Business Context:

Ola faces significant challenges in recruiting and retaining drivers due to high attrition rates. Drivers frequently switch to competitors (like Uber) based on incentives or leave the platform altogether. This high churn negatively impacts operational efficiency, increases acquisition costs, and lowers organizational morale.

Analytical Objective:

As a Data Analyst at Ola, my task is to predict whether a driver will leave the company (churn) or continue working based on historical driver data from 2019 and 2020.

Understanding The Data

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

# importing data and store in variables
olaData = pd.read_csv('ola_driver.csv')

# check dataset dimensions (rows, columns)
f"Shape: {olaData.shape}"
```

```
'Shape: (19104, 14)'
```

```
# top 5 rows of data
olaData.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
```

Data Dictionary Overview

- MMMM-YY: Reporting Date (Monthly)
- Driver_ID: Unique ID for drivers
- Age: Age of the driver
- Gender: Gender of the driver (Male: 0, Female: 1)
- City: City Code of the driver
- Education_Level: Education level (0 for 10+, 1 for 12+, 2 for graduate)
- Income: Monthly average Income of the driver
- Date Of Joining: Joining date for the driver
- LastWorkingDate: Last date of working for the driver
- Joining Designation: Designation of the driver at joining
- Grade: Grade of the driver at reporting
- Total Business Value: Total business value acquired (negative indicates cancellation/refund or car EMI adjustments)
- Quarterly Rating: Quarterly rating of the driver (1-5, higher is better)

```
# printing information about the data
olaData.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
```

```
# total missing values per column
olaData.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
```

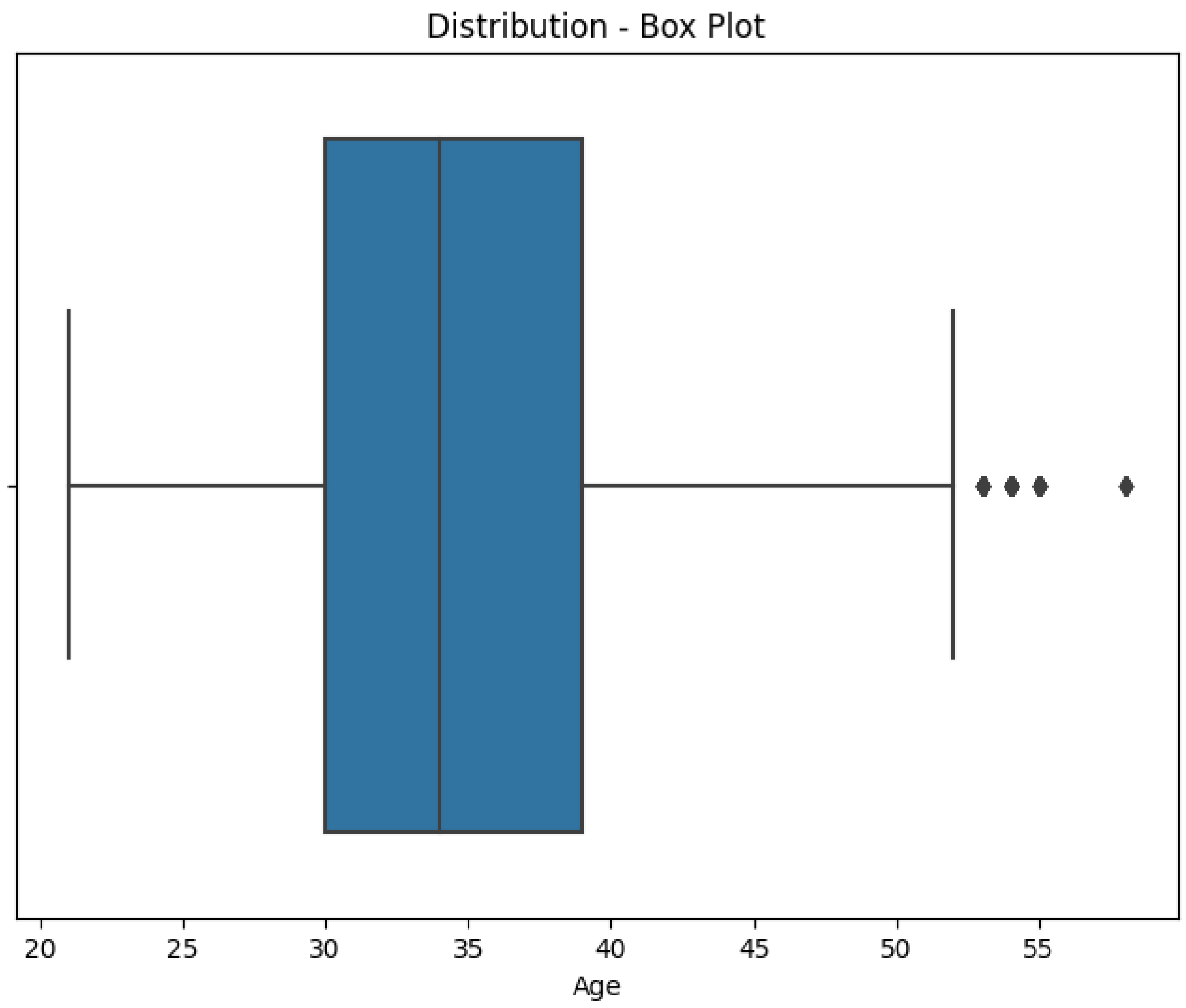
LastWorkingDate shows missing values as the driver hasn't left yet.

```
# generating descriptive statistics
olaData.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
```

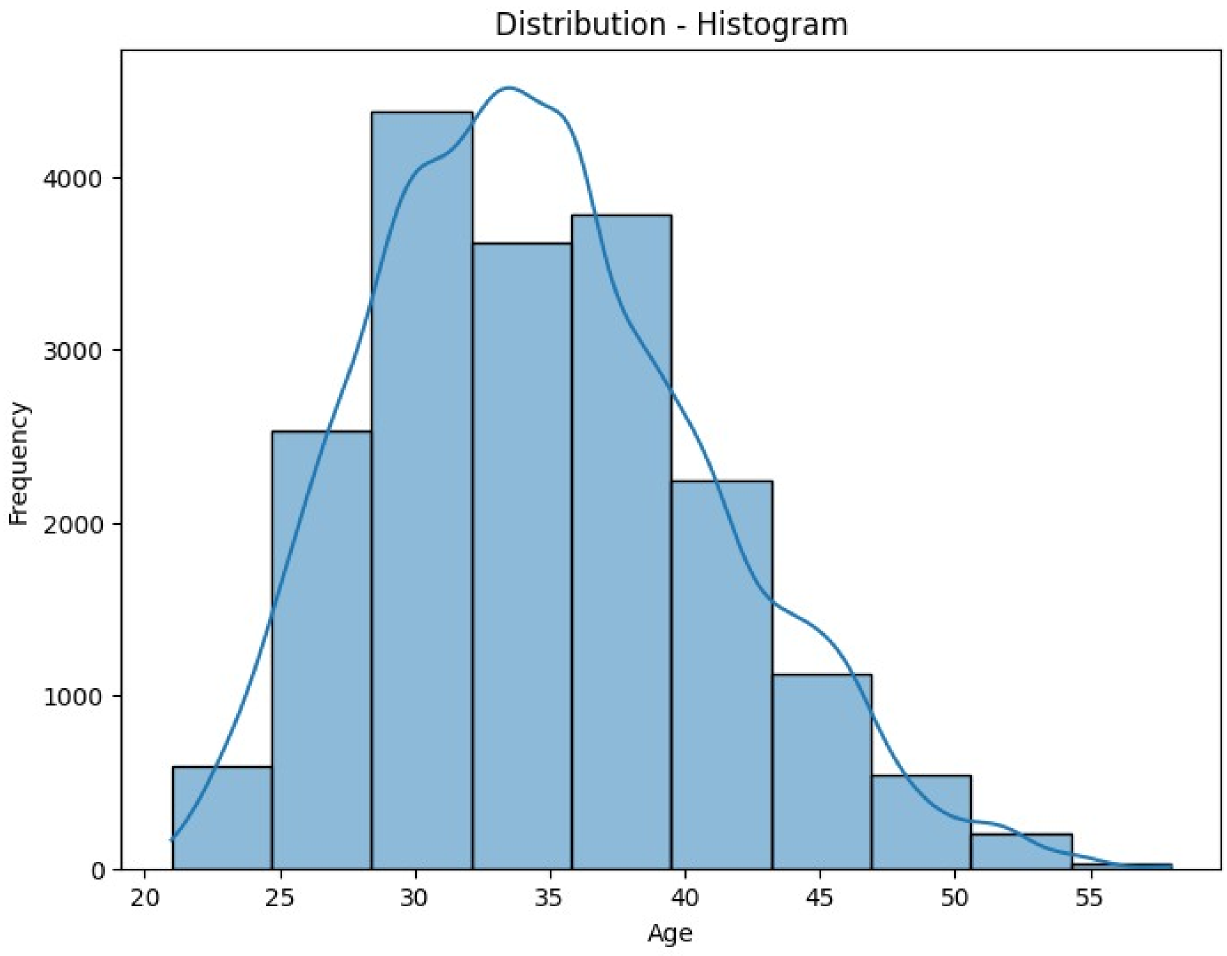
```
3.000000
max              2.000000  188418.000000              5.000000
5.000000

