In [ ]:
```python
import pandas as pd
pd.set_option("display.max_columns", None)
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```python
df = pd.read_csv("dataset/phase1_df.csv")
df.head()
```

Out[ ]:

	Age	Gender	City	Education_Level	Joining Designation	joining date	joining month	joining year	ReportCount	Active	TBV_avg	QR_avg	Income_avg	Grade_avg	income_diff_pattern	tbv_diff_pattern	qr_diff_pattern	grade_diff_pattern
0	28.000000	0.0	C23	2	1	24	12	18	3	0	571860.000000	2.000000	57387.0	1.0	0.0	-2381060.0	0.0	0.0
1	29.000000	0.0	C9	0	1	1	9	19	3	0	40120.000000	1.000000	46368.0	1.0	0.0	0.0	0.0	0.0
2	29.608696	0.0	C19	2	1	28	5	15	23	0	444045.217391	1.260870	119227.0	4.0	0.0	-250000.0	0.0	0.0
3	30.000000	1.0	C23	0	2	30	11	18	2	0	173400.000000	1.000000	52963.0	2.0	0.0	-346800.0	0.0	0.0
4	42.142857	0.0	C20	2	1	3	6	18	7	0	145377.142857	1.428571	51099.0	1.0	0.0	-100000.0	-1.0	0.0

In [ ]:
```python
df.describe(include="object")
```

Out[ ]:

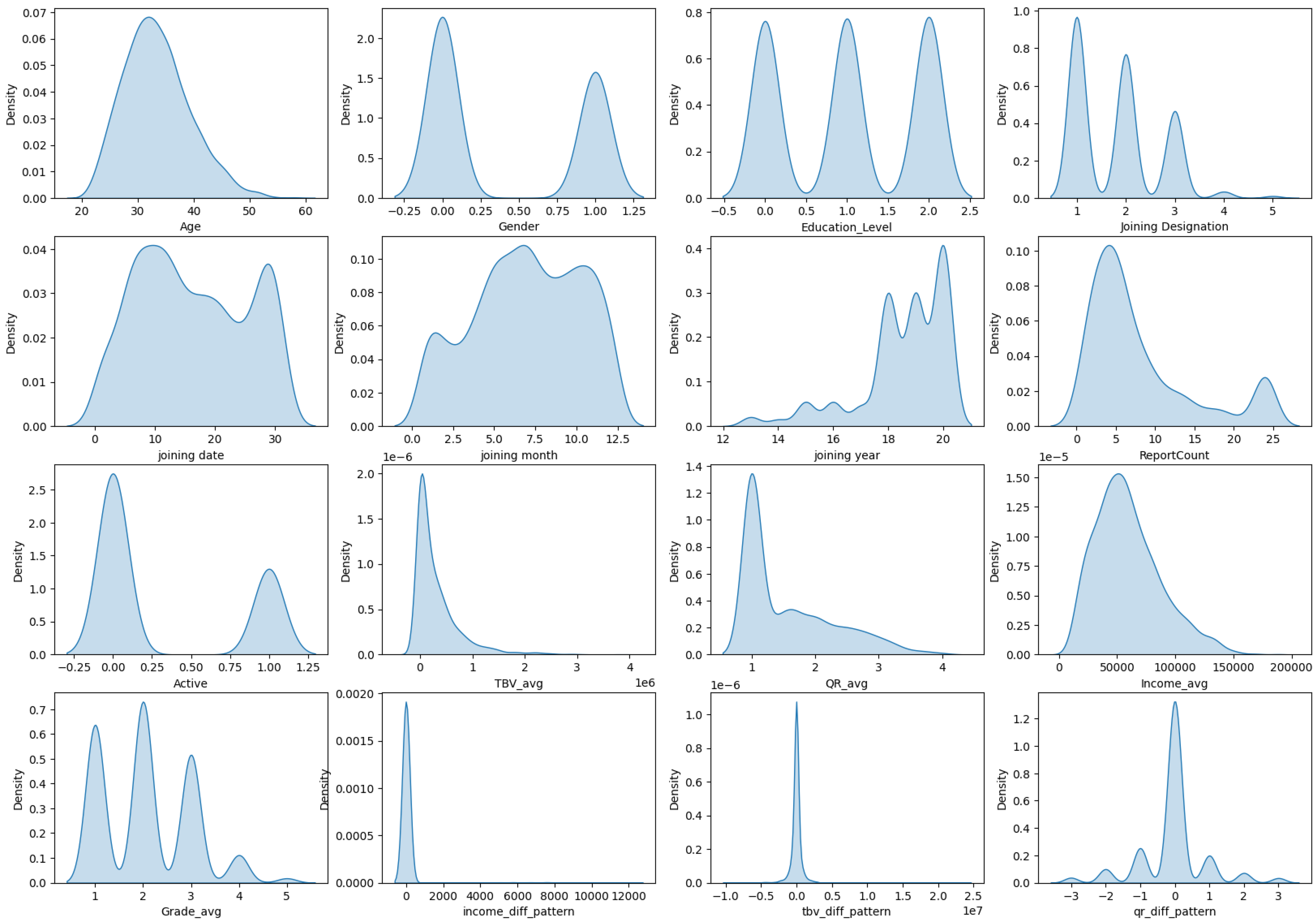
	City
count	2381
unique	29
top	C20
freq	152

In [ ]:
```python
df.describe()
```

Out[ ]:

	Age	Gender	Education_Level	Joining Designation	joining date	joining month	joining year	ReportCount	Active	TBV_avg	QR_avg	Income_avg	Grade_avg	income_diff_pattern	tbv_diff_pattern	qr_diff_pattern	grade_diff_pattern
count	2381.000000	2381.000000	2381.00000	2381.000000	2381.000000	2381.000000	2381.000000	2381.00000	2381.000000	2.381000e+03	2381.000000	2381.000000	2381.000000	2381.000000	2.381000e+03	2381.000000	2381.000000
mean	33.369192	0.410332	1.00756	1.820244	16.186896	6.958001	18.536329	8.02352	0.321294	3.120854e+05	1.566304	59232.460484	2.081713	125.096178	-2.440566e+04	-0.058379	0.018060
std	5.890732	0.491997	0.81629	0.841433	8.959616	3.221762	1.609597	6.78359	0.467071	4.495705e+05	0.719652	28298.214012	0.932257	968.511766	1.182482e+06	0.933703	0.133195
min	21.000000	0.000000	0.00000	1.000000	1.000000	1.000000	13.000000	1.00000	0.000000	-1.979329e+05	1.000000	10747.000000	1.000000	0.000000	-9.658160e+06	-3.000000	0.000000
25%	29.000000	0.000000	0.00000	1.000000	9.000000	5.000000	18.000000	3.00000	0.000000	0.000000e+00	1.000000	39104.000000	1.000000	0.000000	-1.166600e+05	0.000000	0.000000
50%	33.000000	0.000000	1.00000	2.000000	15.000000	7.000000	19.000000	5.00000	0.000000	1.506244e+05	1.000000	55285.000000	2.000000	0.000000	0.000000e+00	0.000000	0.000000
75%	37.000000	1.000000	2.00000	2.000000	24.000000	10.000000	20.000000	10.00000	1.000000	4.294988e+05	2.000000	75835.000000	3.000000	0.000000	0.000000e+00	0.000000	0.000000
max	58.000000	1.000000	2.00000	5.000000	31.000000	12.000000	20.000000	24.00000	1.000000	3.972128e+06	4.000000	188418.000000	5.000000	12155.000000	2.391754e+07	3.000000	1.000000

In [ ]:
```python
fig=plt.figure(figsize=(20,14)) # width*height
for ind_number, col_name in enumerate(df.describe().columns):
    if ind_number<16:
        plt.subplot(4,4,ind_number+1)
        sns.kdeplot(df[col_name], fill=True)
```



In [ ]:
```python
df[df["income_diff_pattern"]>10000]
```

Out[ ]:

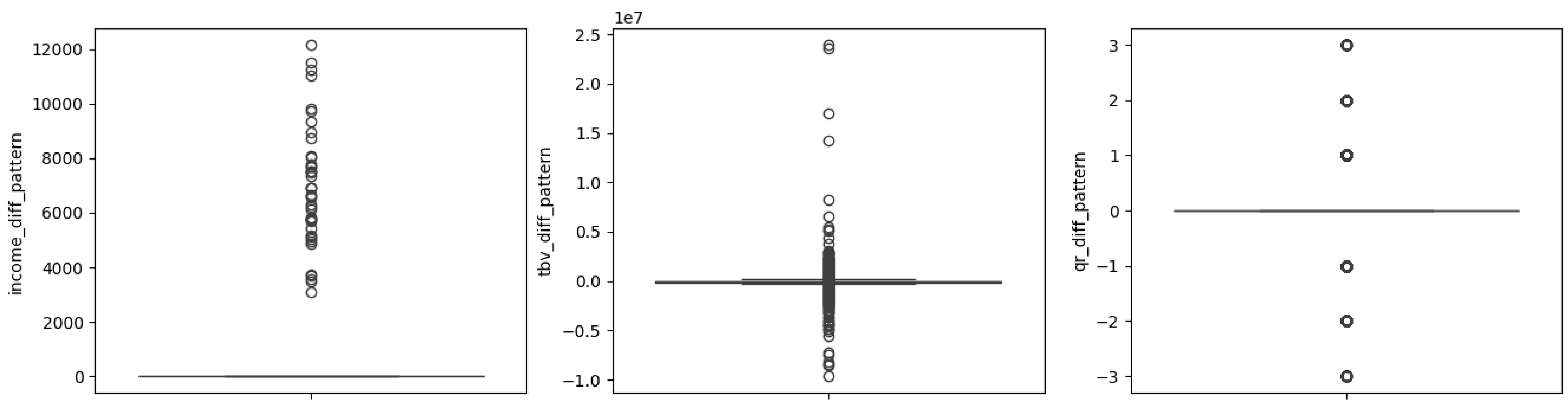
	Age	Gender	City	Education_Level	Joining Designation	joining date	joining month	joining year	ReportCount	Active	TBV_avg	QR_avg	Income_avg	Grade_avg	income_diff_pattern	tbv_diff_pattern	qr_diff_pattern	grade_diff_pattern
10	41.833333	0.0	C14	2	1	5	7	18	24	1	2.911162e+06	3.083333	126132.333333	3.416667	11048.0	1210110.0	-2.0	1.0
240	36.666667	0.0	C16	2	2	13	7	13	24	1	2.283013e+06	3.125000	139347.500000	4.125000	11492.0	1313620.0	-2.0	1.0
456	52.583333	0.0	C15	2	1	6	8	18	24	1	2.124457e+06	3.375000	136744.750000	3.250000	12155.0	-2900660.0	-2.0	1.0
938	48.333333	1.0	C9	2	1	6	3	18	24	1	1.803128e+06	3.250000	126779.250000	3.250000	11269.0	1376900.0	-2.0	1.0

Understanding:

- Mean income lies around 50k for all the drives.
- We have drivers with either reporting count as 5 and it keeps on droping till a certain band and then increase after 20.
- Getting a increase in number of drivers joining on yearly basis.
- June seems to be the favaouable month to join OLA for Drivers.

