```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/ola_driver_scaler.csv')
```

```
df.head()
```

	Unnamed: 0	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjc
0	0	01/01/19	1	28.0	0.0	C23	2	57387	24
1	1	02/01/19	1	28.0	0.0	C23	2	57387	24
2	2	03/01/19	1	28.0	0.0	C23	2	57387	24
3	3	11/01/20	2	31.0	0.0	C7	2	67016	11

```
df.shape
```

```
(19104, 14)
```

```
df.drop(columns='Unnamed: 0', inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   MMM-YY                19104 non-null  object
 1   Driver_ID             19104 non-null  int64
 2   Age                   19043 non-null  float64
 3   Gender                19052 non-null  float64
 4   City                  19104 non-null  object
 5   Education_Level       19104 non-null  int64
 6   Income                19104 non-null  int64
 7   Dateofjoining         19104 non-null  object
 8   LastWorkingDate       1616 non-null   object
 9   Joining Designation   19104 non-null  int64
 10  Grade                 19104 non-null  int64
 11  Total Business Value  19104 non-null  int64
 12  Quarterly Rating      19104 non-null  int64
dtypes: float64(2), int64(7), object(4)
memory usage: 1.9+ MB
```

* Converting features to date time features

```
df['Dateofjoining']=pd.to_datetime(df['Dateofjoining'])
df['LastWorkingDate']=pd.to_datetime(df['LastWorkingDate'])
df['MMM-YY']=pd.to_datetime(df['MMM-YY'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   MMM-YY                19104 non-null  datetime64[ns]
 1   Driver_ID             19104 non-null  int64
 2   Age                   19043 non-null  float64
 3   Gender                19052 non-null  float64
 4   City                  19104 non-null  object
 5   Education_Level       19104 non-null  int64
 6   Income                19104 non-null  int64
 7   Dateofjoining         19104 non-null  datetime64[ns]
 8   LastWorkingDate       1616 non-null   datetime64[ns]
 9   Joining Designation   19104 non-null  int64
 10  Grade                 19104 non-null  int64
 11  Total Business Value  19104 non-null  int64
 12  Quarterly Rating      19104 non-null  int64
dtypes: datetime64[ns](3), float64(2), int64(7), object(1)
memory usage: 1.9+ MB
```

```
df.describe()
```

```
df.describe()
```

	Driver_ID	Age	Gender	Education_Level	Income	J Desig
count	19104.000000	19043.000000	19052.000000	19104.000000	19104.000000	19104.
mean	[illegible]	34.668435	0.418749	1.021671	65652.025126	1.
std	810.705321	6.257912	0.493367	0.800167	30914.515344	0.
min	1.000000	21.000000	0.000000	0.000000	10747.000000	1.
25%	710.000000	30.000000	0.000000	0.000000	42383.000000	1.
50%	1417.000000	34.000000	0.000000	1.000000	60087.000000	1.
75%	2137.000000	39.000000	1.000000	2.000000	83969.000000	2.

Add text cell

```
df.isnull().sum()/len(df) *100
```

```
MMM-YY                  0.000000
Driver_ID               0.000000
Age                     0.319305
Gender                  0.272194
City                    0.000000
Education_Level         0.000000
Income                  0.000000
Dateofjoining           0.000000
LastWorkingDate        91.541039
Joining Designation     0.000000
Grade                   0.000000
Total Business Value    0.000000
Quarterly Rating        0.000000
dtype: float64
```

