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

!gdown
'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/49
2/original/ola_driver_scaler.csv'
```

```
Downloading...
From:
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/492
/original/ola_driver_scaler.csv
To: /content/ola_driver_scaler.csv
   0% 0.00/1.13M [00:00<?, ?B/s] 100% 1.13M/1.13M [00:00<00:00,
16.8MB/s]
```

```
df=pd.read_csv("ola_driver_scaler.csv")
df.head()
```

```
/usr/local/lib/python3.11/dist-packages/google/colab/
_dataframe_summarizer.py:88: UserWarning: Could not infer format, so
each element will be parsed individually, falling back to `dateutil`.
To ensure parsing is consistent and as-expected, please specify a
format.
  cast_date_col = pd.to_datetime(column, errors="coerce")
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 19104,\n  \"fields\":
[\n    {\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 5514,\n        \"min\": 0,\n
\"max\": 19103,\n        \"num_unique_values\": 19104,\n
\"samples\": [\n          18299,\n          9376,\n          4518\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MMM-YY\",\n      \"properties\":
{\n        \"dtype\": \"object\",\n        \"num_unique_values\": 24,\
n        \"samples\": [\n          \"03/01/20\",\n
\"10/01/19\",\n          \"01/01/19\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Driver_ID\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
810,\n        \"min\": 1,\n        \"max\": 2788,\n
\"num_unique_values\": 2381,\n        \"samples\": [\n          1663,\
n          1264,\n          1618\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 6.2579116861907345,\n
\"min\": 21.0,\n        \"max\": 58.0,\n        \"num_unique_values\":
36,\n        \"samples\": [\n          58.0,\n          41.0,\n
24.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"Gender\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.4933670037660394,\n        \"min\": 0.0,\n        \"max\":
1.0,\n        \"num_unique_values\": 2,\n        \"samples\": [\n
1.0,\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"City\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 29,\n        \"samples\": [\n
\"C22\",\n          \"C5\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Education_Level\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 2,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          2,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Income\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 30914,\n        \"min\": 10747,\n
\"max\": 188418,\n        \"num_unique_values\": 2383,\n
\"samples\": [\n          44273,\n          35370\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Dateofjoining\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 869,\n        \"samples\": [\n
\"14/09/19\",\n          \"01/06/18\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"LastWorkingDate\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2018-12-31 00:00:00\",\n        \"max\": \"2020-12-28 00:00:00\",\n
\"num_unique_values\": 493,\n        \"samples\": [\n
\"05/03/20\",\n          \"10/01/19\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Joining Designation\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 1,\n        \"max\": 5,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          2,\n
5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Grade\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 1,\n        \"max\": 5,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          2,\n
5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Business Value\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1128312,\n        \"min\": -6000000,\n
\"max\": 33747720,\n        \"num_unique_values\": 10181,\n
\"samples\": [\n          431090,\n          720180\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Quarterly Rating\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 1,\n        \"max\": 4,\n
```

```
\"num_unique_values\": 4,\n        \"samples\": [\n          1,\n
3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
df.shape
```

```
(19104, 14)
```

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

```
<ipython-input-509-1047511c8b2d>:1: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df['Dateofjoining']=pd.to_datetime(df['Dateofjoining'])
<ipython-input-509-1047511c8b2d>:2: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df['MMM-YY']=pd.to_datetime(df['MMM-YY'])
```

```
Unnamed: 0                     int64
MMM-YY                datetime64[ns]
Driver_ID                      int64
```

```
Age                                 float64
Gender                              float64
City                                 object
Education_Level                       int64
Income                                int64
Dateofjoining                datetime64[ns]
LastWorkingDate                      object
Joining Designation                   int64
Grade                                 int64
Total Business Value                  int64
Quarterly Rating                      int64
dtype: object
```

```
df.describe(include=np.number).T
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10,\n  \"fields\": [\n{\n      \"column\": \"count\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 23.91675377452197,\n\"min\": 19043.0,\n        \"max\": 19104.0,\n\"num_unique_values\": 3,\n        \"samples\": [\n          19104.0,\n          19043.0,\n          19052.0\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"mean\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 179248.01213311238,\n\"min\": 0.4187486878018056,\n        \"max\": 571662.074958124,\n\"num_unique_values\": 10,\n        \"samples\": [\n571662.074958124,\n          1415.5911327470687,\n65652.02512562813\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\":\"std\",\n      \"properties\": {\n        \"dtype\": \"number\",\n\"std\": 355624.8904607991,\n        \"min\": 0.4933670037660394,\n\"max\": 1128312.2184612986,\n        \"num_unique_values\": 10,\n\"samples\": [\n          1128312.2184612986,\n810.7053208637183,\n          30914.515344441774\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"min\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 1897748.089325744,\n\"min\": -6000000.0,\n        \"max\": 10747.0,\n\"num_unique_values\": 5,\n        \"samples\": [\n          1.0,\n-6000000.0,\n          21.0\n        ],\n        \"semantic_type\":\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n\"column\": \"25%\",\n      \"properties\": {\n        \"dtype\":\"number\",\n        \"std\": 13292.327422473838,\n        \"min\":0.0,\n        \"max\": 42383.0,\n        \"num_unique_values\": 6,\n\"samples\": [\n          4775.75,\n          710.0,\n          1.0\n],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n}\n    },\n    {\n      \"column\": \"50%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 78805.98945386561,\n\"min\": 0.0,\n        \"max\": 250000.0,\n\"num_unique_values\": 8,\n        \"samples\": [\n          1417.0,\n
```

```
60087.0,\n          9551.5\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"75%\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 219294.67694114902,\n        \"min\":
1.0,\n        \"max\": 699700.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          2137.0,\n          83969.0,\n
14327.25\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"max\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 10664735.22364782,\n        \"min\": 1.0,\n        \"max\":
33747720.0,\n        \"num_unique_values\": 9,\n        \"samples\":
[\n          33747720.0,\n          2788.0,\n          188418.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
df.describe(include='object').T
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2,\n  \"fields\": [\n
{\n      \"column\": \"count\",\n      \"properties\": {\n
\"dtype\": \"date\",\n        \"min\": \"1616\",\n        \"max\":
\"19104\",\n        \"num_unique_values\": 2,\n        \"samples\": [\
n          \"1616\",\n          \"19104\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"unique\",\n      \"properties\":
{\n        \"dtype\": \"date\",\n        \"min\": 29,\n
\"max\": 493,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          493,\n          29\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"top\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"29/07/20\",\n          \"C20\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"freq\",\n      \"properties\": {\n        \"dtype\": \"date\",\n
\"min\": \"70\",\n        \"max\": \"1008\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"70\",\n
\"1008\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
df.nunique()
```

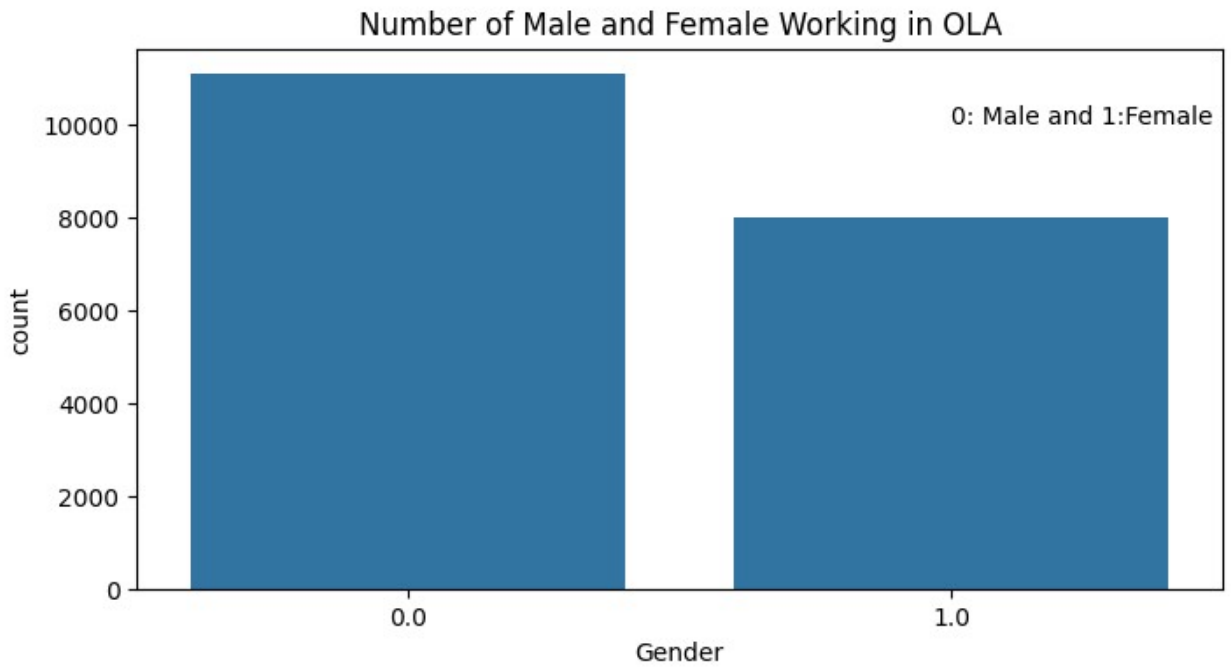
```
Unnamed: 0              19104
MMM-YY                     24
Driver_ID                2381
Age                        36
Gender                      2
City                       29
Education_Level             3
Income                   2383
Dateofjoining             869
```

```
LastWorkingDate            493
Joining Designation          5
Grade                        5
Total Business Value     10181
Quarterly Rating             4
dtype: int64
```

```
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
```

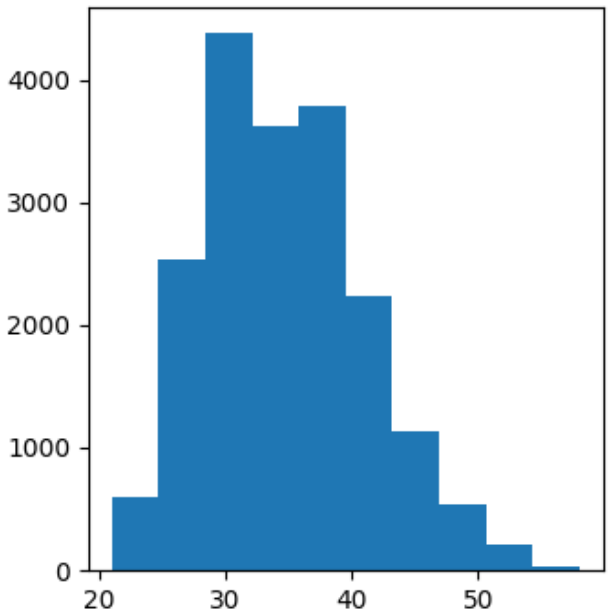
```
df.drop(columns='Unnamed: 0',inplace=True)

plt.figure(figsize=(8,4))
sns.countplot(x='Gender',data=df)
plt.title("Number of Male and Female Working in OLA")
plt.text(1,10000,'''0: Male and 1:Female''')
plt.show()
```

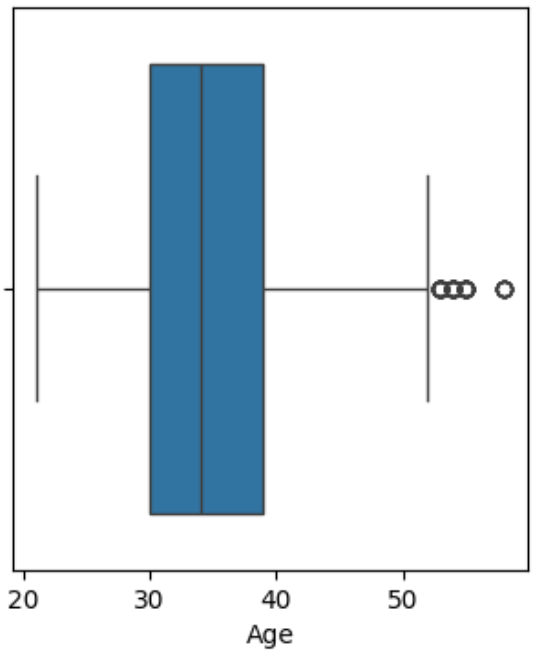


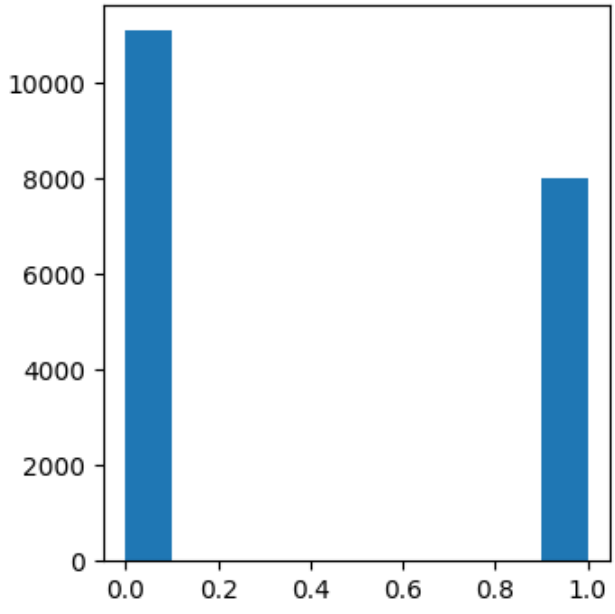

```
num=df.select_dtypes(include=np.number)
num.drop(columns='Driver_ID',inplace=True)
num.columns
```

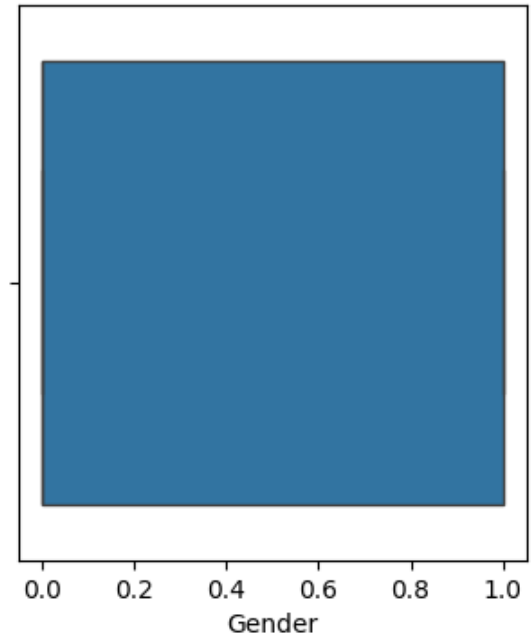
```
Index(['Age', 'Gender', 'Education_Level', 'Income', 'Joining
Designation',
       'Grade', 'Total Business Value', 'Quarterly Rating'],
      dtype='object')
```

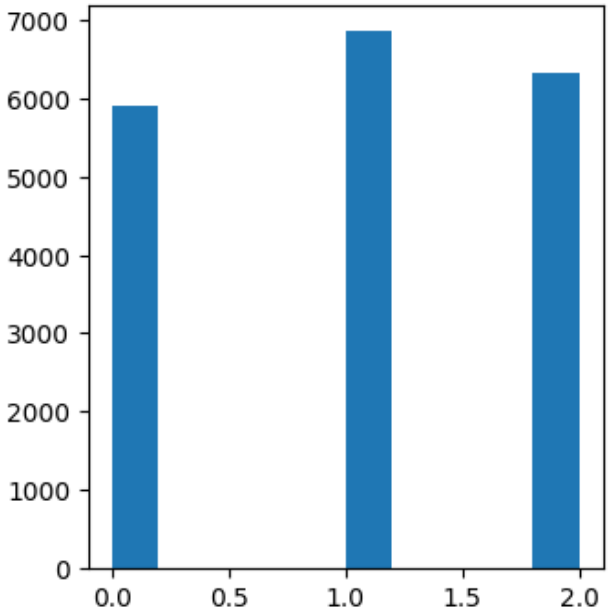
```
for i in num:
  plt.figure(figsize=(8,4))
  plt.subplot(1,2,1)
  plt.hist(x=i,data=df)
  plt.subplot(1,2,2)
  sns.boxplot(x=i,data=df)
  plt.show()
```

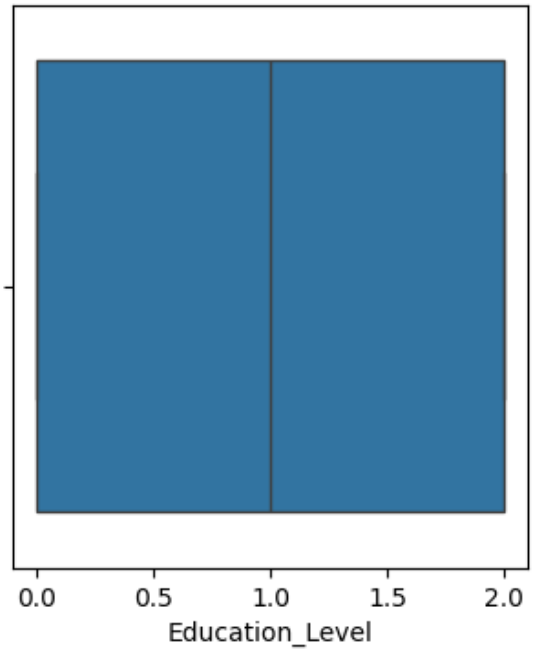

4000
3000
2000
1000
0
20
30
40
50

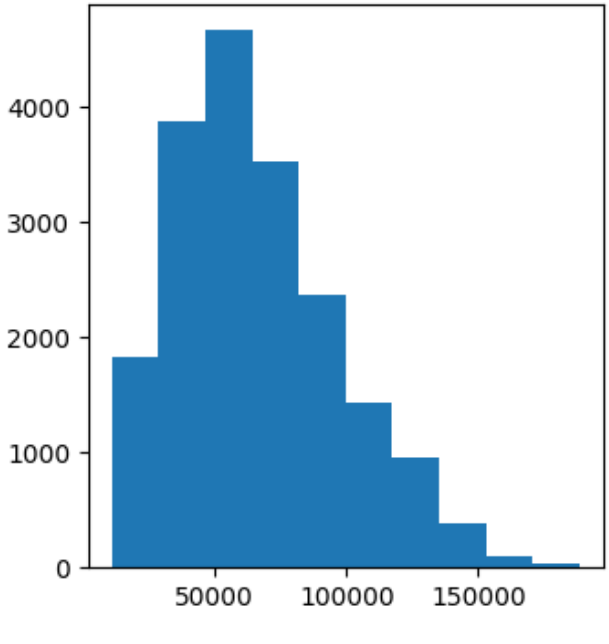

20
30
40
50
Age

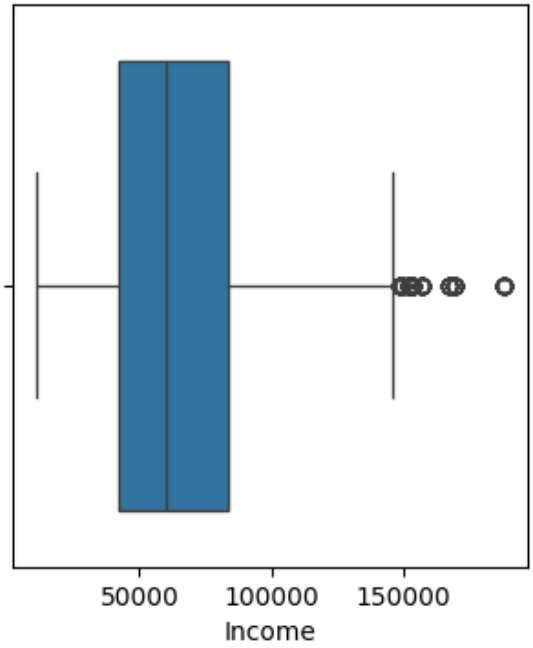

10000
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0

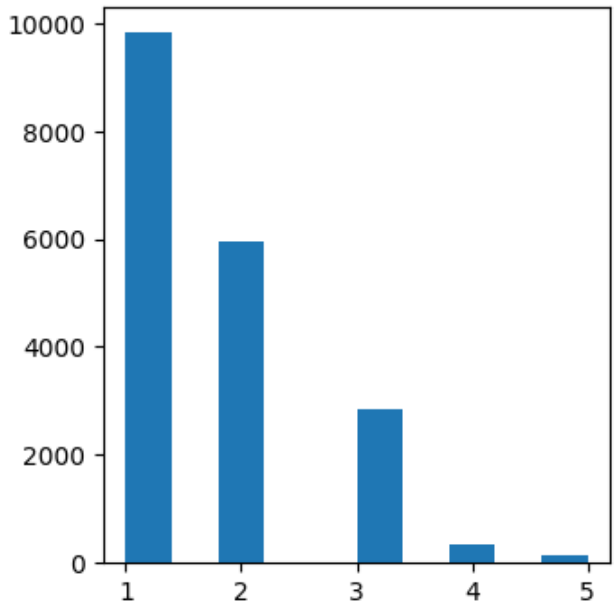

0.0
0.2
0.4
0.6
0.8
1.0
Gender

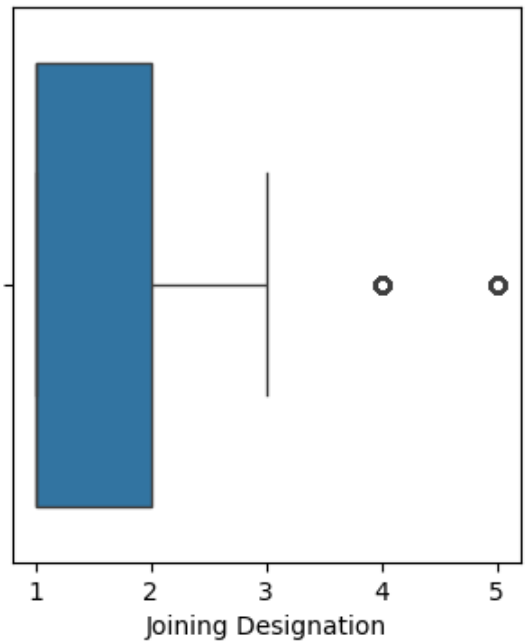

7000
6000
5000
4000
3000
2000
1000
0
0.0
0.5
1.0
1.5
2.0

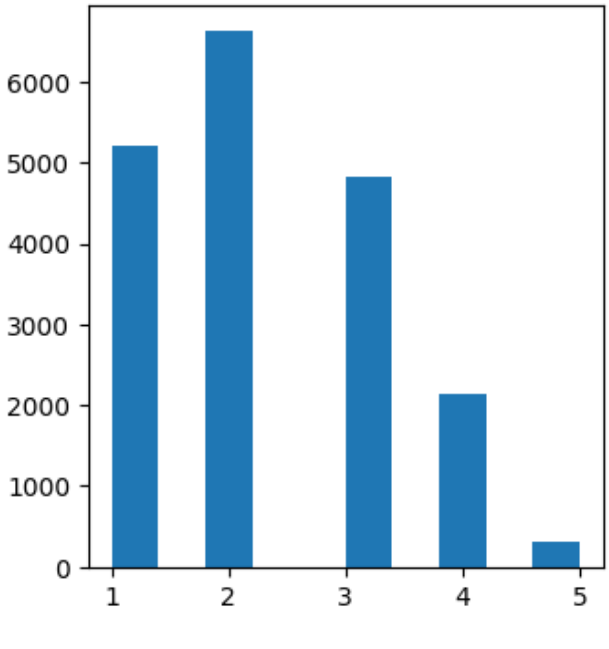

0.0
0.5
1.0
1.5
2.0
Education_Level

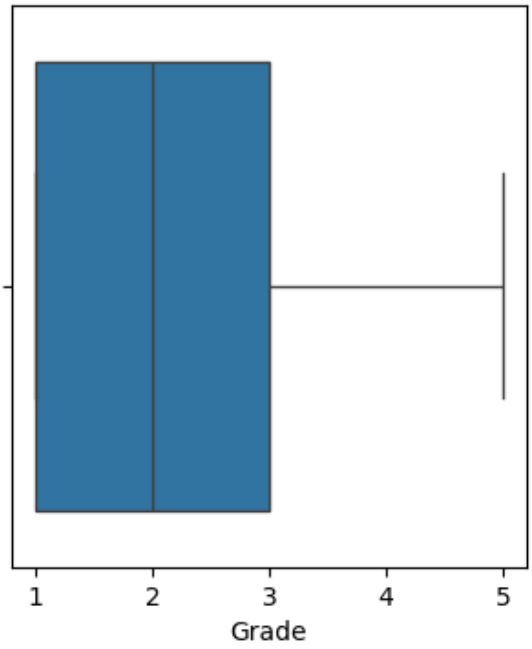

4000
3000
2000
1000
0
50000
100000
150000

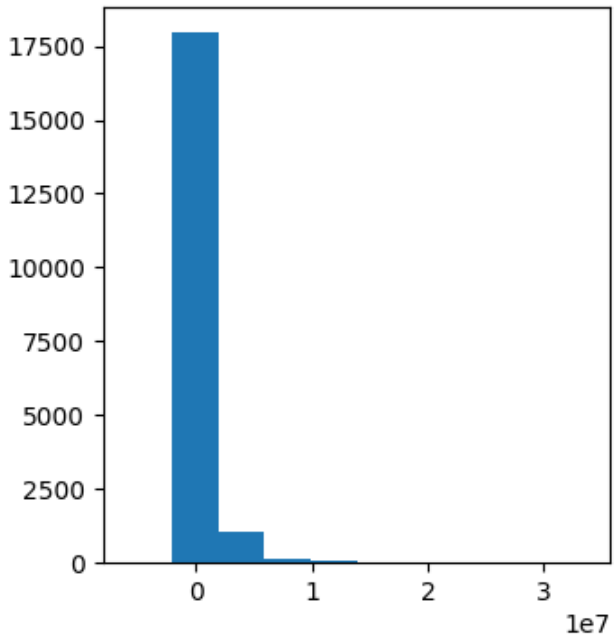

50000
100000
150000
Income

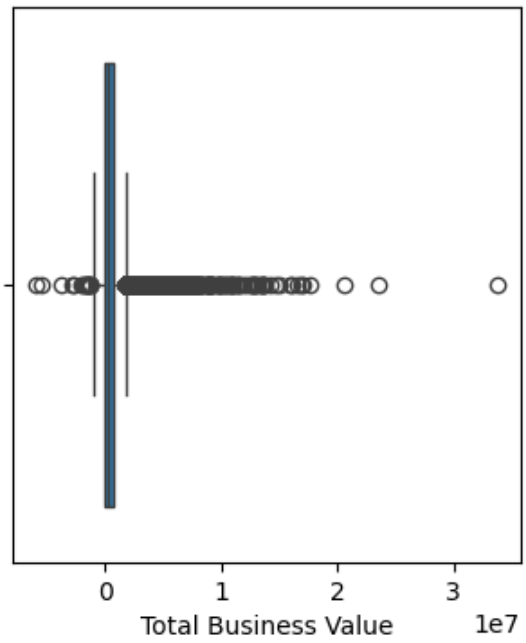

10000
8000
6000
4000
2000
0
1
2
3
4
5

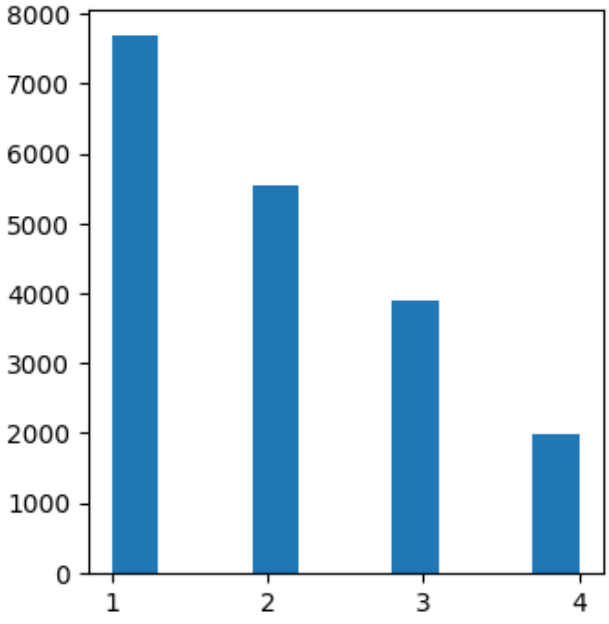

1
2
3
4
5
Joining Designation

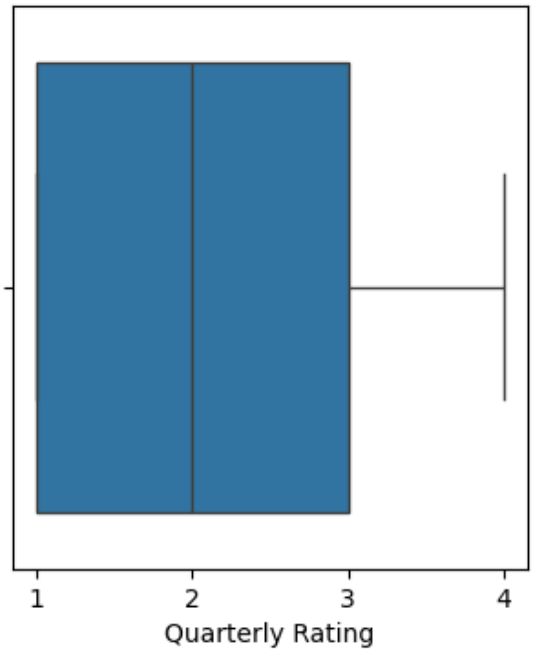

6000
5000
4000
3000
2000
1000
0
1
2
3
4
5


1
2
3
4
5
Grade

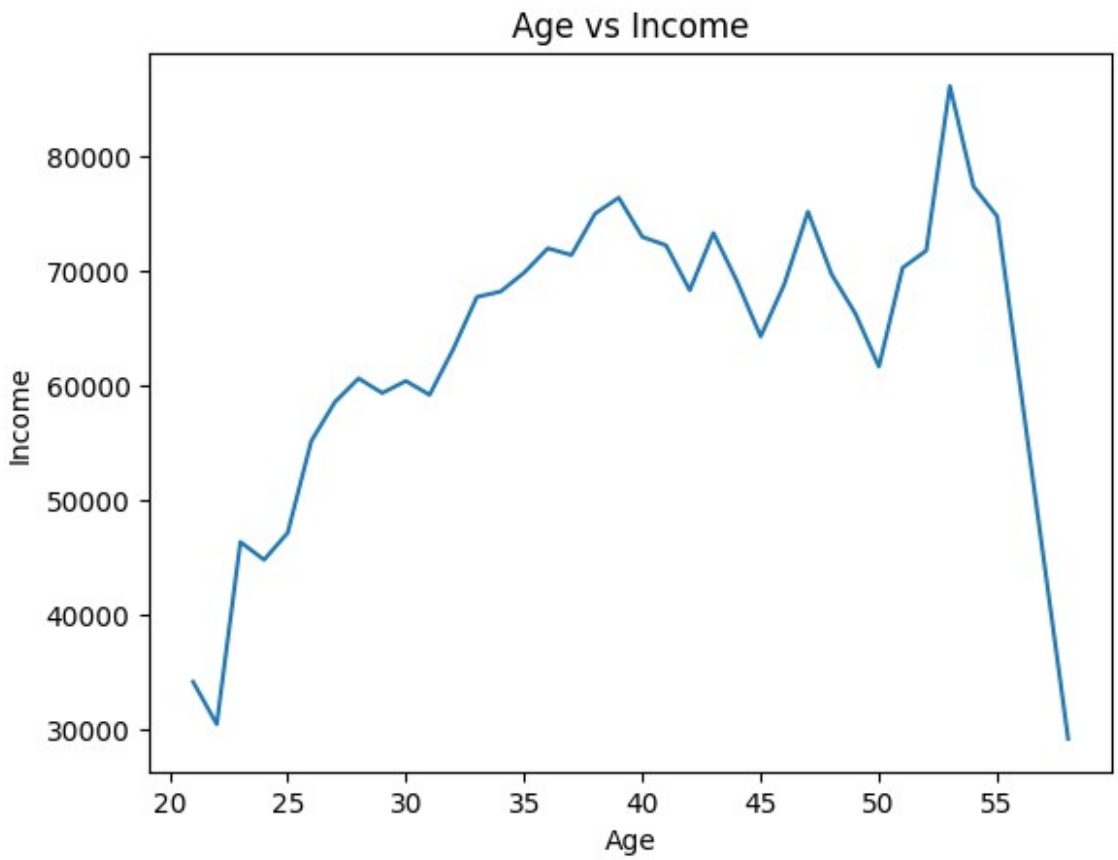




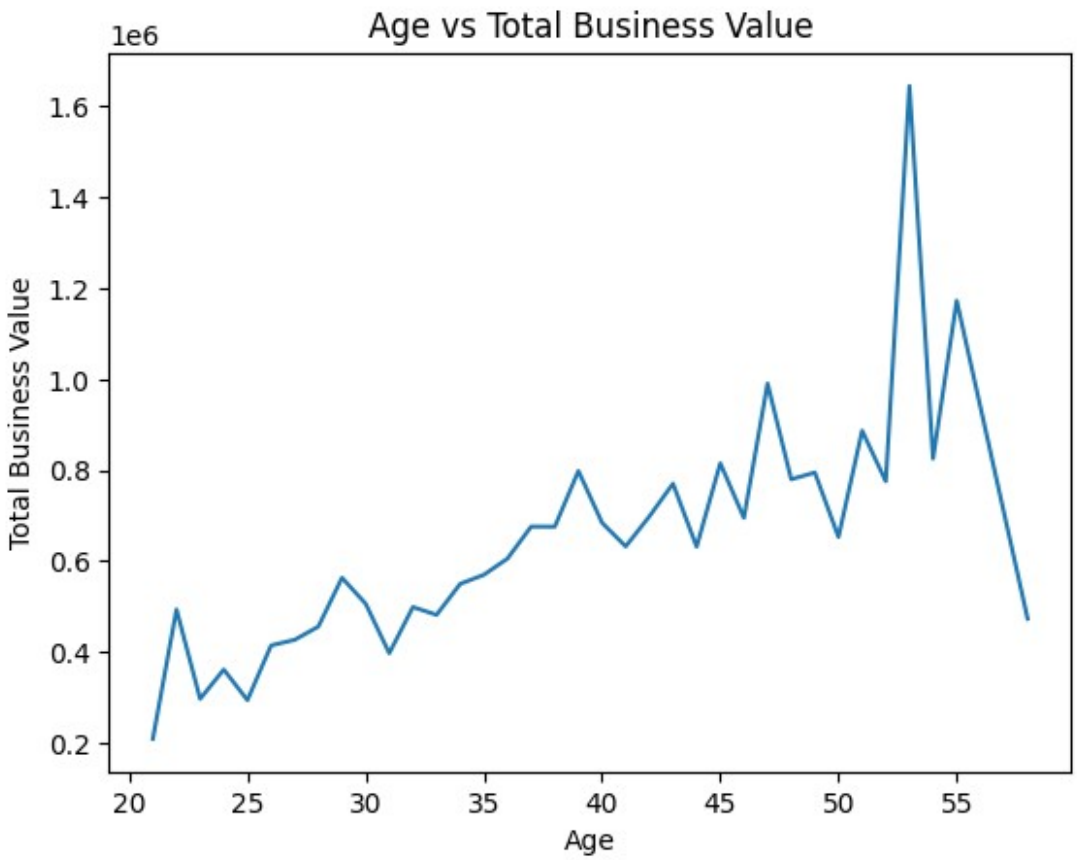




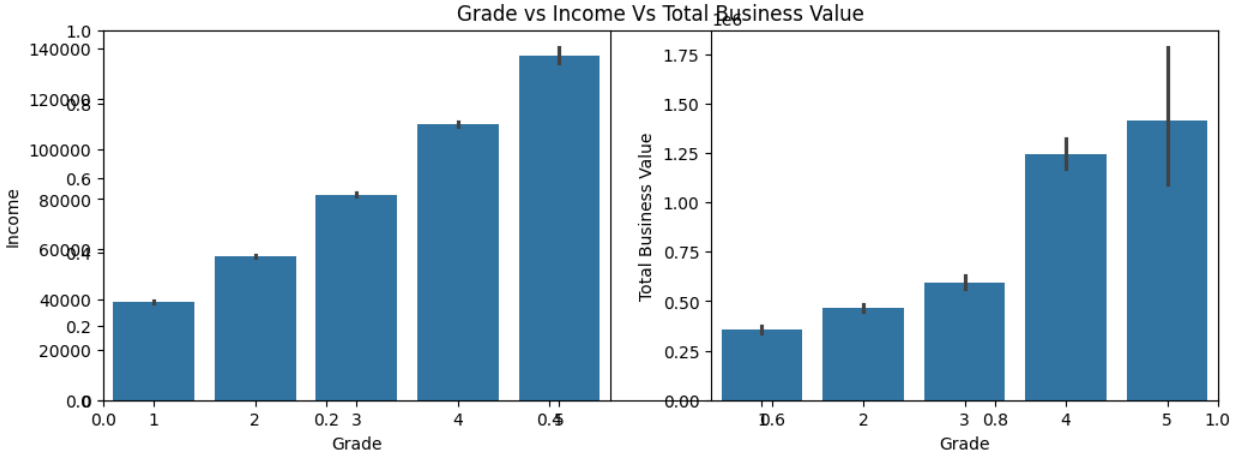

```
plt.title("Age vs Income")
sns.lineplot(x='Age',y='Income',errorbar=None,data=df)
plt.show()
```



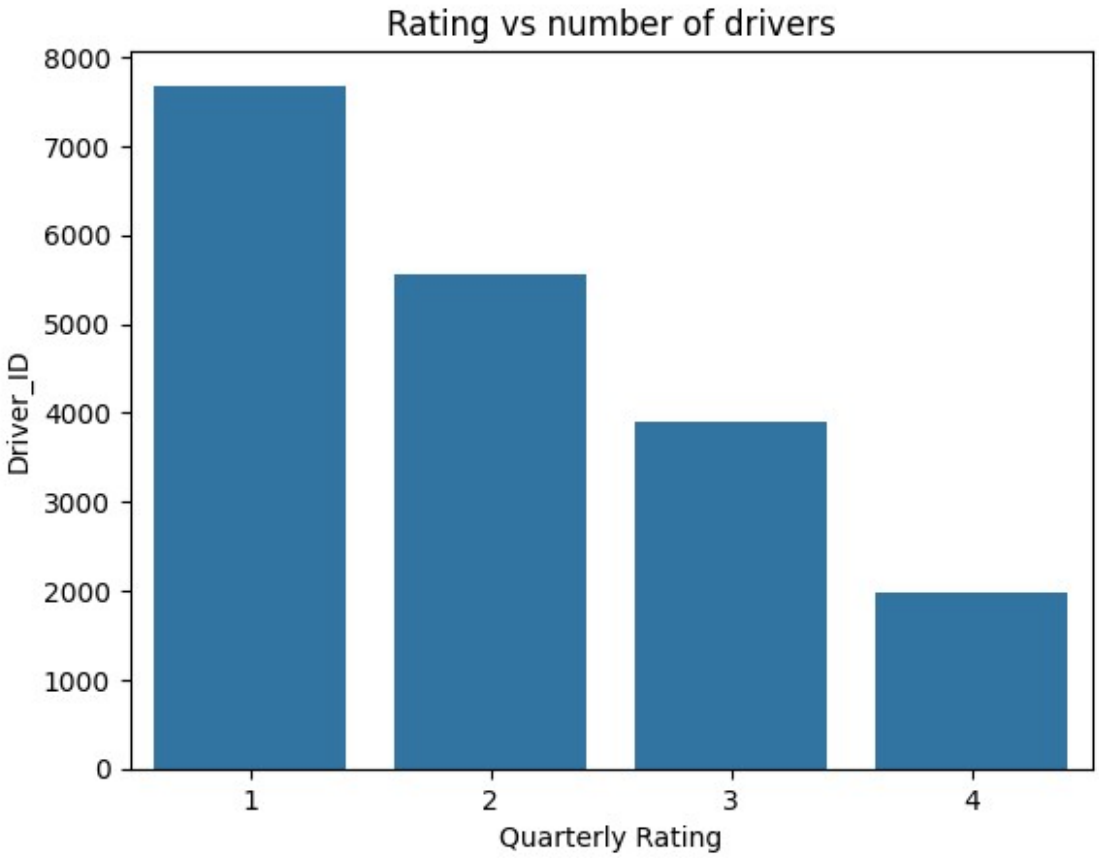

```
plt.title("Age vs Total Business Value")
sns.lineplot(x='Age',y='Total Business Value',errorbar=None,data=df)
plt.show()
```




```
plt.figure(figsize=(12,4))
plt.title("Grade vs Income Vs Total Business Value")
plt.subplot(1,2,1)
sns.barplot(x='Grade',y='Income',data=df)
plt.subplot(1,2,2)
sns.barplot(x='Grade',y='Total Business Value',data=df)
plt.show()
```



```
a=df.groupby('Quarterly Rating')['Driver_ID'].count().reset_index()
plt.title('Rating vs number of drivers')
sns.barplot(x='Quarterly Rating',y='Driver_ID',data=a)
plt.show()
```



Data Preprocessing

KNN Imputation

```
df2=df[['Age', 'Gender', 'Education_Level', 'Income', 'Joining
Designation',
       'Grade', 'Total Business Value', 'Quarterly Rating']].copy()

df2
```


```
{"summary":"{\n  \"name\": \"df2\",\n  \"rows\": 19104,\n  \"fields\":
[\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 6.2579116861907345,\n
\"min\": 21.0,\n        \"max\": 58.0,\n        \"num_unique_values\":
36,\n        \"samples\": [\n          58.0,\n          41.0,\n
24.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
```

\"std\": 0.4933670037660394,\n   \"min\": 0.0,\n   \"max\":
1.0,\n   \"num_unique_values\": 2,\n   \"samples\": [\n
1.0,\n   0.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Education_Level\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   2,\n   0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Income\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 30914,\n   \"min\": 10747,\n
\"max\": 188418,\n   \"num_unique_values\": 2383,\n
\"samples\": [\n   44273,\n   35370\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Joining Designation\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 1,\n   \"max\": 5,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   2,\n
5\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Grade\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1,\n   \"min\": 1,\n   \"max\": 5,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   2,\n
5\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Total Business Value\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 1128312,\n   \"min\": -6000000,\n
\"max\": 33747720,\n   \"num_unique_values\": 10181,\n
\"samples\": [\n   431090,\n   720180\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Quarterly Rating\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1,\n   \"min\": 1,\n   \"max\": 4,\n
\"num_unique_values\": 4,\n   \"samples\": [\n   1,\n
3\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df2"}

```
from sklearn.impute import KNNImputer
X_transformed=KNNImputer(n_neighbors=5).fit_transform(df2)
X_transformed=pd.DataFrame(X_transformed,columns=df2.columns)
X_transformed.isnull().sum()
```

```
Age                     0
Gender                  0
Education_Level         0
Income                  0
Joining Designation     0
Grade                   0
Total Business Value    0
```

```
Quarterly Rating         0
dtype: int64
```

```
for i in X_transformed.columns:
  df[i]=X_transformed[i]
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
```

```
agg_functions = {
    "Age": "max",
    "Gender": "first",
    "Education_Level": "last",
    "Income": "last",
    "Joining Designation": "last",
    "Grade": "last",
    "Total Business Value": "sum",
    "Quarterly Rating": "last",
    "LastWorkingDate": "last",
    "City": "first",
    "Dateofjoining": "last"
}

processed_df = df.groupby(["Driver_ID", "MMM-
YY"]).aggregate(agg_functions).sort_index(ascending = [True, True])

processed_df.head()
```

{"repr_error":"0","type":"dataframe","variable_name":"processed_df"}

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

```
Driver_ID    2381
dtype: int64
```

```
data['Age'] = list(processed_df.groupby('Driver_ID',axis=0).max('MMM-
YY')['Age'])
data['Gender'] =
list(processed_df.groupby('Driver_ID').agg({'Gender':'last'})
['Gender'])
data['City'] =
list(processed_df.groupby('Driver_ID').agg({'City':'last'})['City'])
data['Education'] =
list(processed_df.groupby('Driver_ID').agg({'Education_Level':'last'})
['Education_Level'])
data['Income'] =
list(processed_df.groupby('Driver_ID').agg({'Income':'last'})
['Income'])
data['Joining_Designation'] =
list(processed_df.groupby('Driver_ID').agg({'Joining
Designation':'last'})['Joining Designation'])
data['Grade'] =
list(processed_df.groupby('Driver_ID').agg({'Grade':'last'})['Grade'])
data['Total_Business_Value'] =
list(processed_df.groupby('Driver_ID',axis=0).sum('Total Business
Value')['Total Business Value'])
data['Last_Quarterly_Rating'] =
list(processed_df.groupby('Driver_ID').agg({'Quarterly
Rating':'last'})['Quarterly Rating'])
```

```
<ipython-input-528-63cb045fe890>:1: FutureWarning: The 'axis' keyword
in DataFrame.groupby is deprecated and will be removed in a future
version.
  data['Age'] =
list(processed_df.groupby('Driver_ID',axis=0).max('MMM-YY')['Age'])
<ipython-input-528-63cb045fe890>:8: FutureWarning: The 'axis' keyword
in DataFrame.groupby is deprecated and will be removed in a future
version.
  data['Total_Business_Value'] =
list(processed_df.groupby('Driver_ID',axis=0).sum('Total Business
Value')['Total Business Value'])
```

```
data.head()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 2381,\n  \"fields\":
[\n    {\n      \"column\": \"Driver_ID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 806,\n        \"min\": 1,\n
\"max\": 2788,\n        \"num_unique_values\": 2381,\n
\"samples\": [\n          1663,\n          1264,\n          1618\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 5.933264821155349,\n
\"min\": 21.0,\n        \"max\": 58.0,\n        \"num_unique_values\":
61,\n        \"samples\": [\n          28.0,\n          35.0,\n
30.4\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Gender\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.49149629563998526,\n   \"min\": 0.0,\n   \"max\":
1.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n
1.0,\n   0.4,\n   0.2\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"City\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 29,\n
\"samples\": [\n   \"C22\",\n   \"C5\",\n
\"C24\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Education\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.8162900093072196,\n   \"min\":
0.0,\n   \"max\": 2.0,\n   \"num_unique_values\": 3,\n
\"samples\": [\n   2.0,\n   0.0,\n   1.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Income\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
28383.666384011165,\n   \"min\": 10747.0,\n   \"max\":
188418.0,\n   \"num_unique_values\": 2339,\n   \"samples\":
[\n   58943.0,\n   96076.0,\n   45193.0\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Joining_Designation\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.8414334155102954,\n   \"min\":
1.0,\n   \"max\": 5.0,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   2.0,\n   5.0,\n   3.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Grade\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
0.9415217532867884,\n   \"min\": 1.0,\n   \"max\": 5.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   2.0,\n
5.0,\n   3.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Total_Business_Value\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 9127115.313445697,\n   \"min\": -
1385530.0,\n   \"max\": 95331060.0,\n
\"num_unique_values\": 1629,\n   \"samples\": [\n
479380.0,\n   1628780.0,\n   17866060.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Last_Quarterly_Rating\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.8098389460329392,\n   \"min\": 1.0,\n   \"max\": 4.0,\n
\"num_unique_values\": 4,\n   \"samples\": [\n   1.0,\n
3.0,\n   2.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"data"}

```
first_quarter = processed_df.groupby(["Driver_ID"]).agg({"Quarterly
Rating": "first"})
```

```
last_quarter = processed_df.groupby(["Driver_ID"]).agg({"Quarterly
Rating": "last"})

qr = (last_quarter["Quarterly Rating"] > first_quarter["Quarterly
Rating"]).reset_index()

empid = qr[qr["Quarterly Rating"] == True]["Driver_ID"]

qrl = []
for i in data["Driver_ID"]:
  if i in empid.values:
        qrl.append(1)
  else:
     qrl.append(0)

data["Quarterly_Rating_Increased"] = qrl

lwd = (processed_df.groupby(["Driver_ID"]).agg({"LastWorkingDate":
"last"})["LastWorkingDate"].isna()).reset_index()
lwrid = lwd[lwd["LastWorkingDate"] == True]["Driver_ID"]
target = []

for i in data["Driver_ID"]:
  if i in lwrid.values:
    target.append(0)
  else:
    target.append(1)

data["target"] = target
data.head()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 2381,\n  \"fields\":
[\n    {\n      \"column\": \"Driver_ID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 806,\n        \"min\": 1,\n
\"max\": 2788,\n        \"num_unique_values\": 2381,\n
\"samples\": [\n          1663,\n          1264,\n          1618\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 5.933264821155349,\n
\"min\": 21.0,\n        \"max\": 58.0,\n        \"num_unique_values\":
61,\n        \"samples\": [\n          28.0,\n          35.0,\n
30.4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.49149629563998526,\n        \"min\": 0.0,\n        \"max\":
1.0,\n        \"num_unique_values\": 5,\n        \"samples\": [\n
1.0,\n          0.4,\n          0.2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"City\",\n      \"properties\": {\n
```

```
\"dtype\": \"category\",\n        \"num_unique_values\": 29,\n
\"samples\": [\n          \"C22\",\n          \"C5\",\n
\"C24\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Education\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.8162900093072196,\n        \"min\":
0.0,\n        \"max\": 2.0,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          2.0,\n          0.0,\n          1.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Income\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
28383.666384011165,\n        \"min\": 10747.0,\n        \"max\":
188418.0,\n        \"num_unique_values\": 2339,\n        \"samples\":
[\n          58943.0,\n          96076.0,\n          45193.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Joining_Designation\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.8414334155102954,\n        \"min\":
1.0,\n        \"max\": 5.0,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          2.0,\n          5.0,\n          3.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Grade\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.9415217532867884,\n        \"min\": 1.0,\n        \"max\": 5.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          2.0,\n
5.0,\n          3.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total_Business_Value\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 9127115.313445697,\n        \"min\": -
1385530.0,\n        \"max\": 95331060.0,\n
\"num_unique_values\": 1629,\n        \"samples\": [\n
479380.0,\n          1628780.0,\n          17866060.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Last_Quarterly_Rating\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.8098389460329392,\n        \"min\": 1.0,\n        \"max\": 4.0,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          1.0,\n
3.0,\n          2.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Quarterly_Rating_Increased\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"target\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
```

\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"data"}

```
mrf = processed_df.groupby(["Driver_ID"]).agg({"Income": "first"})

mrl = processed_df.groupby(["Driver_ID"]).agg({"Income": "last"})

mr = (mrl["Income"] > mrf["Income"]).reset_index()

empid = mr[mr["Income"] == True]["Driver_ID"]
income = []
for i in data["Driver_ID"]:
    if i in empid.values:
        income.append(1)
    else:
        income.append(0)
data["Salary_Increased"] = income

data.shape
```

```
(2381, 13)
```