       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
```

Outlier & Pattern Detection

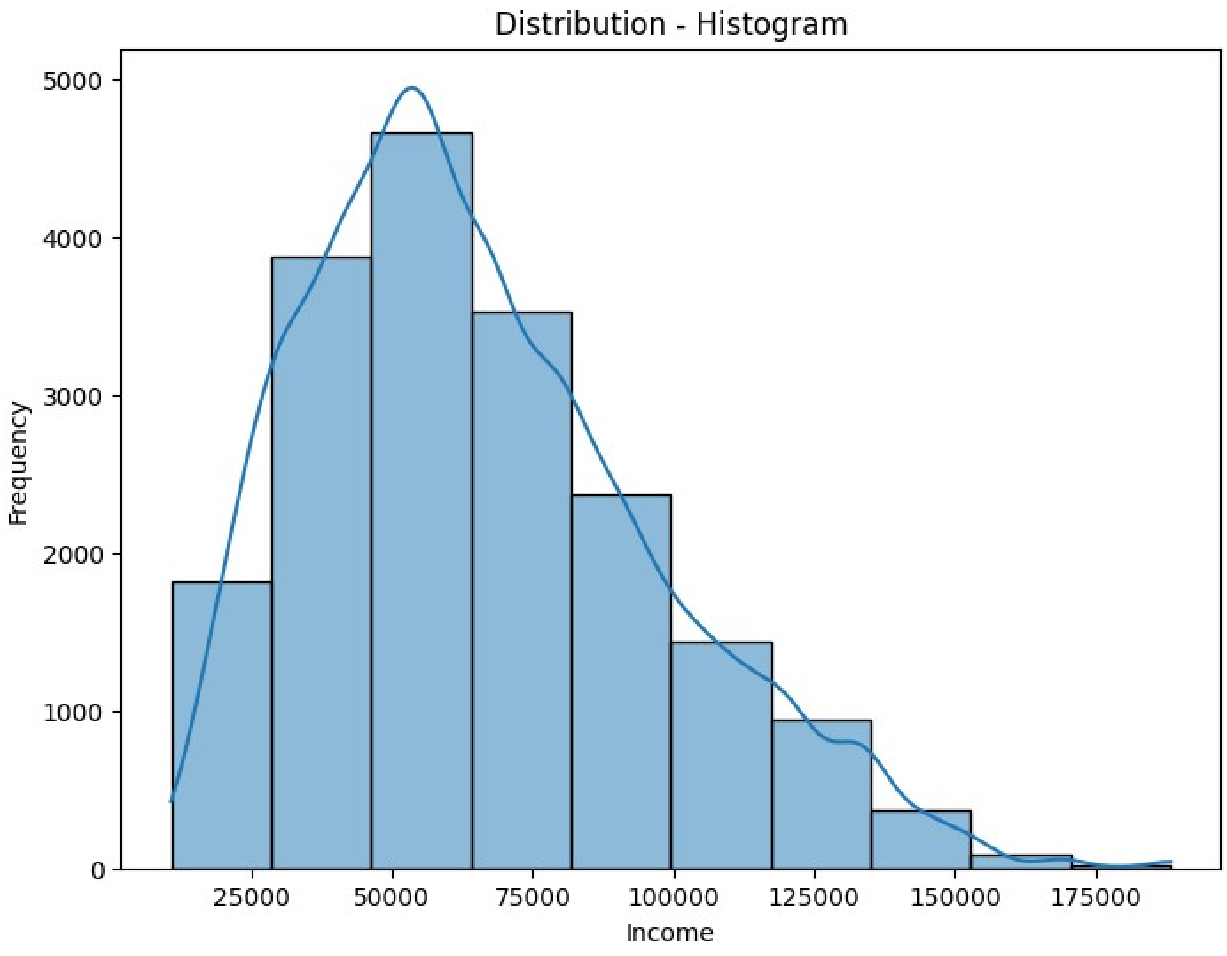
```
# boxplot for age column
plt.figure(figsize=(8, 6))
sns.boxplot(x=olaData['Age'])
plt.title('Distribution - Box Plot')
plt.show()
```



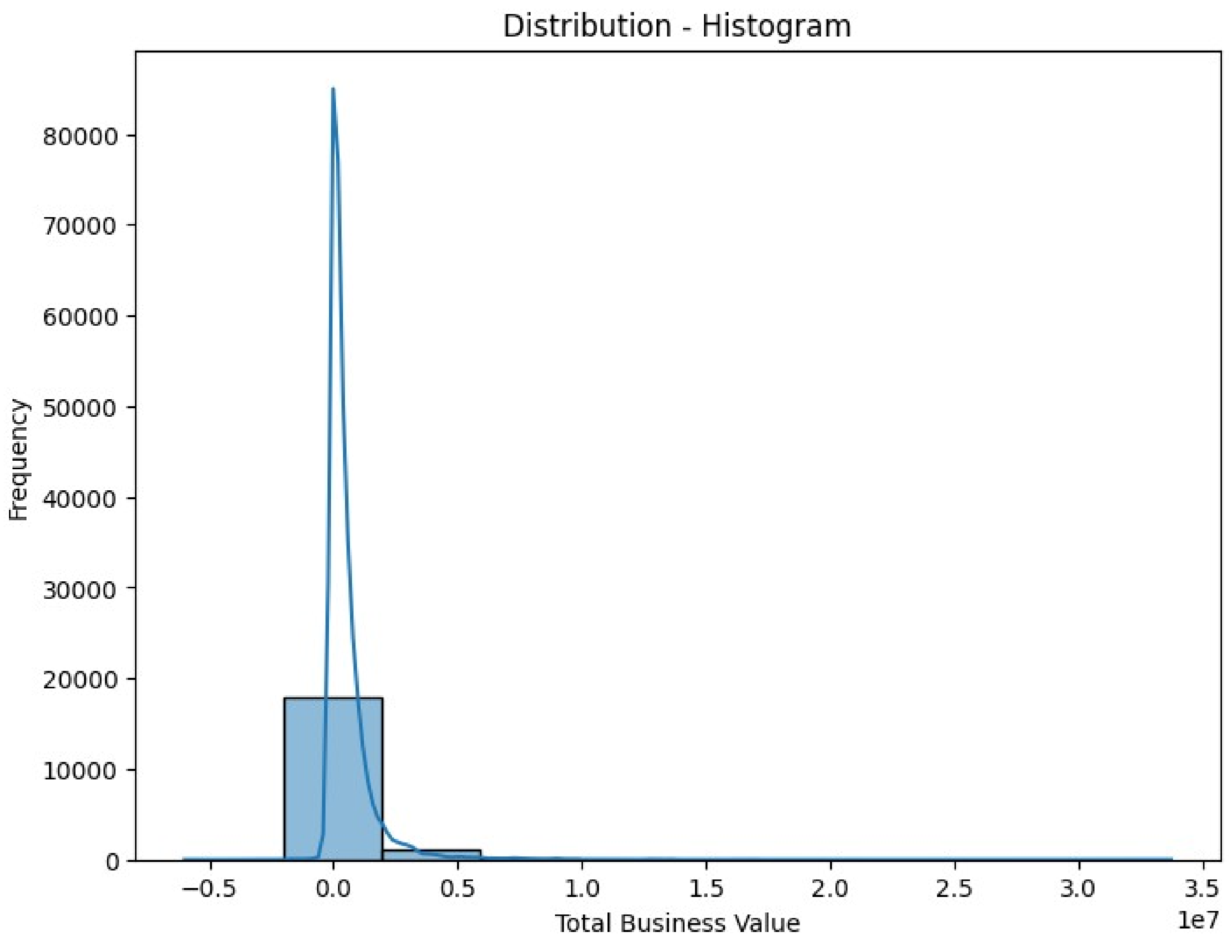
```
# histogram for age column
plt.figure(figsize=(8, 6))
sns.histplot(olaData['Age'], kde=True, bins=10)
plt.title('Distribution - Histogram')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



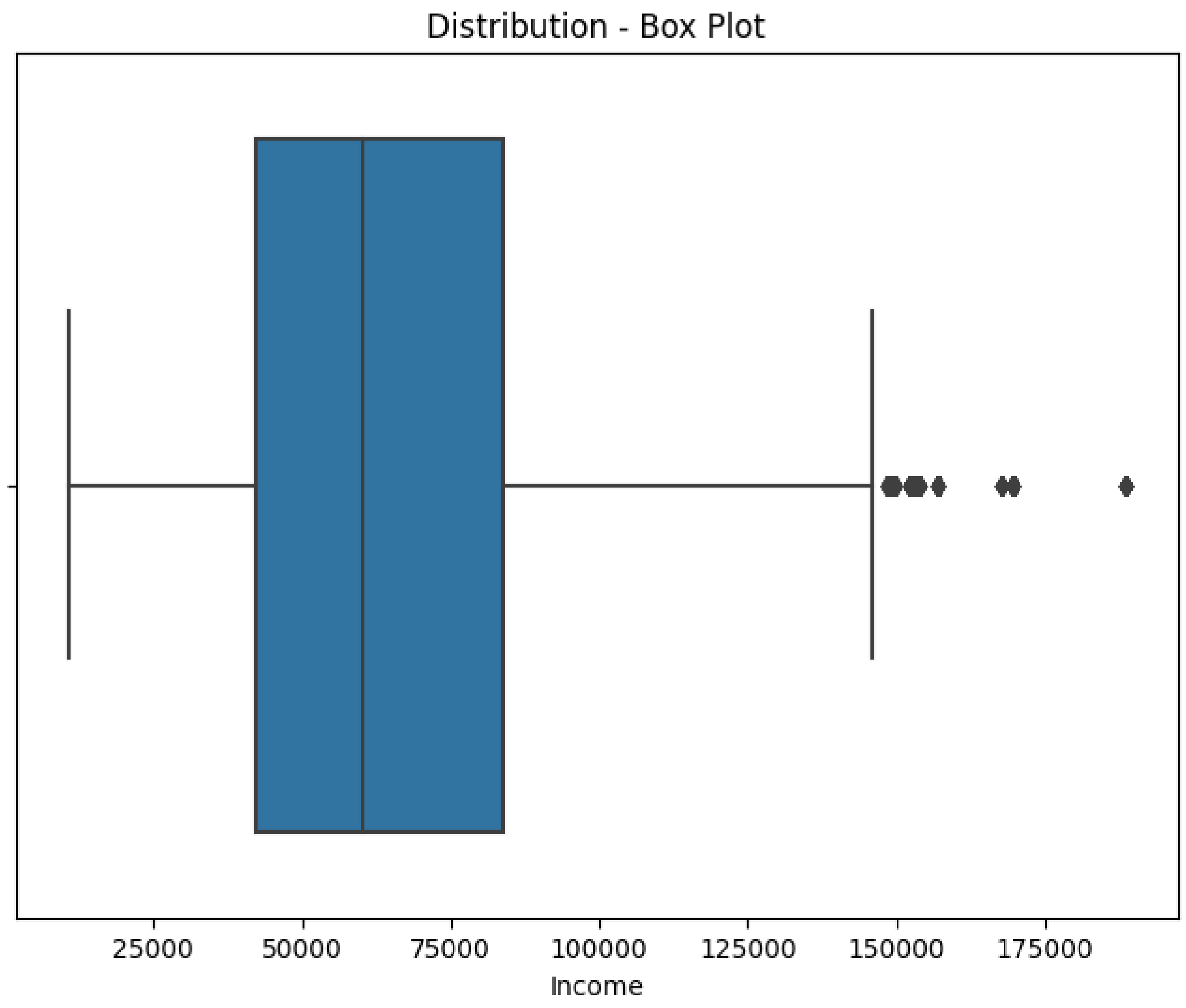
```
# histogram for income column
plt.figure(figsize=(8, 6))
sns.histplot(olaData['Income'], kde=True, bins=10)
plt.title('Distribution - Histogram')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
```



```
# histogram for business values column
plt.figure(figsize=(8, 6))
sns.histplot(olaData['Total Business Value'], kde=True, bins=10)
plt.title('Distribution - Histogram')
plt.xlabel('Total Business Value')
plt.ylabel('Frequency')
plt.show()
```



```
# boxplot for income column
plt.figure(figsize=(8, 6))
sns.boxplot(x=olaData['Income'])
plt.title('Distribution - Box Plot')
plt.show()
```



- 'unnamed' column is index column, removing it.
- we have 24 months of data, basically 2019 and 2020.
- Age and Gender columns contains some null values.
- Going to change the datatypes for particular columns.
- There is no extreme outlier in the columns.

Data Cleaning

This stage involves detecting and addressing issues like missing values, duplicates, incorrect formatting, or irrelevant data, ensuring the data is accurate, complete, and consistent for analysis or decision-making.

Creating duplicate dataset, as it act as a backup and allows us for experimentation without altering the original data.

```
# creating sample ola dataframe
odf = olaData
```

Fix issues like nulls, wrong formats, and duplicated data before doing any analysis.

```
# removing the column
odf = odf.drop('Unnamed: 0', axis=1) # not requied column.

# removing null values
odf.dropna(subset=['Age'], inplace=True) # missing data.
odf.dropna(subset=['Gender'], inplace=True) # missing data.

# convert to datetime
odf['MMM-YY'] = pd.to_datetime(odf['MMM-YY'])
odf['Dateofjoining'] = pd.to_datetime(odf['Dateofjoining'])
odf['LastWorkingDate'] = pd.to_datetime(odf['LastWorkingDate'])

# remove rows where date of joining is after the reporting month
odf = odf[odf['Dateofjoining'] <= odf['MMM-YY']] # a data entry error
or mismatched date formatting

odf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17957 entries, 0 to 19103
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   MMM-YY                17957 non-null  datetime64[ns]
 1   Driver_ID             17957 non-null  int64
 2   Age                   17957 non-null  float64
 3   Gender                17957 non-null  float64
 4   City                  17957 non-null  object
 5   Education_Level       17957 non-null  int64
 6   Income                17957 non-null  int64
 7   Dateofjoining         17957 non-null  datetime64[ns]
 8   LastWorkingDate       1594 non-null   datetime64[ns]
 9   Joining Designation   17957 non-null  int64
 10  Grade                 17957 non-null  int64
 11  Total Business Value  17957 non-null  int64
 12  Quarterly Rating      17957 non-null  int64
dtypes: datetime64[ns](3), float64(2), int64(7), object(1)
memory usage: 1.9+ MB
```

```
odf.describe()
```

```
          Driver_ID           Age        Gender  Education_Level  \
count  17957.000000  17957.000000  17957.000000     17957.000000
mean    1418.764883     34.834438      0.419614         1.022498
std      809.406069      6.290203      0.493509         0.798714
min        1.000000     21.000000      0.000000         0.000000
25%      714.000000     30.000000      0.000000         0.000000
50%     1422.000000     34.000000      0.000000         1.000000
75%     2142.000000     39.000000      1.000000         2.000000
```

```
max     2788.000000     58.000000      1.000000         2.000000

              Income  Joining Designation         Grade  Total
Business Value  \
count   17957.000000         17957.000000  17957.000000
1.795700e+04
mean    66205.172746             1.670324      2.265189
6.046725e+05
std     31113.736660             0.832961      1.035599
1.152512e+06
min     10747.000000             1.000000      1.000000         -
6.000000e+06
25%     42768.000000             1.000000      1.000000
0.000000e+00
50%     60708.000000             1.000000      2.000000
2.917200e+05
75%     84737.000000             2.000000      3.000000
7.370900e+05
max    188418.000000             5.000000      5.000000
3.374772e+07

       Quarterly Rating
count      17957.000000
mean           2.060812
std            1.010074
min            1.000000
25%            1.000000
50%            2.000000
75%            3.000000
max            4.000000
```

Feature Columns

```
#attrition flag (target variable)
odf['Attrition'] = odf['LastWorkingDate'].notna().astype(int)

# tenure in months
# calculate total tenure (until exit) (for drivers who have left)
odf['Total_Tenure_Months'] = np.where(
    odf['LastWorkingDate'].notna(),
    (odf['LastWorkingDate'] - odf['Dateofjoining']) /
np.timedelta64(1, 'M'),
    (odf['MMM-YY'] - odf['Dateofjoining']) / np.timedelta64(1, 'M')
)

odf['Total_Tenure_Months'] = odf['Total_Tenure_Months'].round(1)

# income category
odf['Income_Band'] = pd.cut(odf['Income'],
                           bins=[0, 40000, 60000, 85000, np.inf],
                              labels=['Low', 'Medium', 'High', 'Very 
High'])

# check current income bins evenly distributed
# if one or two categories dominate (say >50%), then it's not well 
balanced
odf['Income_Band'].value_counts(normalize=True)

Medium       0.281227
High         0.259342
Very High    0.247814
Low          0.211617
Name: Income_Band, dtype: float64

# negative business value flag
odf['Negative_Business'] = (odf['Total Business Value'] < 
0).astype(int)

# quarterly rating category
odf['Rating_Level'] = pd.cut(odf['Quarterly Rating'],
                             bins=[0, 2, 3, 4, 5],
                             labels=['Low', 'Average', 'Good', 
'Excellent'])

# experience bucket
# nan value indicates the driver hasn't left yet
odf['Experience_Bucket'] = pd.cut(odf['Total_Tenure_Months'],
                                  bins=[0, 6, 12, 24, float('inf')],
                                  labels=['<6M', '6-12M', '1-2Y', 
'>2Y'],
                                  right=False)

# month and year columns (from reporting date)
odf['Month'] = odf['MMM-YY'].dt.month
odf['Year'] = odf['MMM-YY'].dt.year

#  ensures time order per driver.
odf = odf.sort_values(by=['Driver_ID', 'MMM-YY'])

# quarterly rating increase
# compare current rating to previous month's rating for each driver
odf['Prev_Rating'] = odf.groupby('Driver_ID')['Quarterly 
Rating'].shift(1)

# 1 if rating increased, else 0
odf['Rating_Increased'] = (odf['Quarterly Rating'] > 
odf['Prev_Rating']).astype(int)

# monthly income increase
# compare current income to previous month's income for each driver
odf['Prev_Income'] = odf.groupby('Driver_ID')['Income'].shift(1)
```

```
# 1 if income increased, else 0
odf['Income_Increased'] = (odf['Income'] >
odf['Prev_Income']).astype(int)

# set max columns displayed (e.g., 50)
pd.set_option('display.max_columns', 50)

odf.head()

      MMM-YY  Driver_ID   Age  Gender City  Education_Level  Income  \
0 2019-01-01          1  28.0     0.0  C23                2   57387
1 2019-02-01          1  28.0     0.0  C23                2   57387
2 2019-03-01          1  28.0     0.0  C23                2   57387
4 2020-12-01          2  31.0     0.0   C7                2   67016
6 2020-01-01          4  43.0     0.0  C13                2   65603

  Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0    2018-12-24             NaT                    1      1
1    2018-12-24             NaT                    1      1
2    2018-12-24      2019-03-11                    1      1
4    2020-11-06             NaT                    2      2
6    2019-12-07             NaT                    2      2

   Total Business Value  Quarterly Rating  Attrition
Total_Tenure_Months  \
0               2381060                 2          0
0.3
1               -665480                 2          0
1.3
2                     0                 2          1
2.5
4                     0                 1          0
0.8
6                     0                 1          0
0.8

  Income_Band  Negative_Business Rating_Level Experience_Bucket  Month
Year  \
0      Medium                  0          Low               <6M      1
2019
1      Medium                  1          Low               <6M      2
2019
2      Medium                  0          Low               <6M      3
2019
4        High                  0          Low               <6M     12
2020
6        High                  0          Low               <6M      1
2020
```

```
   Prev_Rating  Rating_Increased  Prev_Income  Income_Increased
0          NaN                 0          NaN                 0
1          2.0                 0      57387.0                 0
2          2.0                 0      57387.0                 0
4          NaN                 0          NaN                 0
6          NaN                 0          NaN                 0
```

Summarizing Data

```
# earliest and latest timestamps recorded
start_datetime = odf['MMM-YY'].min()
end_datetime = odf['MMM-YY'].max()

print(f"Start Date & Time: {start_datetime}")
print(f"End Date & Time: {end_datetime}")
```

```
Start Date & Time: 2019-01-01 00:00:00
End Date & Time: 2020-12-01 00:00:00
```

```
# total distinct values for months, driver id, age, gender, city,
education level, joining designation, grade, and quaterly rating
distinct_counts = {
    'months': odf['MMM-YY'].nunique(),
    'driver_id': odf['Driver_ID'].nunique(),
    'age': odf['Age'].nunique(),
    'gender': odf['Gender'].nunique(),
    'city': odf['City'].nunique(),
    'education_level': odf['Education_Level'].nunique(),
    'joining degination': odf['Joining Designation'].nunique(),
    'grade': odf['Grade'].nunique(),
    'quarterly_rating': odf['Quarterly Rating'].nunique()
}

print("Distinct Value Counts:")
for key, value in distinct_counts.items():
    print(f"{key}: {value}")
```

```
Distinct Value Counts:
months: 24
driver_id: 2308
age: 36
gender: 2
city: 29
education_level: 3
joining degination: 5
grade: 5
quarterly_rating: 4
```

```
# calculate the average number of joins and leaves per month
```

```
monthly_joins = odf.groupby(odf['Dateofjoining'].dt.to_period('M'))
['Driver_ID'].count()
monthly_leaves = odf.groupby(odf['LastWorkingDate'].dt.to_period('M'))
['Driver_ID'].count()

monthly_joins.index = monthly_joins.index.to_timestamp()
monthly_leaves.index = monthly_leaves.index.to_timestamp()

avg_joins_per_month = monthly_joins.mean().round(0)
avg_leaves_per_month = monthly_leaves.mean().round(0)

print(f"Average drivers joined per month: {avg_joins_per_month:.2f}")
print(f"Average drivers left per month: {avg_leaves_per_month:.2f}")
```

```
Average drivers joined per month: 214.00
Average drivers left per month: 64.00
```

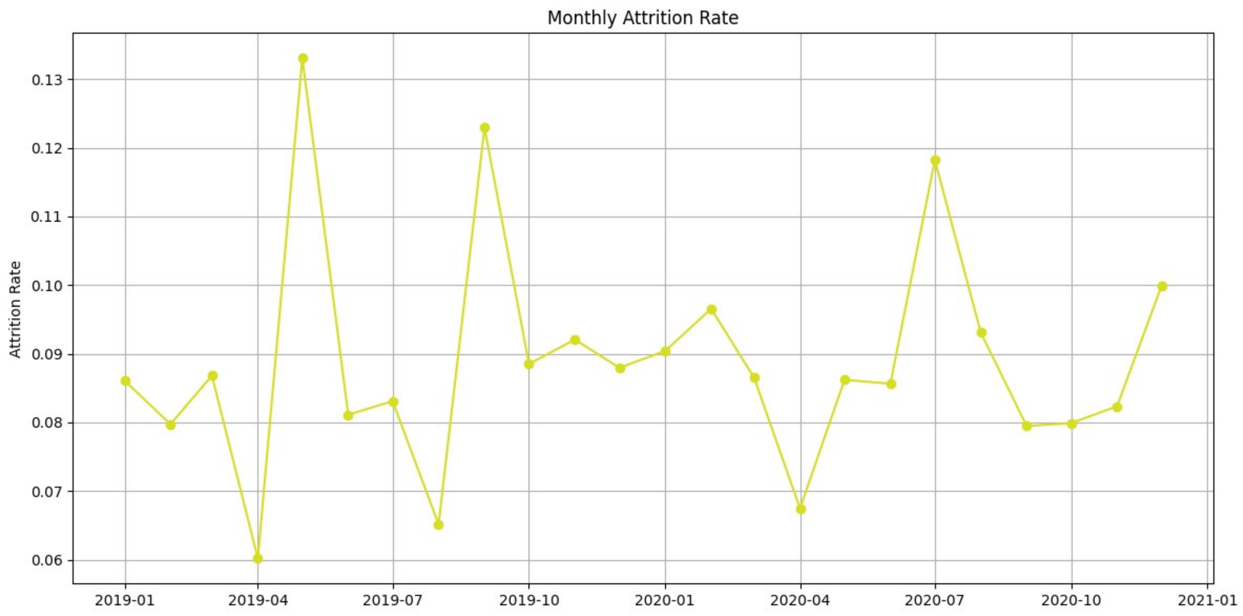
```
#  the average tenure of drivers in months
average_tenure = odf['Total_Tenure_Months'].dropna().mean().round(0)
print(f"Average tenure of drivers: {average_tenure:.2f} months")
```

```
Average tenure of drivers: 21.00 months
```

Data Visualization

Attrition Rate Over Time (MMM-YY)

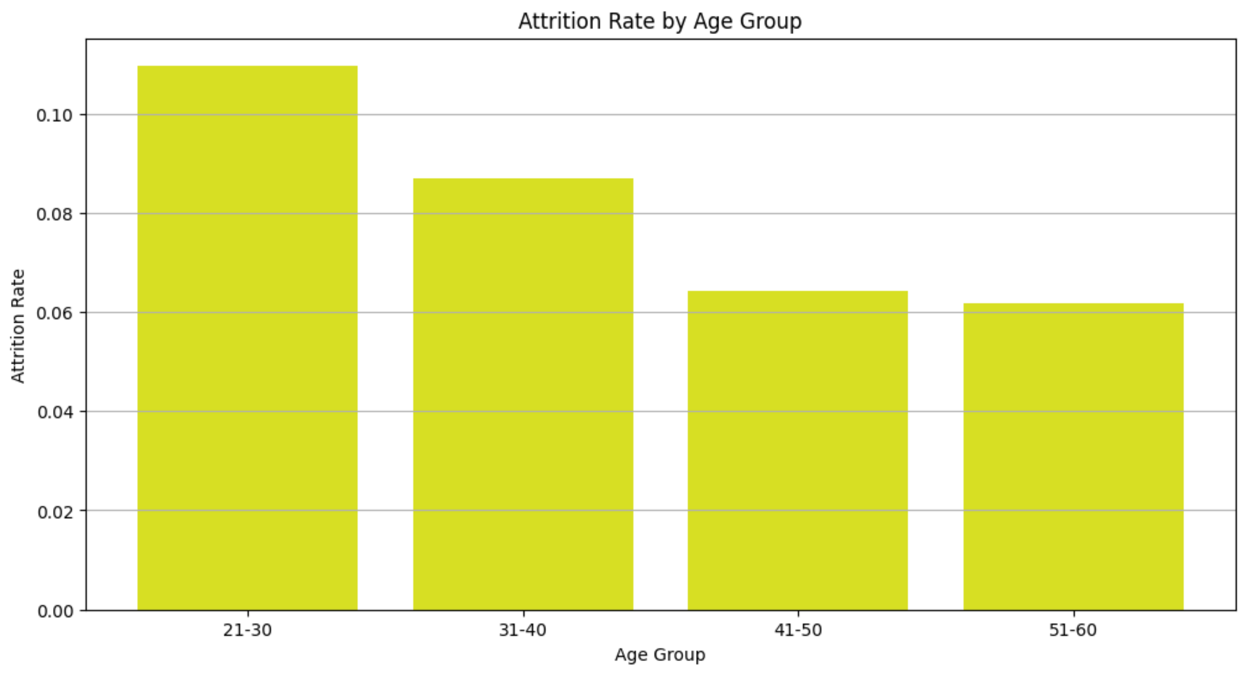
```
# monthly attrition rate
monthly_attrition = odf.groupby('MMM-YY')['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.plot(monthly_attrition.index, monthly_attrition.values,
marker='o', color='#D7DF23')
plt.title('Monthly Attrition Rate')
plt.xlabel(None)
plt.ylabel('Attrition Rate')
plt.xticks(rotation=0)
plt.grid(True)
plt.tight_layout()
plt.show()
```



- Attrition rates showed noticeable fluctuations across 2019 and 2020.
- Peaks were observed in May 2019 (13.3%), September 2019 (11.7%), and July 2020 (10.8%), suggesting possible seasonal or operational factors influencing driver exits during these periods.
- Conversely, months like April 2019 (5.9%) and April 2020 (6.3%) recorded lower attrition, potentially impacted by policy changes, lockdown effects, or reduced operations.

Attrition Rate by Age Group

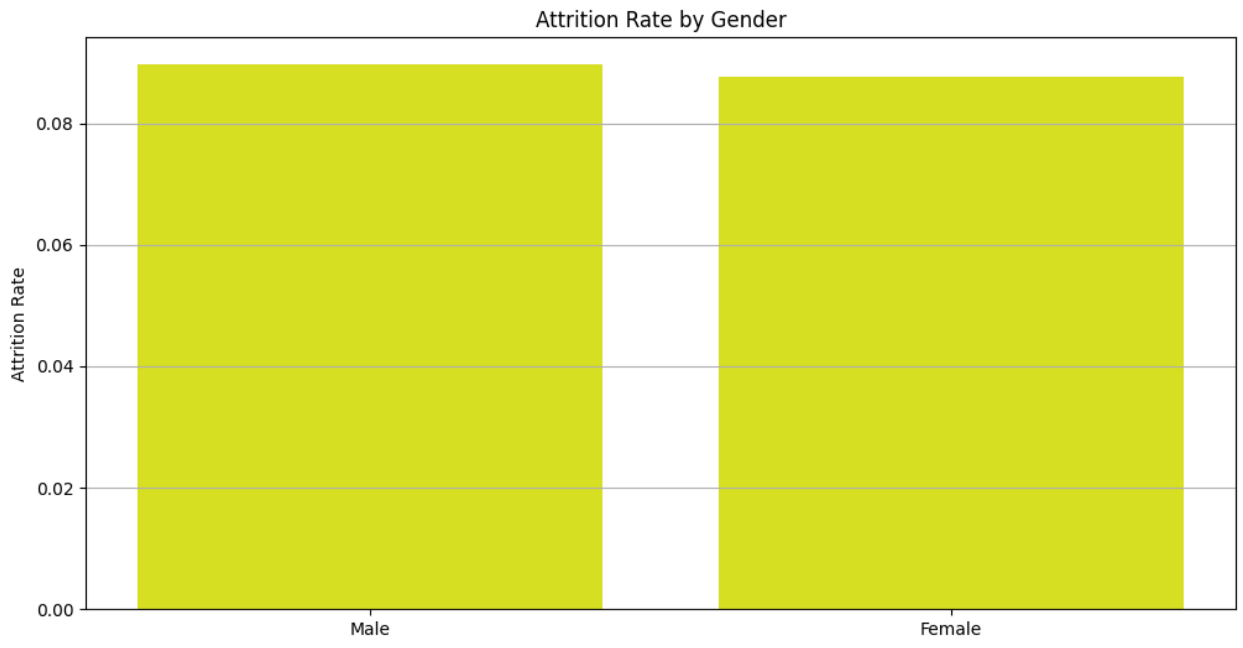
```
odf['Age_Bin'] = pd.cut(odf['Age'], bins=[20, 30, 40, 50, 60],
labels=['21-30', '31-40', '41-50', '51-60'])

age_attrition = odf.groupby('Age_Bin')['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.bar(age_attrition.index.astype(str), age_attrition.values,
color='#D7DF23')
plt.title('Attrition Rate by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Younger drivers in the 21–30 age group exhibit the highest attrition rate at 10.2%, which gradually declines with age — down to 6.1% for the 51–60 group.
- This indicates that younger drivers are more likely to leave, possibly due to job exploration, lack of commitment, or dissatisfaction with early earnings.

Attrition by Gender

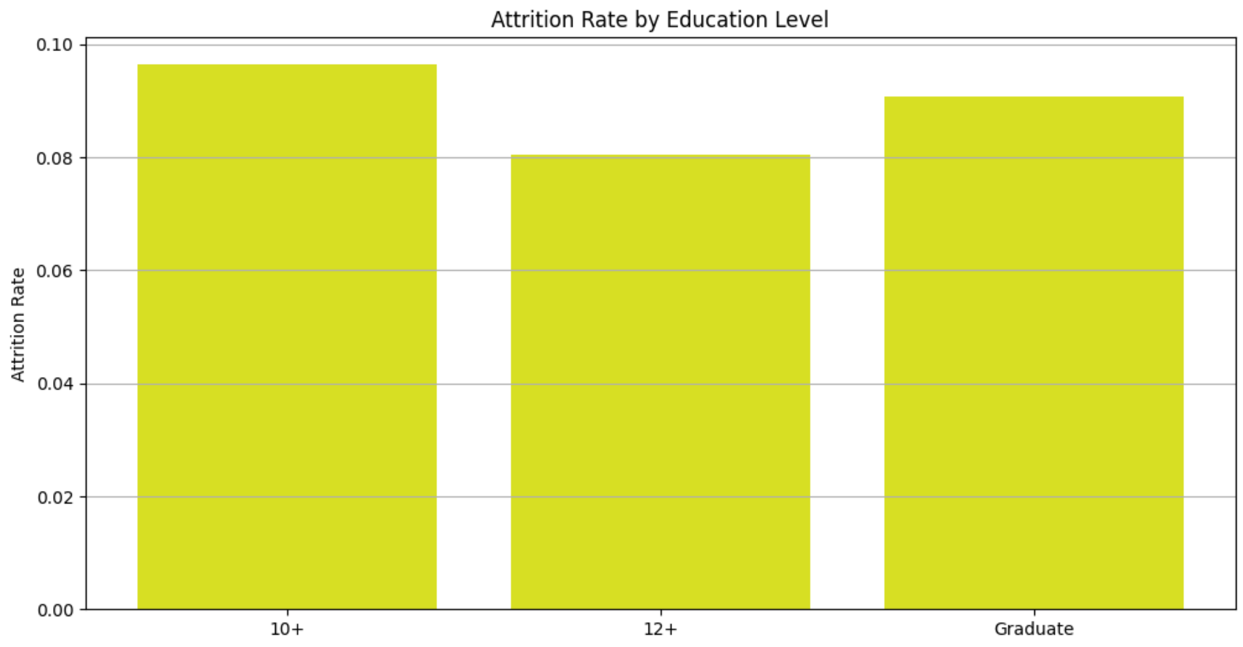
```
gender_attrition = odf.groupby('Gender')['Attrition'].mean()
labels = ['Male', 'Female']

plt.figure(figsize=(12,6))
plt.bar(labels, gender_attrition.values, color='#D7DF23')
plt.title('Attrition Rate by Gender')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- The attrition rate is slightly higher for male drivers (8.5%) compared to female drivers (8.3%), though the difference is marginal.
- This suggests that gender is not a significant differentiator in driver attrition for this dataset.

Attrition by Education Level

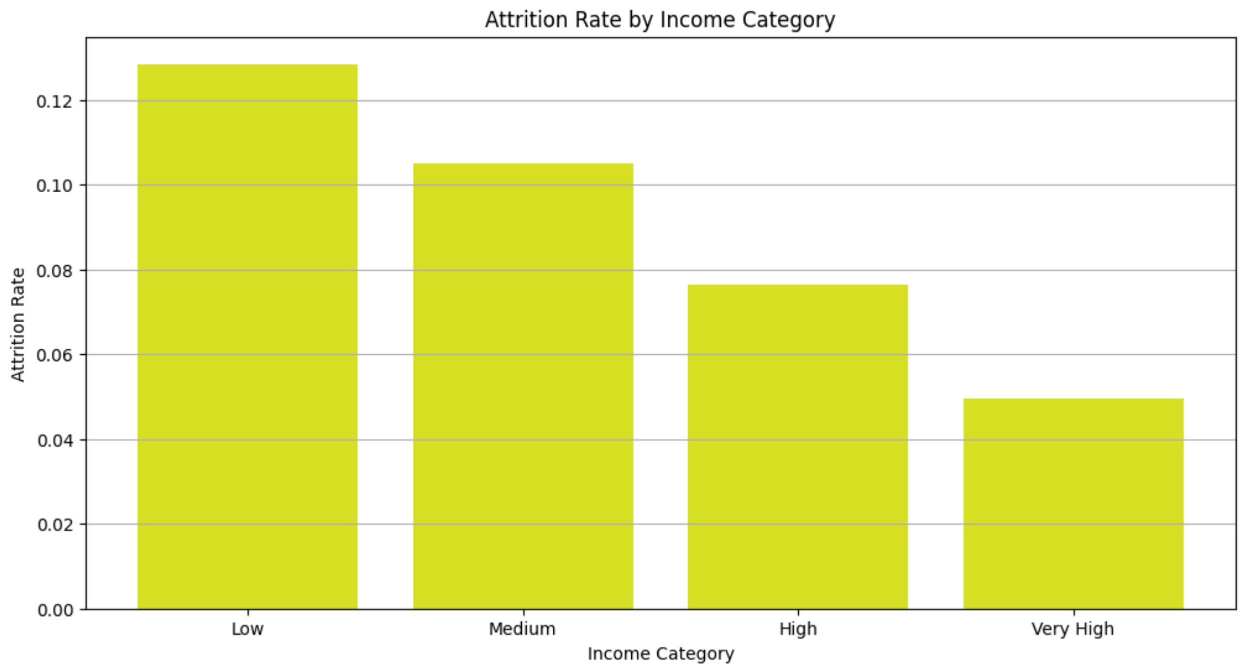
```
education_attrition = odf.groupby('Education_Level')
['Attrition'].mean()
labels = ['10+', '12+', 'Graduate']

plt.figure(figsize=(12,6))
plt.bar(labels, education_attrition.values, color='#D7DF23')
plt.title('Attrition Rate by Education Level')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Drivers with the lowest education level (10+) show the highest attrition rate (9.2%), followed by graduates (8.6%), while those with 12+ education have the lowest attrition rate (7.7%).
- This suggests that moderate education (12+) might align better with job expectations and satisfaction in this role.

Attrition by Income Category

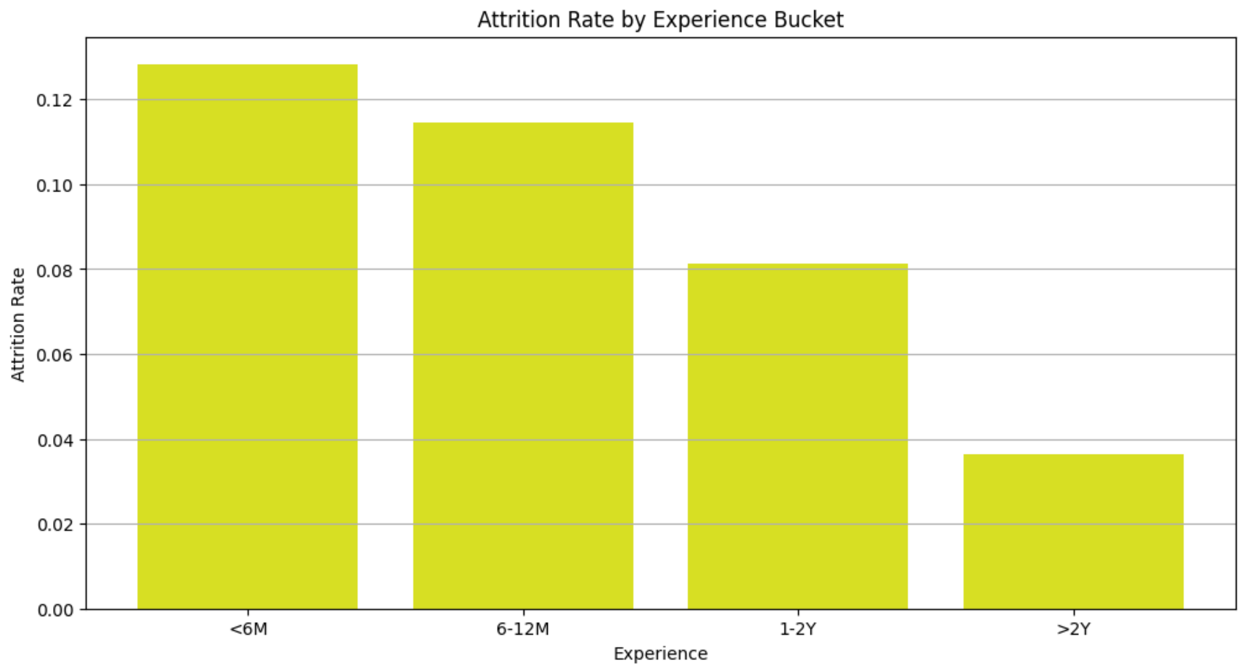
```
income_attrition = odf.groupby('Income_Band')['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.bar(income_attrition.index.astype(str), income_attrition.values,
color='#D7DF23')
plt.title('Attrition Rate by Income Category')
plt.xlabel('Income Category')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Attrition shows a clear inverse relationship with income.
- Drivers in the Low-income group face the highest attrition rate (12.1%), while those in the Very High-income group have a significantly lower attrition rate of 4.8%.
- This suggests that financial satisfaction plays a critical role in driver retention.

Attrition by Experience Bucket

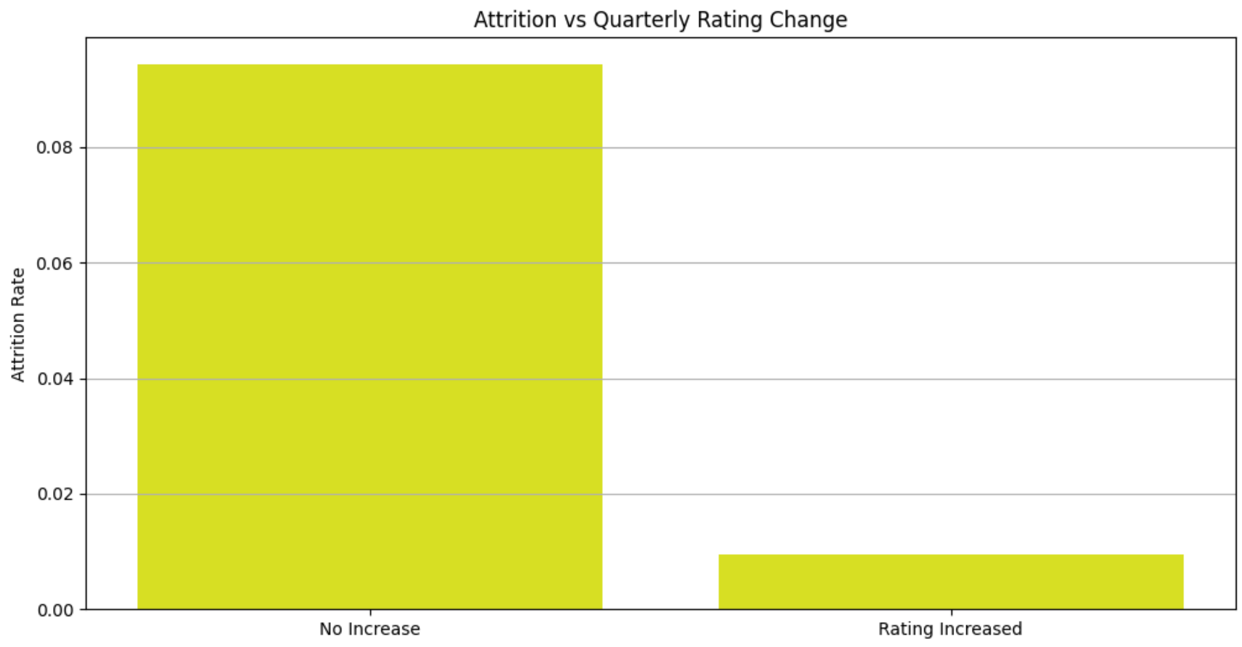
```
exp_attrition = odf.groupby('Experience_Bucket')['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.bar(exp_attrition.index.astype(str), exp_attrition.values,
color='#D7DF23')
plt.title('Attrition Rate by Experience Bucket')
plt.xlabel('Experience')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Drivers with less than 6 months of experience have the highest attrition rate (15.2%), which steadily declines as experience increases, dropping to just 3.6% for drivers with over 2 years of experience.
- This strongly indicates that early-stage drivers are more prone to quitting, possibly due to unmet expectations or adjustment issues.

Attrition by Rating Increase

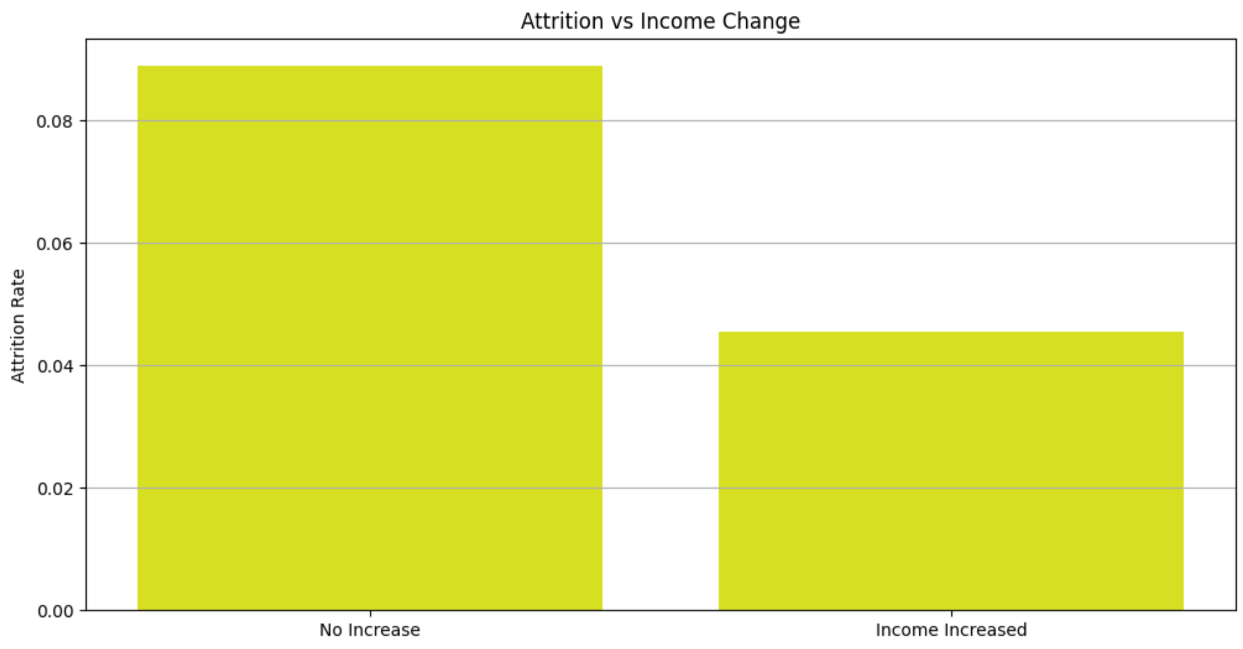
```
rating_increase_attrition = odf.groupby('Rating_Increased')
['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.bar(['No Increase', 'Rating Increased'],
rating_increase_attrition.values, color='#D7DF23')
plt.title('Attrition vs Quarterly Rating Change')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Drivers whose quarterly ratings increased had a significantly lower attrition rate (0.88%) compared to those whose ratings did not improve (8.99%).
- This suggests that performance growth and recognition play a vital role in driver motivation and retention.

Attrition by Income Increase

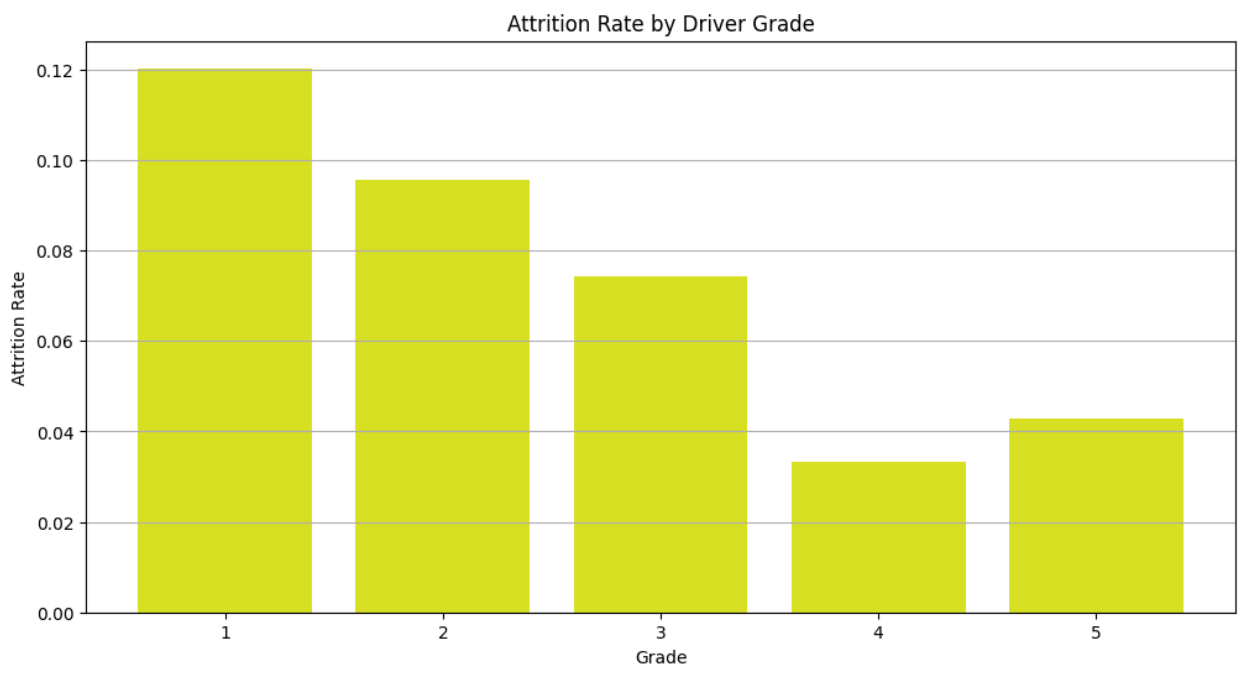
```
income_increase_attrition = odf.groupby('Income_Increased')
['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.bar(['No Increase', 'Income Increased'],
income_increase_attrition.values, color='#D7DF23')
plt.title('Attrition vs Income Change')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Drivers who experienced an increase in income had a significantly lower attrition rate (4.5%) compared to those whose income did not increase (8.5%).
- This highlights that financial progression strongly contributes to driver retention.

Attrition by Grade

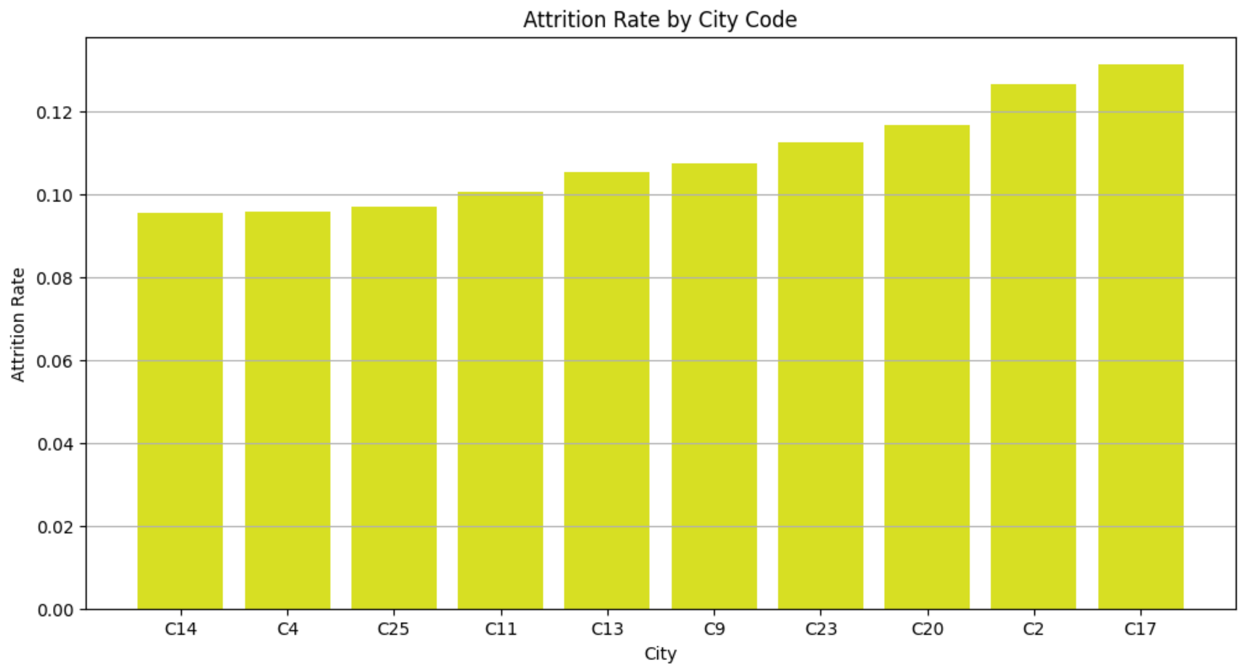
```
grade_attrition = odf.groupby('Grade')['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.bar(grade_attrition.index.astype(str), grade_attrition.values,
color='#D7DF23')
plt.title('Attrition Rate by Driver Grade')
plt.xlabel('Grade')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Attrition decreases progressively with higher grades. Grade 1 drivers show the highest attrition rate (11.4%), while Grade 4 and 5 drivers have significantly lower rates (3.3% and 4.3%) respectively.
- This suggests that higher-grade drivers are more stable and committed, likely due to experience, performance, or better incentives.

Attrition by City

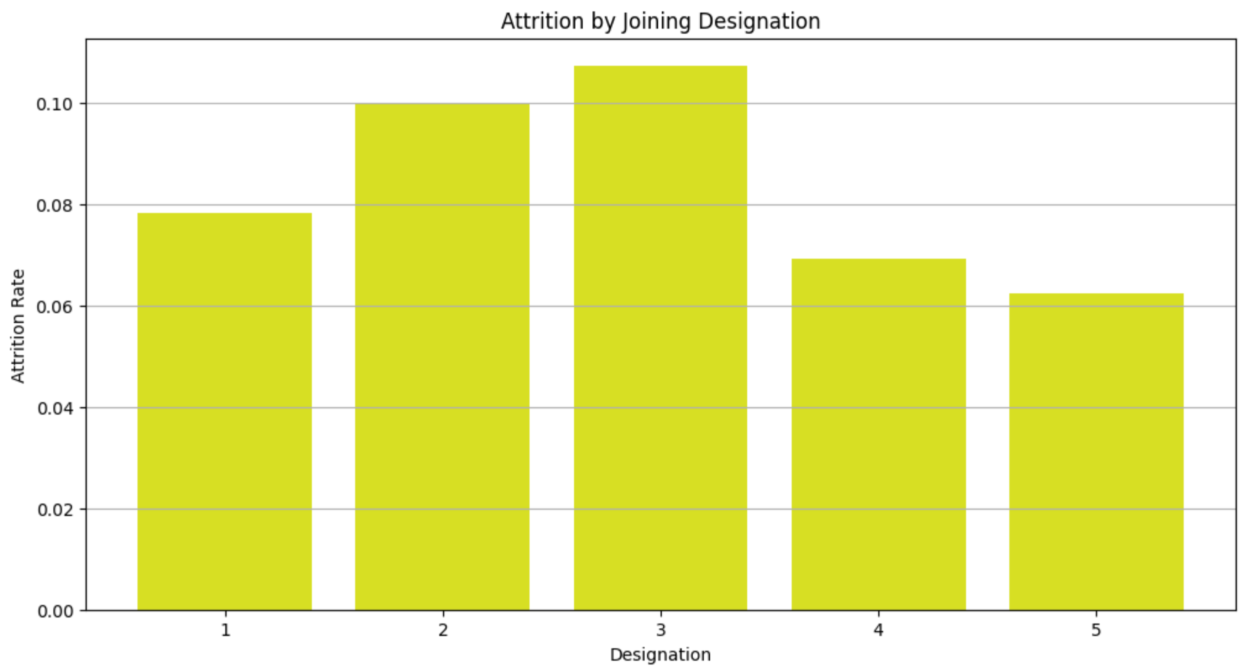
```
city_attrition = odf.groupby('City')
['Attrition'].mean().sort_values().tail(10)

plt.figure(figsize=(12,6))
plt.bar(city_attrition.index.astype(str), city_attrition.values,
color='#D7DF23')
plt.title('Attrition Rate by City Code')
plt.xlabel('City')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Attrition rates vary significantly by city, ranging from 5.7% in City C29 to over 12.6% in City C17.
- Cities like C2, C20, C23, and C17 report notably higher attrition, indicating possible regional challenges such as poor local support, market competition, or lower earnings potential

Attrition by Joining Designation

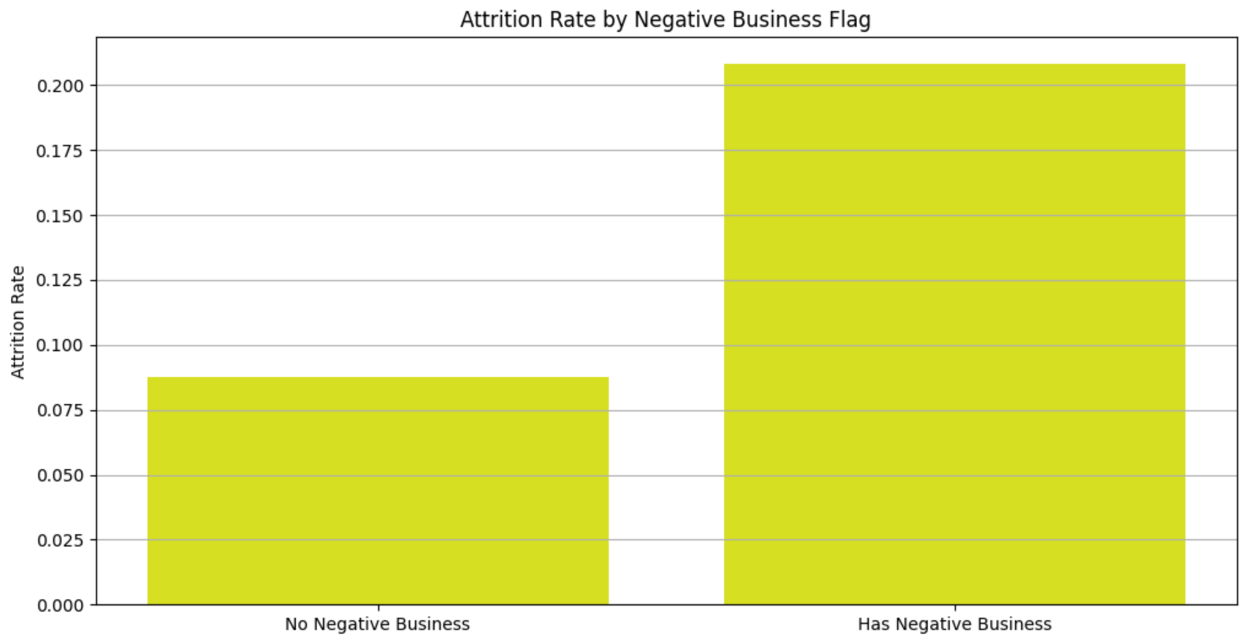
```
designation_attrition = odf.groupby('Joining Designation')
['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.bar(designation_attrition.index.astype(str),
designation_attrition.values, color='#D7DF23')
plt.title('Attrition by Joining Designation')
plt.xlabel('Designation')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Attrition rates differ by the driver's joining designation.
- Drivers who joined with designation 1 show the lowest attrition rate (7.6%), while those with designations 2 and 3 have higher attrition rates of 9.4% and 9.7% respectively.
- Designations 4 and 5 also have relatively lower attrition (6.5% and 6.2%).

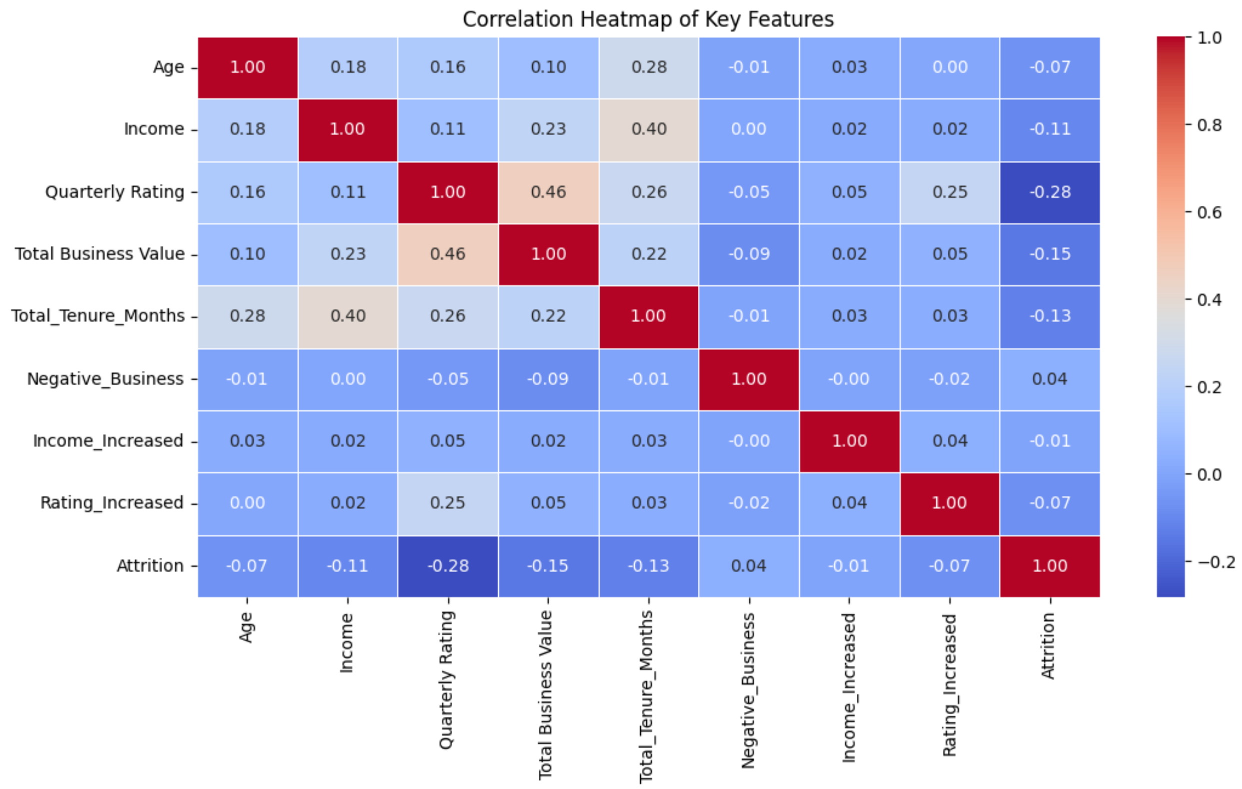
Attrition by Negative Business

```
negative_attrition = odf.groupby('Negative_Business')
['Attrition'].mean()

plt.figure(figsize=(12,6))
plt.bar(['No Negative Business', 'Has Negative Business'],
negative_attrition.values, color='#D7DF23')
plt.title('Attrition Rate by Negative Business Flag')
plt.ylabel('Attrition Rate')
plt.grid(axis='y')
plt.show()
```



- Drivers with negative business transactions (e.g., cancellations, refunds, or EMI adjustments) exhibit a much higher attrition rate (20.8%) compared to those without negative business (8.4%).
- This indicates that financial setbacks or losses are strongly linked to driver churn.

```
# Create a DataFrame with relevant numeric & binary features
corr_df = odf[['Age', 'Income', 'Quarterly Rating', 'Total Business 
Value',
              'Total_Tenure_Months', 'Negative_Business', 
'Income_Increased',
              'Rating_Increased', 'Attrition']]

# Calculate correlation matrix
corr_matrix = corr_df.corr()

plt.figure(figsize=(12,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", 
linewidths=0.5)
plt.title('Correlation Heatmap of Key Features')
plt.show()
```



	Age	Income	Quarterly Rating	Total Business Value	Total_Tenure_Months	Negative_Business	Income_Increased	Rating_Increased	Attrition
Age	1.00	0.18	0.16	0.10	0.28	-0.01	0.03	0.00	-0.07
Income	0.18	1.00	0.11	0.23	0.40	0.00	0.02	0.02	-0.11
Quarterly Rating	0.16	0.11	1.00	0.46	0.26	-0.05	0.05	0.25	-0.28
Total Business Value	0.10	0.23	0.46	1.00	0.22	-0.09	0.02	0.05	-0.15
Total_Tenure_Months	0.28	0.40	0.26	0.22	1.00	-0.01	0.03	0.03	-0.13
Negative_Business	-0.01	0.00	-0.05	-0.09	-0.01	1.00	-0.00	-0.02	0.04
Income_Increased	0.03	0.02	0.05	0.02	0.03	-0.00	1.00	0.04	-0.01
Rating_Increased	0.00	0.02	0.25	0.05	0.03	-0.02	0.04	1.00	-0.07
Attrition	-0.07	-0.11	-0.28	-0.15	-0.13	0.04	-0.01	-0.07	1.00

How We Detect High-Risk Drivers

After digging deep into the data, I found that certain traits are commonly seen in drivers who end up leaving Ola. So, to spot high-risk drivers, I just checked if they match any of these patterns.

Here's what I observed:

- **Younger drivers** (especially age 21–30) tend to leave more often.
- **Less experience** (drivers with <6 months or 6–12 months experience) showed higher attrition.
- If a driver's **quarterly rating didn't improve**, they're more likely to churn.
- If a driver's **income didn't go up**, they might be unsatisfied and leave.
- **Negative business value** (like refunds or cancellations) also connects with higher attrition.
- Drivers in **lower grades or lower income groups** also showed higher risk.
- **Lower education levels** slightly increased risk but not as much as other factors.

So the plan is: if any driver falls into one or more of these categories, I'll mark them as a "high-risk" driver. This list can help the company target specific people for retention maybe offer them support, guidance, or incentives to stay.

```
# define high-risk conditions
high_risk_conditions = (
    (odf['Age_Bin'] == '21-30') |
    (odf['Experience_Bucket'].isin(['<6M', '6-12M'])) |
    (odf['Rating_Increased'] == 0) |
    (odf['Income_Increased'] == 0) |
    (odf['Negative_Business'] == 1)
)

# filter the dataframe
high_risk_drivers = odf[high_risk_conditions]

# get unique list of high-risk driver ids
high_risk_driver_ids = high_risk_drivers['Driver_ID'].unique()

# display result
print("High-risk Driver IDs (potential attrition):")
print(high_risk_driver_ids)

# Export to CSV
pd.DataFrame({'High_Risk_Driver_ID':
high_risk_driver_ids}).to_csv("high_risk_drivers.csv", index=False)
```

```
High-risk Driver IDs (potential attrition):
[   1    2    4 ... 2786 2787 2788]
```

```
# save the dataframe to a csv file
odf.to_csv('ola_driver_attrition_features.csv', index=False)
```

Recommendations

Implement Targeted Retention Strategies for Younger and Less Experienced Drivers

- Provide mentorship, onboarding support, and early incentives to reduce churn in the 21-30 age group and drivers with less than 6 months experience.

Encourage Continuous Performance Improvement and Recognition

- Introduce regular performance feedback, rating improvement incentives, and recognition programs to motivate drivers and lower attrition.

Create Clear Income Growth Pathways

- Offer structured income increases, bonuses, and milestone rewards to drivers to boost financial motivation and retention.

Design Grade and Designation-Based Career Progression Plans

- Develop transparent promotion policies with benefits tied to driver grades and joining designations to increase loyalty.

Focus on City-Specific Interventions

- Analyze high-attrition cities closely and implement localized engagement plans, improved support services, and incentives tailored to each city's needs.

Provide Support for Drivers Facing Financial Challenges

- Monitor drivers with negative business transactions and offer counseling, financial assistance, or dispute resolution to mitigate losses and reduce churn.

Maintain Consistent Communication and Feedback Loops

- Engage drivers through frequent communication channels to address grievances, provide updates, and foster a sense of belonging.

Use Data-Driven Insights for Proactive Attrition Prediction

- Regularly analyze driver data to identify at-risk segments and intervene early with personalized retention efforts.