Most null values are in LastWorkingDate with 91% followed by Age feature has 31% null values and gender has 27% null values

```
!pip install fancyimpute
```

```
Requirement already satisfied: fancyimpute in /usr/local/lib/python3.10/dist-packages (0.7.0)
Requirement already satisfied: knnimpute>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from fancyimpute) (0.1.0)
Requirement already satisfied: scikit-learn>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from fancyimpute) (1.2.2)
Requirement already satisfied: cvxpy in /usr/local/lib/python3.10/dist-packages (from fancyimpute) (1.3.3)
Requirement already satisfied: cvxopt in /usr/local/lib/python3.10/dist-packages (from fancyimpute) (1.3.2)
Requirement already satisfied: pytest in /usr/local/lib/python3.10/dist-packages (from fancyimpute) (7.4.4)
Requirement already satisfied: nose in /usr/local/lib/python3.10/dist-packages (from fancyimpute) (1.3.7)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from knnimpute>=0.1.0->fancyimpute) (1.16.0)
Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.10/dist-packages (from knnimpute>=0.1.0->fancyimpute) (1.25.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.24.2->fancyimpute) (1.1
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.24.2->fancyimpute) (1
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.24.2->fancyimpu
Requirement already satisfied: osqp>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from cvxpy->fancyimpute) (0.6.2.post8)
Requirement already satisfied: ecos>=2 in /usr/local/lib/python3.10/dist-packages (from cvxpy->fancyimpute) (2.0.13)
Requirement already satisfied: scs>=1.1.6 in /usr/local/lib/python3.10/dist-packages (from cvxpy->fancyimpute) (3.2.4.post1)
Requirement already satisfied: setuptools>65.5.1 in /usr/local/lib/python3.10/dist-packages (from cvxpy->fancyimpute) (67.7.2)
Requirement already satisfied: iniconfig in /usr/local/lib/python3.10/dist-packages (from pytest->fancyimpute) (2.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from pytest->fancyimpute) (23.2)
Requirement already satisfied: pluggy<2.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from pytest->fancyimpute) (1.4.0)
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /usr/local/lib/python3.10/dist-packages (from pytest->fancyimpute) (1.2.
Requirement already satisfied: tomli>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pytest->fancyimpute) (2.0.1)
Requirement already satisfied: qdldl in /usr/local/lib/python3.10/dist-packages (from osqp>=0.4.1->cvxpy->fancyimpute) (0.1.7.post0
```

Using KNN imputation to fill null values in age and gender feature

```
from fancyimpute import KNN
knn=KNN()
```

```
df_num_cols= df.select_dtypes(np.number)
```

```
df_num_cols.head()
```

	Driver_ID	Age	Gender	Education_Level	Income	Joining Designation	Grade	Total Business Value	Qua
0	1	28.0	0.0	2	57387	1	1	2381060	
1	[illegible]	[illegible]	0.0	2	57387	1	1	-665480	
2	1	28.0	0.0	2	57387	1	1	0	
3	2	31.0	0.0	2	67016	2	2	0	

Add text cell

```
df_num_cols.isnull().sum()
```

```
Driver_ID                 0
Age                      61
Gender                   52
Education_Level           0
Income                    0
Joining Designation       0
Grade                     0
Total Business Value      0
Quarterly Rating          0
dtype: int64
```

```
df_num_knn= pd.DataFrame(knn.fit_transform(df_num_cols))
```

```
Imputing row 1/19104 with 0 missing, elapsed time: 87.019
Imputing row 101/19104 with 0 missing, elapsed time: 87.020
Imputing row 201/19104 with 0 missing, elapsed time: 87.021
Imputing row 301/19104 with 0 missing, elapsed time: 87.023
Imputing row 401/19104 with 0 missing, elapsed time: 87.023
Imputing row 501/19104 with 0 missing, elapsed time: 87.025
Imputing row 601/19104 with 0 missing, elapsed time: 87.025
Imputing row 701/19104 with 0 missing, elapsed time: 87.025
Imputing row 801/19104 with 0 missing, elapsed time: 87.027
Imputing row 901/19104 with 0 missing, elapsed time: 87.029
Imputing row 1001/19104 with 0 missing, elapsed time: 87.029
Imputing row 1101/19104 with 0 missing, elapsed time: 87.029
Imputing row 1201/19104 with 0 missing, elapsed time: 87.031
Imputing row 1301/19104 with 0 missing, elapsed time: 87.031
Imputing row 1401/19104 with 0 missing, elapsed time: 87.032
Imputing row 1501/19104 with 0 missing, elapsed time: 87.033
Imputing row 1601/19104 with 0 missing, elapsed time: 87.034
Imputing row 1701/19104 with 0 missing, elapsed time: 87.035
Imputing row 1801/19104 with 0 missing, elapsed time: 87.035
Imputing row 1901/19104 with 0 missing, elapsed time: 87.036
Imputing row 2001/19104 with 0 missing, elapsed time: 87.037
Imputing row 2101/19104 with 0 missing, elapsed time: 87.038
Imputing row 2201/19104 with 0 missing, elapsed time: 87.038
Imputing row 2301/19104 with 0 missing, elapsed time: 87.039
Imputing row 2401/19104 with 0 missing, elapsed time: 87.040
Imputing row 2501/19104 with 0 missing, elapsed time: 87.041
Imputing row 2601/19104 with 0 missing, elapsed time: 87.041
Imputing row 2701/19104 with 0 missing, elapsed time: 87.042
Imputing row 2801/19104 with 0 missing, elapsed time: 87.043
Imputing row 2901/19104 with 0 missing, elapsed time: 87.044
Imputing row 3001/19104 with 0 missing, elapsed time: 87.045
Imputing row 3101/19104 with 0 missing, elapsed time: 87.046
Imputing row 3201/19104 with 0 missing, elapsed time: 87.046
Imputing row 3301/19104 with 0 missing, elapsed time: 87.047
Imputing row 3401/19104 with 0 missing, elapsed time: 87.047
Imputing row 3501/19104 with 0 missing, elapsed time: 87.048
Imputing row 3601/19104 with 0 missing, elapsed time: 87.049
Imputing row 3701/19104 with 0 missing, elapsed time: 87.050
Imputing row 3801/19104 with 0 missing, elapsed time: 87.051
Imputing row 3901/19104 with 0 missing, elapsed time: 87.051
Imputing row 4001/19104 with 0 missing, elapsed time: 87.051
Imputing row 4101/19104 with 0 missing, elapsed time: 87.052
Imputing row 4201/19104 with 0 missing, elapsed time: 87.053
Imputing row 4301/19104 with 0 missing, elapsed time: 87.054
Imputing row 4401/19104 with 0 missing, elapsed time: 87.055
Imputing row 4501/19104 with 0 missing, elapsed time: 87.055
Imputing row 4601/19104 with 0 missing, elapsed time: 87.056
Imputing row 4701/19104 with 0 missing, elapsed time: 87.056
Imputing row 4801/19104 with 0 missing, elapsed time: 87.058
Imputing row 4901/19104 with 0 missing, elapsed time: 87.059
Imputing row 5001/19104 with 0 missing, elapsed time: 87.059
Imputing row 5101/19104 with 0 missing, elapsed time: 87.059
Imputing row 5201/19104 with 0 missing, elapsed time: 87.059
Imputing row 5301/19104 with 0 missing, elapsed time: 87.061
Imputing row 5401/19104 with 0 missing, elapsed time: 87.062
Imputing row 5501/19104 with 0 missing, elapsed time: 87.062
Imputing row 5601/19104 with 0 missing, elapsed time: 87.063
Imputing row 5701/19104 with 0 missing, elapsed time: 87.063
```

```
df_num_knn.columns= df_num_cols.columns
```

```
df_num_knn.head()
```

	Driver_ID	Age	Gender	Education_Level	Income	Joining Designation	Grade	Total Business Value	Qu
0	[illegible]	[illegible]	0.0	2.0	57387.0	1.0	1.0	2381060.0	
1	1.0	28.0	0.0	2.0	57387.0	1.0	1.0	-665480.0	
2	1.0	28.0	0.0	2.0	57387.0	1.0	1.0	0.0	
3	2.0	31.0	0.0	2.0	67016.0	2.0	2.0	0.0	

```
df_num_knn.isnull().sum()
```

```
Driver_ID              0
Age                    0
Gender                 0
Education_Level        0
Income                 0
Joining Designation    0
Grade                  0
Total Business Value   0
Quarterly Rating       0
dtype: int64
```

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

```
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
```

replacing Age and Gender column in df with imputed values

```
df['Age']=df_num_knn['Age']
df['Gender']=df_num_knn['Gender']
```

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

```
MMM-YY                      0
Driver_ID                   0
Age                         0
Gender                      0
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
```

creating a new dataframe df1 with aggregation and drivers level

```
df1=pd.DataFrame()
df1['Driver_ID']= df['Driver_ID'].unique()
```

```
df1
```

	Driver_ID
0	1
1	2
2	
3	5
4	6
...	...
2376	2784
2377	2785
2378	2786
2379	2787
2380	2788

2381 rows × 1 columns

```
df1['Age']= list(df.groupby("Driver_ID").agg({'Age':'last'})['Age'])
df1['Gender']= list(df.groupby("Driver_ID").agg({'Gender':'last'})['Gender'])
df1['City']= list(df.groupby("Driver_ID").agg({'City':'last'})['City'])
df1['Education_Level']= list(df.groupby("Driver_ID").agg({'Education_Level':'last'})['Education_Level'])
df1['Income']= list(df.groupby("Driver_ID").agg({'Income':'last'})['Income'])
df1['Dateofjoining']= list(df.groupby("Driver_ID").agg({'Dateofjoining':'last'})['Dateofjoining'])
df1['LastWorkingDate']= list(df.groupby("Driver_ID").agg({'LastWorkingDate':'last'})['LastWorkingDate'])
df1['Joining Designation']= list(df.groupby("Driver_ID").agg({'Joining Designation':'last'})['Joining Designation'])
df1['Grade']= list(df.groupby("Driver_ID").agg({'Grade':'last'})['Grade'])
df1['Total Business Value']= list(df.groupby("Driver_ID").agg({'Total Business Value':'sum'})['Total Business Value'])
df1['Quarterly Rating']= list(df.groupby("Driver_ID").agg({'Quarterly Rating':'last'})['Quarterly Rating'])
```

```
df1.head()
```

	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingD
0	1	28.0	0.0	C23	2	57387	2018-12-24	2019-0
1	2	31.0	0.0	C7	2	67016	2020-11-06	
2	4	43.0	0.0	C13	2	65603	2019-12-07	2020-0
3	5	29.0	0.0	C9	0	46368	2019-01-09	2019-0

```
df1.shape
```

```
(2381, 12)
```

Creating Target feature which shows whether the driver is churned or not

```
df1['Target']= np.where(pd.notnull(df1['LastWorkingDate']),1,0)
```

creating new feature which shows whether Quarterly rating of a driver has increased or not

```
df1['qrf']= list(df.groupby('Driver_ID').agg({'Quarterly Rating':"first"})['Quarterly Rating'])
df1['Increase_in_QR']= np.where(df1["Quarterly Rating"]>df1['qrf'],1,0)
```

```
df1['Increase_in_QR'].value_counts()
```

```
0    2023
1     358
Name: Increase_in_QR, dtype: int64
```

creating new feature which shows whether Income of a driver has increased or not

```
df1['Income_first']= list(df.groupby('Driver_ID').agg({'Income':"first"})['Income'])
df1['Increase_in_Income']= np.where(df1["Income"]>df1['Income_first'],1,0)
```

```
df1['Target'].value_counts()
```

```
1    1616
0     765
Name: Target, dtype: int64
```

```
df1.drop(columns=['LastWorkingDate','Dateofjoining'],inplace=True)
```

Add text cell

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Driver_ID             2381 non-null   int64
 1   Age                   2381 non-null   float64
 2   Gender                2381 non-null   float64
 3   City                  2381 non-null   object
 4   Education_Level       2381 non-null   int64
 5   Income                2381 non-null   int64
 6   Joining Designation   2381 non-null   int64
 7   Grade                 2381 non-null   int64
 8   Total Business Value  2381 non-null   int64
 9   Quarterly Rating      2381 non-null   int64
 10  Target                2381 non-null   int64
 11  qrf                   2381 non-null   int64
 12  Increase_in_QR        2381 non-null   int64
 13  Income_first          2381 non-null   int64
 14  Increase_in_Income    2381 non-null   int64
dtypes: float64(2), int64(12), object(1)
memory usage: 279.1+ KB
```

