In [30]:
```
#required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import KNNImputer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, BaggingClassifier
from sklearn.metrics import classification_report, roc_auc_score, roc_curve, confusion_matrix
from imblearn.over_sampling import SMOTE
```

In [31]:
```
#loading the dataset
df = pd.read_csv('ola_driver_scaler.csv')
```

In [32]:
```
#shape of the data set
df.shape
```

Out[32]:
```
(19104, 14)
```

In [33]:
```
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")
```

Out[33]:

	Unnamed: 0	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Bu
0	0	01/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	23
1	1	02/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	-6
2	2	03/01/19	1	28.0	0.0	C23	2	57387	24/12/18	03/11/19	1	1	
3	3	11/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	
4	4	12/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	

In [34]:
```
df.drop('Unnamed: 0', axis=1, inplace=True)
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")
```

Out[34]:

	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Total Business Value	Qua R
0	01/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	2381060	
1	02/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	-665480	
2	03/01/19	1	28.0	0.0	C23	2	57387	24/12/18	03/11/19	1	1	0	
3	11/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	
4	12/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	

In [35]:
```
#data types of the column
df.info()
```

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

In [36]:
```
# statistical summary
df.describe()
```

Out[36]:

	Driver_ID	Age	Gender	Education_Level	Income	Joining Designation	Grade	Total Business Value	Quarterly Rating
count	19104.000000	19043.000000	19052.000000	19104.000000	19104.000000	19104.000000	19104.000000	1.910400e+04	19104.000000
mean	1415.591133	34.668435	0.418749	1.021671	65652.025126	1.690536	2.252670	5.716621e+05	2.008899
std	810.705321	6.257912	0.493367	0.800167	30914.515344	0.836984	1.026512	1.128312e+06	1.009832
min	1.000000	21.000000	0.000000	0.000000	10747.000000	1.000000	1.000000	-6.000000e+06	1.000000
25%	710.000000	30.000000	0.000000	0.000000	42383.000000	1.000000	1.000000	0.000000e+00	1.000000
50%	1417.000000	34.000000	0.000000	1.000000	60087.000000	1.000000	2.000000	2.500000e+05	2.000000
75%	2137.000000	39.000000	1.000000	2.000000	83969.000000	2.000000	3.000000	6.997000e+05	3.000000
max	2788.000000	58.000000	1.000000	2.000000	188418.000000	5.000000	5.000000	3.374772e+07	4.000000

In [37]:
```
#Checking for any missing values
df.isna().sum()
```

Out[37]:

	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

In [38]:
```
#duplciate records
duplicated_count = df.duplicated().sum()
print(duplicated_count)
```

```
0
```

In [39]:
```
# Convert date columns to datetime objects
df['MMM-YY'] = pd.to_datetime(df['MMM-YY'])
df['Dateofjoining'] = pd.to_datetime(df['Dateofjoining'])
df['LastWorkingDate'] = pd.to_datetime(df['LastWorkingDate'])
df.head()
```

```
<ipython-input-39-b7c08f3b1e50>:2: UserWarning: Could not infer format, so each element will be parsed individually, falling ba
ck to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df['MMM-YY'] = pd.to_datetime(df['MMM-YY'])
<ipython-input-39-b7c08f3b1e50>:3: UserWarning: Could not infer format, so each element will be parsed individually, falling ba
ck to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df['Dateofjoining'] = pd.to_datetime(df['Dateofjoining'])
<ipython-input-39-b7c08f3b1e50>:4: UserWarning: Could not infer format, so each element will be parsed individually, falling ba
ck to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df['LastWorkingDate'] = pd.to_datetime(df['LastWorkingDate'])
```

Out[39]:

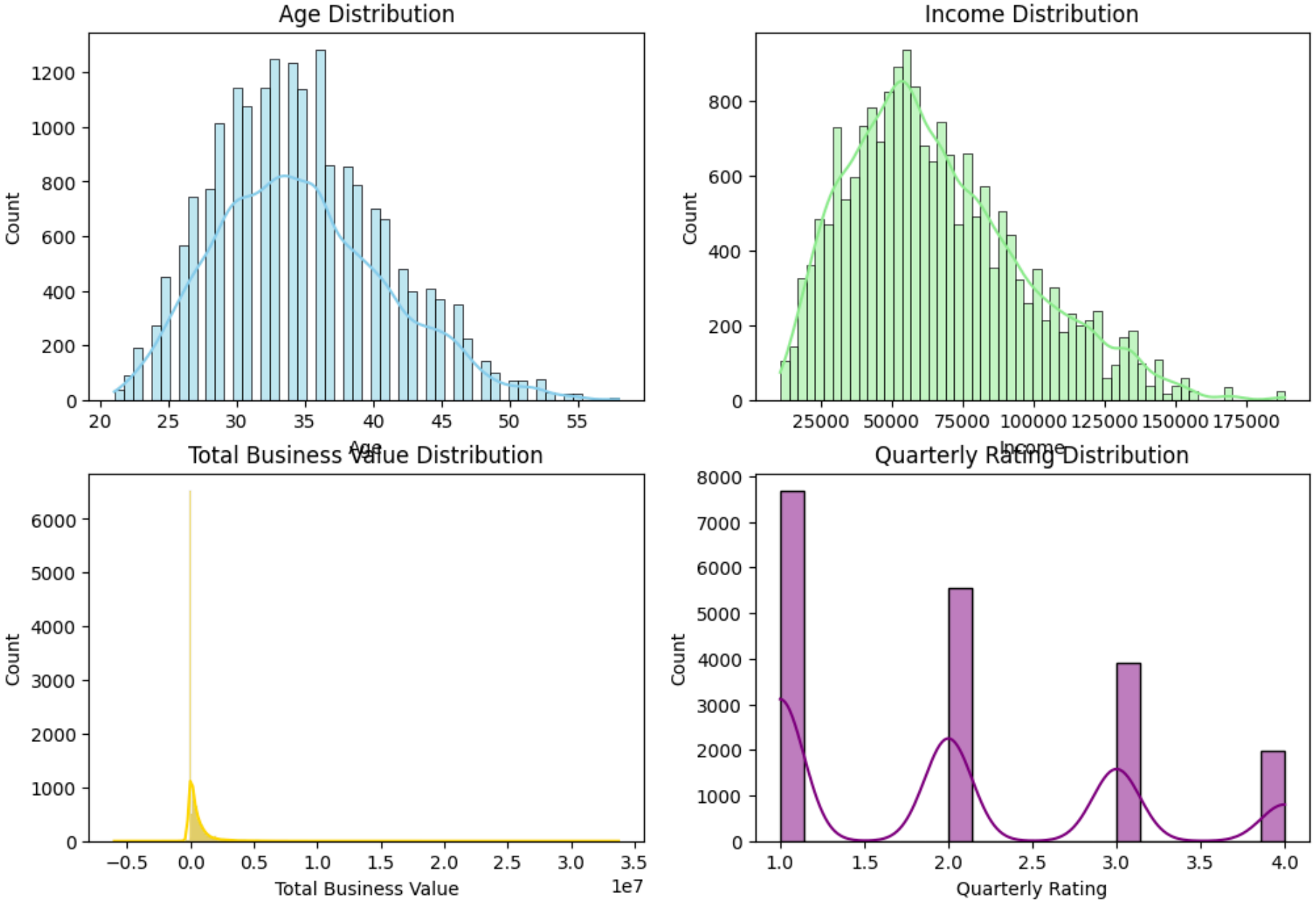
	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Total Business Value	Quart Rat
0	2019-01-01	1	28.0	0.0	C23	2	57387	2018-12-24	NaT	1	1	2381060	
1	2019-02-01	1	28.0	0.0	C23	2	57387	2018-12-24	NaT	1	1	-665480	
2	2019-03-01	1	28.0	0.0	C23	2	57387	2018-12-24	2019-03-11	1	1	0	
3	2020-11-01	2	31.0	0.0	C7	2	67016	2020-11-06	NaT	2	2	0	
4	2020-12-01	2	31.0	0.0	C7	2	67016	2020-11-06	NaT	2	2	0	

In [40]:

```
# Univariate Analysis Numerical Analysis
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))

sns.histplot(df['Age'],kde=True, ax = axes[0,0], color='skyblue')
axes[0,0].set_title('Age Distribution')

sns.histplot(df['Income'],kde=True, ax = axes[0,1], color='lightgreen')
axes[0,1].set_title('Income Distribution')

sns.histplot(df['Total Business Value'],kde=True, ax = axes[1,0], color='gold')
```

```
axes[1,0].set_title('Total Business Value Distribution')

sns.histplot(df['Quarterly Rating'],kde=True, ax = axes[1,1], color='purple')
axes[1,1].set_title('Quarterly Rating Distribution')
```

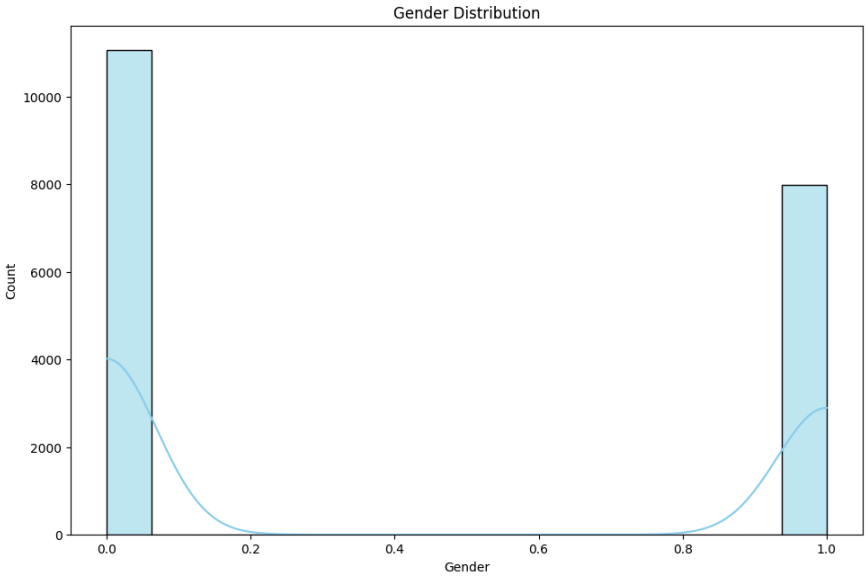
Out[40]: Text(0.5, 1.0, 'Quarterly Rating Distribution')