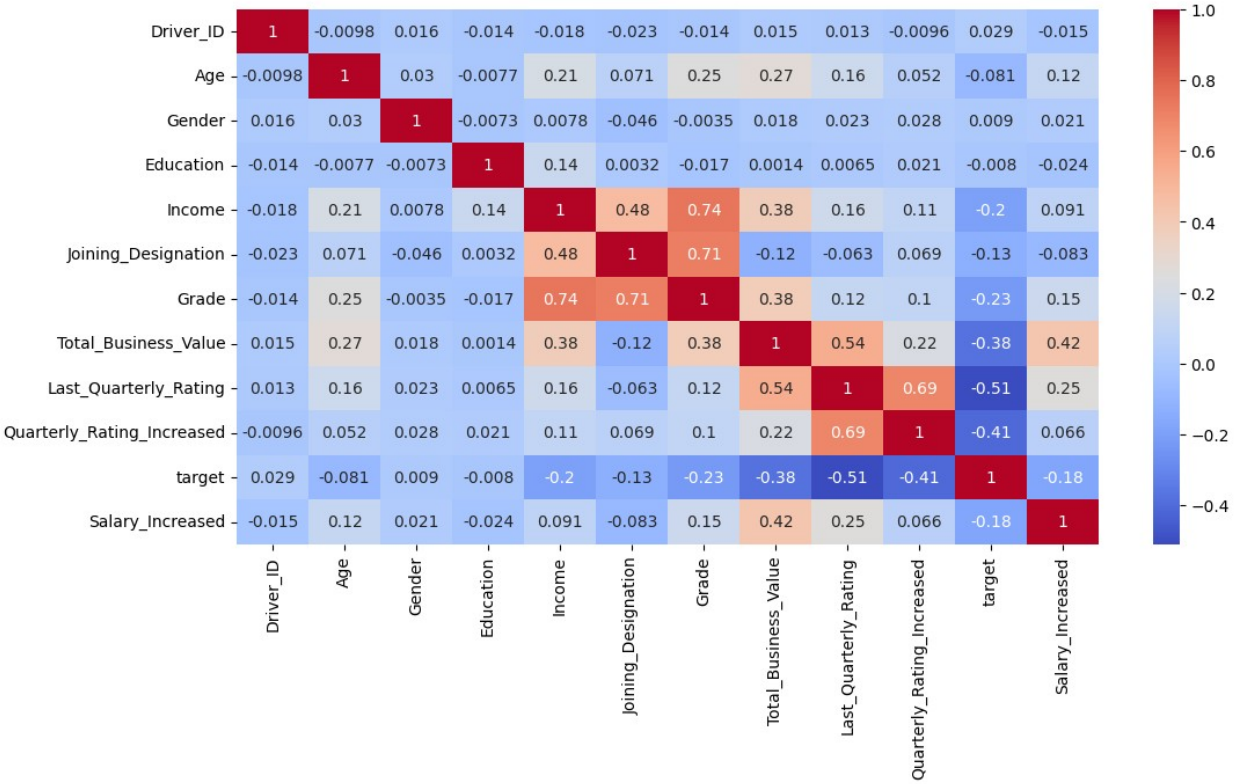
```
data.describe().T
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 12,\n \"fields\":
[\n   {\n   \"column\": \"count\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.0,\n   \"min\":
2381.0,\n   \"max\": 2381.0,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   2381.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"mean\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 1322592.958350505,\n
\"min\": 0.018059638807223857,\n   \"max\": 4586741.8227635445,\n
\"num_unique_values\": 12,\n   \"samples\": [\n
0.6787064258714826\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"std\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 2634017.5002043652,\n   \"min\": 0.1331951173987259,\n
\"max\": 9127115.313445697,\n   \"num_unique_values\": 12,\n
\"samples\": [\n   0.467071340254642\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"min\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 400262.6693780048,\n
\"min\": -1385530.0,\n   \"max\": 10747.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   21.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"25%\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 11271.004870368868,\
n   \"min\": 0.0,\n   \"max\": 39104.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   30.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n

```
}\n    },\n    {\n      \"column\": \"50%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 235090.42334288708,\
n        \"min\": 0.0,\n        \"max\": 817680.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          1400.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"75%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 1202975.8684523925,\
n        \"min\": 0.0,\n        \"max\": 4173650.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          37.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"max\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 27514739.83969129,\n
\"min\": 1.0,\n        \"max\": 95331060.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          58.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
plt.figure(figsize=(12,6))
sns.heatmap(data.corr(numeric_only=True),annot=True,cmap='coolwarm')
```

```
<Axes: >
```




```
categorical_columns =
data.select_dtypes(include=['object']).columns.tolist()
```

```
from sklearn.preprocessing import OneHotEncoder
encoder=OneHotEncoder(sparse_output=False)
a=encoder.fit_transform(data[categorical_columns])

one_hot_df = pd.DataFrame(a,
columns=encoder.get_feature_names_out(categorical_columns))
data=pd.concat([data, one_hot_df], axis=1)
data.head()
```

{"type":"dataframe","variable_name":"data"}

```
data.drop(columns='City',inplace=True)
data.shape
```

```
(2381, 41)
```

Model building

```
from sklearn.model_selection import train_test_split
X=data[['Driver_ID', 'Age', 'Gender', 'Education',
'Income','Joining_Designation', 'Grade', 'Total_Business_Value',
'Last_Quarterly_Rating', 'Quarterly_Rating_Increased',
   'Salary_Increased', 'City_C1', 'City_C10', 'City_C11', 'City_C12',
       'City_C13', 'City_C14', 'City_C15', 'City_C16', 'City_C17',
'City_C18',
       'City_C19', 'City_C2', 'City_C20', 'City_C21', 'City_C22',
'City_C23',
       'City_C24', 'City_C25', 'City_C26', 'City_C27', 'City_C28',
'City_C29',
       'City_C3', 'City_C4', 'City_C5', 'City_C6', 'City_C7',
'City_C8',
       'City_C9']].copy()
Y=data['target'].copy()
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.25,rand
om_state=18)

X_train,X_val,y_train,y_val=train_test_split(X_train,y_train,test_size
=0.25,random_state=18)
from imblearn.over_sampling import SMOTE
smt=SMOTE()
x_sm,y_sm=smt.fit_resample(X_train,y_train)

y_sm.value_counts()
```

```
target
1    896
0    896
Name: count, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
st=StandardScaler()
```

```
x_sm=st.fit_transform(x_sm)
X_val=st.transform(X_val)
X_test=st.transform(X_test)

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import
ConfusionMatrixDisplay,confusion_matrix,classification_report
from sklearn.model_selection import KFold, cross_validate,
cross_val_score

kfold = KFold(n_splits=10)

depths = [3,4,5,6,9,11,13,15]

for depth in depths:
    tree_clf = RandomForestClassifier(random_state=7, max_depth=depth)

    cv_acc_results = cross_validate(tree_clf, x_sm, y_sm, cv = kfold,
scoring = 'accuracy', return_train_score = True)

    print(f"K-Fold for depth:{depth} Accuracy Mean: Train:
{cv_acc_results['train_score'].mean()*100} Validation:
{cv_acc_results['test_score'].mean()*100}")
    print(f"K-Fold for depth: {depth} Accuracy Std: Train:
{cv_acc_results['train_score'].std()*100} Validation:
{cv_acc_results['test_score'].std()*100}")
    print('***************')
```

```
K-Fold for depth:3 Accuracy Mean: Train: 79.79298165187012 Validation:
75.32712600869027
K-Fold for depth: 3 Accuracy Std: Train: 0.7626801845660832
Validation: 6.072521226812834
***************
K-Fold for depth:4 Accuracy Mean: Train: 81.61592996727889 Validation:
77.39540657976411
K-Fold for depth: 4 Accuracy Std: Train: 1.0871009867777808
Validation: 3.9529646968581043
***************
K-Fold for depth:5 Accuracy Mean: Train: 83.401649747169 Validation:
78.5667287399131
K-Fold for depth: 5 Accuracy Std: Train: 1.2115369512351835
Validation: 3.799126554653145
***************
K-Fold for depth:6 Accuracy Mean: Train: 85.1005708888236 Validation:
79.62818125387957
K-Fold for depth: 6 Accuracy Std: Train: 1.3668925606245732
Validation: 3.7398339725461796
***************
K-Fold for depth:9 Accuracy Mean: Train: 89.95545267283964 Validation:
```

```
82.4205462445686
K-Fold for depth: 9 Accuracy Std: Train: 1.3485284288327275
Validation: 5.756401330600069
***************
K-Fold for depth:11 Accuracy Mean: Train: 92.98122112673241
Validation: 83.70639354438237
K-Fold for depth: 11 Accuracy Std: Train: 1.2125392825817165
Validation: 6.810621862597498
***************
K-Fold for depth:13 Accuracy Mean: Train: 96.05664044772698
Validation: 84.09652389819988
K-Fold for depth: 13 Accuracy Std: Train: 0.8998831641274456
Validation: 7.109728098099372
***************
K-Fold for depth:15 Accuracy Mean: Train: 98.1771093734376 Validation:
84.43202979515829
K-Fold for depth: 15 Accuracy Std: Train: 0.512812079795211
Validation: 7.942377518642904
***************

params = {
          'n_estimators' : [100,200,300,400],
          'max_depth' : [3,5,10,11,15],
          'criterion' : ['gini', 'entropy'],
          'bootstrap' : [True, False],
          'max_features' : [8,9,10]
         }

grid = GridSearchCV(estimator = RandomForestClassifier(),
                    param_grid = params,
                    scoring = 'accuracy',
                    cv = 3,
                    n_jobs=-1
                    )
grid.fit(x_sm,y_sm)

print("Best params: ", grid.best_params_)
print("Best score: ", grid.best_score_)

Best params:  {'bootstrap': True, 'criterion': 'gini', 'max_depth':
15, 'max_features': 8, 'n_estimators': 400}
Best score:  0.8387730178204288

clf2 = RandomForestClassifier(random_state=18, bootstrap=True,
criterion='gini',
                              max_depth=15, max_features=9,
n_estimators=100)

kfold = KFold(n_splits=10)
cv_acc_results = cross_validate(clf2, x_sm, y_sm, cv=kfold,
```

```
scoring='accuracy', return_train_score=True)

print(f"K-Fold Accuracy Mean: \n Train:
{cv_acc_results['train_score'].mean()*100:.3f} \n Validation:
{cv_acc_results['test_score'].mean()*100:.3f}")
print(f"K-Fold Accuracy Std: \n Train:
{cv_acc_results['train_score'].std()*100:.3f}, \n Validation:
{cv_acc_results['test_score'].std()*100:.3f}")
```

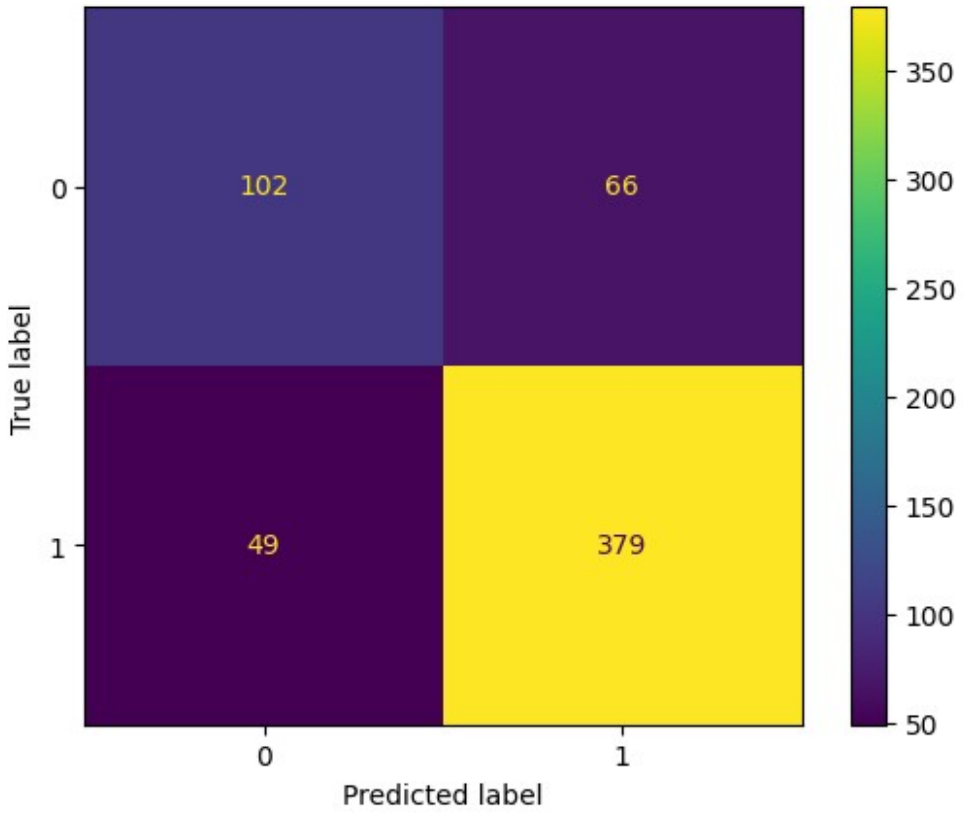
```
K-Fold Accuracy Mean:
 Train: 99.399
 Validation: 84.935
K-Fold Accuracy Std:
 Train: 0.326,
 Validation: 7.926
```

```
clf2=RandomForestClassifier(random_state=18,max_depth=15)
clf2.fit(x_sm,y_sm)
```

```
RandomForestClassifier(max_depth=15, random_state=18)
```

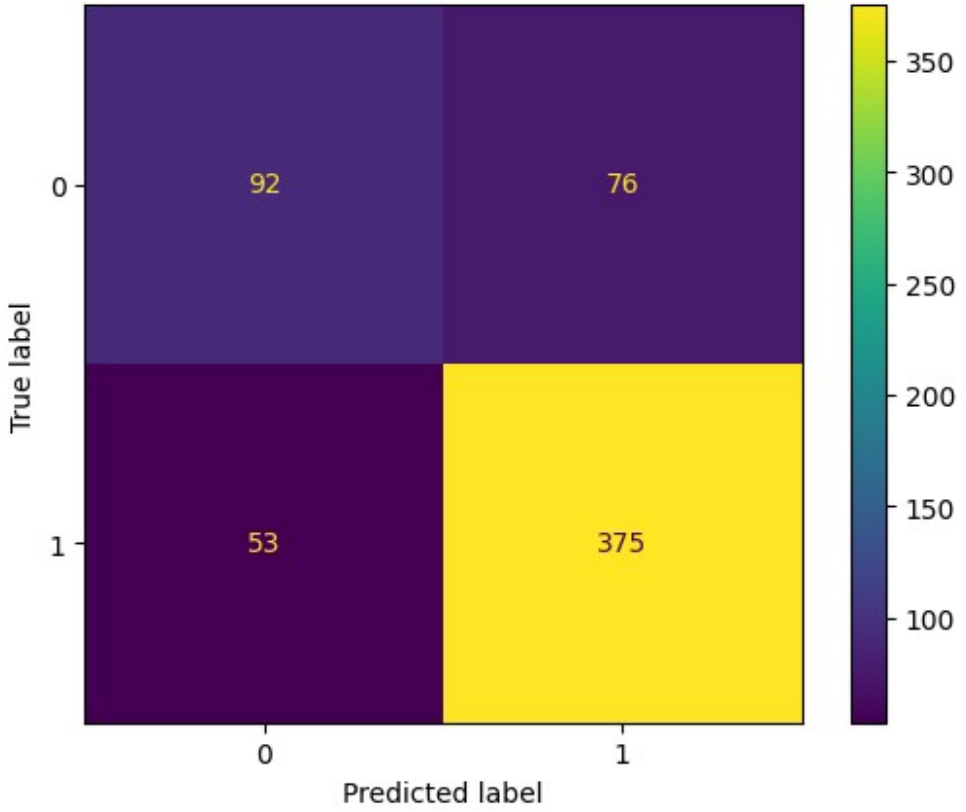
```
y_pred = clf2.predict(X_test)

print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)

ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=clf2.classes_).plot()
```

```
              precision    recall  f1-score   support

           0       0.68      0.61      0.64       168
           1       0.85      0.89      0.87       428

    accuracy                           0.81       596
   macro avg       0.76      0.75      0.75       596
weighted avg       0.80      0.81      0.80       596


<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x78cfb0494390>
```




```
from sklearn.ensemble import GradientBoostingClassifier
params = {
    "max_depth": [2, 3, 4],
    "loss": ["log_loss", "exponential"],
    "subsample": [0.1, 0.2, 0.5, 0.8, 1],
    "learning_rate": [0.1, 0.2, 0.3],
    "n_estimators": [50,100,150,200]
}

gbdt = GradientBoostingClassifier()
C = GridSearchCV(estimator=gbdt, cv=3, n_jobs=-1, verbose=True,
param_grid=params)


gbc=GradientBoostingClassifier(learning_rate= 0.1, loss='exponential',
max_depth= 4, n_estimators=150)
gbc.fit(x_sm, y_sm)
y_pred = gbc.predict(X_test)
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)

ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=gbc.classes_).plot()
```

```
              precision    recall  f1-score   support

           0       0.63      0.55      0.59       168
           1       0.83      0.88      0.85       428

    accuracy                           0.78       596
   macro avg       0.73      0.71      0.72       596
weighted avg       0.78      0.78      0.78       596


<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x78cf9d2144d0>
```



Actionable Insights and Recommendation

1.Out of 2381 drivers 1616 have left the company.

2.We need to incentivise the drivers overtime or other perks to overcome churning

3.The employees whose quarterly rating has increased are less likely to leave the organization.

4.Company needs to implement the reward system for the customer who provide the feedback and rate drivers

5.The employees whose monthly salary has not increased are more likely to leave the organization.

6.Company needs to get in touch with those drivers whose monthly salary has not increased and help them out to earn more by provider bonus and perks.

7.Out of 2381 employees, 1744 employees had their last quarterly rating as 1.