```
#EDA
df1.describe()
```

	Driver_ID	Age	Gender	Education_Level	Income	Joini Designati
count	2381.000000	2381.000000	2381.000000	2381.00000	2381.000000	2381.000(
mean	1397.559009	33.668219	0.409995	1.00756	59334.157077	1.820
std	806.161628	5.978597	0.491589	0.81629	28383.666384	0.841
min	1.000000	21.000000	0.000000	0.00000	10747.000000	1.000(
25%	695.000000	29.000000	0.000000	0.00000	39104.000000	1.000(
50%	1400.000000	33.000000	0.000000	1.00000	55315.000000	2.000(
75%	2100.000000	37.000000	1.000000	2.00000	75986.000000	2.000(
max	2788.000000	58.000000	1.000000	2.00000	188418.000000	5.000(

```
n=['City', 'Gender', 'Education_Level','Quarterly Rating', 'Increase_in_Income', 'Increase_in_QR']
for i in n:
  print(df1[i].value_counts(normalize=True)*100)
  print('-'*60)
```

```
C20    6.383872
C15    4.241915
C29    4.031919
C26    3.905922
C8     3.737925
C27    3.737925
C10    3.611928
C16    3.527929
C22    3.443931
C3     3.443931
C28    3.443931
C12    3.401932
C5     3.359933
C1     3.359933
C21    3.317934
C14    3.317934
C6     3.275934
C4     3.233935
C7     3.191936
C9     3.149937
C25    3.107938
C23    3.107938
C24    3.065939
C19    3.023940
C2     3.023940
C17    2.981940
```

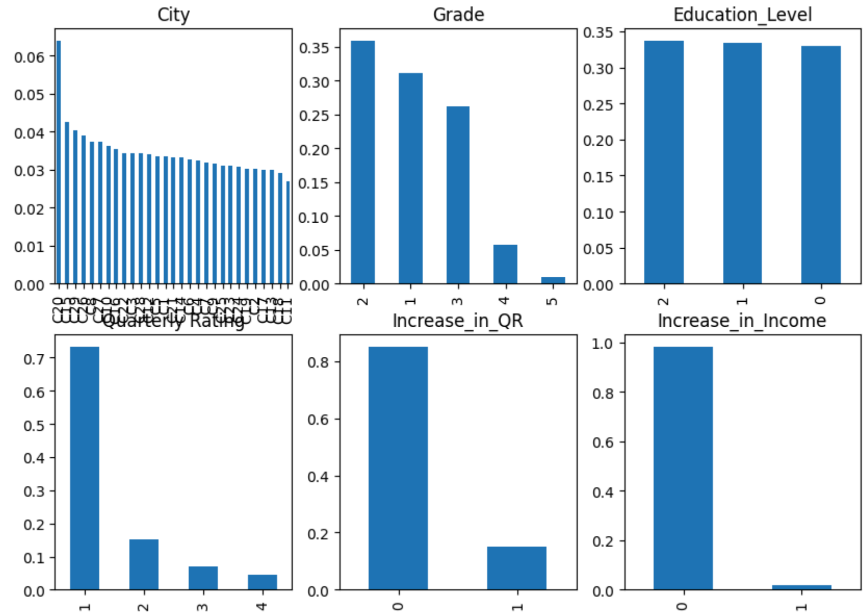
```
C13    2.981940
C18    2.897942
C11    2.687946
Name: City, dtype: float64
------------------------------------------------------------
0.000000e+00    58.798824
1.000000e-      5183
7.814222e-      1999
5.444811e-01     0.041999
9.999999e-01     0.041999
2.056423e-11     0.041999
8.175254e-05     0.041999
2.142856e-01     0.041999
6.505287e-12     0.041999
6.572146e-01     0.041999
Name: Gender, dtype: float64
------------------------------------------------------------
2    33.683326
1    33.389332
0    32.927341
Name: Education_Level, dtype: float64
------------------------------------------------------------
1    73.246535
2    15.203696
3     7.055859
4     4.493910
Name: Quarterly Rating, dtype: float64
------------------------------------------------------------
0    98.194036
1     1.805964
Name: Increase_in_Income, dtype: float64
------------------------------------------------------------
```

Add text cell

```
plt.subplots(figsize=(10,7))

plt.subplot(231)
df1['City'].value_counts(normalize=True).plot.bar(title='City')
plt.subplot(232)
df1['Grade'].value_counts(normalize=True).plot.bar(title='Grade')
plt.subplot(233)
df1['Education_Level'].value_counts(normalize=True).plot.bar(title='Education_Level')
plt.subplot(234)
df1['Quarterly Rating'].value_counts(normalize=True).plot.bar(title='Quarterly Rating')
plt.subplot(235)
df1['Increase_in_QR'].value_counts(normalize=True).plot.bar(title='Increase_in_QR')
plt.subplot(236)
df1['Increase_in_Income'].value_counts(normalize=True).plot.bar(title='Increase_in_Income')
```

```
<ipython-input-37-5cdd1cd5b52a>:3: MatplotlibDeprecationWarning: Auto-removal of over
  plt.subplot(231)
<Axes: title={'center': 'Increase_in_Income'}>
```



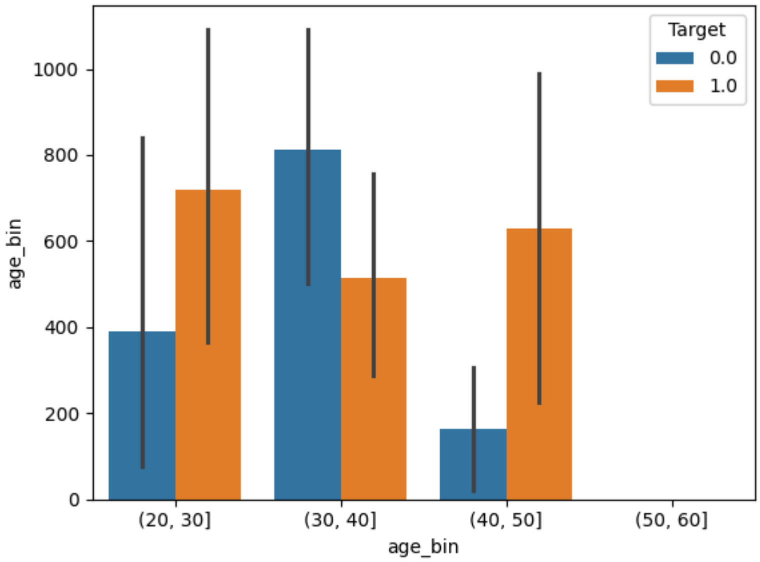
- Around 35% of drivers have grade 2 and 30% have grade 1
- the percentage of drivers with education level 0, 1, 2 are almost equal with 33% each
- Most of the drivers have quarterly rating 1 which is around 73% folowed by 2 with 15% and only 4% have 4 rating
- Only 1.8% Add text cell ve increase in there income
- 85% of drivers have no increse in quarterly rating

```
df1['age_bin']=pd.cut(df1['Age'], bins= [20,30,40,50,60])
agebin= pd.crosstab(df1['age_bin'], df1['Target'])
agebin
```

Target	0	1
age_bin		
(20, 30]	220	543
(30, 40]	434	857
(40, 50]	105	202
(50, 60]	6	14

```
sns.barplot(x=df1['age_bin'], y=df1['age_bin'].value_counts(),hue=df1['Target'])
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_core/data.py:265: RuntimeWarning: '<
  frame = pd.DataFrame(plot_data)
/usr/local/lib/python3.10/dist-packages/seaborn/_core/data.py:265: RuntimeWarning: '<
  frame = pd.DataFrame(plot_data)
<Axes: xlabel='age_bin', ylabel='age_bin'>
```

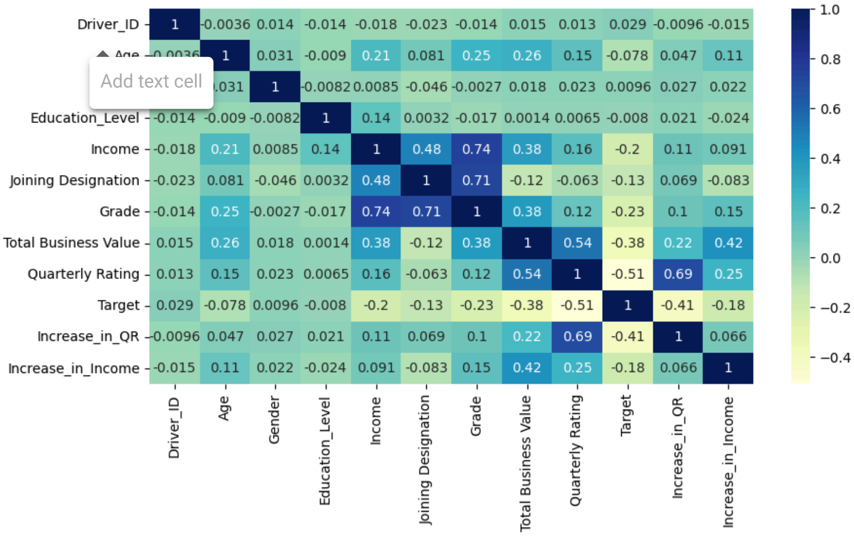


- The churn rate is much higher among drivers of age between 40 to 50 followed by 20 to 30
- While churn rate is lower for drivers of age between 30 to 40

```
df1.drop(columns='Income_first', inplace=True)
df1.drop(columns='qrf', inplace=True)
```

```
plt.figure(figsize=(10,5))
sns.heatmap(df1.corr(), annot=True, cmap='YlGnBu')
```

```
<ipython-input-41-a5690c0109c4>:2: FutureWarning: The default value of numeric_only i
  sns.heatmap(df1.corr(), annot=True, cmap='YlGnBu')
<Axes: >
```



- Churn rate which is Target feature is negatively correalted to quarterly rating and increase in income which mean drivers with more quarterly rating are less likely to churn.
- Total business value is positively correalted to quarterly rating
- Income of driver is positively correlated to grade, which means better the grade better the income

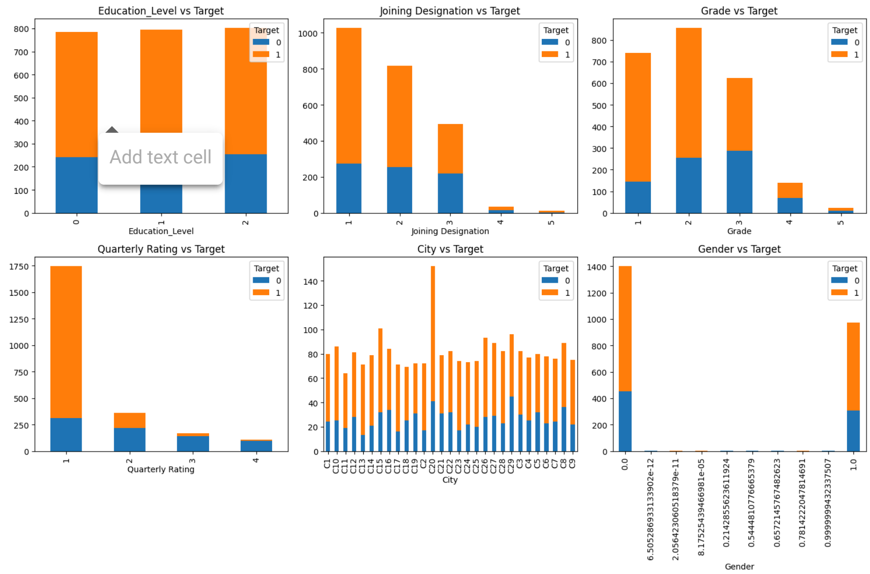
```
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# Flatten the axes array for easier iteration
axes = axes.flatten()

# Plot each crosstab on a separate subplot
crosstabs = [
    ('Education_Level', 'Target'),
    ('Joining Designation', 'Target'),
    ('Grade', 'Target'),
    ('Quarterly Rating', 'Target'),
    ('City', 'Target'),
    ('Gender', 'Target')
]

for ax, (column1, column2) in zip(axes, crosstabs):
    pd.crosstab(df1[column1], df1[column2]).plot(kind='bar', stacked=True, ax=ax)
    ax.set_title(f'{column1} vs {column2}')

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



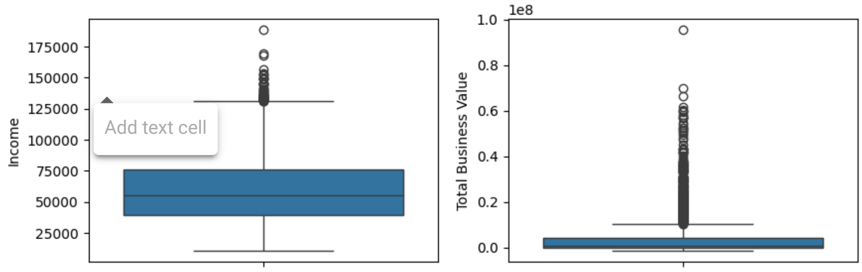
- The number of drivers churned with different Education level are almost equal
- Drivers with joining designation as 1 and 2 are the most churned
- Drivers with less Grade(1 & 2) churned more when compared to higher grade drivers(3& 4)
- Drivers with qurterly rating 1 have churned more
- The churn rate of drivers is hgihest in city c20

```
#Handling outliers
plt.figure(figsize=(10,7))

plt.subplot(221)
sns.boxplot(df1['Income'])

plt.subplot(222)
sns.boxplot(df1['Total Business Value'])
```

```
<Axes: ylabel='Total Business Value'>
```



- There are few outliers in income and many outliers in total business value
- As we do not have huge dataset, not treating outliers

MODEL BUILDING

```
#OHE
OHE= pd.get_dummies(df1['City'],prefix='City')
df1=pd.concat([df1,OHE], axis=1)
df1.drop(columns='City', inplace=True)
```

One hot encoding on city feature as it not ordinal

```
df1.drop(columns='Driver_ID', inplace=True)
```

```
df1.drop(columns='age_bin', inplace=True)
```

Spliting the data into train and test with 80% and 20% respectively

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(df1.drop(columns=['Target']), df1['Target'], train_size=0.2, random_state=2)
```

Scaling the data

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
X_scaled= scaler.fit_transform(X_train)
```

Buidling model using Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
tree= RandomForestClassifier(class_weight='balanced')
params= {'max_depth':[2,3,4,5], 'n_estimators': [50, 100, 150, 200]}

clf=GridSearchCV(tree, params, cv=5, scoring='f1')
clf.fit(X_train, y_train)
clf.best_params_
clf.best_score_
```

```
0.8534325644011259
```

```
from sklearn.metrics import classification_report
y_pred=clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.62      0.66       617
           1       0.83      0.88      0.85      1288

    accuracy                           0.79      1905
   macro avg       0.77      0.75      0.75      1905
weighted avg       0.79      0.79      0.79      1905
```

The accuracy of the Random Forest model is 80% and F1 score is 86%

Buidling a model using Gradient boosting algorithm

```
from sklearn.ensemble import GradientBoostingClassifier
params= {'max_depth':[3,4,5,6,7], 'n_estimators':[50,75,100,125,150], 'learning_rate':[0.1,0.2,0.3,0.4,0.5]}
GBDT= GradientBoostingClassifier()
GBGS= GridSearchCV(GBDT, params, cv=5)
GBGS.fit(X_train, y_train)
GBGS.best_params_
GBGS.best_score_
print(classification_report(y_test, GBGS.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.76      0.55      0.64       617
           1       0.81      0.91      0.86      1288

    accuracy                           0.80      1905
   macro avg       0.78      0.73      0.75      1905
weighted avg       0.79      0.80      0.79      1905
```

The accuracy of the Gradient boosting model is 80% and F1 score is 86% which is similar to Random Forest model

Buidling a model using Extreme Gradient boosting algorithm

```
from xgboost import XGBClassifier
XGB= XGBClassifier(class_weight='balanced')
XGB.fit(X_train, y_train)
y_pred=XGB.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.64      0.54      0.59       617
           1       0.80      0.85      0.82      1288

    accuracy                           0.75      1905
   macro avg       0.72      0.70      0.70      1905
```