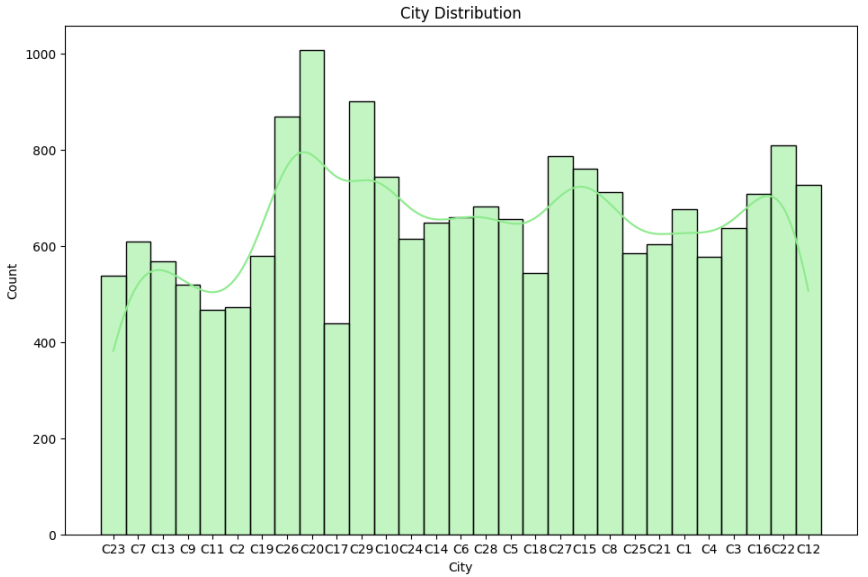


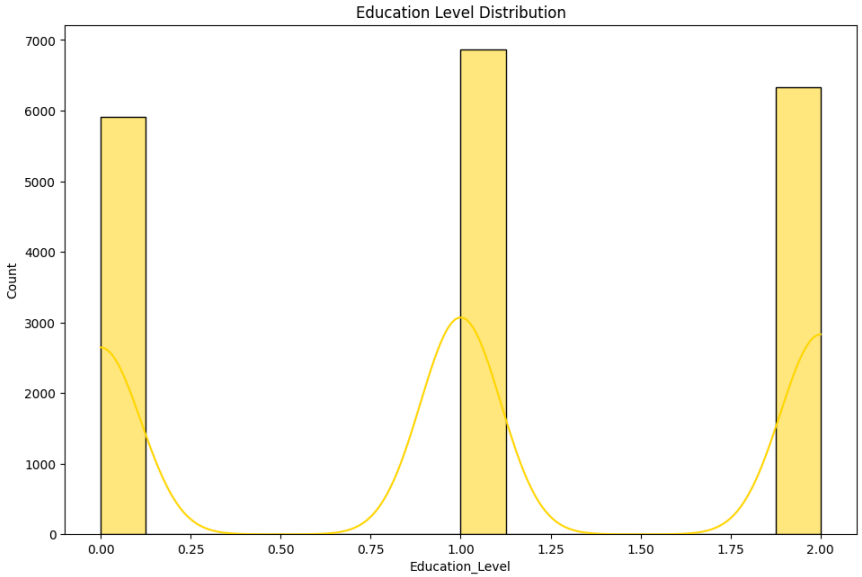
In [41]:
```
#unvariate analysis of categorical_cols
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(25, 25))
```

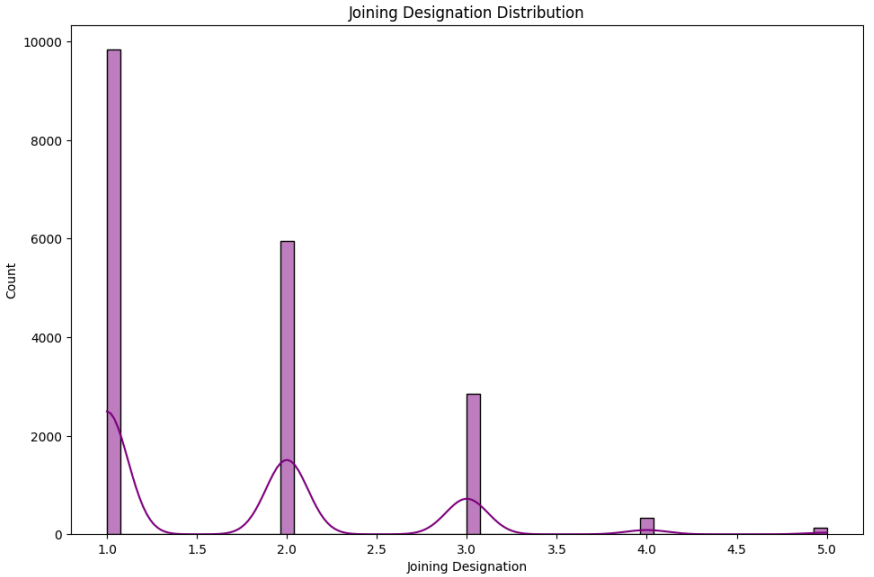
```
sns.histplot(df['Gender'],kde=True, ax = axes[0,0], color='skyblue')
axes[0,0].set_title('Gender Distribution')

sns.histplot(df['City'],kde=True, ax = axes[0,1], color='lightgreen')
axes[0,1].set_title('City Distribution')

sns.histplot(df['Education_Level'],kde=True, ax = axes[1,0], color='gold')
axes[1,0].set_title('Education Level Distribution')

sns.histplot(df['Joining Designation'],kde=True, ax = axes[1,1], color='purple')
axes[1,1].set_title('Joining Designation Distribution')

sns.histplot(df['Grade'],kde=True, ax = axes[2,0], color='teal')
axes[2,0].set_title('Grade Distribution')
```

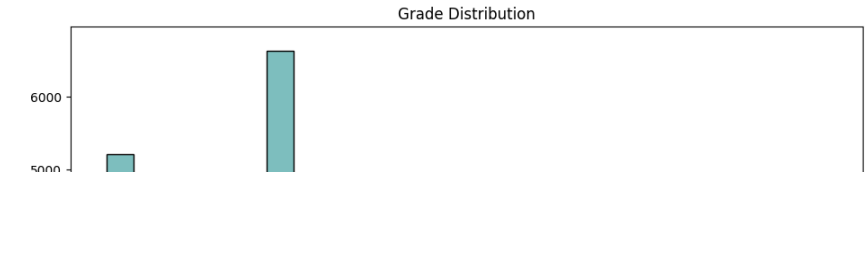
Out[41]: Text(0.5, 1.0, 'Grade Distribution')


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

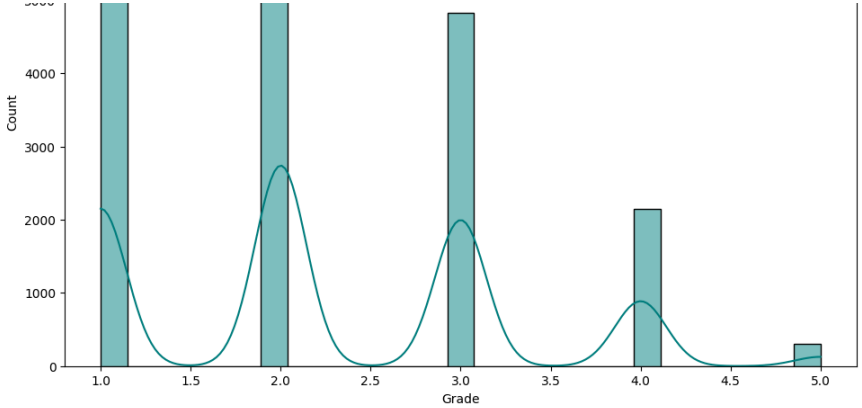

City Distribution
Count
City
0
200
400
600
800
1000
C23 C7 C13 C9 C11 C2 C19 C26 C20 C17 C29 C10 C24 C14 C6 C28 C5 C18 C27 C15 C8 C25 C21 C1 C4 C3 C16 C22 C12

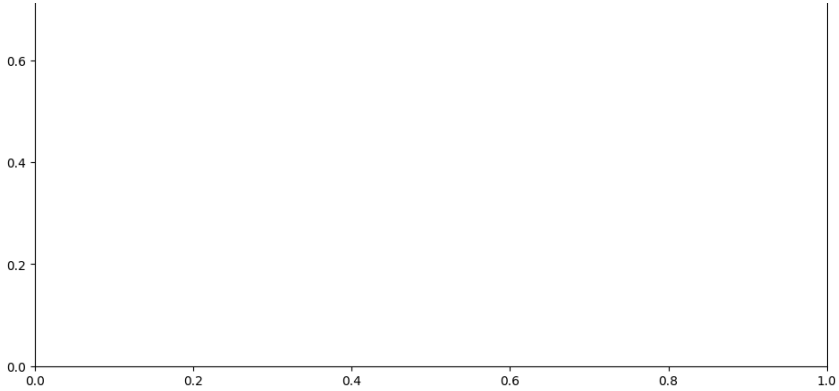

Education Level Distribution
Count
Education_Level
0
1000
2000
3000
4000
5000
6000
7000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Joining Designation Distribution
Count
Joining Designation
0
2000
4000
6000
8000
10000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Grade Distribution
6000
5000

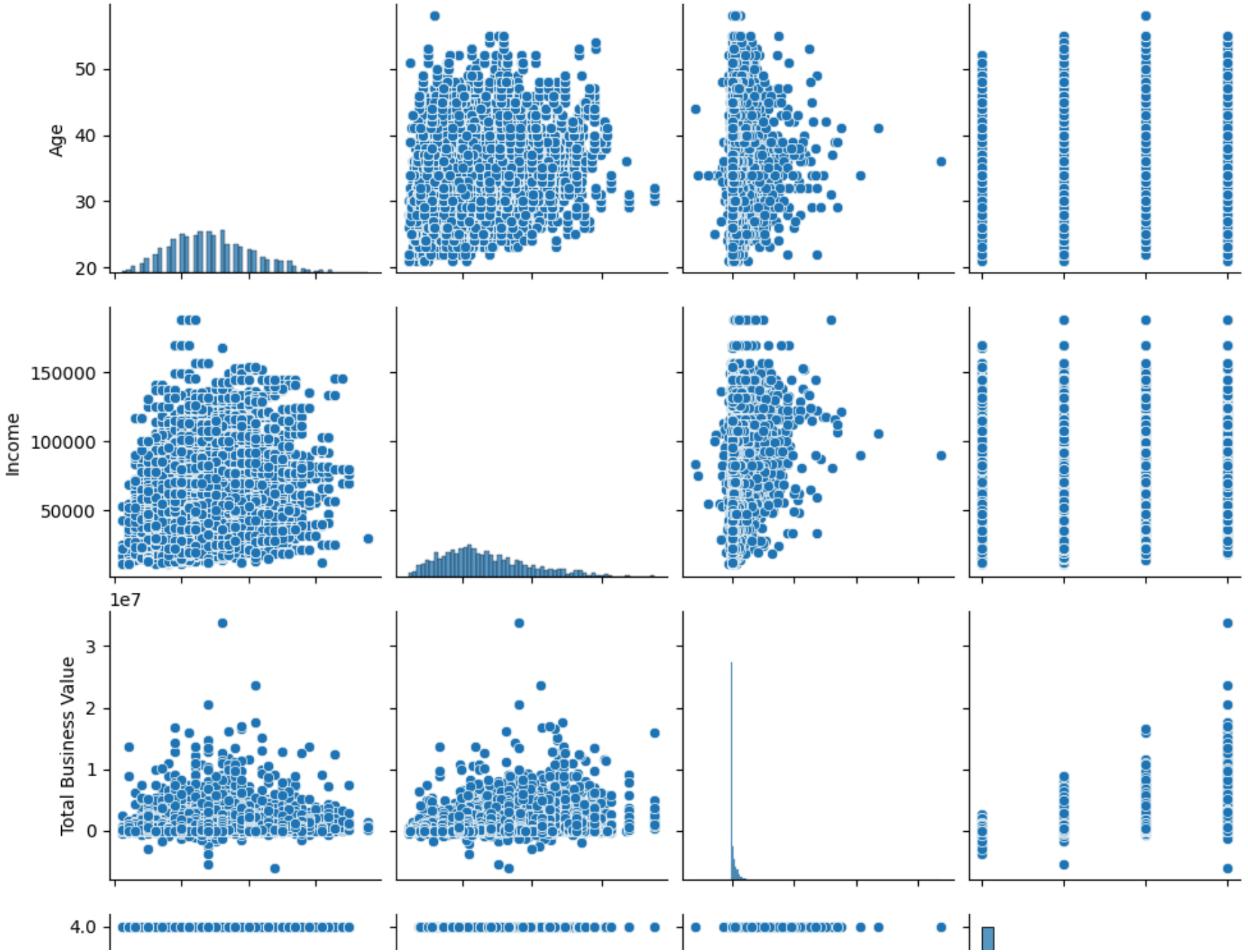

1.0
0.8

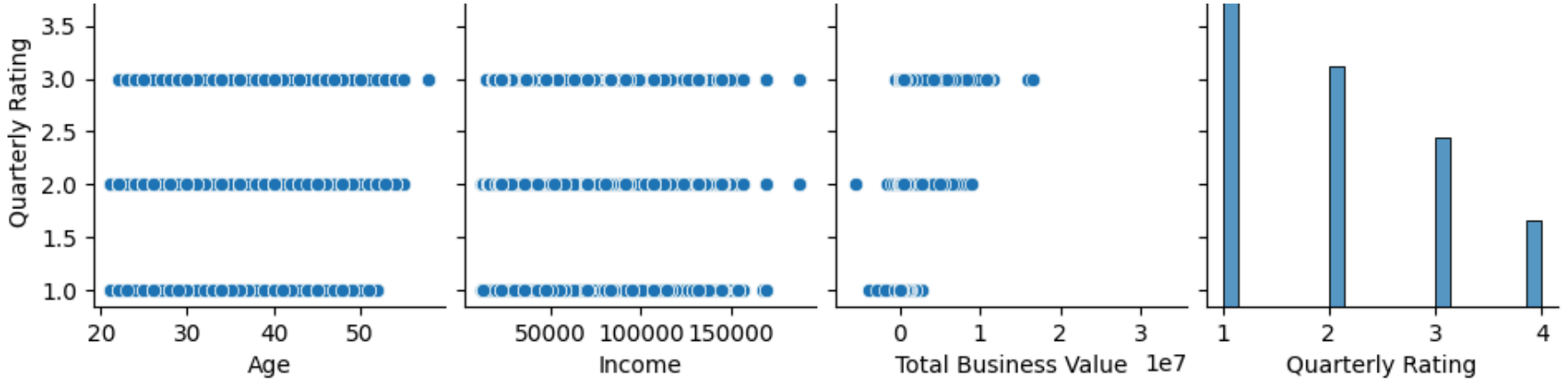


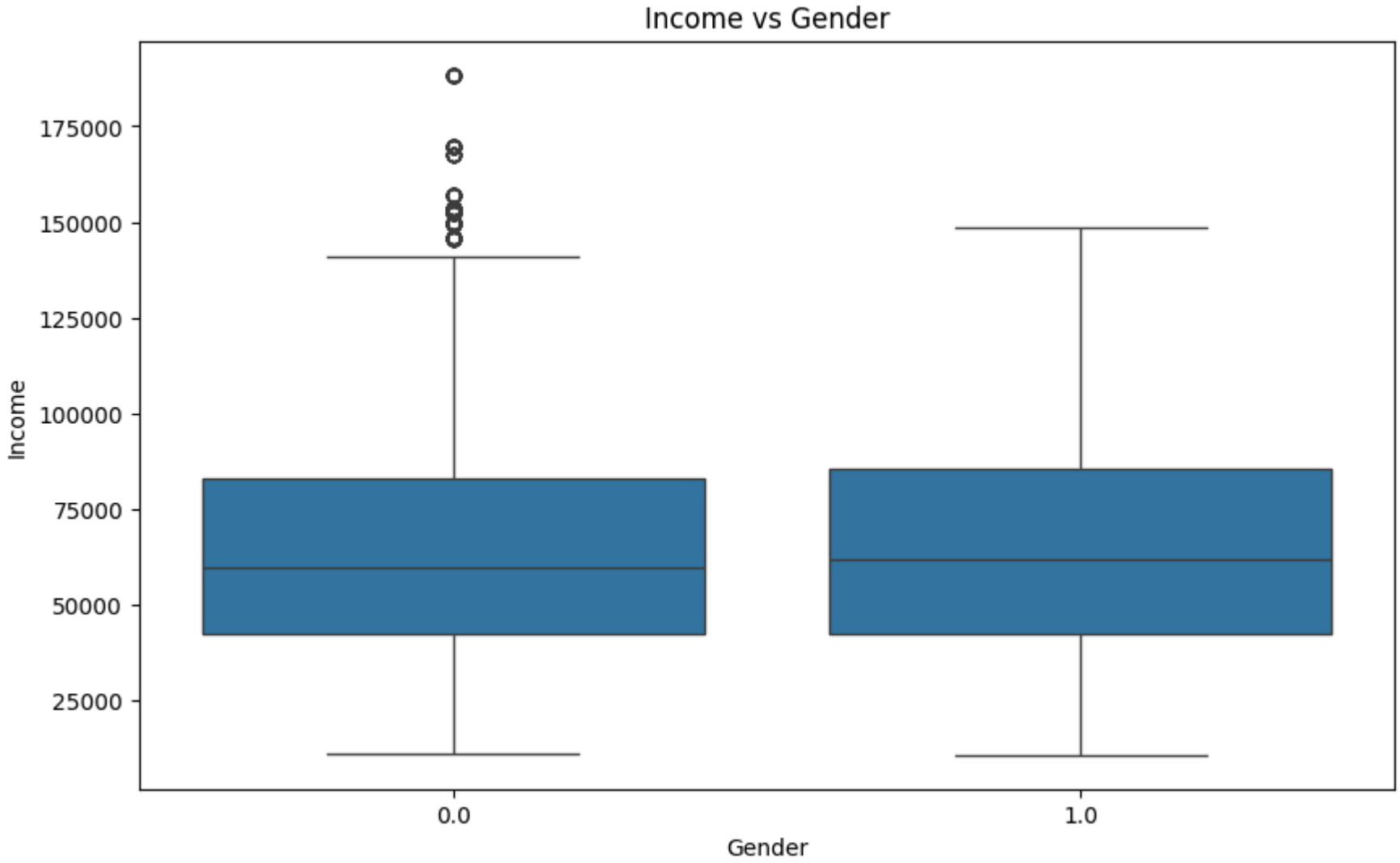


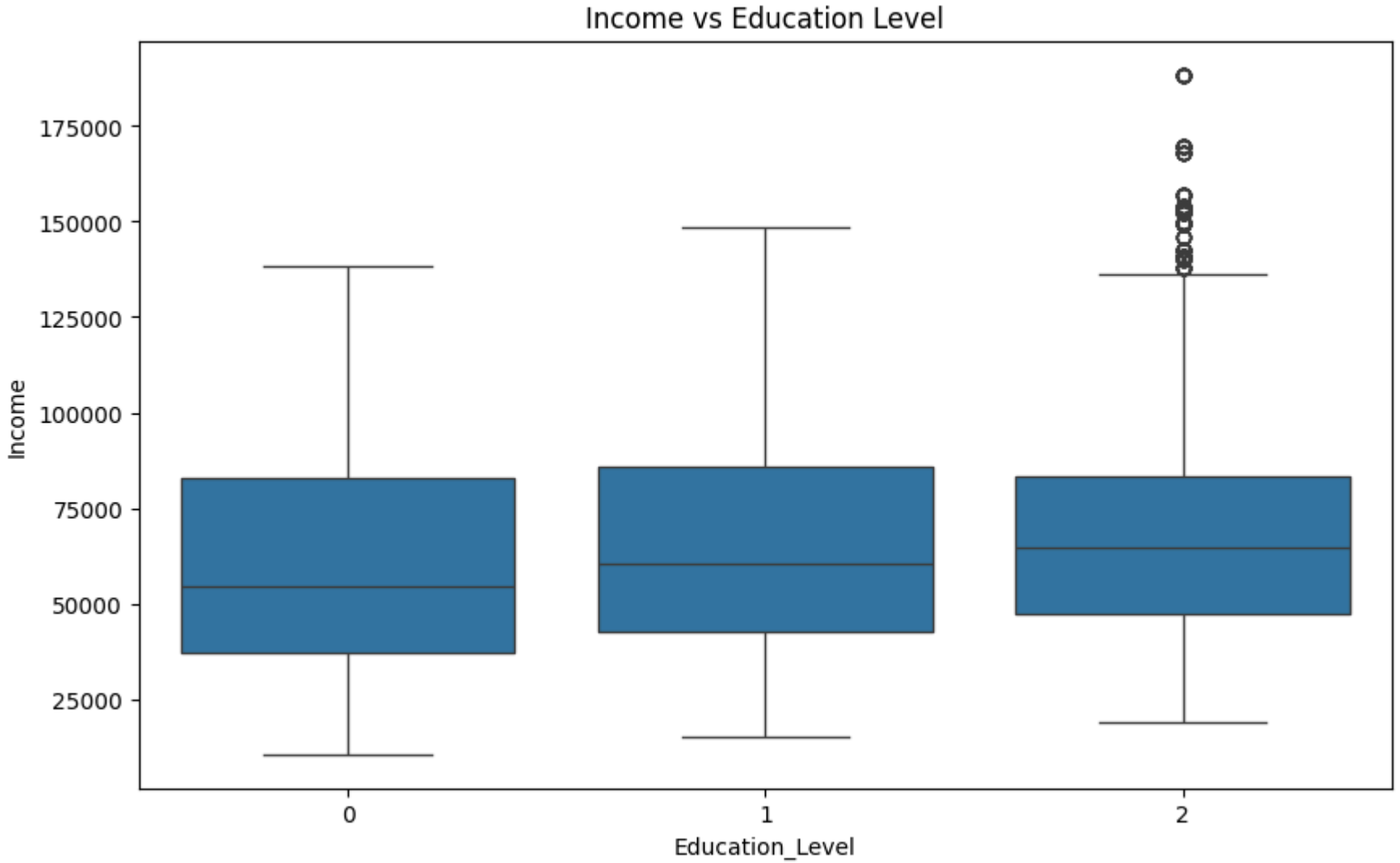
In [42]:

```
# Bivariate Analysis
numerical_cols = ['Age', 'Income', 'Total Business Value', 'Quarterly Rating']

sns.pairplot(df[numerical_cols])
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='Income', data=df)
plt.title('Income vs Gender')
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(x='Education_Level', y='Income', data=df)
plt.title('Income vs Education Level')
plt.show()
```


Age
50
40
30
20
Income
150000
100000
50000
1e7
Total Business Value
3
2
1
0
4.0


Quarterly Rating
3.5
3.0
2.5
2.0
1.5
1.0
20
30
40
50
Age
50000
100000
150000
Income
0
1
2
3
1e7
Total Business Value
1
2
3
4
Quarterly Rating


Income vs Gender
Income
175000
150000
125000
100000
75000
50000
25000
0.0
1.0
Gender



In [43]:
```
# 2. Data Preprocessing
# KNN Imputation
numerical_for_impute = df[numerical_cols]
imputer = KNNImputer(n_neighbors=5)
df[numerical_cols] = imputer.fit_transform(numerical_for_impute)
```

In [44]:
```
# Feature Engineering
driver_data = pd.DataFrame(df['Driver_ID'].unique(), columns=['Driver_ID'])

def aggregate_driver_data(group):
    return pd.Series({
        'Age': group['Age'].mean(),
        'Gender': group['Gender'].mode().iloc[0],
        'City': group['City'].mode().iloc[0],
        'Education_Level': group['Education_Level'].mode().iloc[0],
        'Income': group['Income'].mean(),
        'Joining_Designation': group['Joining Designation'].mode().iloc[0],
        'Grade': group['Grade'].mode().iloc[0],
        'Total_Business_Value': group['Total Business Value'].mean(),
        'Quarterly_Rating': group['Quarterly Rating'].mean(),
        'Joining_Date': group['Dateofjoining'].min(),
        'Last_Working_Date': group['LastWorkingDate'].max()
    })

driver_agg = df.groupby('Driver_ID').apply(aggregate_driver_data).reset_index()
driver_data = driver_data.merge(driver_agg, on='Driver_ID', how='left')

driver_data['Target'] = driver_data['Last_Working_Date'].notna().astype(int)

driver_data['Income_Increase'] = driver_data.groupby('Driver_ID')['Income'].pct_change().fillna(0).apply(lambda x: 1 if x > 0

driver_data['Rating_Increase'] = driver_data.groupby('Driver_ID')['Quarterly_Rating'].diff().fillna(0).apply(lambda x: 1 if x
```

```
<ipython-input-44-22d8fd36e9f5>:19: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
  driver_agg = df.groupby('Driver_ID').apply(aggregate_driver_data).reset_index()
```

In [45]:
```
# One-Hot Encoding
driver_data = pd.get_dummies(driver_data, columns=['Gender', 'City', 'Education_Level', 'Joining_Designation', 'Grade'], drop_
```

In [46]:
```
# Class Imbalance Treatment
X = driver_data.drop(['Driver_ID', 'Last_Working_Date', 'Target', 'Joining_Date'], axis=1)
y = driver_data['Target']

smote = SMOTE(random_state=42)
```

```
X_resampled, y_resampled = smote.fit_resample(X, y)

# Standardization
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_resampled)
```

In [47]:
```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_resampled, test_size=0.2, random_state=42)
```

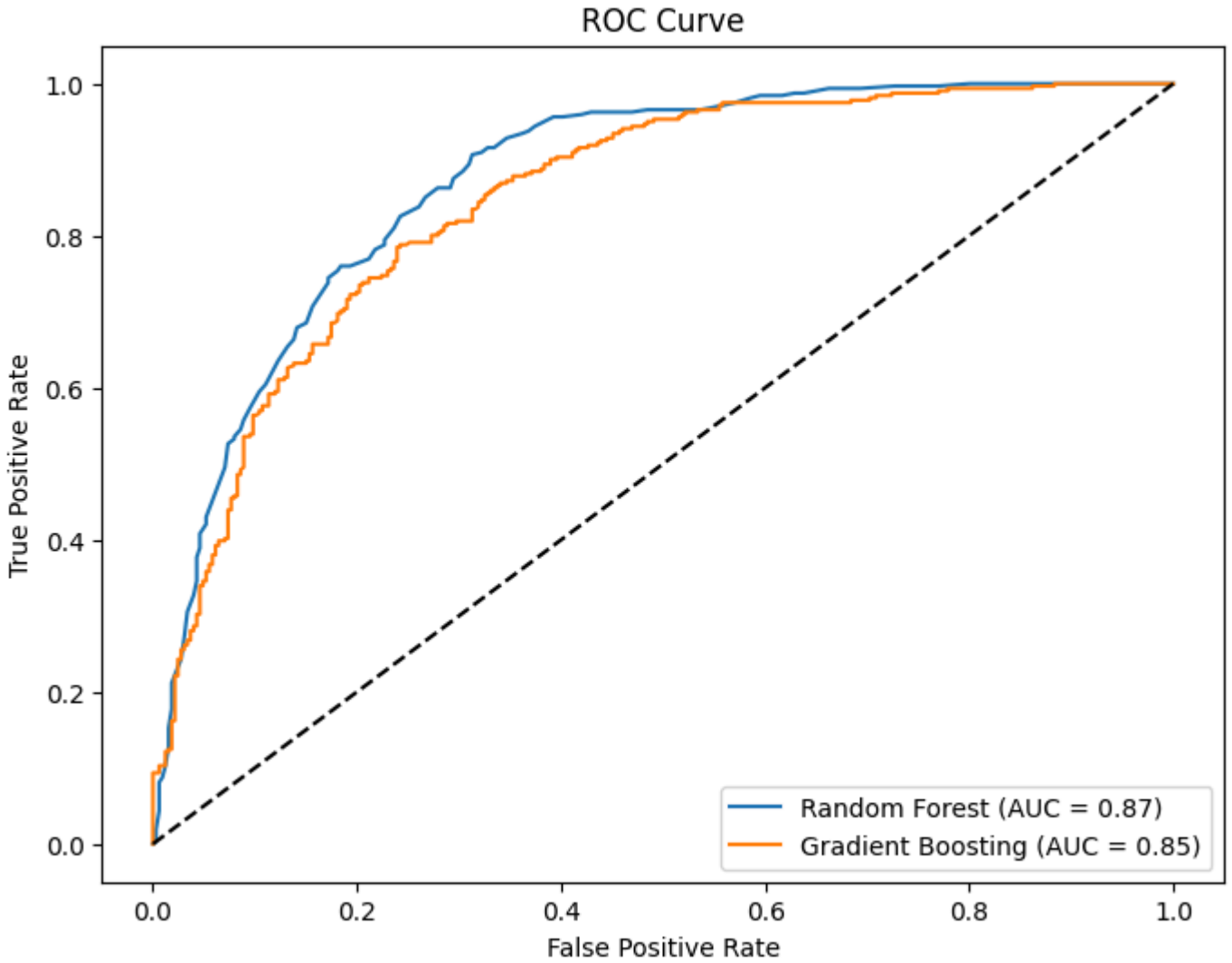
In [48]:
```
# 3. Model Building
# Bagging (Random Forest)
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Boosting (Gradient Boosting)
gb = GradientBoostingClassifier(random_state=42)
gb.fit(X_train, y_train)
y_pred_gb = gb.predict(X_test)
```

In [50]:
```
# 4. Results Evaluation
# Classification Report
print("Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
print("Gradient Boosting Classification Report:\n", classification_report(y_test, y_pred_gb))
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.79      0.78       326
           1       0.78      0.77      0.78       321

    accuracy                           0.78       647
   macro avg       0.78      0.78      0.78       647
weighted avg       0.78      0.78      0.78       647

Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.73      0.75       326
           1       0.74      0.79      0.77       321

    accuracy                           0.76       647
   macro avg       0.76      0.76      0.76       647
weighted avg       0.76      0.76      0.76       647
```

In [51]:

```python
# ROC AUC Curve
y_pred_proba_rf = rf.predict_proba(X_test)[:, 1]
y_pred_proba_gb = gb.predict_proba(X_test)[:, 1]

roc_auc_rf = roc_auc_score(y_test, y_pred_proba_rf)
roc_auc_gb = roc_auc_score(y_test, y_pred_proba_gb)

fpr_rf, tpr_rf, _ = roc_curve(y_test, y_pred_proba_rf)
fpr_gb, tpr_gb, _ = roc_curve(y_test, y_pred_proba_gb)

plt.figure(figsize=(8, 6))
plt.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {roc_auc_rf:.2f})')
plt.plot(fpr_gb, tpr_gb, label=f'Gradient Boosting (AUC = {roc_auc_gb:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

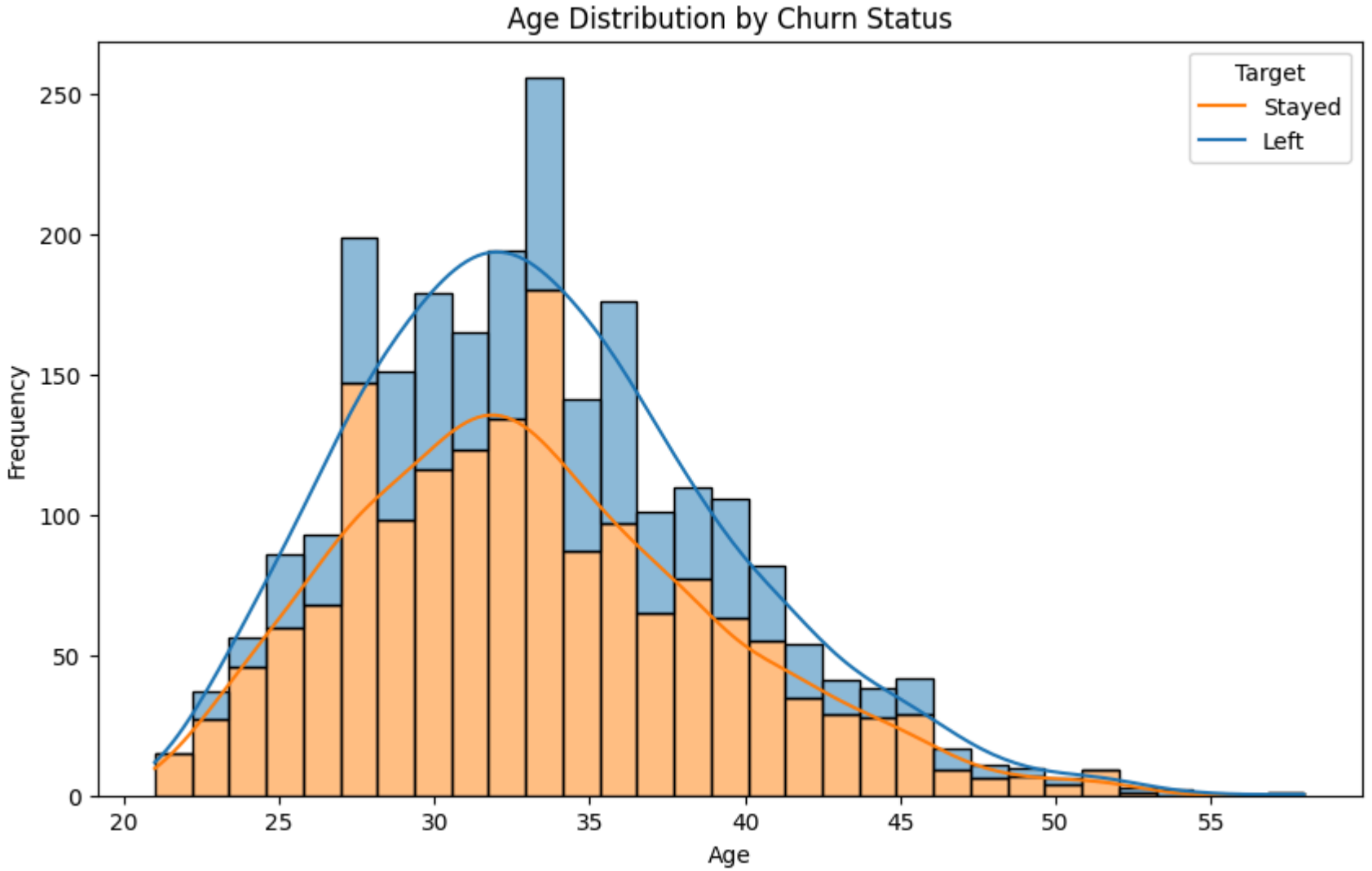


In [52]:
```
# Confusion Matrix
print("Random Forest Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("Gradient Boosting Confusion Matrix:\n", confusion_matrix(y_test, y_pred_gb))
```

```
Random Forest Confusion Matrix:
 [[257  69]
 [ 74 247]]
Gradient Boosting Confusion Matrix:
 [[237  89]
 [ 67 254]]
```

In [59]:

```python
plt.figure(figsize=(10, 6))
sns.histplot(data=driver_data, x='Age', hue='Target', kde=True, multiple='stack')
plt.title('Age Distribution by Churn Status')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend(title='Target', labels=['Stayed', 'Left'])
plt.show()
```



In [61]:
```
!jupyter nbconvert --to html /content/OLA_business_case (1).ipynb
```

```
/bin/bash: -c: line 1: syntax error near unexpected token `('
/bin/bash: -c: line 1: `jupyter nbconvert --to html /content/OLA_business_case (1).ipynb'
```

In [ ]: