In [167]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import seaborn
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [1]:
```
!gdown 'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/492/original/ola_driver_scaler.csv'
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/492/original/ola_driver_scaler.csv
 (https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/492/original/ola_driver_scaler.csv)
To: /Users/apple/Documents/ola_driver_scaler.csv
100%|████████████████████████████████████████| 1.13M/1.13M [00:00<00:00, 2.09MB/s]
```

In [228]:
```
df = pd.read_csv('ola_driver_scaler.csv')

df.shape
```

Out[228]:
```
(19104, 14)
```

In [5]:
```
df.head()
```

Out[5]:

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

Define Problem Statement and perform Exploratory Data Analysis

Recruiting and retaining drivers is seen by industry watchers as a tough battle for Ola. Churn among drivers is high and it's very easy for drivers to stop working for the service on the fly or jump to Uber depending on the rates.

As the companies get bigger, the high churn could become a bigger problem. To find new drivers, Ola is casting a wide net, including people who don't have cars for jobs. But this acquisition is really costly. Losing drivers frequently impacts the morale of the organization and acquiring new drivers is more expensive than retaining existing ones.

In [6]:
```
df.shape
```

Out[6]:
```
(19104, 14)
```

In [7]:
```
df.dtypes
```

Out[7]:
```
Unnamed: 0                 int64
MMM-YY                    object
Driver_ID                  int64
Age                      float64
Gender                   float64
City                      object
Education_Level            int64
Income                     int64
Dateofjoining             object
LastWorkingDate           object
Joining Designation        int64
Grade                      int64
Total Business Value       int64
Quarterly Rating           int64
dtype: object
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

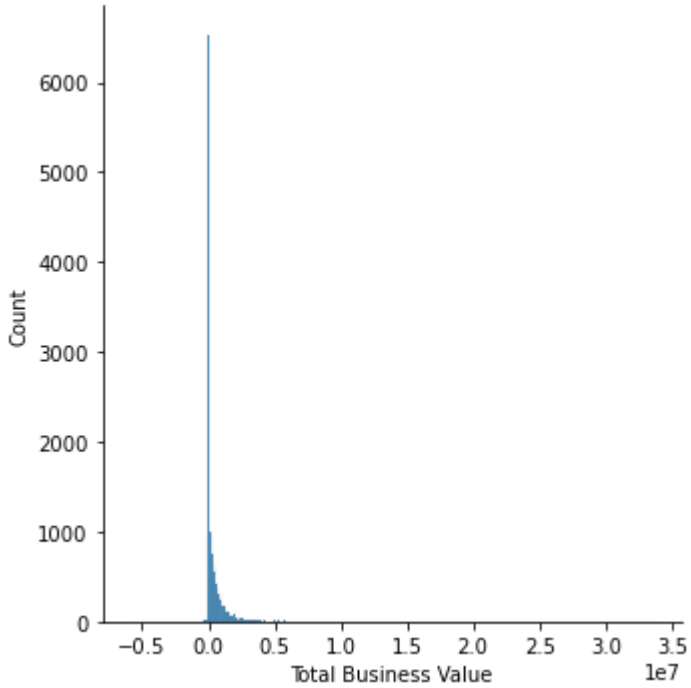
In [9]:
```
df.describe()
```

Out[9]:

	Unnamed: 0	Driver_ID	Age	Gender	Education_Level	Income	Joining Designation	Grade	Total Business Value	Qua R
count	19104.000000	19104.000000	19043.000000	19052.000000	19104.000000	19104.000000	19104.000000	19104.000000	1.910400e+04	19104.0(
mean	9551.500000	1415.591133	34.668435	0.418749	1.021671	65652.025126	1.690536	2.252670	5.716621e+05	2.0(
std	5514.994107	810.705321	6.257912	0.493367	0.800167	30914.515344	0.836984	1.026512	1.128312e+06	1.0(
min	0.000000	1.000000	21.000000	0.000000	0.000000	10747.000000	1.000000	1.000000	-6.000000e+06	1.0(
25%	4775.750000	710.000000	30.000000	0.000000	0.000000	42383.000000	1.000000	1.000000	0.000000e+00	1.0(
50%	9551.500000	1417.000000	34.000000	0.000000	1.000000	60087.000000	1.000000	2.000000	2.500000e+05	2.0(
75%	14327.250000	2137.000000	39.000000	1.000000	2.000000	83969.000000	2.000000	3.000000	6.997000e+05	3.0(
max	19103.000000	2788.000000	58.000000	1.000000	2.000000	188418.000000	5.000000	5.000000	3.374772e+07	4.0(

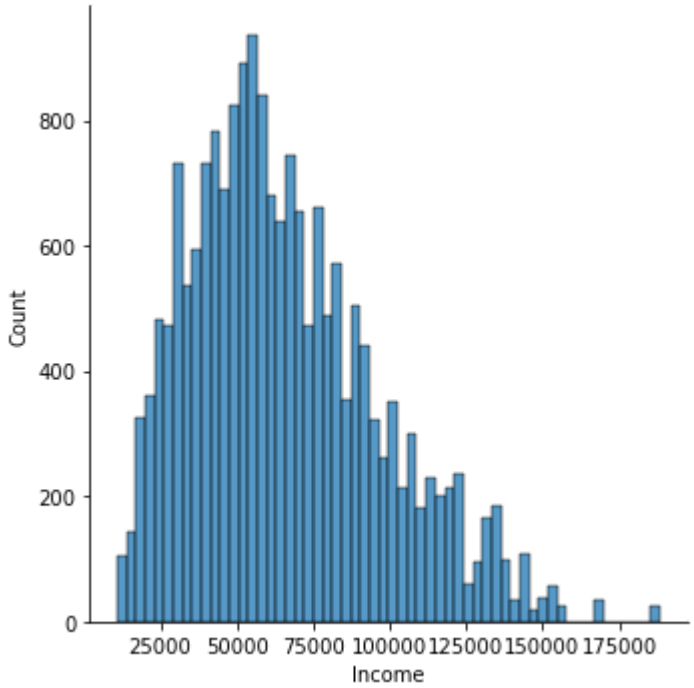
In [10]:
```
seaborn.displot(df['Total Business Value'])
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa65162b880>
```



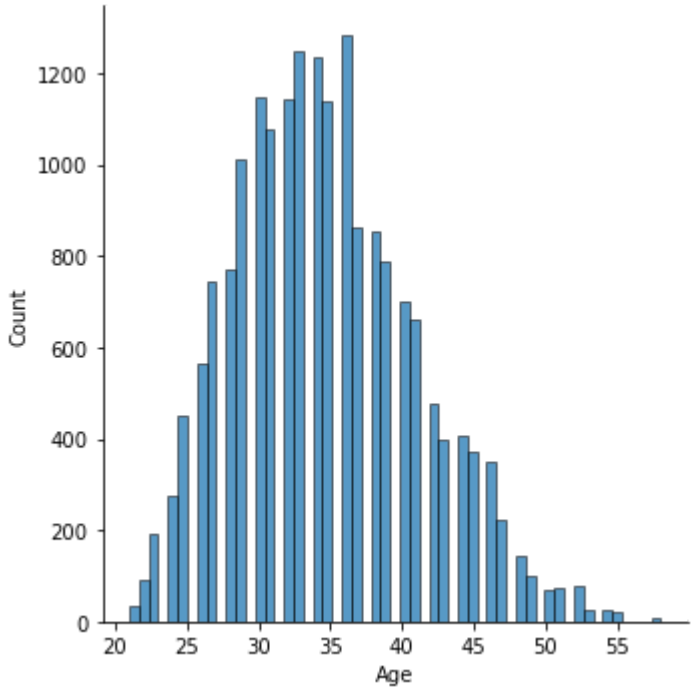
In [11]:
```
seaborn.displot(df['Income'])
```

Out[11]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa65689a400>
```



In [12]:
```
seaborn.displot(df['Age'])
```

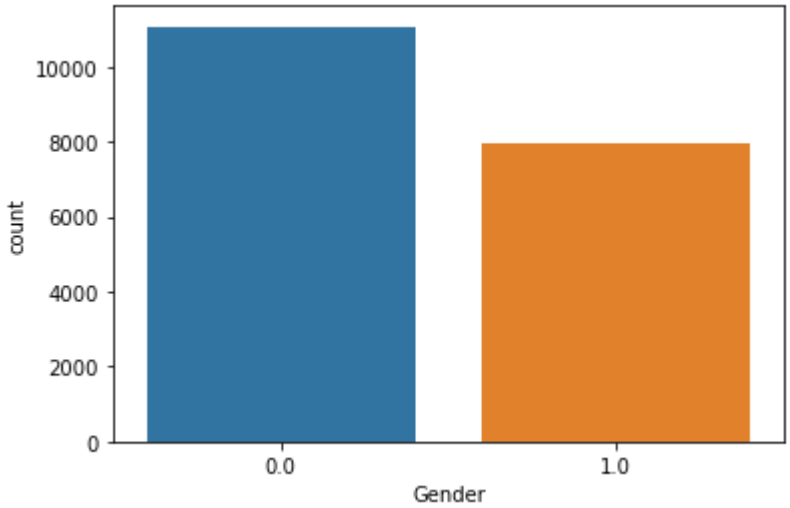
Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa6512459a0>
```



In [17]:
```
seaborn.countplot(df['Gender'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

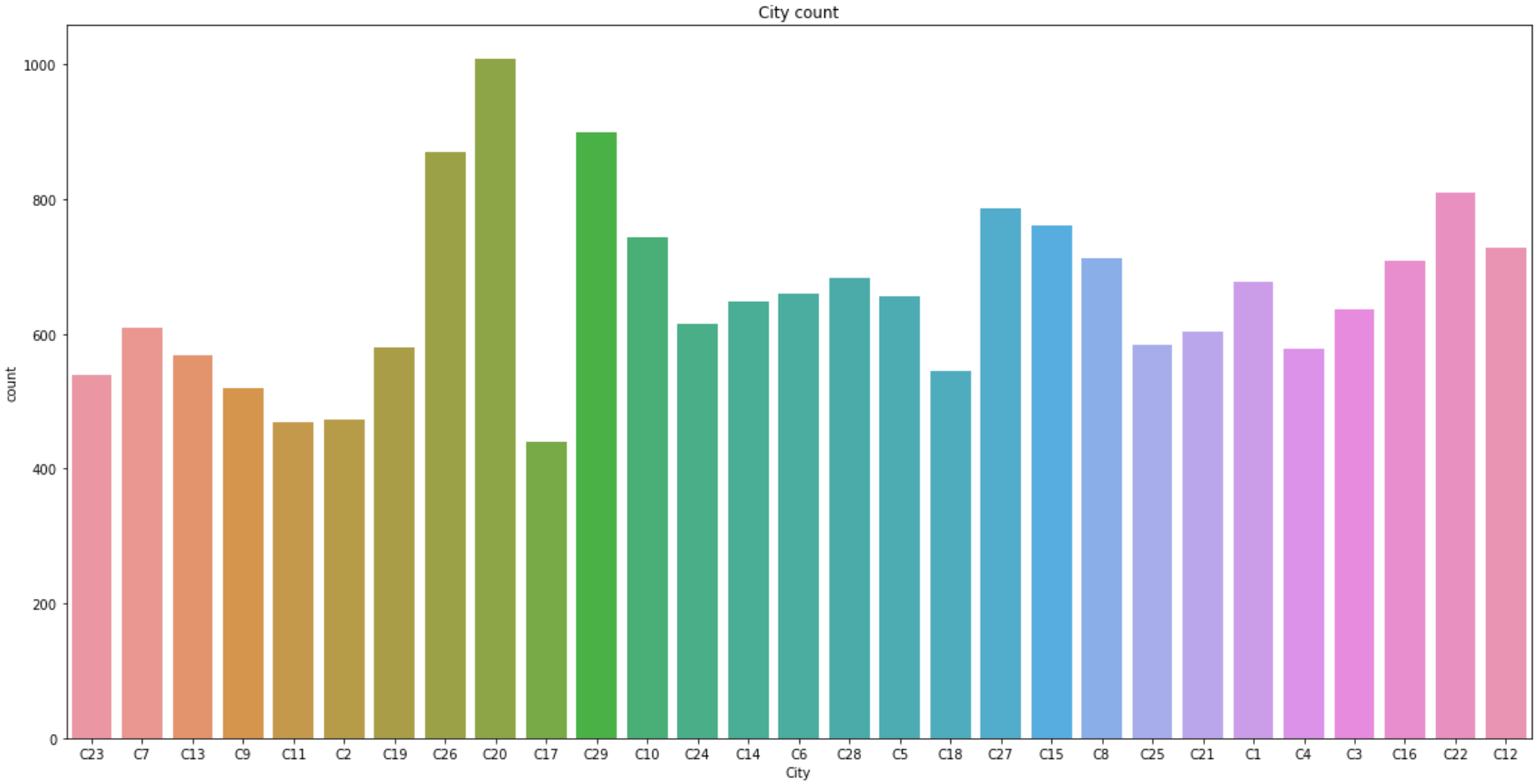
Out[17]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



In [21]:
```
fig = plt.figure(figsize = (20, 10))
seaborn.countplot(df['City'])
plt.title("City count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

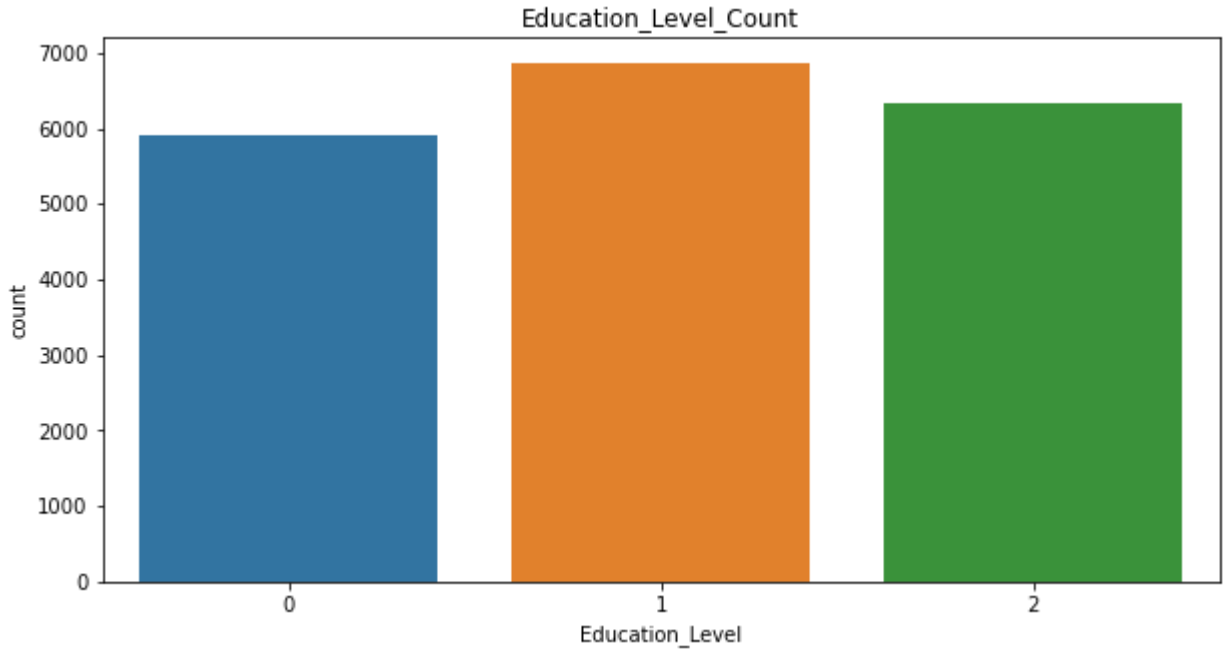
Out[21]: Text(0.5, 1.0, 'City count')



In [25]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.countplot(df['Education_Level'])
plt.title("Education_Level_Count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

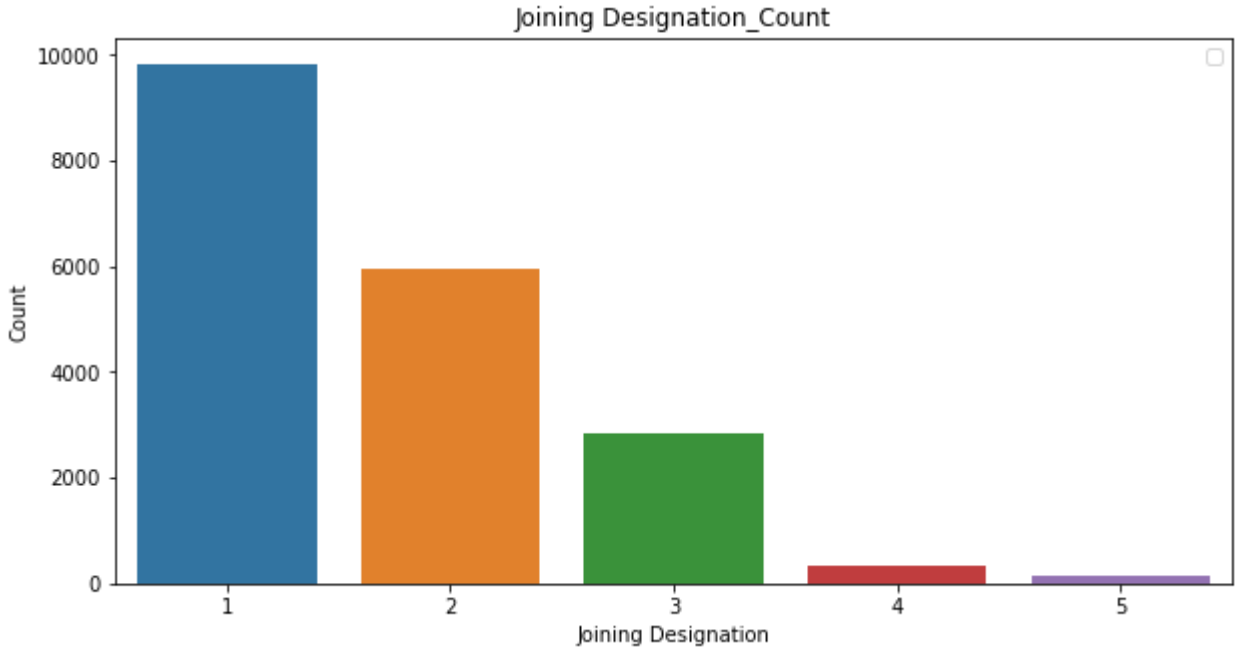
Out[25]: Text(0.5, 1.0, 'Education_Level_Count')



In [27]:
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (10, 5))
seaborn.countplot(df['Joining Designation'])
plt.xlabel("Joining Designation")
plt.ylabel("Count")
plt.title("Joining Designation_Count")
plt.legend()
plt.show()
```

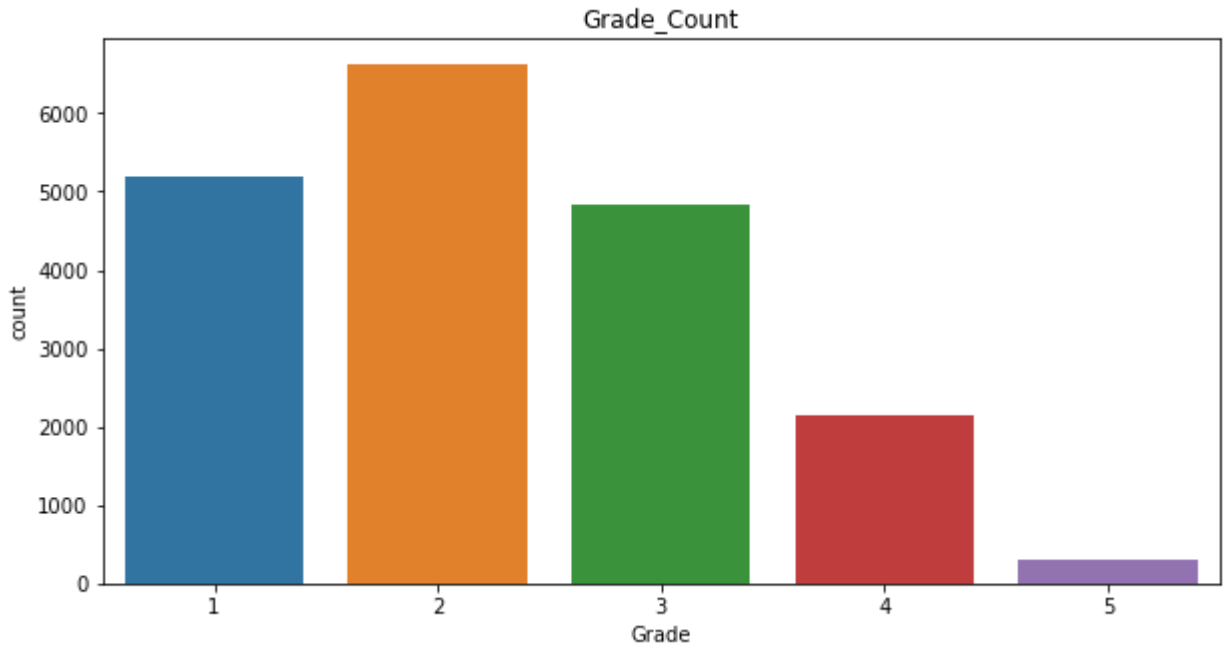
```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ig
nored when legend() is called with no argument.
```



In [29]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.countplot(df['Grade'])
plt.title("Grade_Count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

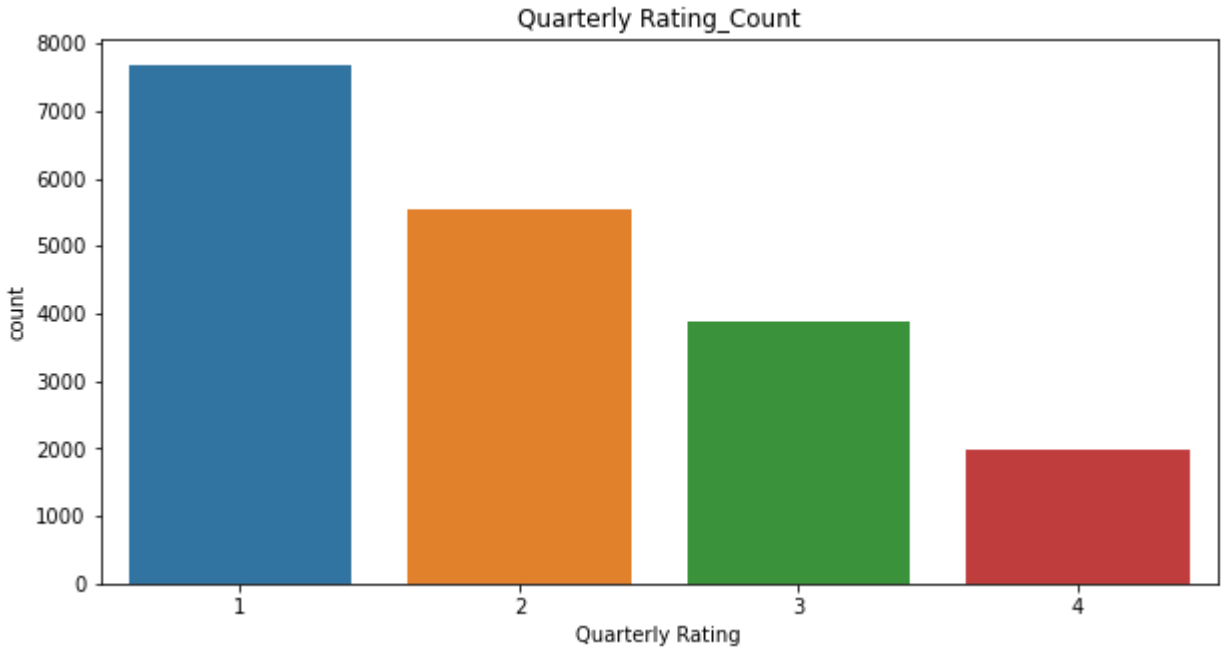
Out[29]:
```
Text(0.5, 1.0, 'Grade_Count')
```



In [37]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.countplot(df['Quarterly Rating'])
plt.title("Quarterly Rating_Count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

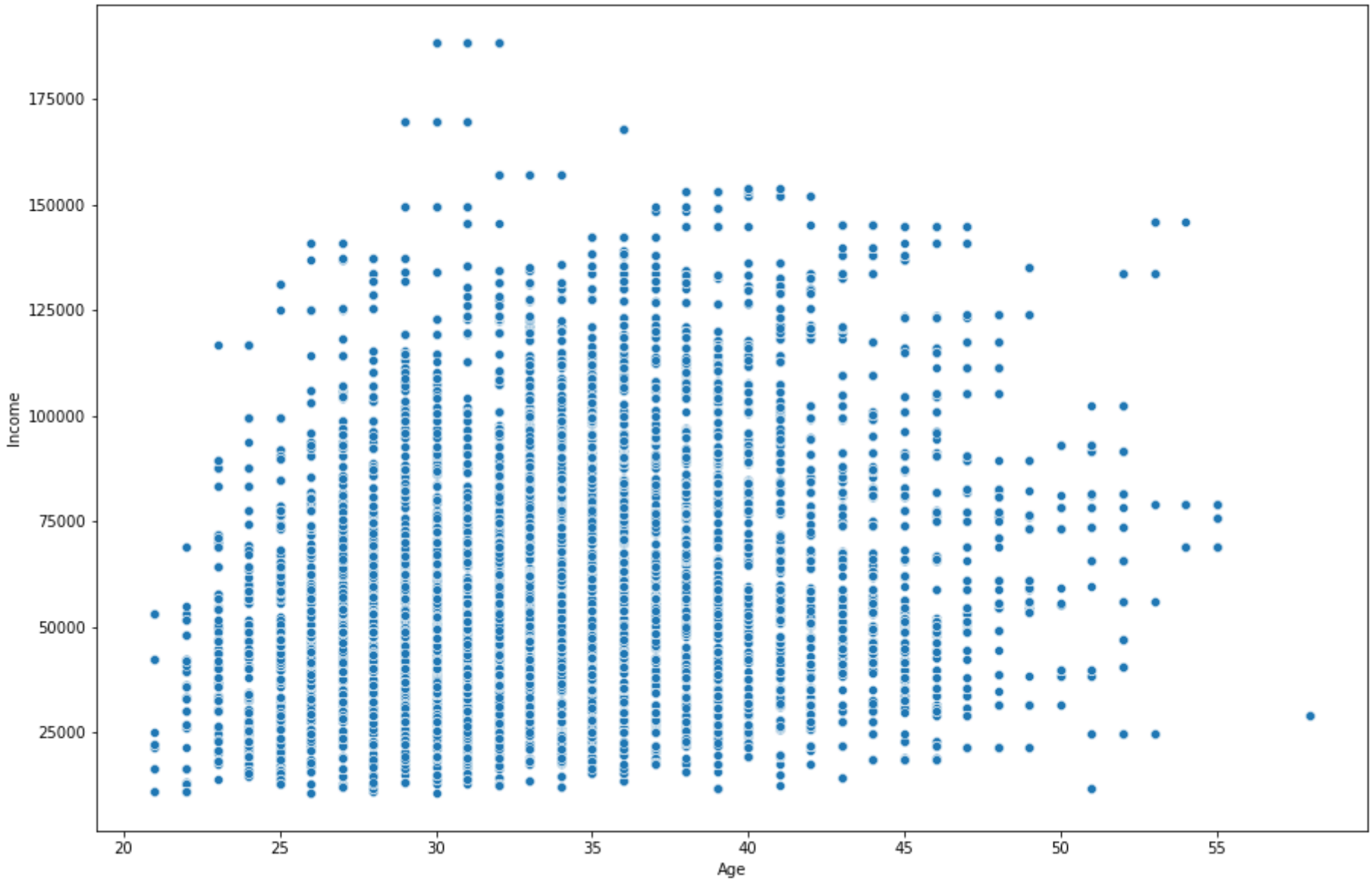
Out[37]: Text(0.5, 1.0, 'Quarterly Rating_Count')



In [65]:
```
fig = plt.figure(figsize = (15, 10))
seaborn.scatterplot(df['Age'], df['Income'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

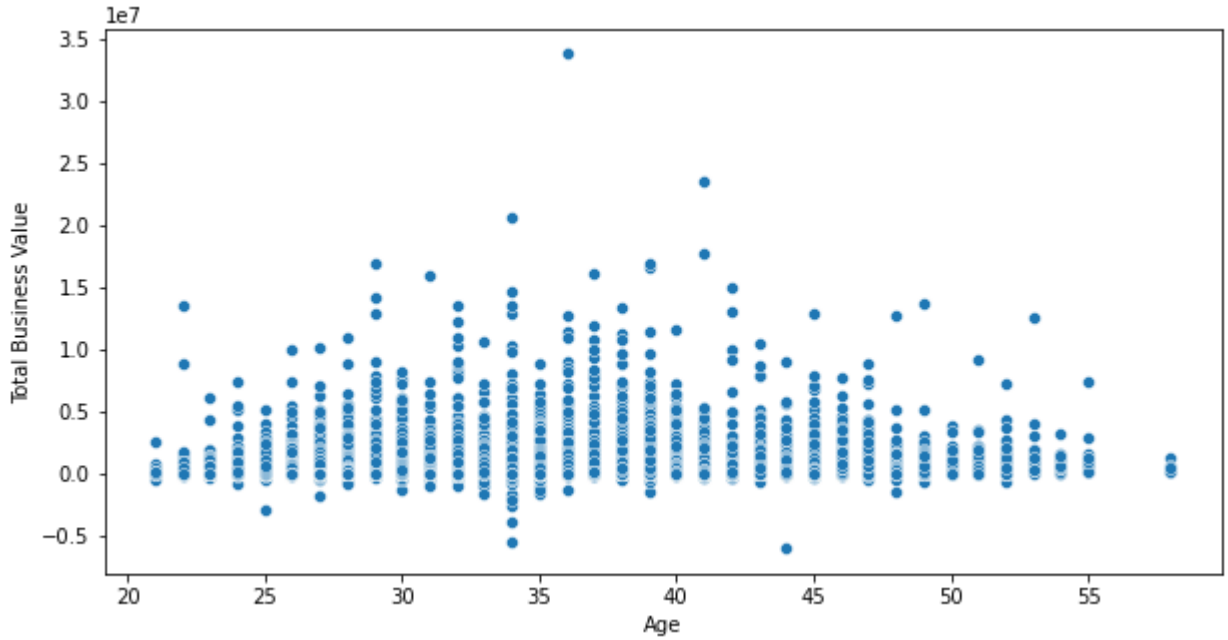
Out[65]:
```
<AxesSubplot:xlabel='Age', ylabel='Income'>
```


175000
150000
125000
100000
75000
50000
25000
Income
20
25
30
35
40
45
50
55
Age

In [43]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.scatterplot(df['Age'], df['Total Business Value'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

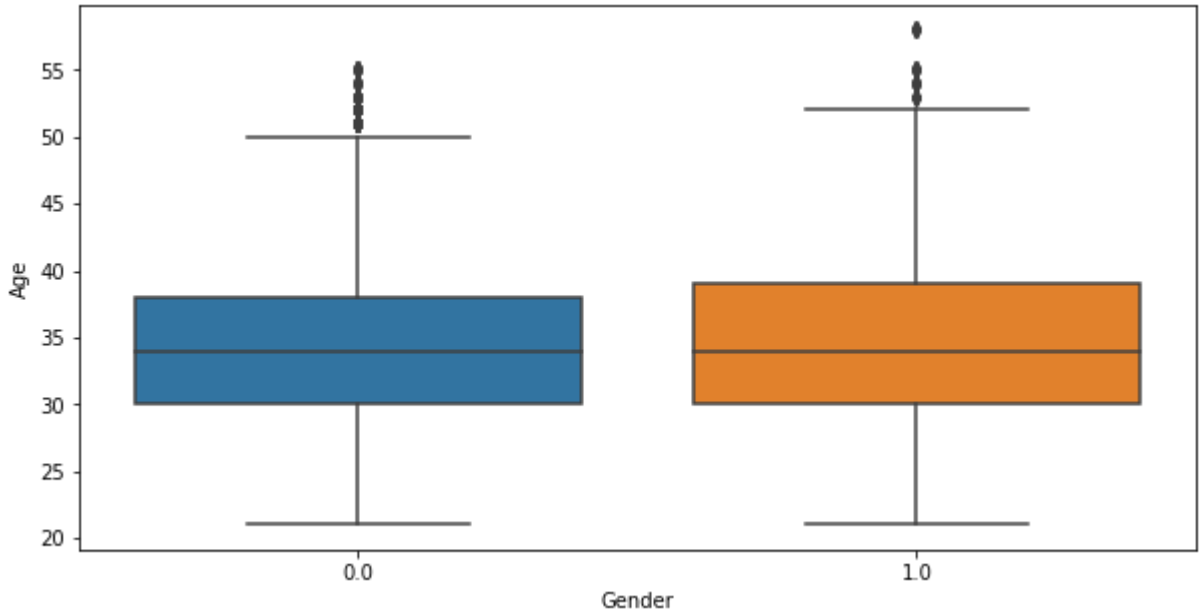
Out[43]: <AxesSubplot:xlabel='Age', ylabel='Total Business Value'>



In [45]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.boxplot(df['Gender'], df['Age'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

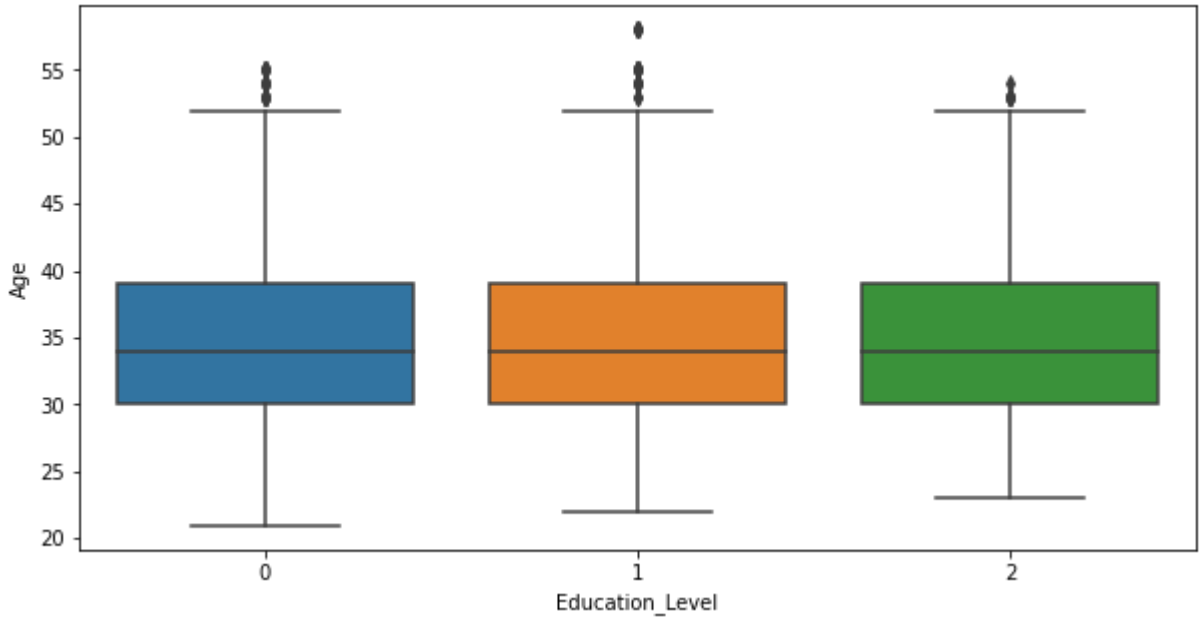
Out[45]: <AxesSubplot:xlabel='Gender', ylabel='Age'>



In [48]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.boxplot(df['Education_Level'], df['Age'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

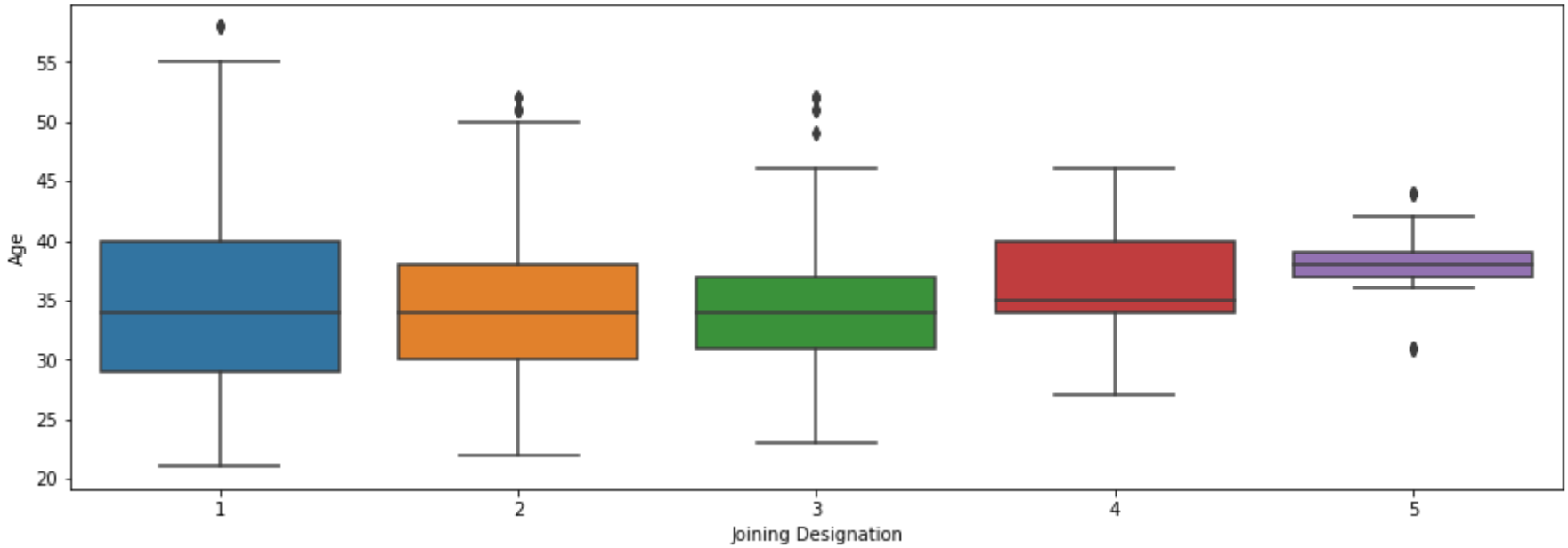
Out[48]: <AxesSubplot:xlabel='Education_Level', ylabel='Age'>



In [52]:
```
fig = plt.figure(figsize = (15, 5))
seaborn.boxplot(df['Joining Designation'], df['Age'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

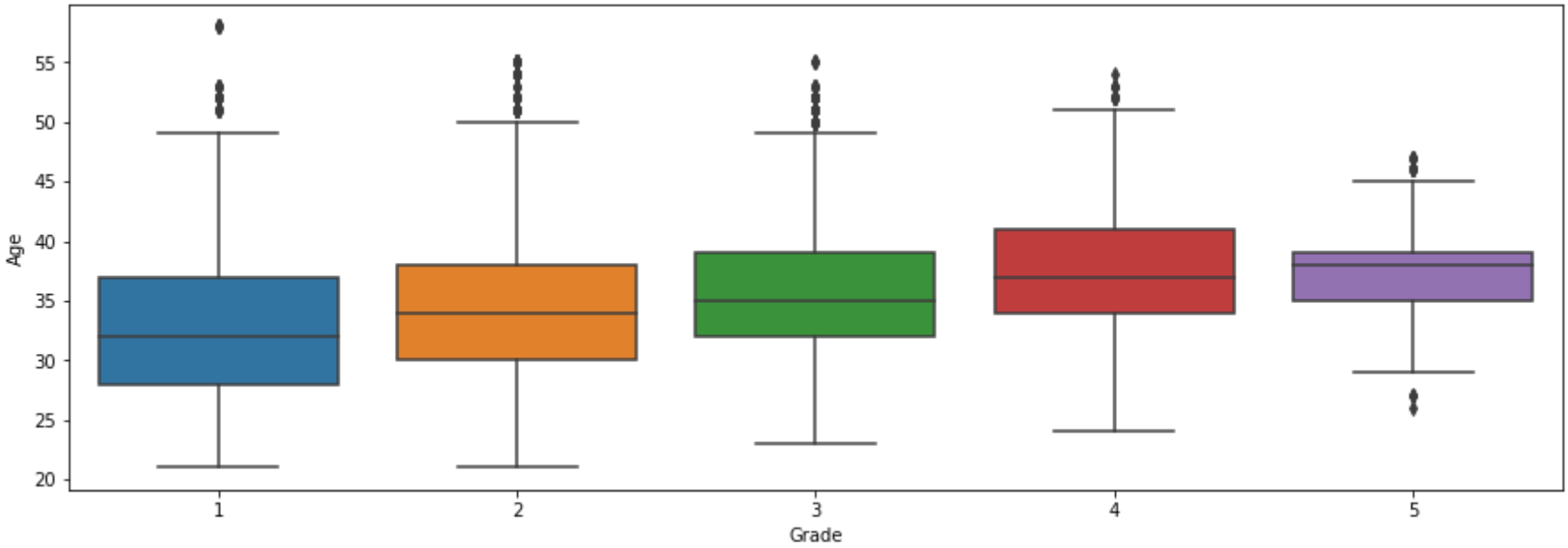
Out[52]: <AxesSubplot:xlabel='Joining Designation', ylabel='Age'>



In [53]:
```
fig = plt.figure(figsize = (15, 5))
seaborn.boxplot(df['Grade'], df['Age'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

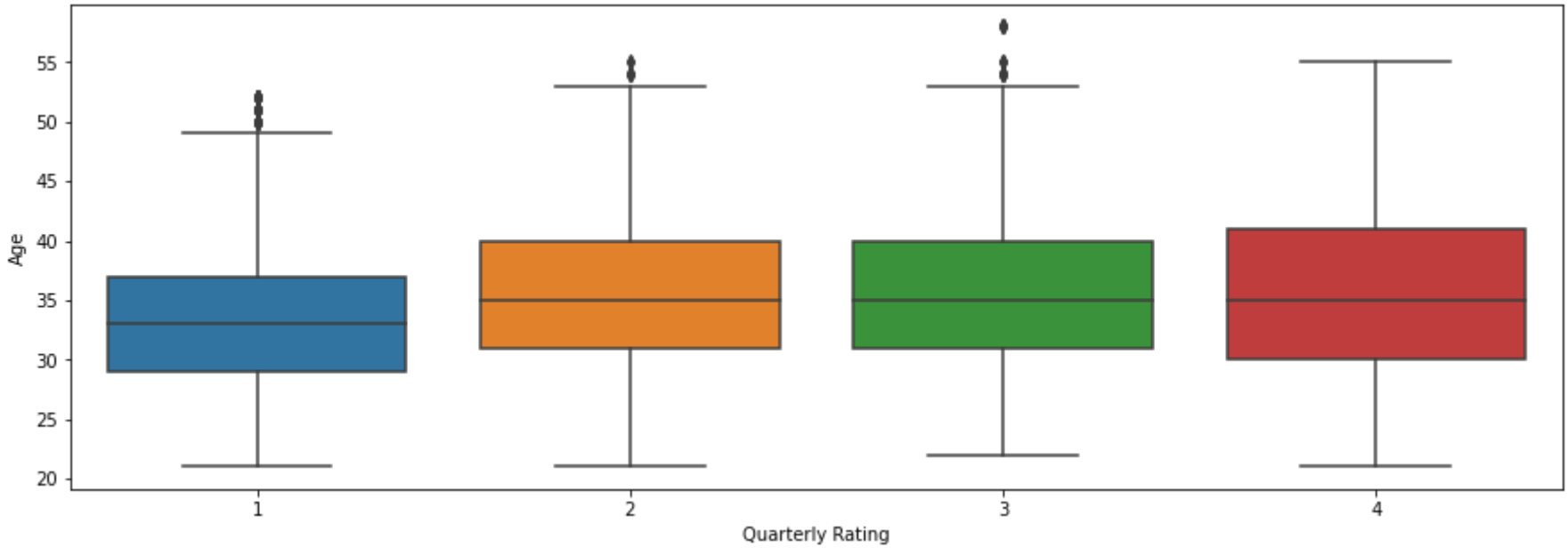
Out[53]: <AxesSubplot:xlabel='Grade', ylabel='Age'>



In [55]:
```
fig = plt.figure(figsize = (15, 5))
seaborn.boxplot(df['Quarterly Rating'], df['Age'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

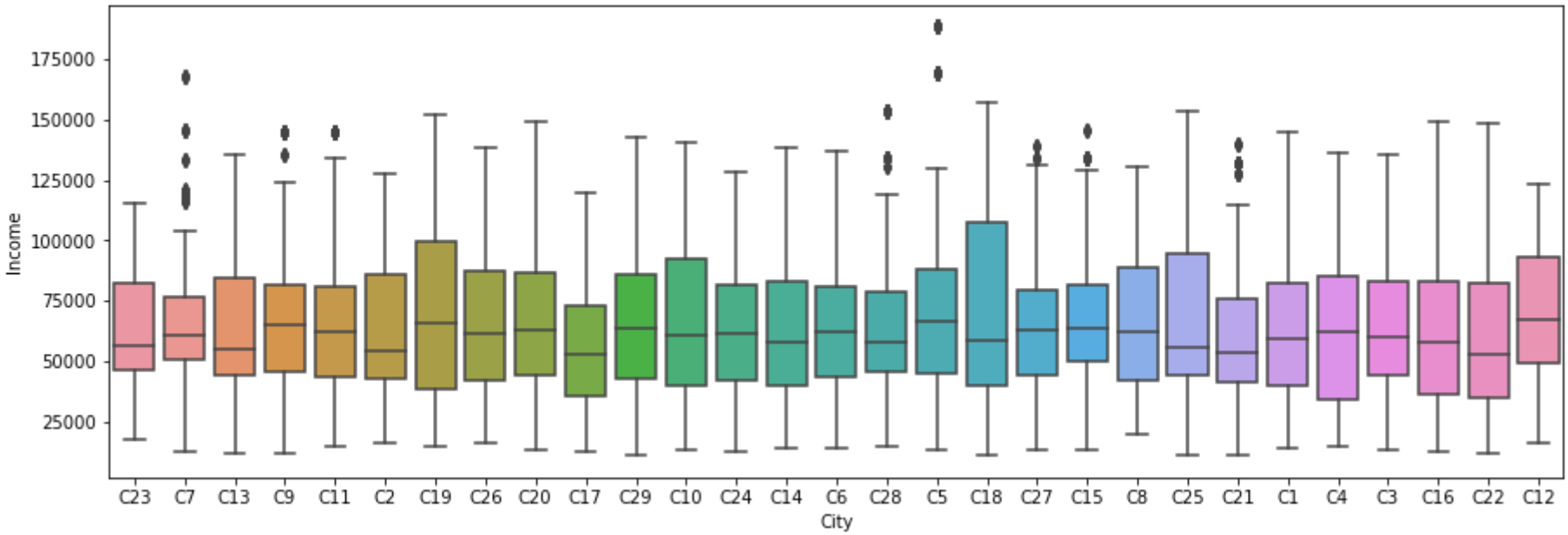
Out[55]: <AxesSubplot:xlabel='Quarterly Rating', ylabel='Age'>



In [57]:
```
fig = plt.figure(figsize = (15, 5))
seaborn.boxplot(df['City'], df['Income'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

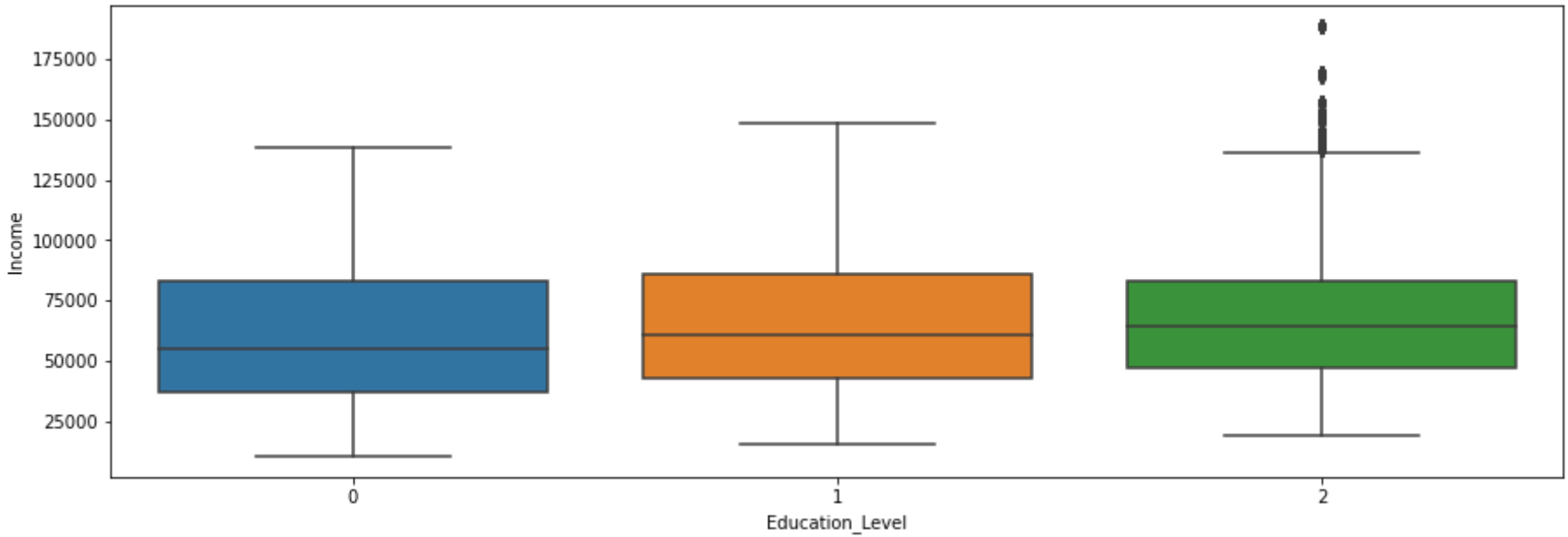
Out[57]: <AxesSubplot:xlabel='City', ylabel='Income'>



In [67]:
```
fig = plt.figure(figsize = (15, 5))
seaborn.boxplot(df['Education_Level'], df['Income'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

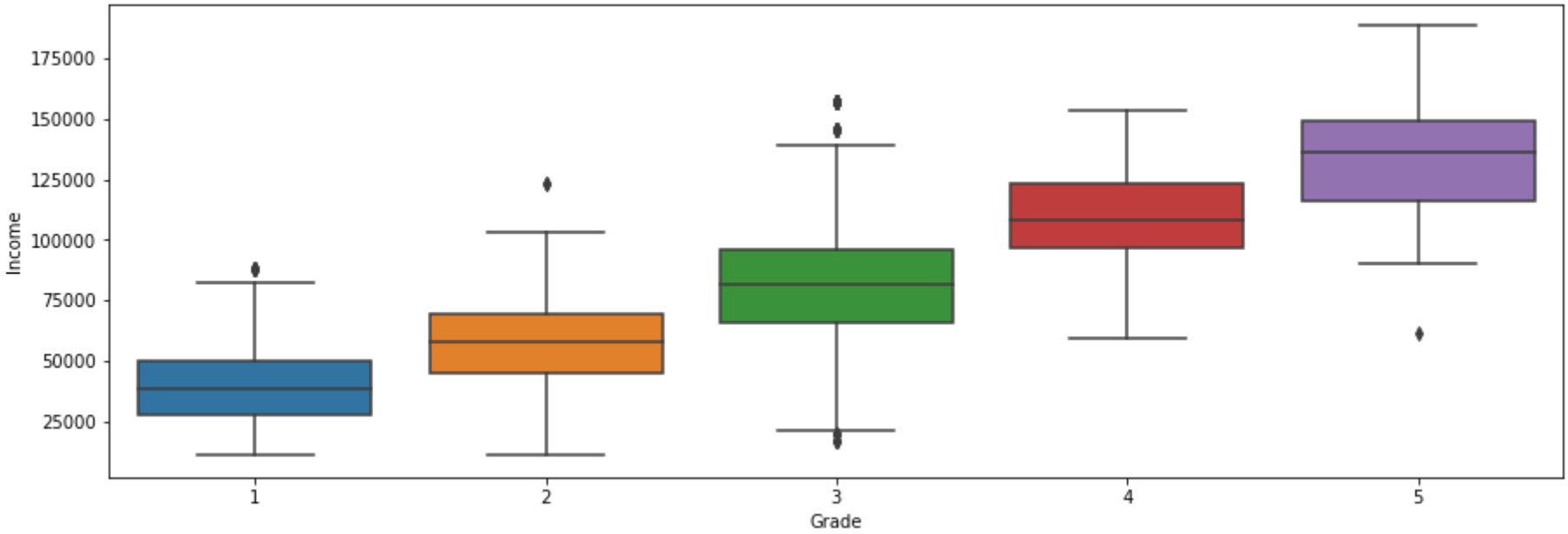
Out[67]: <AxesSubplot:xlabel='Education_Level', ylabel='Income'>



In [68]:
```
fig = plt.figure(figsize = (15, 5))
seaborn.boxplot(df['Grade'], df['Income'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

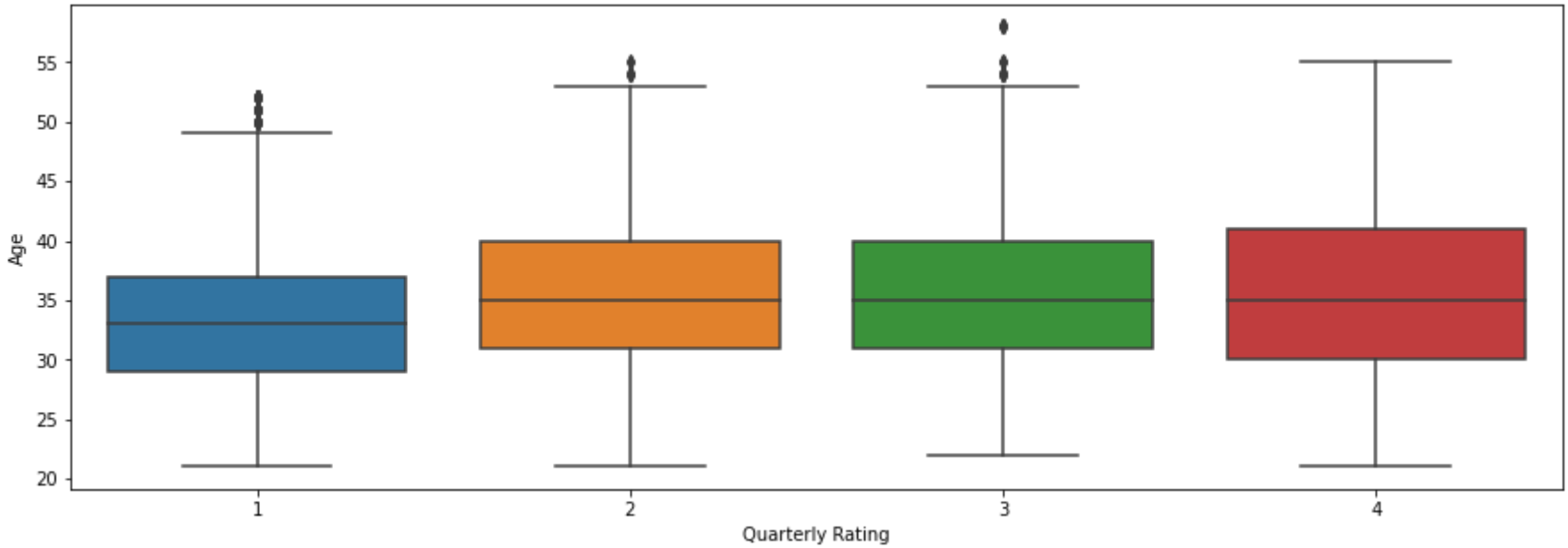
Out[68]: <AxesSubplot:xlabel='Grade', ylabel='Income'>



In [71]:
```
fig = plt.figure(figsize = (15, 5))
seaborn.boxplot(df['Quarterly Rating'], df['Age'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[71]: <AxesSubplot:xlabel='Quarterly Rating', ylabel='Age'>



Data Preprocessing

KNN Imputation Feature Engineering Class Imbalance treatment Standardization Encoding

In [229]:
```
df['MMM-YY'] = pd.to_datetime(df['MMM-YY'])
df['LastWorkingDate'] = pd.to_datetime(df['LastWorkingDate'])
df['Dateofjoining'] = pd.to_datetime(df['Dateofjoining'])
```

In [230]:
```
# knn imputation transform for the horse colic dataset
from numpy import isnan
from pandas import read_csv
from sklearn.impute import KNNImputer
# define imputer
imputer = KNNImputer(n_neighbors=5, weights='uniform', metric='nan_euclidean')
```

In [231]:
```
imputer.fit(df['Age'].values.reshape(-1,1))
```

Out[231]:
```
KNNImputer()
```

In [232]:
```
df['Age'] = imputer.transform(df['Age'].values.reshape(-1,1))
```

In [233]:
```
imputer.fit(df['Gender'].values.reshape(-1,1))
df['Gender'] = imputer.transform(df['Gender'].values.reshape(-1,1))
```

In [234]:
```
df['LastWorkingDate'][~df['LastWorkingDate'].isnull()] = 1
df['LastWorkingDate'][df['LastWorkingDate'].isnull()] = 0
```

```
/var/folders/2m/svsbyfss4h53t29lk1vcnvf40000gn/T/ipykernel_58911/647458482.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['LastWorkingDate'][~df['LastWorkingDate'].isnull()] = 1
/var/folders/2m/svsbyfss4h53t29lk1vcnvf40000gn/T/ipykernel_58911/647458482.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['LastWorkingDate'][df['LastWorkingDate'].isnull()] = 0
```

In [235]:
```
df['Target']=df['LastWorkingDate']
df.drop(['LastWorkingDate','MMM-YY','Dateofjoining'], axis=1, inplace=True)
```

In [236]:
```
df.drop(['Unnamed: 0'], axis=1, inplace=True)
```

In [237]:
```
df['Target'].value_counts()
```

Out[237]:
```
0    17488
1     1616
Name: Target, dtype: int64
```

In [238]:
```
df['Total Business Value'] = StandardScaler().fit_transform(np.array(df['Total Business Value']).reshape(-1, 1))
```

In [239]:
```
df['Income'] = StandardScaler().fit_transform(np.array(df['Income']).reshape(-1, 1))
```

In [240]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le = preprocessing.LabelEncoder()
le.fit(df['City'])

df['City']=le.transform(df['City'])
```

In [246]:
```
df.head()
```

Out[246]:

	Driver_ID	Age	Gender	City	Education_Level	Income	Joining Designation	Grade	Total Business Value	Quarterly Rating	Target
0	1	28.0	0.0	15	2	-0.267358	1	1	1.603674	2	0
1	1	28.0	0.0	15	2	-0.267358	1	1	-1.096482	2	0
2	1	28.0	0.0	15	2	-0.267358	1	1	-0.506666	2	1
3	2	31.0	0.0	26	2	0.044122	2	2	-0.506666	1	0
4	2	31.0	0.0	26	2	0.044122	2	2	-0.506666	1	0

Model building

In [247]:
```
X=df.drop(['Target'], axis=1)
y=df['Target']
```

In [257]:
```
y=y.astype('int')
```

In [258]:
```
# Split features and target into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1, stratify=y)
```

In [259]:
```
# Instantiate and fit the RandomForestClassifier
forest = RandomForestClassifier()
forest.fit(X_train, y_train)
```

Out[259]:
```
RandomForestClassifier()
```

In [260]:
```
# Make predictions for the test set
y_pred_test = forest.predict(X_test)
```

In [261]:
```
# View accuracy score
accuracy_score(y_test, y_pred_test)
```

Out[261]:
```
0.8777219430485762
```

In [262]:
```
# View confusion matrix for test data and predictions
confusion_matrix(y_test, y_pred_test)
```

Out[262]:
```
array([[4180,  192],
       [ 392,   12]])
```

In [263]:
```
# View the classification report for test data and predictions
print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.91      0.96      0.93      4372
           1       0.06      0.03      0.04       404

    accuracy                           0.88      4776
   macro avg       0.49      0.49      0.49      4776
weighted avg       0.84      0.88      0.86      4776
```

In [274]:
```
y1_pred_proba=forest.predict_proba(X_test)[::,1]
```

In [275]:

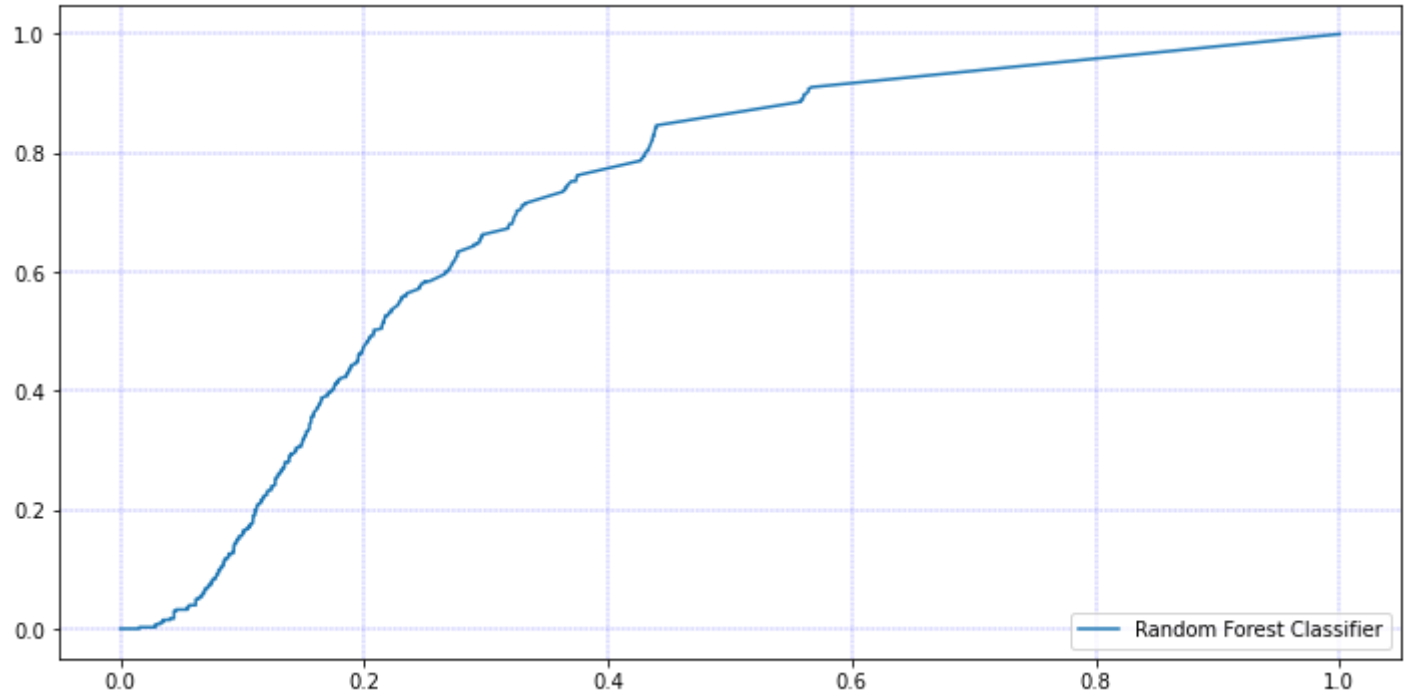
```
import matplotlib.pyplot as plt
from sklearn import metrics

fpr1, tpr1, _a = metrics.roc_curve(y_test,  y1_pred_proba)

plt.figure(figsize=(12,6))

plt.plot(fpr1,tpr1,label="Random Forest Classifier")


plt.legend(loc=4)
plt.grid(color='b', ls = '-.', lw = 0.25)
plt.show()
```



Boosting

XGBClassifier

In [265]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from xgboost import XGBClassifier
# define dataset
#fine the model
model = XGBClassifier()
# evaluate the model
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (np.mean(n_scores), np.std(n_scores)))
```

```
Accuracy: 0.908 (0.003)
```

In [266]:
```
# make predictions using xgboost for classification
from numpy import asarray
from sklearn.datasets import make_classification
from xgboost import XGBClassifier
# define dataset
# define the model
model = XGBClassifier()
# fit the model on the whole dataset
model.fit(X_train, y_train)

yhat = model.predict(X_test)
```

```
Predicted Class: 0
```

In [267]:
```
accuracy_score(y_test, yhat)
```

Out[267]:
```
0.9036850921273032
```

In [268]:
```
# View confusion matrix for test data and predictions
confusion_matrix(y_test, yhat)
```

Out[268]:
```
array([[4311,   61],
       [ 399,    5]])
```

In [276]:
```
y2_pred_prob=model.predict_proba(X_test)[::,1]
```

GradientBoostingClassifier

In [283]:
```
# for classification
from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier()
model.fit(X_train,y_train)
```

Out[283]:
```
GradientBoostingClassifier()
```

In [287]:
```
yhat = model.predict(X_test)
```

In [288]:
```
accuracy_score(y_test, yhat)
```

Out[288]:
```
0.9139447236180904
```

In [289]:
```
# View confusion matrix for test data and predictions
confusion_matrix(y_test, yhat)
```

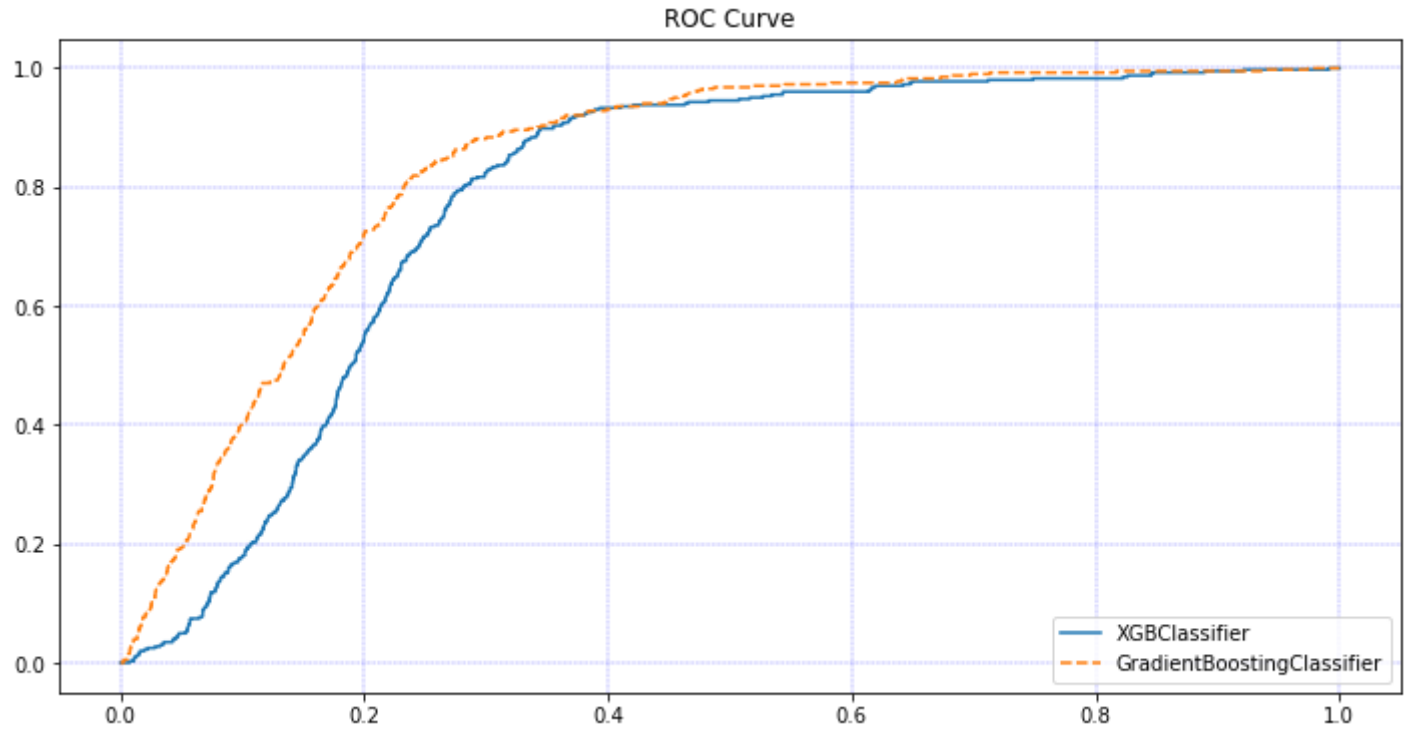
Out[289]:
```
array([[4365,    7],
       [ 404,    0]])
```

In [290]:
```
y3_pred_prob=model.predict_proba(X_test)[::,1]
```

In [293]:
```
import matplotlib.pyplot as plt
fpr2,tpr2,_b= metrics.roc_curve(y_test,  y2_pred_prob)
fpr3,tpr3,_c= metrics.roc_curve(y_test,  y3_pred_prob)

plt.figure(figsize=(12,6))


plt.plot(fpr2,tpr2,label="XGBClassifier")
plt.plot(fpr3,tpr3,'--',label="GradientBoostingClassifier")
plt.title('ROC Curve')
plt.legend(loc=4)
plt.grid(color='b', ls = '-.', lw = 0.25)
plt.show()
```



Actionable Insights & Recommendations

In [ ]:
```
Income and age seems randomly distributed
2 most of the drivers starts with joining desingnation 1
3. most of the drive at Grade 2
4 income of the driver increases after grade increase
5.s
```