In [ ]:
```python
fig=plt.figure(figsize=(16,4)) # width*height

plt.subplot(1,3,1)
sns.boxplot(data=df["income_diff_pattern"])

plt.subplot(1,3,2)
sns.boxplot(data=df["tbv_diff_pattern"])

plt.subplot(1,3,3)
sns.boxplot(data=df["qr_diff_pattern"])
```

Out[ ]: <Axes: ylabel='qr_diff_pattern'>



- Let us categorises the values based on their magnitute.

```
In [ ]: # income_diff_pattern : based on box plot let us categorize in 5 bins 1,2,3,4,5

        # Define bin edges
        bin_edges = [0, 2000, 4000, 8000, 10000, df['income_diff_pattern'].max()]

        # Define bin labels
        bin_labels = [1, 2, 3, 4, 5]

        # Create a new column 'income_diff_pattern' based on the bins
        df['income_diff_pattern'] = pd.cut(df['income_diff_pattern'], bins=bin_edges, labels=bin_labels, include_lowest=True).astype(int)
```

```
In [ ]: # tbv_diff_pattern : based on box plot

        Q1 = df["tbv_diff_pattern"].quantile(0.05)
        Q2 = df["tbv_diff_pattern"].quantile(0.15)
        Q3 = df["tbv_diff_pattern"].quantile(0.85)
        Q4 = df["tbv_diff_pattern"].quantile(0.95)

        print(Q1,Q2,Q3,Q4, df['tbv_diff_pattern'].max())


        # Define bin edges
        bin_edges = [df['tbv_diff_pattern'].min(), Q1, Q2, 0, Q3, Q4, df['tbv_diff_pattern'].max()]

        # Define bin labels
        bin_labels = [-3, -2, -1, 1, 2, 3]

        # Create a new column 'income_diff_pattern' based on the bins
        df['tbv_diff_pattern'] = pd.cut(df['tbv_diff_pattern'], bins=bin_edges, labels=bin_labels, include_lowest=True).astype(int)
```

```
-1048330.0 -398770.0 261100.0 950000.0 23917540.0
```

```
In [ ]: df.head()
```

Out[ ]:

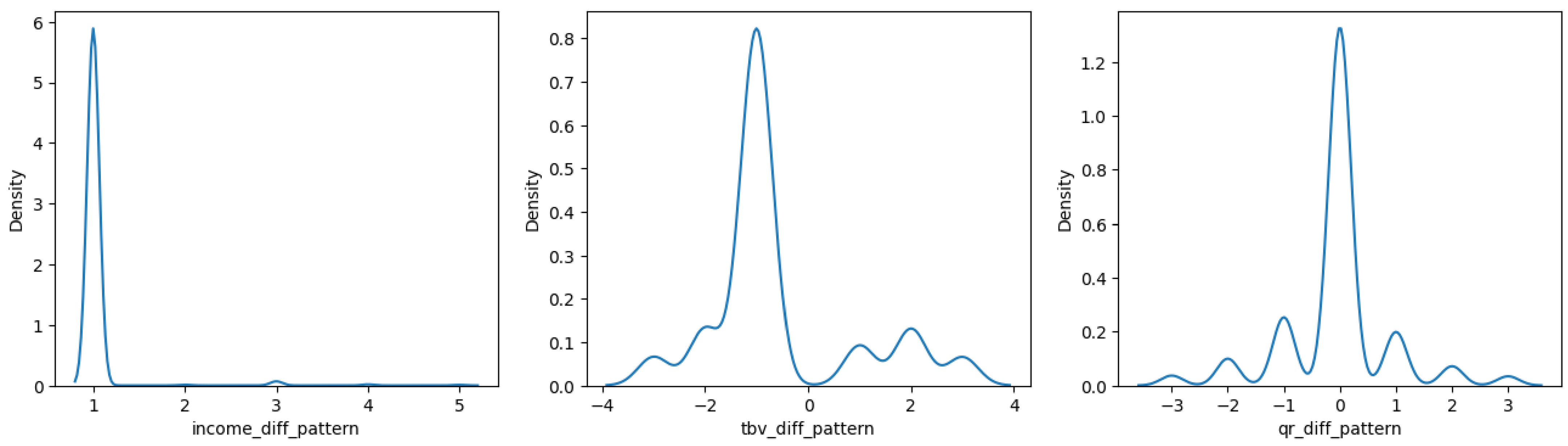
	Age	Gender	City	Education_Level	Joining Designation	joining date	joining month	joining year	ReportCount	Active	TBV_avg	QR_avg	Income_avg	Grade_avg	income_diff_pattern	tbv_diff_pattern	qr_diff_pattern	grade_diff_pattern
0	28.000000	0.0	C23	2	1	24	12	18	3	0	571860.000000	2.000000	57387.0	1.0	1	-3	0.0	0.0
1	29.000000	0.0	C9	0	1	1	9	19	3	0	40120.000000	1.000000	46368.0	1.0	1	-1	0.0	0.0
2	29.608696	0.0	C19	2	1	28	5	15	23	0	444045.217391	1.260870	119227.0	4.0	1	-1	0.0	0.0
3	30.000000	1.0	C23	0	2	30	11	18	2	0	173400.000000	1.000000	52963.0	2.0	1	-1	0.0	0.0
4	42.142857	0.0	C20	2	1	3	6	18	7	0	145377.142857	1.428571	51099.0	1.0	1	-1	-1.0	0.0

```
In [ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Age                  2381 non-null   float64
 1   Gender               2381 non-null   float64
 2   City                 2381 non-null   object 
 3   Education_Level      2381 non-null   int64  
 4   Joining Designation  2381 non-null   int64  
 5   joining date         2381 non-null   int64  
 6   joining month        2381 non-null   int64  
 7   joining year         2381 non-null   int64  
 8   ReportCount          2381 non-null   int64  
 9   Active               2381 non-null   int64  
 10  TBV_avg              2381 non-null   float64
 11  QR_avg               2381 non-null   float64
 12  Income_avg           2381 non-null   float64
 13  Grade_avg            2381 non-null   float64
 14  income_diff_pattern  2381 non-null   int64  
 15  tbv_diff_pattern     2381 non-null   int64  
 16  qr_diff_pattern      2381 non-null   float64
 17  grade_diff_pattern   2381 non-null   float64
dtypes: float64(8), int64(9), object(1)
memory usage: 335.0+ KB
```

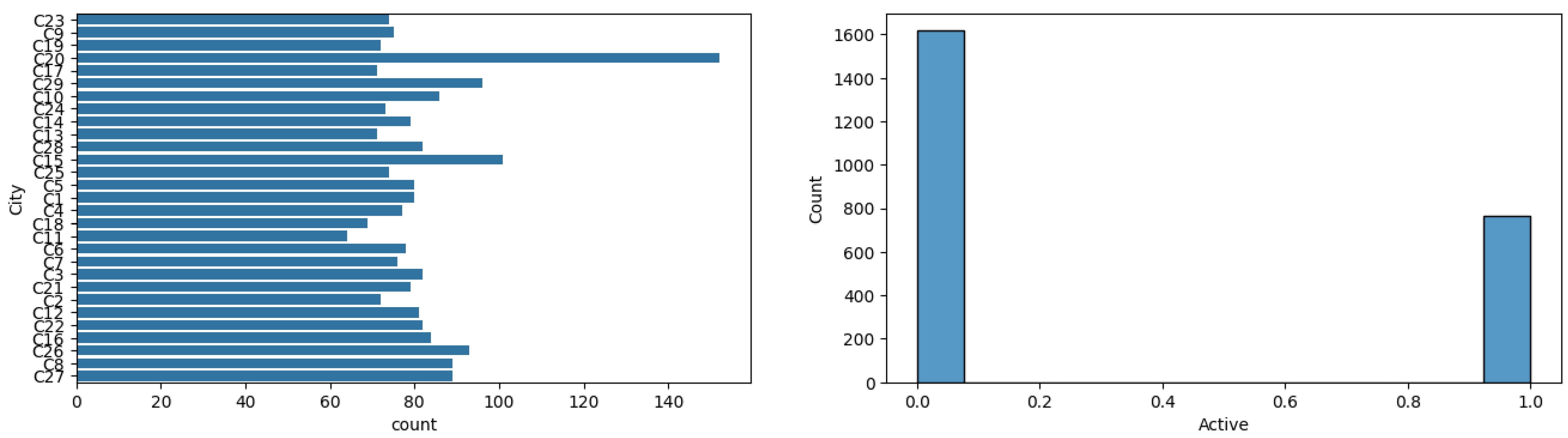
```
In [ ]: fig=plt.figure(figsize=(16,4)) # width*height

        plt.subplot(1,3,1)
        sns.kdeplot(data=df["income_diff_pattern"])

        plt.subplot(1,3,2)
        sns.kdeplot(data=df["tbv_diff_pattern"])

        plt.subplot(1,3,3)
        sns.kdeplot(data=df["qr_diff_pattern"])
```

```
Out[ ]: <Axes: xlabel='qr_diff_pattern', ylabel='Density'>
```




```
In [ ]: fig=plt.figure(figsize=(16,4)) # width*height

        plt.subplot(1,2,1)
        sns.countplot(df["City"], fill=True)

        plt.subplot(1,2,2)
        sns.histplot(df["Active"], fill=True)
```

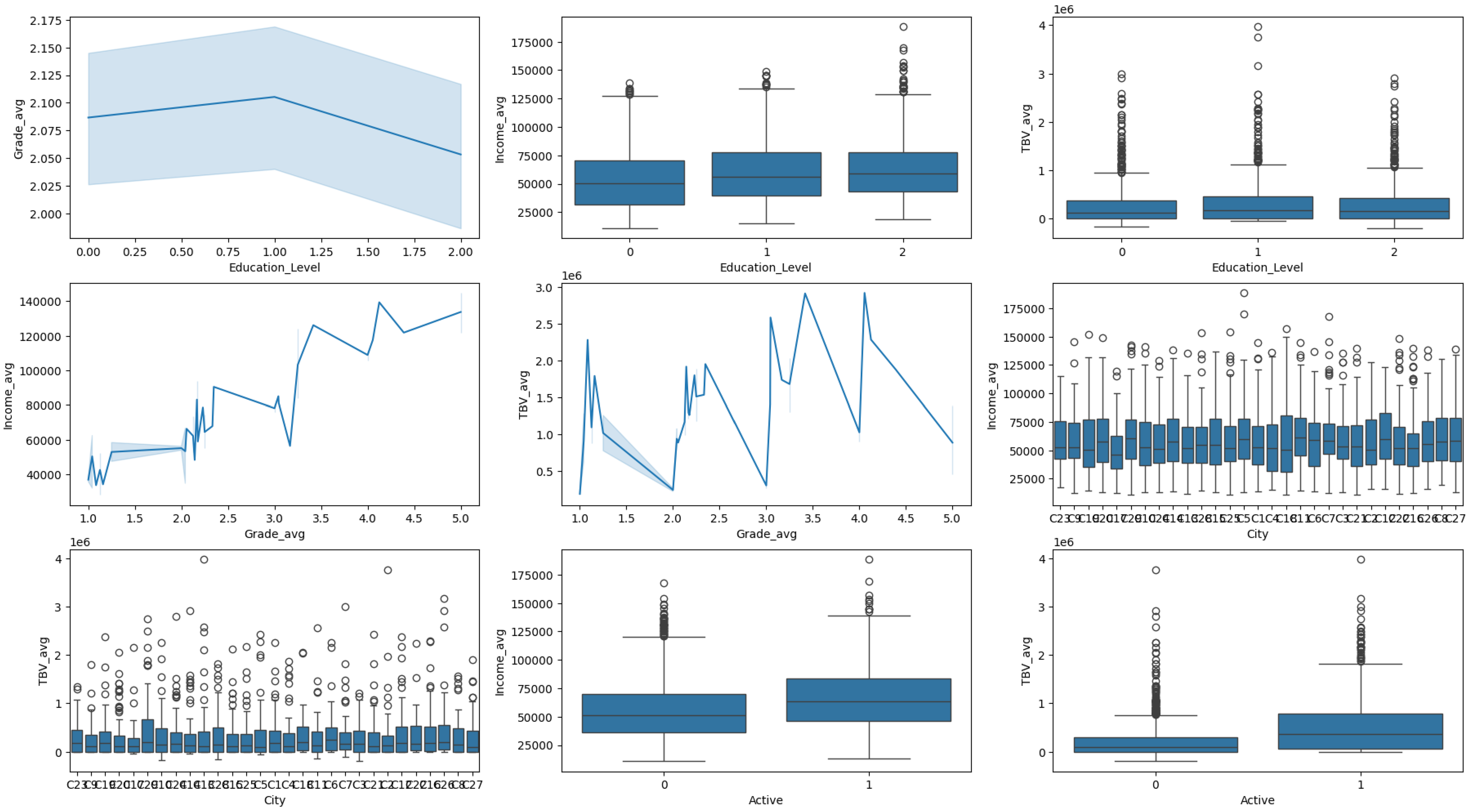
```
Out[ ]: <Axes: xlabel='Active', ylabel='Count'>
```



Understanding:

- Data seems unbalanced for both Active and Non-Active Drivers, but I think there is no need for Scaling.
- Data seems balances for all the cities we have.

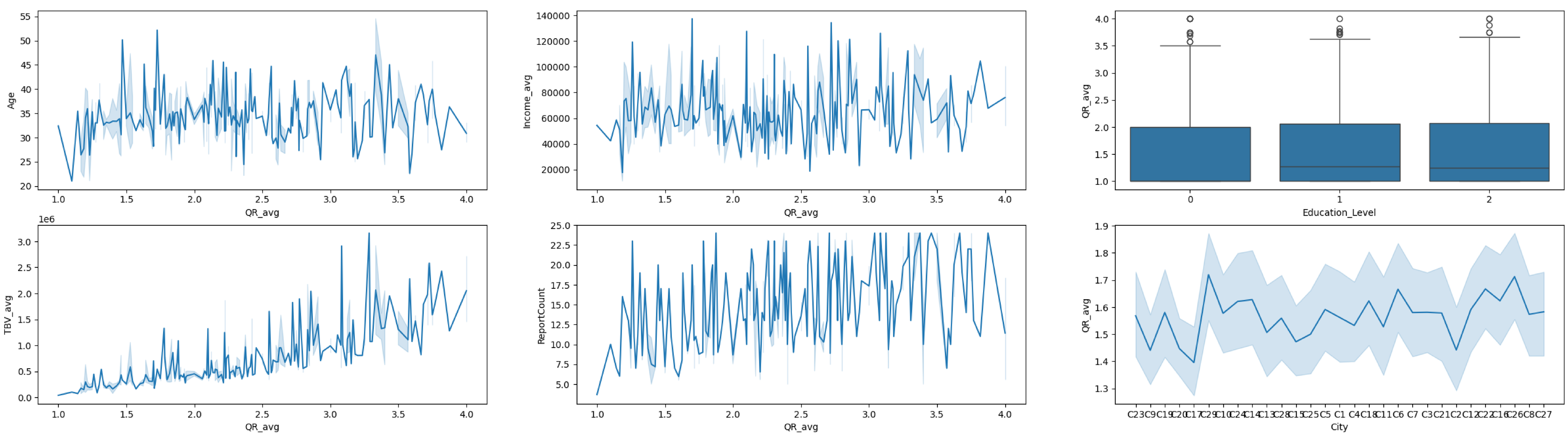
```
In [ ]: fig=plt.figure(figsize=(22,12))

        plt.subplot(3,3,1)
        sns.lineplot(y='Grade_avg', x='Education_Level', data=df)
        plt.subplot(3,3,2)
        sns.boxplot(y='Income_avg', x='Education_Level', data=df)
        plt.subplot(3,3,3)
        sns.boxplot(y='TBV_avg', x='Education_Level', data=df)

        plt.subplot(3,3,4)
        sns.lineplot(y='Income_avg', x='Grade_avg', data=df)
        plt.subplot(3,3,5)
        sns.lineplot(y='TBV_avg', x='Grade_avg', data=df)

        plt.subplot(3,3,6)
        sns.boxplot(y='Income_avg', x='City', data=df)

        plt.subplot(3,3,7)
        sns.boxplot(y='TBV_avg', x='City', data=df)

        plt.subplot(3,3,8)
        sns.boxplot(y='Income_avg', x='Active', data=df)

        plt.subplot(3,3,9)
        sns.boxplot(y='TBV_avg', x='Active', data=df)

        plt.show()
```

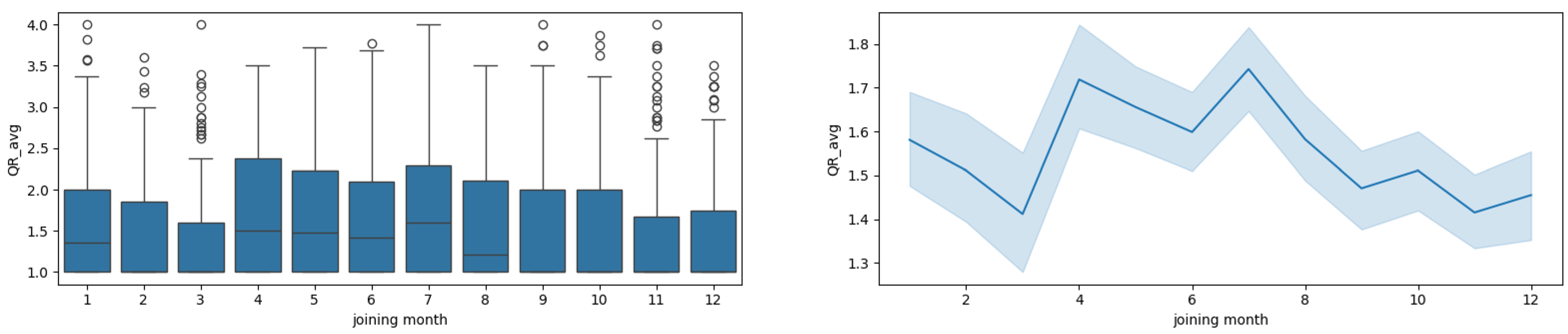


- Correlation between driver ratings and factors like age, income, and education.

In [ ]:
```python
fig=plt.figure(figsize=(30,12))

plt.subplot(3,3,1)
sns.lineplot(y='Age', x='QR_avg', data=df)

plt.subplot(3,3,2)
sns.lineplot(y='Income_avg', x='QR_avg', data=df)

plt.subplot(3,3,3)
sns.boxplot(y='QR_avg', x='Education_Level', data=df)

plt.subplot(3,3,4)
sns.lineplot(y='TBV_avg', x='QR_avg', data=df)

plt.subplot(3,3,5)
sns.lineplot(y='ReportCount', x='QR_avg', data=df)

plt.subplot(3,3,6)
sns.lineplot(y='QR_avg', x='City', data=df)

fig=plt.figure(figsize=(30,12))

plt.subplot(3,3,7)
sns.boxplot(y='QR_avg', x='joining month', data=df)

plt.subplot(3,3,8)
sns.lineplot(y='QR_avg', x='joining month', data=df)
```

Out[ ]:
```
<Axes: xlabel='joining month', ylabel='QR_avg'>
```





In [ ]:
```python
df.groupby("QR_avg")["TBV_avg"].mean()
```

Out[ ]:
```
QR_avg
1.000000    3.802113e+04
1.100000    1.000130e+05
1.142857    7.172000e+04
1.166667    1.754842e+05
1.187500    1.477394e+05
                ...
3.750000    1.588392e+06
3.769231    1.798186e+06
3.818182    2.426973e+06
3.875000    1.277854e+06
4.000000    2.051391e+06
Name: TBV_avg, Length: 163, dtype: float64
```

In [ ]:
```python
print("Grade <-> Income")
# Calculate Pearson correlation
pearson_corr = df['Grade_avg'].corr(df['Income_avg'])
print(f"Pearson Correlation: {pearson_corr}")
# Calculate Spearman correlation
spearman_corr = df['Grade_avg'].corr(df['Income_avg'], method='spearman')
print(f"Spearman Correlation: {spearman_corr}")

print("\nGrade <-> Total Business Value")
# Calculate Pearson correlation
pearson_corr = df['Grade_avg'].corr(df['TBV_avg'])
print(f"Pearson Correlation: {pearson_corr}")
# Calculate Spearman correlation
spearman_corr = df['Grade_avg'].corr(df['TBV_avg'], method='spearman')
print(f"Spearman Correlation: {spearman_corr}")
```

```
Grade <-> Income
Pearson Correlation: 0.7396736903173439
Spearman Correlation: 0.7102193516509495

Grade <-> Total Business Value
Pearson Correlation: 0.33291162712609945
Spearman Correlation: 0.19359942881616332
```

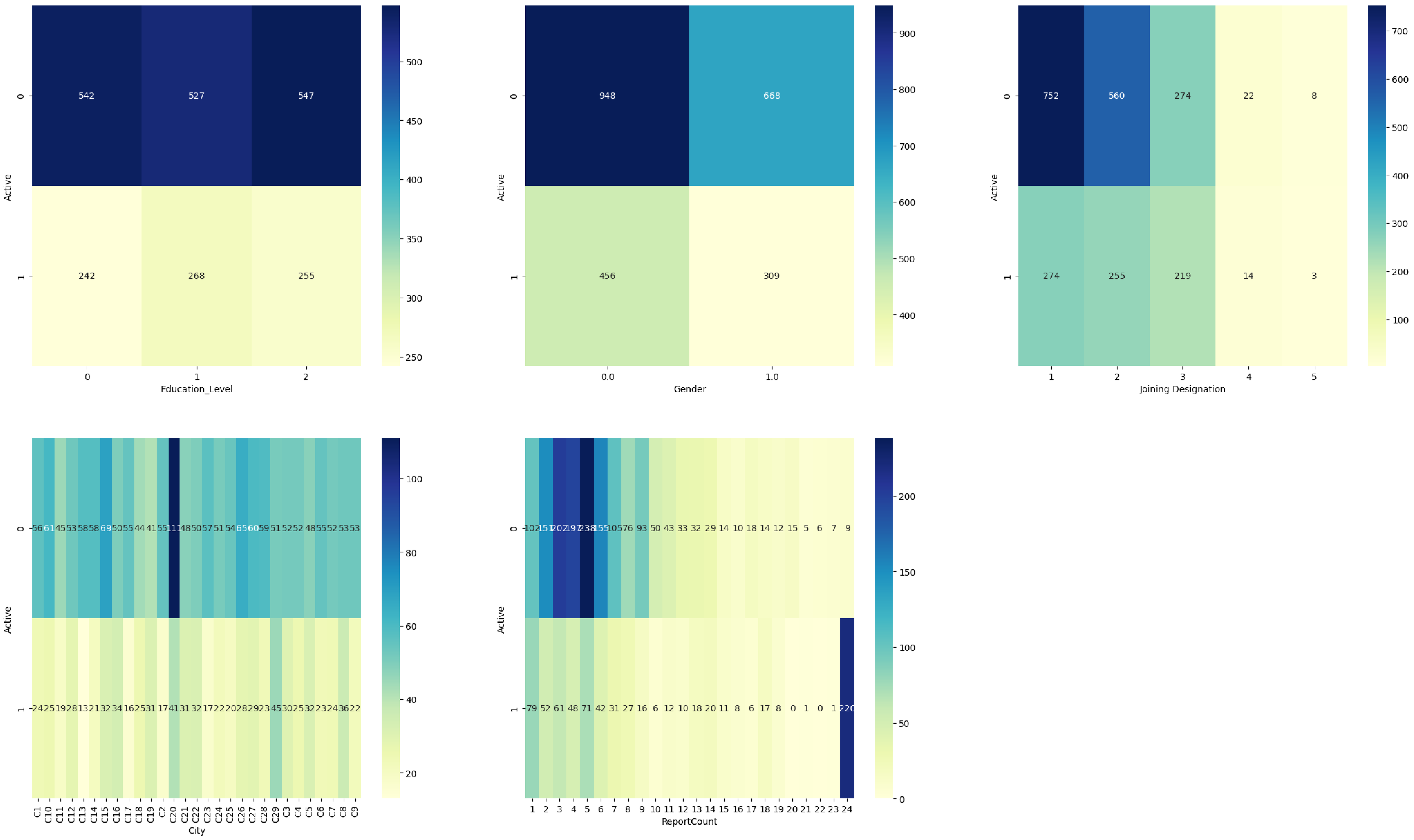
Understandings:

- Whenever Driver's Grade changes, total Business Values has imapct. Which shows Grading has positive correlation with Business values the Driver is providing.
- With change in Grade, you income increases.

In [ ]:
```python
print("\Age <-> Quarterly Rating")
# Calculate Pearson correlation
pearson_corr = df['Age'].corr(df['QR_avg'])
print(f"Pearson Correlation: {pearson_corr}")
# Calculate Spearman correlation
spearman_corr = df['Age'].corr(df['QR_avg'], method='spearman')
print(f"Spearman Correlation: {spearman_corr}")
```

```
\Age <-> Quarterly Rating
Pearson Correlation: 0.19510169050688675
Spearman Correlation: 0.17632230499418575
```

In [ ]:
```python
# 3. CATEGORICAL VS CATEGORICAL
categorical_col = ["Education_Level", "Gender", "Joining Designation", "City","ReportCount"]
fig=plt.figure(figsize=(28,16))
for ind_val, columns in enumerate(categorical_col):
 if ind_val<6:
    plt.subplot(2,3,ind_val+1)
    cross_tab = pd.crosstab(df["Active"], df[columns])
    sns.heatmap(cross_tab,annot=True, cmap="YlGnBu", fmt='d', cbar=True)
    # Set labels and title
    plt.xlabel(columns)
    plt.ylabel('Active')
    # Show the plot
    # plt.show()
```



Trade-Off Analysis:

- From the heatmaps plotted above, it doesn't seem to have any correlation between the Education and Driver's Attrition.
- From the heatmaps it clearly indicated that if we able to hold our Driver for long, the Churn is very less.

Recommendations based on understanding

Categorical Understandings:

- Their seems to be big Churn in City C20 as compare to other cities present, we should work and understand the reason and keep the business inpact from upcoming challenges.
- Designation 1 and 5 have high changes of Leaving as compare to other Designations.
- Gender and Education seems unbiased.

Visualization understanding

- No impact of cities on Driver's Income and Business Value.
- Active Drivers seems to have better Income and contributing more as compare to Non-Active Drivers.
- With increase in Grade there is increase in Income.
- No relationship between Driver's Education and Income/Business Value.

Outliers and Influence :

- It seems that there are outliers in Drivers Income and Business Values, but after understanding business perspective we should not remove them.
- They can act as valuable information when building the prediction models.

Skeness and Correlation

- It seems that both Income and Business Value are left skewed.
- Grade and Income are positively correlated, similarly Grade and Average_Business_Value is positively correlated

In [ ]:
```python
df.shape
```

Out[ ]: (2381, 18)

In [ ]:
```python
df.describe()
```

Out[ ]:

	Age	Gender	Education_Level	Joining Designation	joining date	joining month	joining year	ReportCount	Active	TBV_avg	QR_avg	Income_avg	Grade_avg	income_diff_pattern	tbv_diff_pattern	qr_diff_pattern	grade_diff_pattern
count	2381.000000	2381.000000	2381.00000	2381.000000	2381.000000	2381.000000	2381.000000	2381.00000	2381.000000	2.381000e+03	2381.000000	2381.000000	2381.000000	2381.000000	2381.000000	2381.000000	2381.000000
mean	33.369192	0.410332	1.00756	1.820244	16.186896	6.958001	18.536329	8.02352	0.321294	3.120854e+05	1.566304	59232.460484	2.081713	1.040319	-0.559849	-0.058379	0.018060
std	5.890732	0.491997	0.81629	0.841433	8.959616	3.221762	1.609597	6.78359	0.467071	4.495705e+05	0.719652	28298.214012	0.932257	0.314983	1.445759	0.933703	0.133195
min	21.000000	0.000000	0.00000	1.000000	1.000000	1.000000	13.000000	1.00000	0.000000	-1.979329e+05	1.000000	10747.000000	1.000000	1.000000	-3.000000	-3.000000	0.000000
25%	29.000000	0.000000	0.00000	1.000000	9.000000	5.000000	18.000000	3.00000	0.000000	0.000000e+00	1.000000	39104.000000	1.000000	1.000000	-1.000000	0.000000	0.000000
50%	33.000000	0.000000	1.00000	2.000000	15.000000	7.000000	19.000000	5.00000	0.000000	1.506244e+05	1.000000	55285.000000	2.000000	1.000000	-1.000000	0.000000	0.000000
75%	37.000000	1.000000	2.00000	2.000000	24.000000	10.000000	20.000000	10.00000	1.000000	4.294988e+05	2.000000	75835.000000	3.000000	1.000000	-1.000000	0.000000	0.000000
max	58.000000	1.000000	2.00000	5.000000	31.000000	12.000000	20.000000	24.00000	1.000000	3.972128e+06	4.000000	188418.000000	5.000000	5.000000	3.000000	3.000000	1.000000

In [ ]:
```python
df.describe(include="object")
```

Out[ ]:

	City
count	2381
unique	29
top	C20
freq	152

Data Modelling

In [ ]:
```python
# Standard Scaler
from sklearn.preprocessing import MinMaxScaler

# numeric_columns = ["TBV_avg", "Income_avg"]
numeric_columns = df.describe().columns
scaler = MinMaxScaler()
scaler.fit(df[numeric_columns])
df[numeric_columns] = scaler.transform(df[numeric_columns])
```

In [ ]:
```python
df.head()
```

Out[ ]:

	Age	Gender	City	Education_Level	Joining Designation	joining date	joining month	joining year	ReportCount	Active	TBV_avg	QR_avg	Income_avg	Grade_avg	income_diff_pattern	tbv_diff_pattern	qr_diff_pattern	grade_diff_pattern
0	0.189189	0.0	C23	1.0	0.00	0.766667	1.000000	0.714286	0.086957	0.0	0.184600	0.333333	0.262508	0.00	0.0	0.000000	0.500000	0.0
1	0.216216	0.0	C9	0.0	0.00	0.000000	0.727273	0.857143	0.086957	0.0	0.057086	0.000000	0.200489	0.00	0.0	0.333333	0.500000	0.0
2	0.232667	0.0	C19	1.0	0.00	0.900000	0.363636	0.285714	0.956522	0.0	0.153949	0.086957	0.610567	0.75	0.0	0.333333	0.500000	0.0
3	0.243243	1.0	C23	0.0	0.25	0.966667	0.909091	0.714286	0.043478	0.0	0.089047	0.000000	0.237608	0.25	0.0	0.333333	0.500000	0.0
4	0.571429	0.0	C20	1.0	0.00	0.066667	0.454545	0.714286	0.260870	0.0	0.082327	0.142857	0.227116	0.00	0.0	0.333333	0.333333	0.0

- Since we will be using Tree based approach I will be doing OHE on Cities

0: Non-Active users 1: Active users

In [ ]:
```python
df["Active"].value_counts()
```

Out[ ]:
```
Active
0.0    1616
1.0     765
Name: count, dtype: int64
```

In [ ]:
```python
df.shape
```

Out[ ]:
```
(2381, 18)
```

In [ ]:
```python
df = pd.get_dummies(df,columns=["City"], dtype=int)
df.shape
```

Out[ ]:
```
(2381, 46)
```

In [ ]:
```python
df.head()
```

Out[ ]:

	Age	Gender	Education_Level	Joining Designation	joining date	joining month	joining year	ReportCount	Active	TBV_avg	QR_avg	Income_avg	Grade_avg	income_diff_pattern	tbv_diff_pattern	qr_diff_pattern	grade_diff_pattern	City_C1	City_C10	City_C11	City_C12	City_C13	City_C14	City_C15	City_C16
0	0.189189	0.0	1.0	0.00	0.766667	1.000000	0.714286	0.086957	0.0	0.184600	0.333333	0.262508	0.00	0.0	0.000000	0.500000	0.0	0	0	0	0	0	0	0	0
1	0.216216	0.0	0.0	0.00	0.000000	0.727273	0.857143	0.086957	0.0	0.057086	0.000000	0.200489	0.00	0.0	0.333333	0.500000	0.0	0	0	0	0	0	0	0	0
2	0.232667	0.0	1.0	0.00	0.900000	0.363636	0.285714	0.956522	0.0	0.153949	0.086957	0.610567	0.75	0.0	0.333333	0.500000	0.0	0	0	0	0	0	0	0	0
3	0.243243	1.0	0.0	0.25	0.966667	0.909091	0.714286	0.043478	0.0	0.089047	0.000000	0.237608	0.25	0.0	0.333333	0.500000	0.0	0	0	0	0	0	0	0	0
4	0.571429	0.0	1.0	0.00	0.066667	0.454545	0.714286	0.260870	0.0	0.082327	0.142857	0.227116	0.00	0.0	0.333333	0.333333	0.0	0	0	0	0	0	0	0	0

In [ ]:
```python
df.to_csv("dataset/phase2_df.csv", index=False)
```

In [ ]: