='blue',palette='plasma',bw=0.5,linewidth=1.5,inner='quartile',legend=('HasleftCompany'))
plt.xlabel("Churn", fontsize = 14)
plt.ylabel("Tenure", fontsize = 14)
plt.title("Churn Rate By Tenure", fontsize = 16)
plt.xticks([0,1],["Stay", "Left"], fontsize = 12)
plt.grid()
plt.show()
```



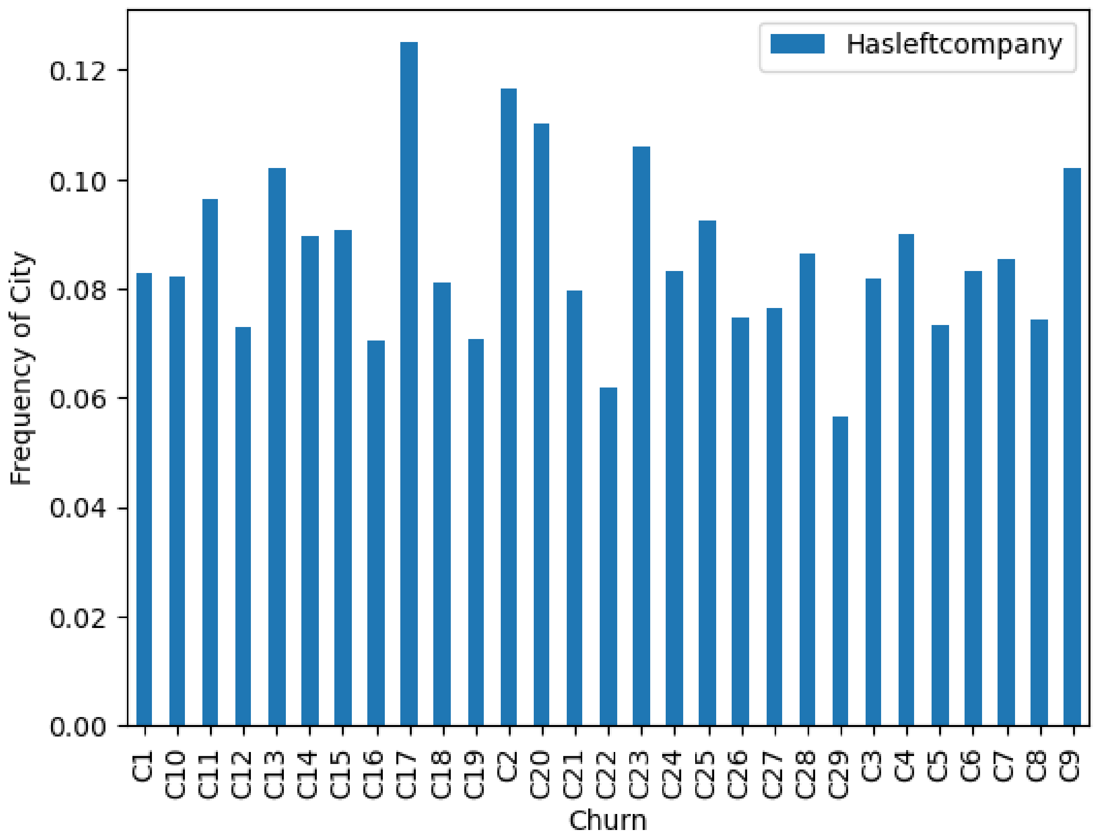

```
sns.boxplot(x='Hasleftcompany',y='Quarterly
Rating',data=df,color='pink')
plt.title('Quarterly Rating vs Churn')
plt.show()
```




```
City_churn=df.groupby('City')['Hasleftcompany'].mean()
print(City_churn)
City_churn.plot(kind='bar',legend='Hasleftcompany')
plt.xlabel("Churn")
plt.ylabel("Frequency of City")
plt.legend()
plt.show()
```

```
City
C1     0.082718
C10    0.081989
C11    0.096154
C12    0.072902
C13    0.101933
C14    0.089506
C15    0.090670
C16    0.070522
C17    0.125000
C18    0.080882
C19    0.070812
C2     0.116525
C20    0.110119
C21    0.079602
```

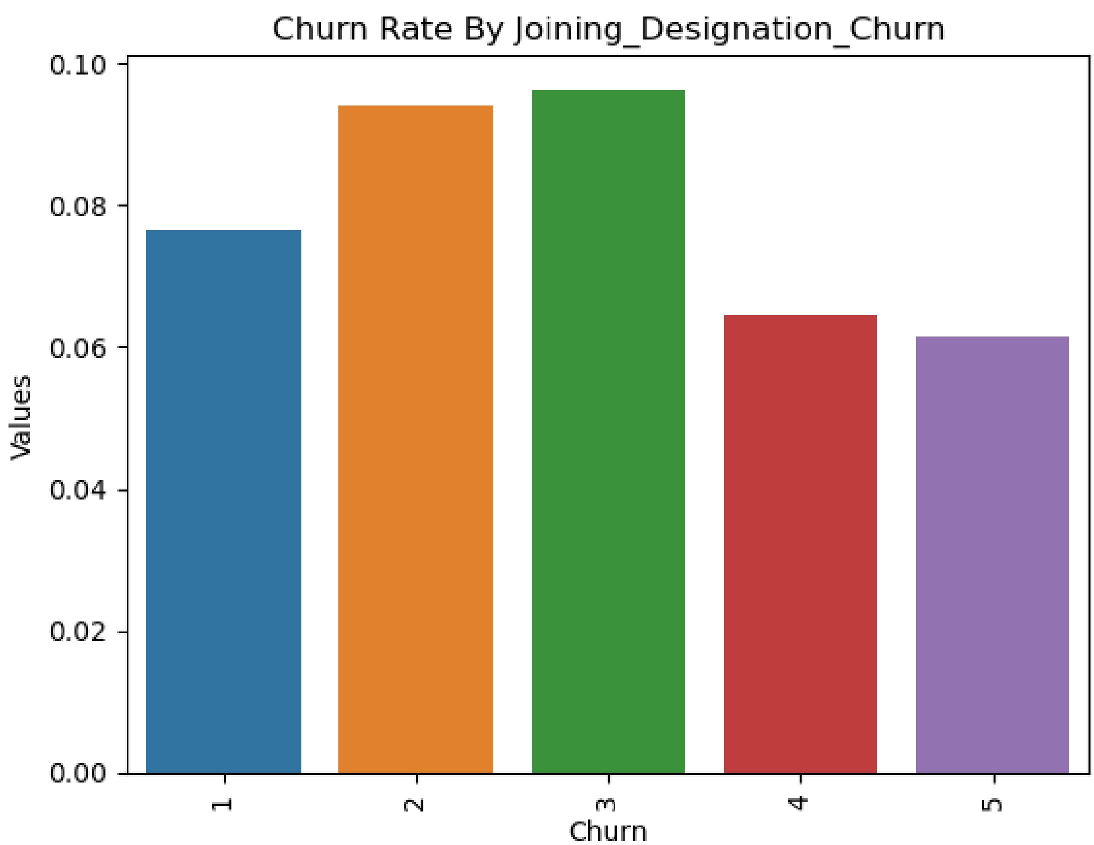
```
C22    0.061805
C23    0.105948
C24    0.083062
C25    0.092466
C26    0.074799
C27    0.076336
C28    0.086384
C29    0.056667
C3     0.081633
C4     0.089965
C5     0.073171
C6     0.083333
C7     0.085386
C8     0.074438
C9     0.101923
Name: Hasleftcompany, dtype: float64
```




```python
Joining_Designation_churn=df.groupby('Joining Designation')
['Hasleftcompany'].mean()
print(Joining_Designation_churn)
sns.barplot(x=Joining_Designation_churn.index,y=Joining_Designation_ch
urn.values)
plt.xlabel("Churn")
```

```
plt.ylabel("Values")
plt.title("Churn Rate By Joining_Designation_Churn")
plt.xticks(rotation = 90)
plt.show()
```

```
Joining Designation
1    0.076493
2    0.094039
3    0.096242
4    0.064516
5    0.061538
Name: Hasleftcompany, dtype: float64
```



Standardization

```
Numerical_Features=df.select_dtypes(include=[np.number])

from sklearn.preprocessing import StandardScaler

Scaler=StandardScaler()
Standarized_data=Scaler.fit_transform(Numerical_Features)
print(Standarized_data)
```

```
[[-1.73196015 -1.74493508 -1.06733399 ... -0.08989762 -0.26403172
  -0.04804685]
 [-1.73177882 -1.74493508 -1.06733399 ... -0.08989762 -0.26403172
  -0.04804685]
 [-1.73159749 -1.74493508 -1.06733399 ... -2.45948835 -0.26403172
  -0.04804685]
 ...
 [ 1.73159749  1.69290216 -0.74721868 ... -0.6765064  -0.26403172
  -0.04804685]
 [ 1.73177882  1.69290216 -0.74721868 ... -0.6765064  -0.26403172
  -0.04804685]
 [ 1.73196015  1.69290216 -0.74721868 ... -0.6765064  -0.26403172
  -0.04804685]]
```

Encoding

```python
Encoded_df=pd.get_dummies(df[['City','Education_Level','Joining
Designation']],drop_first=True)
Encoded_df
```

```
       Education_Level  Joining Designation  City_C10  City_C11
City_C12  \
0                    2                    1     False     False
False
1                    2                    1     False     False
False
2                    2                    1     False     False
False
3                    2                    2     False     False
False
4                    2                    2     False     False
False
...                ...                  ...       ...       ...
...
19099                2                    2     False     False
False
19100                2                    2     False     False
False
19101                2                    2     False     False
False
19102                2                    2     False     False
False
19103                2                    2     False     False
False

       City_C13  City_C14  City_C15  City_C16  City_C17  ...  City_C27
\
0         False     False     False     False     False  ...     False

1         False     False     False     False     False  ...     False
```

```
2          False     False     False     False     False  ...     False
3          False     False     False     False     False  ...     False
4          False     False     False     False     False  ...     False
...          ...       ...       ...       ...       ...  ...       ...
19099      False     False     False     False     False  ...      True
19100      False     False     False     False     False  ...      True
19101      False     False     False     False     False  ...      True
19102      False     False     False     False     False  ...      True
19103      False     False     False     False     False  ...      True

       City_C28  City_C29  City_C3  City_C4  City_C5  City_C6  City_C7
\
0          False     False    False    False    False    False    False
1          False     False    False    False    False    False    False
2          False     False    False    False    False    False    False
3          False     False    False    False    False    False     True
4          False     False    False    False    False    False     True
...          ...       ...      ...      ...      ...      ...      ...
19099      False     False    False    False    False    False    False
19100      False     False    False    False    False    False    False
19101      False     False    False    False    False    False    False
19102      False     False    False    False    False    False    False
19103      False     False    False    False    False    False    False

       City_C8  City_C9
0         False    False
1         False    False
2         False    False
3         False    False
4         False    False
```

```
...        ...      ...
19099    False    False
19100    False    False
19101    False    False
19102    False    False
19103    False    False

[19104 rows x 30 columns]
```

Actionable Insights & Recommendations

INSIGHTS

Recommendations

Questions

Data Structure and Overview

```
df.shape
```

```
(19104, 23)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 19104 entries, 0 to 19103
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            19104 non-null  int64
 1   MMM-YY                19104 non-null  datetime64[ns]
 2   Driver_ID             19104 non-null  int64
 3   Age                   19104 non-null  float64
 4   Gender                19104 non-null  float64
 5   City                  19104 non-null  object
 6   Education_Level       19104 non-null  int64
 7   Income                19104 non-null  int64
 8   Dateofjoining         19104 non-null  datetime64[ns]
 9   LastWorkingDate       1616 non-null   datetime64[ns]
 10  Joining Designation   19104 non-null  int64
 11  Grade                 19104 non-null  int64
 12  Total Business Value  19104 non-null  int64
 13  Quarterly Rating      19104 non-null  int64
 14  Hasleftcompany        19104 non-null  int64
 15  Birthyear             19104 non-null  int64
 16  AgeAtJoining          19104 non-null  int64
 17  Tenure                19104 non-null  int64
```

```
 18   Salary_Range           19104 non-null   category
 19   Quarterly_Range        11425 non-null   category
 20   Riders_Age_Category    19069 non-null   category
 21   Rating_Increased       19104 non-null   int32
 22   Salary_Increased       19104 non-null   int64
dtypes: category(3), datetime64[ns](3), float64(2), int32(1),
int64(13), object(1)
memory usage: 3.0+ MB
```

Descriptive Statistics

```
temp = df[["Age", "Income", "Total Business Value", "Quarterly
Rating"]].aggregate([np.mean, np.median, np.std])
temp

              Age        Income  Total Business Value  Quarterly
Rating
mean    34.668435  65652.025126          5.716621e+05
2.008899
median  34.000000  60087.000000          2.500000e+05
2.000000
std      6.247912  30914.515344          1.128312e+06
1.009832

df['Driver_ID'].nunique()

2381
```

Temporal Analysis

```
df['Hasleftcompany'].value_counts()

Hasleftcompany
0    17488
1     1616
Name: count, dtype: int64

df['Dateofjoining']=pd.to_datetime(df['Dateofjoining'])
df['Month_Year_Joining']=df['Dateofjoining'].dt.to_period('M')
per_month_jooining_count=df.groupby('Month_Year_Joining')
['Driver_ID'].count()
per_month_jooining_count

Month_Year_Joining
2013-04     31
2013-05     24
2013-06     59
2013-07     63
2013-08     33
          ...
2020-08    325
```

```
2020-09    314
2020-10    139
2020-11     93
2020-12     59
Freq: M, Name: Driver_ID, Length: 85, dtype: int64
```

```
Average_Tenure_of_driver=df['Tenure'].mean()
print(f"Average Tenure of Driver is: {Average_Tenure_of_driver:.2f}")
```

```
Average Tenure of Driver is: 2307.53
```

Feature Engineering

```
df['Hasleftcompany'].value_counts()
```

```
Hasleftcompany
0    17488
1     1616
Name: count, dtype: int64
```

```
df['Year_of_Joining']=df['Dateofjoining'].dt.year
df['Year_of_Joining'].value_counts(ascending=False)
```

```
Year_of_Joining
2018    4936
2019    4515
2020    3667
2015    1965
2016    1625
2017    1100
2013     693
2014     603
Name: count, dtype: int64
```

```
df['Month_of_Joining']=df['Dateofjoining'].dt.month_name()
df['Month_of_Joining'].value_counts(ascending=False)
```

```
Month_of_Joining
July         2730
May          2362
October      2095
June         1973
August       1886
November     1867
September    1449
January      1381
December     1261
April        1014
February      684
March         402
Name: count, dtype: int64
```

```
df['Quarter_of_Joining']=df['Dateofjoining'].dt.quarter
df['Quarter_of_Joining'].value_counts(ascending=False)
```

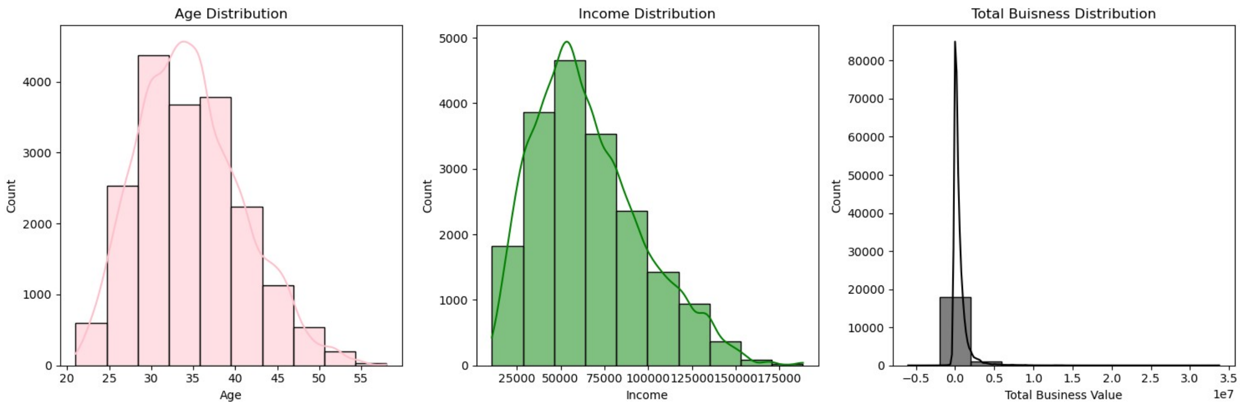
```
Quarter_of_Joining
3    6065
2    5349
4    5223
1    2467
Name: count, dtype: int64
```

```
df['Day_of_Joining']=df['Dateofjoining'].dt.day_name()
df['Day_of_Joining'].value_counts(ascending=False)
```

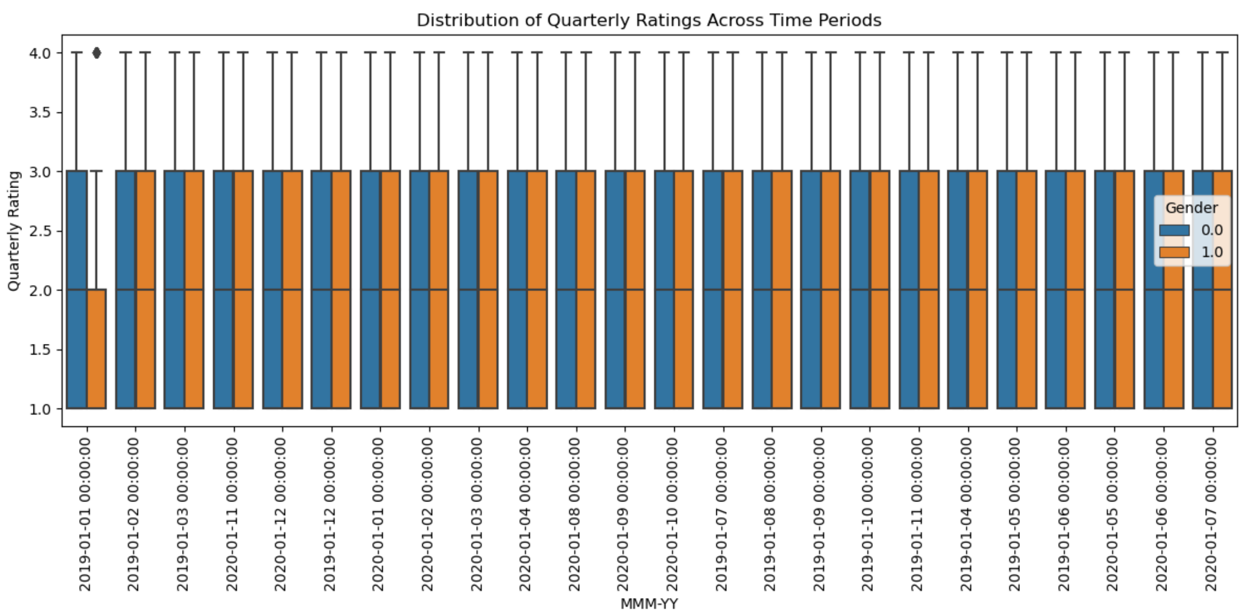
```
Day_of_Joining
Friday       4439
Sunday       3153
Thursday     2823
Monday       2769
Saturday     2705
Tuesday      2103
Wednesday    1112
Name: count, dtype: int64
```

EDA

```
fig,axes=plt.subplots(1,3,figsize=(15,5))

sns.histplot(x="Age",data=df,kde=True,bins=10,ax=axes[0],color='pink')
axes[0].set_title("Age Distribution")

sns.histplot(x='Income',data=df,kde=True,bins=10,ax=axes[1],color='gre
en')
axes[1].set_title("Income Distribution")

sns.histplot(x='Total Business
Value',data=df,kde=True,bins=10,ax=axes[2],color='black')
axes[2].set_title("Total Buisness Distribution")

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

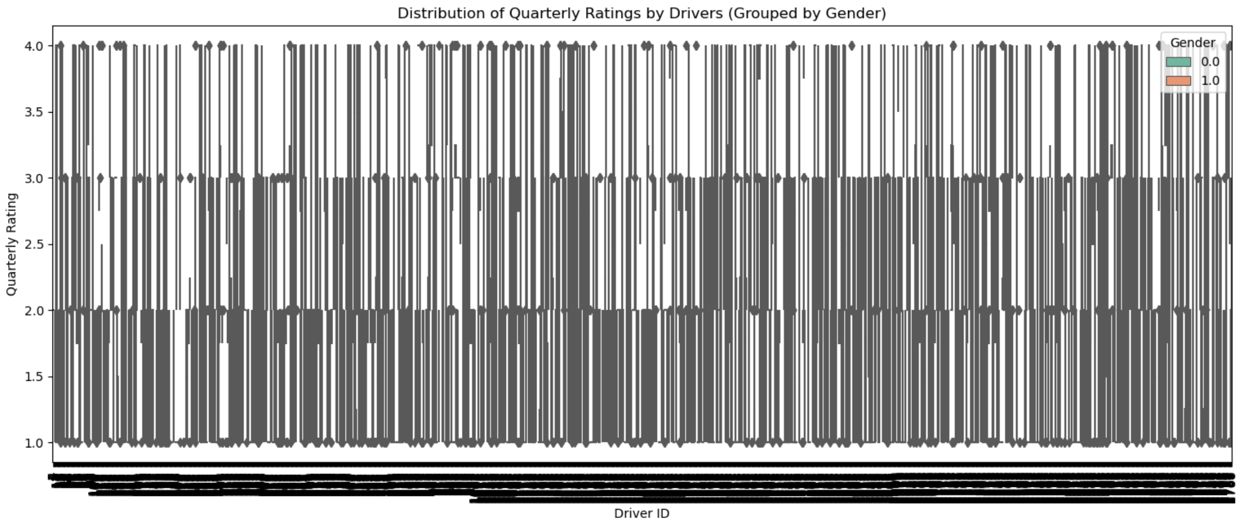



```
plt.figure(figsize=(12,6))
sns.boxplot(x='MMM-YY',y='Quarterly Rating',hue='Gender',data=df)
plt.title('Distribution of Quarterly Ratings Across Time Periods')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

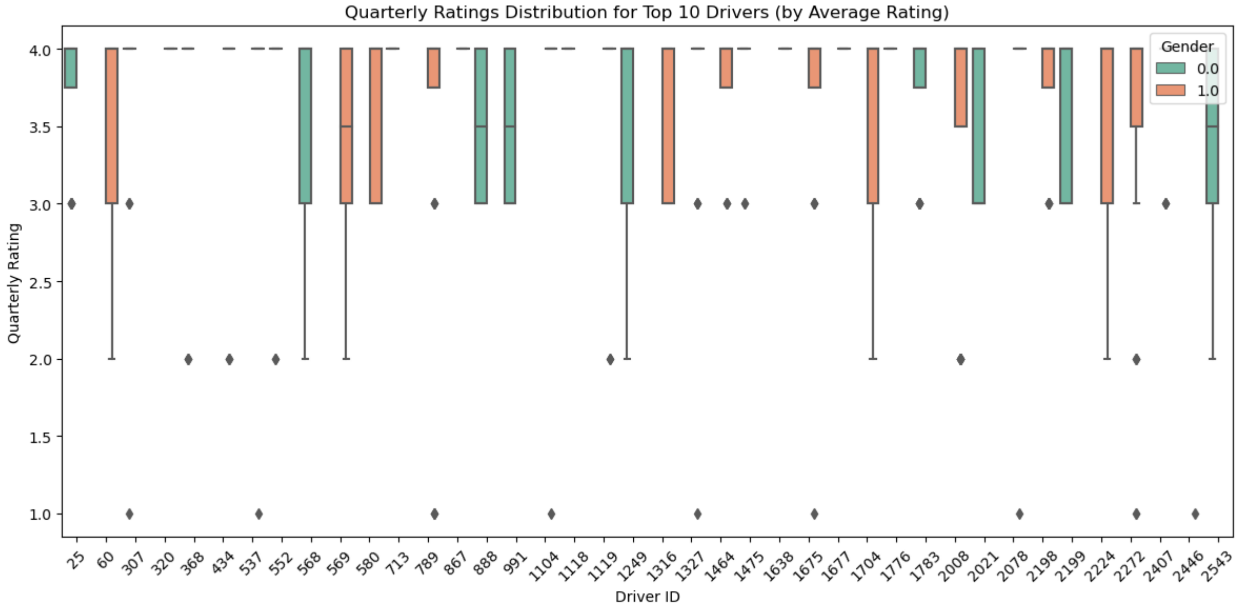



```
plt.figure(figsize=(14, 6))
sns.boxplot(x='Driver_ID', y='Quarterly Rating', hue='Gender',
data=df_filtered, palette="Set2")
plt.xticks(rotation=90)
plt.title('Distribution of Quarterly Ratings by Drivers (Grouped by
Gender)')
plt.xlabel('Driver ID')
plt.ylabel('Quarterly Rating')
plt.legend(title='Gender', loc='upper right')
```

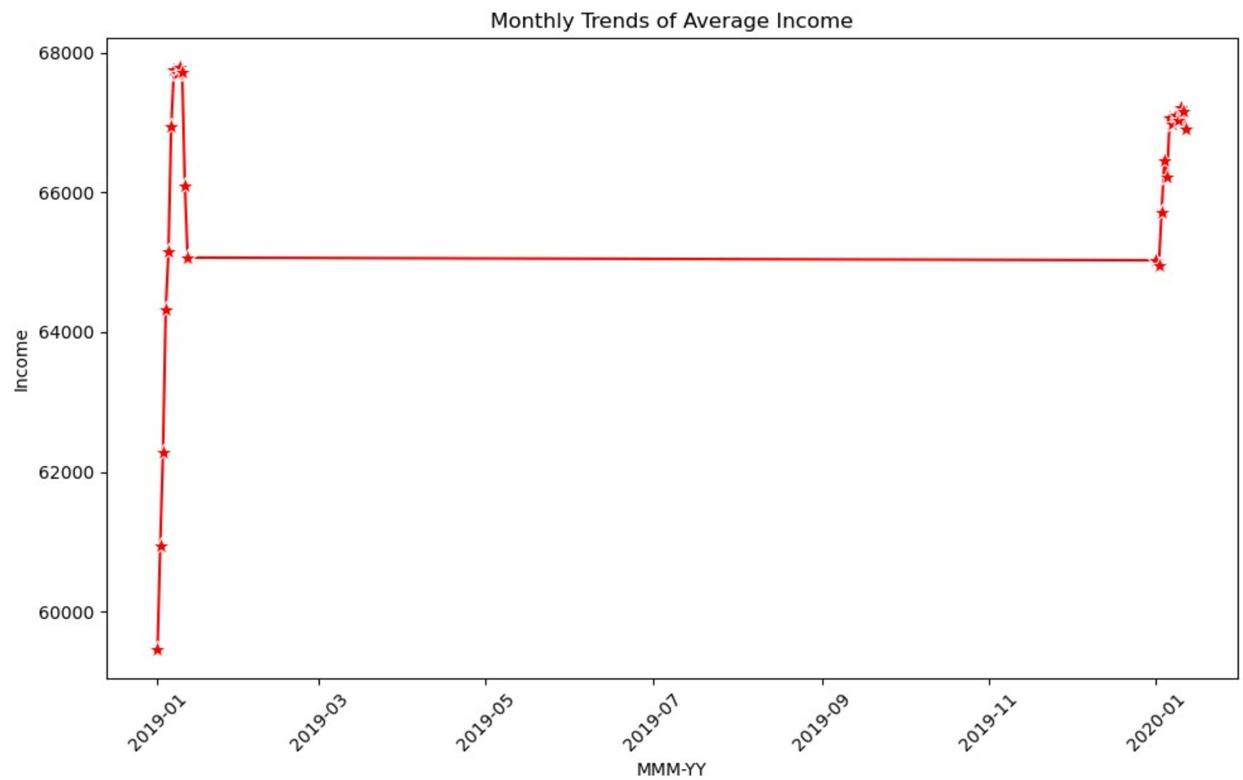
```
plt.tight_layout()
plt.show()
```

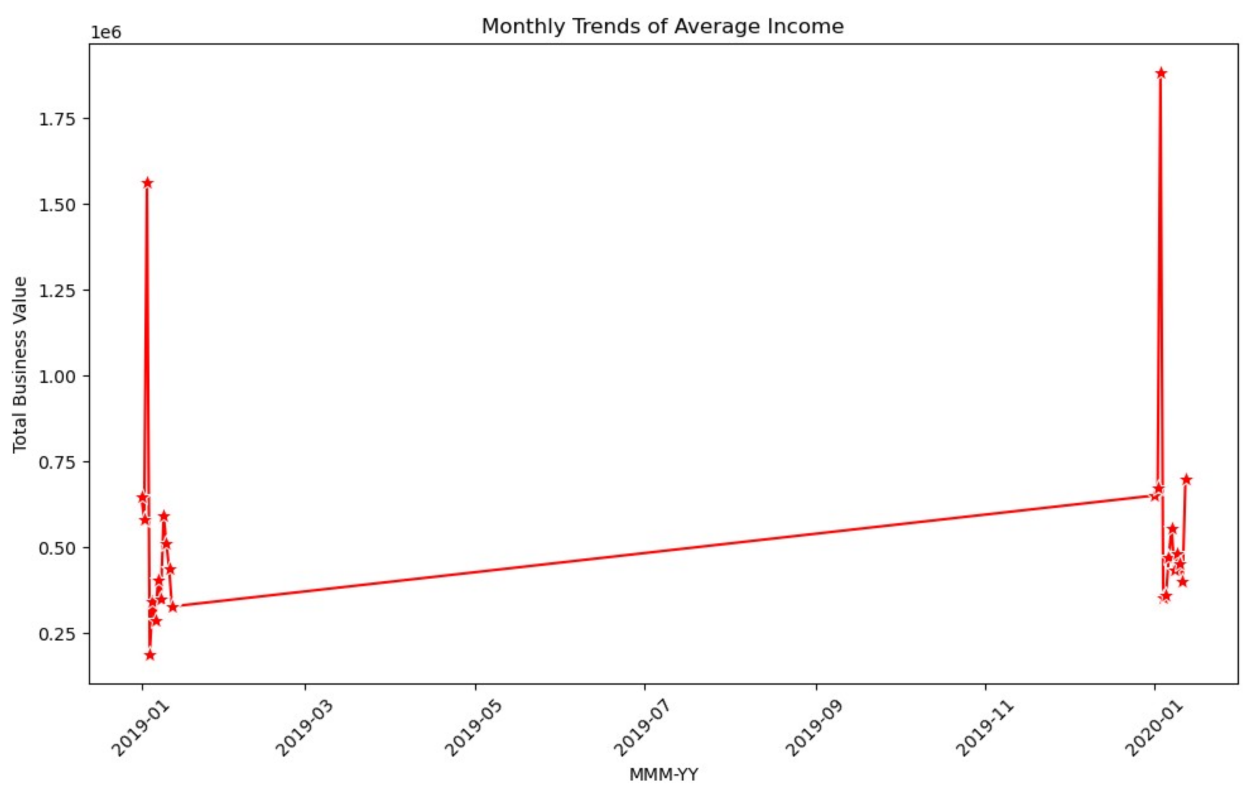



```
top_drivers = (
    df.groupby('Driver_ID')['Quarterly Rating']
    .mean()
    .sort_values(ascending=False)
    .head(40)  # Select top 10 drivers
    .index
)
df_top = df[df['Driver_ID'].isin(top_drivers)]

# Step 3: Create a boxplot for the top 10 drivers
plt.figure(figsize=(12, 6))
sns.boxplot(x='Driver_ID', y='Quarterly Rating', hue='Gender',
data=df_top, palette="Set2")

# Customize the plot
plt.title('Quarterly Ratings Distribution for Top 10 Drivers (by
Average Rating)')
plt.xlabel('Driver ID')
plt.ylabel('Quarterly Rating')
plt.xticks(rotation=45)
plt.legend(title='Gender', loc='upper right')
plt.tight_layout()

# Display the plot
plt.show()
```




```
plt.figure(figsize=(10,6))
monthly_income=df.groupby('MMM-YY')['Income'].mean().reset_index()

sns.lineplot(x='MMM-YY',y='Income',data=monthly_income,marker='*',color='red',markersize=10)
plt.title("Monthly Trends of Average Income")
plt.tight_layout()
plt.xticks(rotation = 45)
plt.show()

plt.figure(figsize=(10,6))
monthly_income=df.groupby('MMM-YY')['Total Business Value'].mean().reset_index()

sns.lineplot(x='MMM-YY',y='Total Business Value',data=monthly_income,marker='*',color='red',markersize=10)
plt.title("Monthly Trends of Average Income")
plt.tight_layout()
plt.xticks(rotation = 45)
plt.show()
```


Monthly Trends of Average Income
Income
MMM-YY
68000
66000
64000
62000
60000
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01


Monthly Trends of Average Income
1e6
Total Business Value
MMM-YY
1.75
1.50
1.25
1.00
0.75
0.50
0.25
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01

Missing Values Handling

```
no_null_Lastworking_date=df.LastWorkingDate.dropna().values
df['LastWorkingDate']=df['LastWorkingDate'].fillna(pd.to_datetime('tod
ay').normalize())


df.LastWorkingDate.isnull().sum()
```

```
0
```

Correlation and Relationships

```
Correlation_age_income=df['Age'].corr(df['Income'])
Correlation_age_income
```

```
0.19084585445978905
```

```
Education_status=df.groupby('Education_Level')['Total Business
Value'].describe()
print("Statics How the Educaton_Level affecting the Total Business
Value")
Education_status
```

```
Statics How the Educaton_Level affecting the Total Business Value

                  count           mean           std        min
25%  \
Education_Level

0                5913.0  565410.657872  1.092937e+06 -2628700.0  0.0

1                6864.0  601287.867133  1.227469e+06 -5483890.0  0.0

2                6327.0  545364.175755  1.044904e+06 -6000000.0  0.0


                      50%       75%         max
Education_Level
0                239180.0  689660.0  23550000.0
1                270885.0  721917.5  33747720.0
2                246450.0  676765.0  17651940.0
```

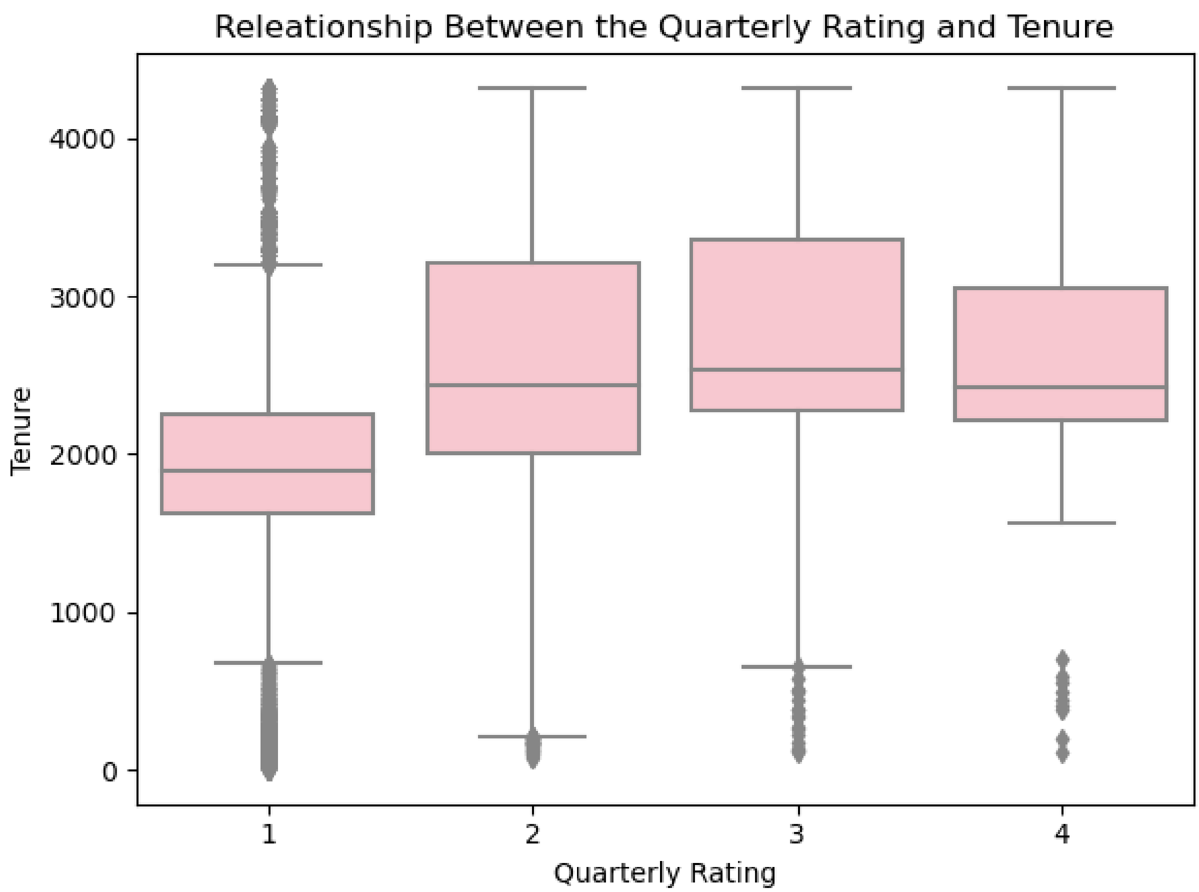
```
Education_status=df.groupby('City')['Total Business Value'].describe()
print("Statics How the City affecting the Total Business Value")
Education_status
```

```
Statics How the City affecting the Total Business Value

       count           mean           std        min  25%       50%
75%  \
City
```

```
C1      677.0  531560.280650  1.028461e+06  -442150.0  0.0  263930.0
602580.0
C10     744.0  540753.736559  9.124536e+05  -500000.0  0.0  251815.0
676185.0
C11     468.0  538549.145299  1.099731e+06  -439300.0  0.0  203935.0
627980.0
C12     727.0  667282.310867  1.246261e+06  -484900.0  0.0  299930.0
848230.0
C13     569.0  796263.075571  2.160790e+06 -1304840.0  0.0  266330.0
794250.0
C14     648.0  607931.635802  1.321692e+06  -582010.0  0.0  203950.0
743022.5
C15     761.0  553266.636005  9.613450e+05  -250000.0  0.0  250000.0
705790.0
C16     709.0  632585.712271  1.186105e+06 -1477940.0  0.0  271470.0
798670.0
C17     440.0  429160.204545  9.620125e+05 -1496650.0  0.0  157025.0
504522.5
C18     544.0  550106.250000  1.112772e+06 -6000000.0  0.0  300280.0
713742.5
C19     579.0  630978.151986  1.393624e+06 -1704230.0  0.0  301750.0
749550.0
C2      472.0  553365.084746  1.147130e+06  -232800.0  0.0  201305.0
565440.0
C20    1008.0  468535.605159  9.012326e+05 -3791250.0  0.0  153470.0
614690.0
C21     603.0  572684.776119  9.606052e+05 -1629620.0  0.0  250000.0
705390.0
C22     809.0  559749.431397  9.485263e+05 -1850000.0  0.0  300000.0
707770.0
C23     538.0  423986.561338  7.554442e+05  -665480.0  0.0  151080.0
499980.0
C24     614.0  584712.426710  1.115906e+06  -647520.0  0.0  252000.0
654282.5
C25     584.0  507575.119863  9.842604e+05  -500000.0  0.0  206200.0
571935.0
C26     869.0  661837.445339  1.386692e+06 -5483890.0  0.0  276610.0
796830.0
C27     786.0  572039.312977  9.689232e+05 -2628700.0  0.0  321580.0
752167.5
C28     683.0  591406.778917  1.144723e+06 -2910060.0  0.0  250000.0
765605.0
C29     900.0  736637.511111  1.332774e+06  -619680.0  0.0  369300.0
854320.0
C3      637.0  458003.940345  7.576680e+05 -1590270.0  0.0  242160.0
574090.0
C4      578.0  556092.266436  1.004997e+06  -500000.0  0.0  203515.0
669765.0
C5      656.0  634855.975610  1.271910e+06  -831520.0  0.0  250000.0
```

```
750000.0
C6     660.0  566042.954545  1.033945e+06  -921510.0  0.0  300000.0
686437.5
C7     609.0  484569.228243  8.848838e+05  -344010.0  0.0  200000.0
600000.0
C8     712.0  566328.539326  9.622600e+05  -154360.0  0.0  273900.0
756330.0
C9     520.0  467914.865385  8.592157e+05  -242830.0  0.0  241755.0
558292.5

             max
City
C1    13737020.0
C10   10703580.0
C11   13435570.0
C12   16873160.0
C13   33747720.0
C14   17651940.0
C15   12506660.0
C16   11454170.0
C17   12836130.0
C18   12849540.0
C19   23550000.0
C2    10951600.0
C20    7799990.0
C21    9000000.0
C22   13588750.0
C23    6350000.0
C24   11040770.0
C25   15059230.0
C26   16979740.0
C27   10305800.0
C28   12200000.0
C29   16606860.0
C3     7435230.0
C4     8560000.0
C5    15993610.0
C6    11894240.0
C7     9100910.0
C8    12941160.0
C9    12701500.0
```

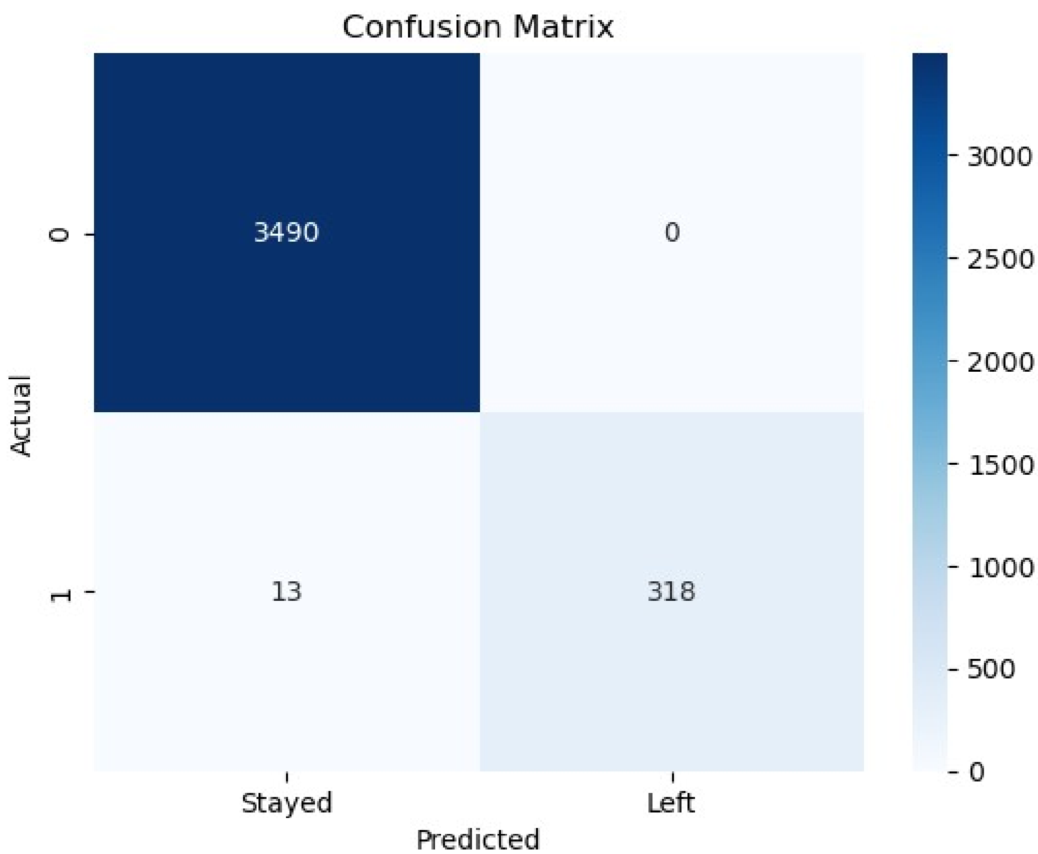
```
sns.boxplot(x='Quarterly Rating',y='Tenure',data=df,color='pink')
plt.title("Releationship Between the Quarterly Rating and Tenure")
plt.tight_layout()
plt.show()
```

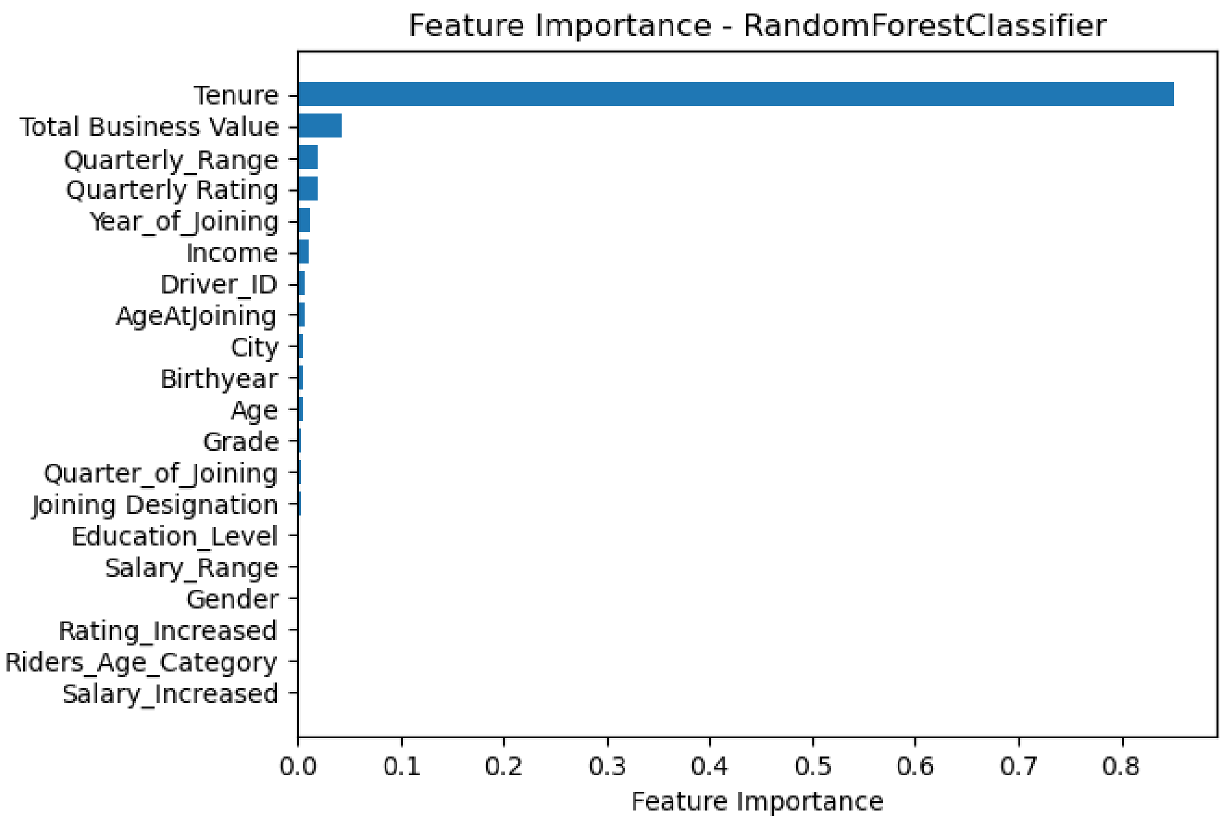



```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix

df=df.drop(columns=['Unnamed: 0'])

label_encoder=LabelEncoder()
df['Gender']=label_encoder.fit_transform(df['Gender'])
df['City']=label_encoder.fit_transform(df['City'])
df['Education_Level'] =
label_encoder.fit_transform(df['Education_Level'])
df['Joining Designation'] = label_encoder.fit_transform(df['Joining
Designation'])
df['Grade'] = label_encoder.fit_transform(df['Grade'])
df['Salary_Range'] = label_encoder.fit_transform(df['Salary_Range'])
df['Quarterly_Range'] =
label_encoder.fit_transform(df['Quarterly_Range'])
df['Riders_Age_Category'] =
label_encoder.fit_transform(df['Riders_Age_Category'])
```

```
X=df.drop(columns=['Hasleftcompany', 'Dateofjoining',
'LastWorkingDate','MMM-
YY','Month_Year_Joining','Month_of_Joining','Day_of_Joining'])
y=df['Hasleftcompany']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,rando
m_state=42)

scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)

model=RandomForestClassifier(random_state=42)
model.fit(X_train_scaled,y_train)

y_pred=model.predict(X_test_scaled)

print('Accuracy',accuracy_score(y_test,y_pred))
print('Classification Report:')
print(classification_report(y_test,y_pred))
print('Confusion Matrix:')
conf_matrix=confusion_matrix(y_test,y_pred)

sns.heatmap(conf_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=['
Stayed','Left'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

features_Importance=model.feature_importances_
feature_name=X.columns
sorted_idx=features_Importance.argsort()

plt.barh(feature_name[sorted_idx],features_Importance[sorted_idx])
plt.xlabel("Feature Importance")
plt.title("Feature Importance - RandomForestClassifier")
plt.show()
```

```
Accuracy 0.9965977492802931
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3490
           1       1.00      0.96      0.98       331

    accuracy                           1.00      3821
   macro avg       1.00      0.98      0.99      3821
weighted avg       1.00      1.00      1.00      3821

Confusion Matrix:
```


Confusion Matrix
3490
0
13
318
0
1
Actual
Stayed
Left
Predicted
3000
2500
2000
1500
1000
500
0



Recommendations

Actionable Insights & Recommdations

```
import os

df.to_csv('Cleaned_Dataset.csv',index=False)

https://app.powerbi.com/links/IfRjsPKjgS?ctid=ed77d40f-8c11-413d-96e2-
467edfd73d60&pbi_source=linkShare
```