In [ ]:
```
# Project Name: Gulf Countries Oil Market Analysis

# Pravir Mishra ( E22BCAU0143 )
# Harsh Chaudhary ( E22BCAU0029 )
# Nakul Chauhan ( E22BCAU0013 )

# Course Name : Digital Marketing and Trend Analysis
# Course Code : CBCA311


# Project Analysis using Python
```

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

df1 = pd.read_csv('gcc_oil_export_data.csv')

df1
```

Out[ ]:

	Country	Year	Export Volume (barrels)	Export Value (USD)	Destination	Type of Oil	Price per Barrel (USD)	Export Revenue (USD)
0	Saudi Arabia	2001	689355	3.112270e+07	Japan	Crude Oil	75.46	5.201873e+07
1	UAE	2001	1370340	9.725328e+07	USA	Refined	32.02	4.387829e+07
2	Qatar	2001	731178	2.275677e+07	USA	Refined	75.05	5.487491e+07
3	Kuwait	2001	907548	6.533426e+07	China	Crude Oil	35.48	3.219980e+07
4	Bahrain	2001	1759528	1.223758e+08	South Korea	Refined	55.04	9.684442e+07
...	...	...	...	...	...	...	...	...
127	UAE	2022	1163200	3.668886e+07	South Korea	Refined	94.02	1.093641e+08
128	Qatar	2022	1725395	6.561293e+07	USA	Crude Oil	49.27	8.501021e+07
129	Kuwait	2022	1731988	1.137177e+08	India	LNG	37.81	6.548647e+07
130	Bahrain	2022	1012640	6.632194e+07	Japan	Crude Oil	81.46	8.248965e+07
131	Oman	2022	1938559	1.517464e+08	China	Crude Oil	74.51	1.444420e+08

132 rows × 8 columns

In [ ]:
```
print(df1.isnull().sum())
```

```
Country                   0
Year                      0
Export Volume (barrels)   0
Export Value (USD)        0
Destination               0
Type of Oil               0
Price per Barrel (USD)    0
Export Revenue (USD)      0
dtype: int64
```

In [ ]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Country                  132 non-null    object
 1   Year                     132 non-null    int64
 2   Export Volume (barrels)  132 non-null    int64
 3   Export Value (USD)       132 non-null    float64
 4   Destination              132 non-null    object
 5   Type of Oil              132 non-null    object
 6   Price per Barrel (USD)   132 non-null    float64
 7   Export Revenue (USD)     132 non-null    float64
dtypes: float64(3), int64(2), object(3)
memory usage: 8.4+ KB
```

In [ ]:
```
df1.describe()
```

Out[ ]:

	Year	Export Volume (barrels)	Export Value (USD)	Price per Barrel (USD)	Export Revenue (USD)
count	132.000000	1.320000e+02	1.320000e+02	132.000000	1.320000e+02
mean	2011.500000	1.240962e+06	6.679338e+07	63.483106	7.884493e+07
std	6.368458	4.499220e+05	3.142591e+07	21.271765	3.999089e+07
min	2001.000000	5.106100e+05	1.753278e+07	30.550000	1.753435e+07
25%	2006.000000	8.310152e+05	4.338713e+07	42.355000	4.652921e+07
50%	2011.500000	1.247350e+06	5.844245e+07	64.440000	7.223156e+07
75%	2017.000000	1.703771e+06	8.311628e+07	82.142500	1.094798e+08
max	2022.000000	1.971838e+06	1.517464e+08	99.690000	1.836111e+08

In [ ]:
```
X1 = df1[['Export Volume (barrels)', 'Price per Barrel (USD)']]
y1 = df1['Export Revenue (USD)']

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2, random_state=42)

model1 = LinearRegression()
model1.fit(X1_train, y1_train)

y1_pred = model1.predict(X1_test)

print("Predictions:", y1_pred)
print("Actual values:", y1_test.values)
```

```
Predictions: [ 1.42762024e+08  6.13060404e+07  9.62525465e+07  2.07900947e+07
  1.10714866e+08  8.13473979e+07  2.15489622e+07  9.11708054e+07
  1.04393245e+08  7.64080111e+07  4.42922839e+07  1.16346201e+08
  1.09255334e+08  1.00715670e+08  7.16177736e+07  1.17806150e+08
  1.15390295e+08  1.35410853e+08 -4.68925237e+06  2.65073671e+07
  8.26526721e+07  1.01760810e+08  8.86713581e+07  1.07944994e+07
  5.35301939e+07  6.93131166e+07  1.12958132e+08]
Actual values: [1.54498567e+08 4.82622202e+07 9.62269128e+07 3.19082794e+07
 1.09827173e+08 7.91719162e+07 3.24477106e+07 8.50102117e+07
 9.83582746e+07 7.64891457e+07 4.77338645e+07 1.20030885e+08
 1.11126990e+08 9.68444211e+07 6.01933376e+07 1.21820267e+08
 1.10967564e+08 1.44598202e+08 1.75343474e+07 3.37710778e+07
 7.15307804e+07 1.01147921e+08 6.93298241e+07 2.59535615e+07
 4.61004388e+07 6.89761958e+07 1.09364064e+08]
```

In [ ]:
```
df2 = pd.read_csv('revenue_utilization.csv')

df2.head()
```

Out[ ]:

	Country	Year	Export Revenue (USD)	Infrastructure	Healthcare	Education	Other
0	Saudi Arabia	2001	52018728.30	20807491.32	10403745.66	10403745.66	10403745.66
1	UAE	2001	43878286.80	21939143.40	8775657.36	8775657.36	4387828.68
2	Qatar	2001	54874908.90	16462472.67	21949963.56	5487490.89	10974981.78
3	Kuwait	2001	32199803.04	19319881.82	3219980.30	3219980.30	6439960.61
4	Bahrain	2001	96844421.12	38737768.45	29053326.34	19368884.22	9684442.11

In [ ]:

```
print(df2.isnull().sum())

df2.info()
```

```
Country                 0
Year                    0
Export Revenue (USD)    0
Infrastructure          0
Healthcare              0
Education               0
Other                   0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Country               132 non-null    object 
 1   Year                  132 non-null    int64  
 2   Export Revenue (USD)  132 non-null    float64
 3   Infrastructure        132 non-null    float64
 4   Healthcare            132 non-null    float64
 5   Education             132 non-null    float64
 6   Other                 132 non-null    float64
dtypes: float64(5), int64(1), object(1)
memory usage: 7.3+ KB
```

In [ ]:

```
df2.describe()
```

Out[ ]:

	Year	Export Revenue (USD)	Infrastructure	Healthcare	Education	Other
count	132.000000	1.320000e+02	1.320000e+02	1.320000e+02	1.320000e+02	1.320000e+02
mean	2011.500000	7.884493e+07	3.300682e+07	1.807373e+07	1.460801e+07	1.315637e+07
std	6.368458	3.999089e+07	2.001093e+07	1.138115e+07	9.318508e+06	7.933202e+06
min	2001.000000	1.753435e+07	6.752622e+06	2.449652e+06	2.250874e+06	1.753435e+06
25%	2006.000000	4.652921e+07	1.799834e+07	9.234077e+06	6.533871e+06	7.583402e+06
50%	2011.500000	7.223156e+07	2.837672e+07	1.601659e+07	1.332925e+07	1.090655e+07
75%	2017.000000	1.094798e+08	4.401238e+07	2.394543e+07	2.034540e+07	1.677721e+07
max	2022.000000	1.836111e+08	1.091195e+08	6.966578e+07	4.333261e+07	3.672222e+07

In [ ]:

```
X2 = df2[['Infrastructure', 'Healthcare', 'Education']]
y2 = df2['Export Revenue (USD)']

X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.2, random_state=42)

model2 = LinearRegression()
model2.fit(X2_train, y2_train)

y2_pred = model2.predict(X2_test)

print("Predictions:", y2_pred)
print("Actual values:", y2_test.values)
```

```
Predictions: [1.47214713e+08 4.68570134e+07 1.03560389e+08 3.50389982e+07
 1.18074572e+08 7.58946109e+07 3.56136773e+07 8.14727357e+07
 1.05853628e+08 7.34110724e+07 4.62060963e+07 1.14522545e+08
 1.06529257e+08 1.04240507e+08 5.79938324e+07 1.16679486e+08
 1.06178772e+08 1.37748405e+08 1.97298640e+07 3.29960957e+07
 6.88155251e+07 1.08826199e+08 6.67302923e+07 2.56812784e+07
 4.47222790e+07 6.65190498e+07 1.17555958e+08]
Actual values: [1.54498567e+08 4.82622202e+07 9.62269128e+07 3.19082794e+07
 1.09827173e+08 7.91719162e+07 3.24477106e+07 8.50102117e+07
 9.83582746e+07 7.64891457e+07 4.77338645e+07 1.20030885e+08
 1.11126990e+08 9.68444211e+07 6.01933376e+07 1.21820267e+08
 1.10967564e+08 1.44598202e+08 1.75343474e+07 3.37710778e+07
 7.15307804e+07 1.01147921e+08 6.93298241e+07 2.59535615e+07
 4.61004388e+07 6.89761958e+07 1.09364064e+08]
```

In [ ]:

```
df3 = pd.read_csv('tax_utilization_modified.csv')

df3.head()
```

Out[ ]:

	Country	Tax Rate per Year	Year Period	Tax Generated (USD)	Sector	Tax Utilization (USD)
0	USA	0.20	2001 - 2005	2.729335e+08	Infrastructure	5.458671e+08
1	USA	0.20	2001 - 2005	2.729335e+08	Healthcare	4.094003e+08
2	USA	0.20	2001 - 2005	2.729335e+08	Education	2.729335e+08
3	USA	0.20	2001 - 2005	2.729335e+08	Other	1.364668e+08
4	USA	0.25	2006 - 2010	3.411669e+08	Infrastructure	5.458671e+08

In [ ]:

```
print(df3.isnull().sum())
```

```
Country                  0
Tax Rate per Year        0
Year Period              0
Tax Generated (USD)      0
Sector                   0
Tax Utilization (USD)    0
dtype: int64
```

In [ ]:

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Country                100 non-null    object 
 1   Tax Rate per Year      100 non-null    float64
 2   Year Period            100 non-null    object 
 3   Tax Generated (USD)    100 non-null    float64
 4   Sector                 100 non-null    object 
 5   Tax Utilization (USD)  100 non-null    float64
dtypes: float64(3), object(3)
memory usage: 4.8+ KB
```

In [ ]:

```
df3.describe()
```

Out[ ]:

	Tax Rate per Year	Tax Generated (USD)	Tax Utilization (USD)
count	100.000000	1.000000e+02	1.000000e+02
mean	0.224000	4.662574e+08	5.203766e+08
std	0.050292	1.516922e+08	3.431613e+08
min	0.150000	2.047001e+08	1.364668e+08
25%	0.200000	3.453703e+08	2.569185e+08
50%	0.220000	4.604938e+08	4.747857e+08
75%	0.250000	5.742991e+08	5.778559e+08
max	0.300000	7.831351e+08	1.381481e+09

In [ ]:

```python
X3 = df3[['Tax Rate per Year']]
y3 = df3['Tax Generated (USD)']

X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.2, random_state=42)

model3 = LinearRegression()
model3.fit(X3_train, y3_train)

y3_pred = model3.predict(X3_test)

print("Predictions:", y3_pred)
print("Actual values:", y3_test.values)
```

```
Predictions: [4.14599333e+08 6.38604270e+08 3.02596864e+08 5.26601801e+08
 5.26601801e+08 4.59400320e+08 4.14599333e+08 4.14599333e+08
 3.02596864e+08 4.14599333e+08 4.59400320e+08 3.02596864e+08
 6.38604270e+08 6.38604270e+08 3.02596864e+08 5.26601801e+08
 4.59400320e+08 4.59400320e+08 6.38604270e+08 3.02596864e+08]
Actual values: [4.89077689e+08 5.05366727e+08 3.45370333e+08 4.21138940e+08
 4.21138940e+08 5.74299065e+08 5.22090059e+08 4.89077689e+08
 2.04700144e+08 2.72933526e+08 3.00226878e+08 3.91567544e+08
 6.90740666e+08 7.83135089e+08 3.66808267e+08 3.41166907e+08
 5.06543155e+08 5.06543155e+08 4.09400289e+08 3.91567544e+08]
```

In [3]:

```python
import pandas as pd
import plotly.express as px

df1 = pd.read_csv('gcc_oil_export_data.csv')

fig1 = px.scatter(df1, x="Export Volume (barrels)", y="Price per Barrel (USD)", size="Export Revenue (USD)", trendline="ols")
fig1.show()
```

In [4]:

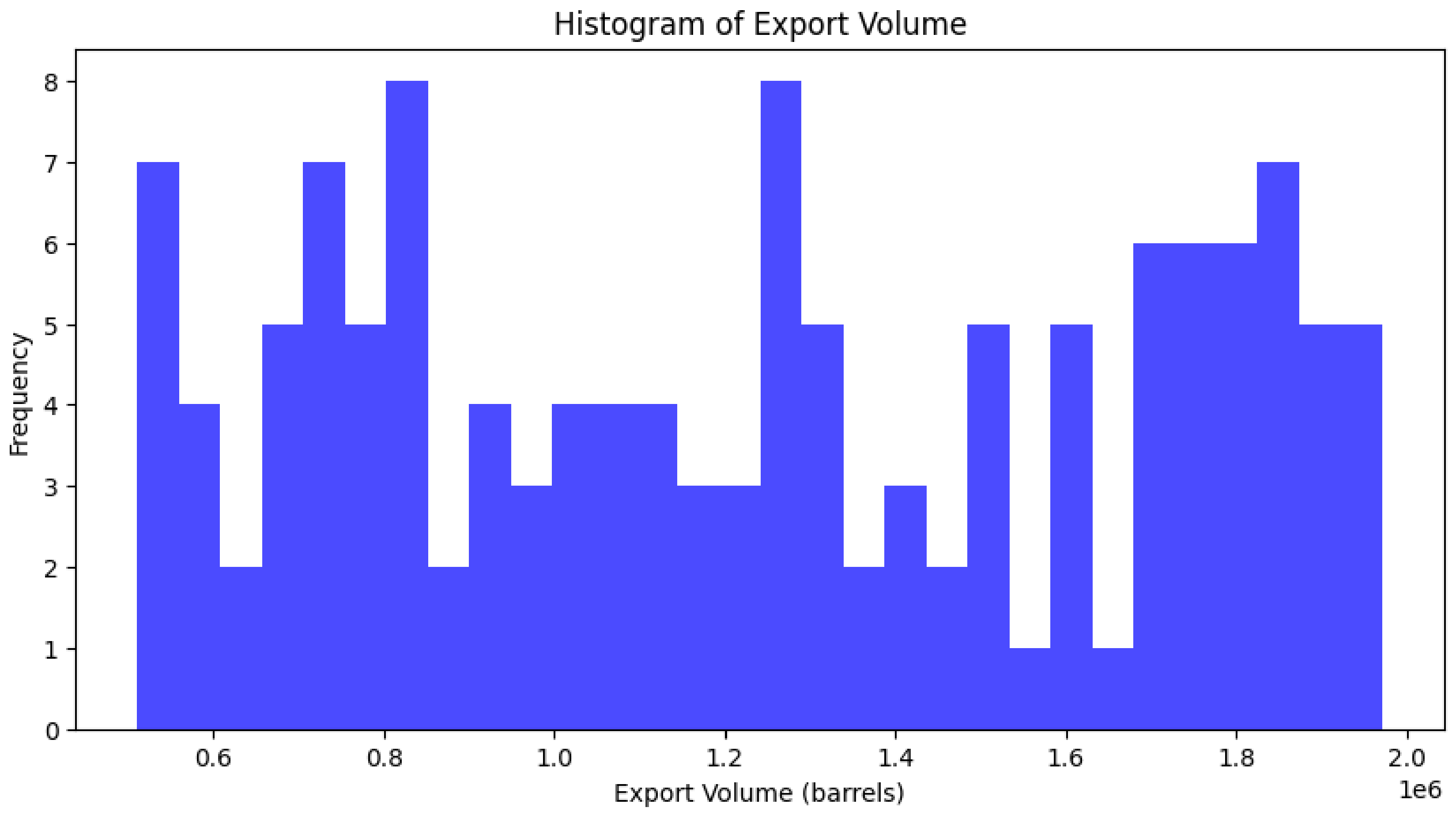
```python
df2 = pd.read_csv('revenue_utilization.csv')

fig2 = px.scatter(df2, x="Infrastructure", y="Healthcare", size="Education", trendline="ols")
fig2.show()
```

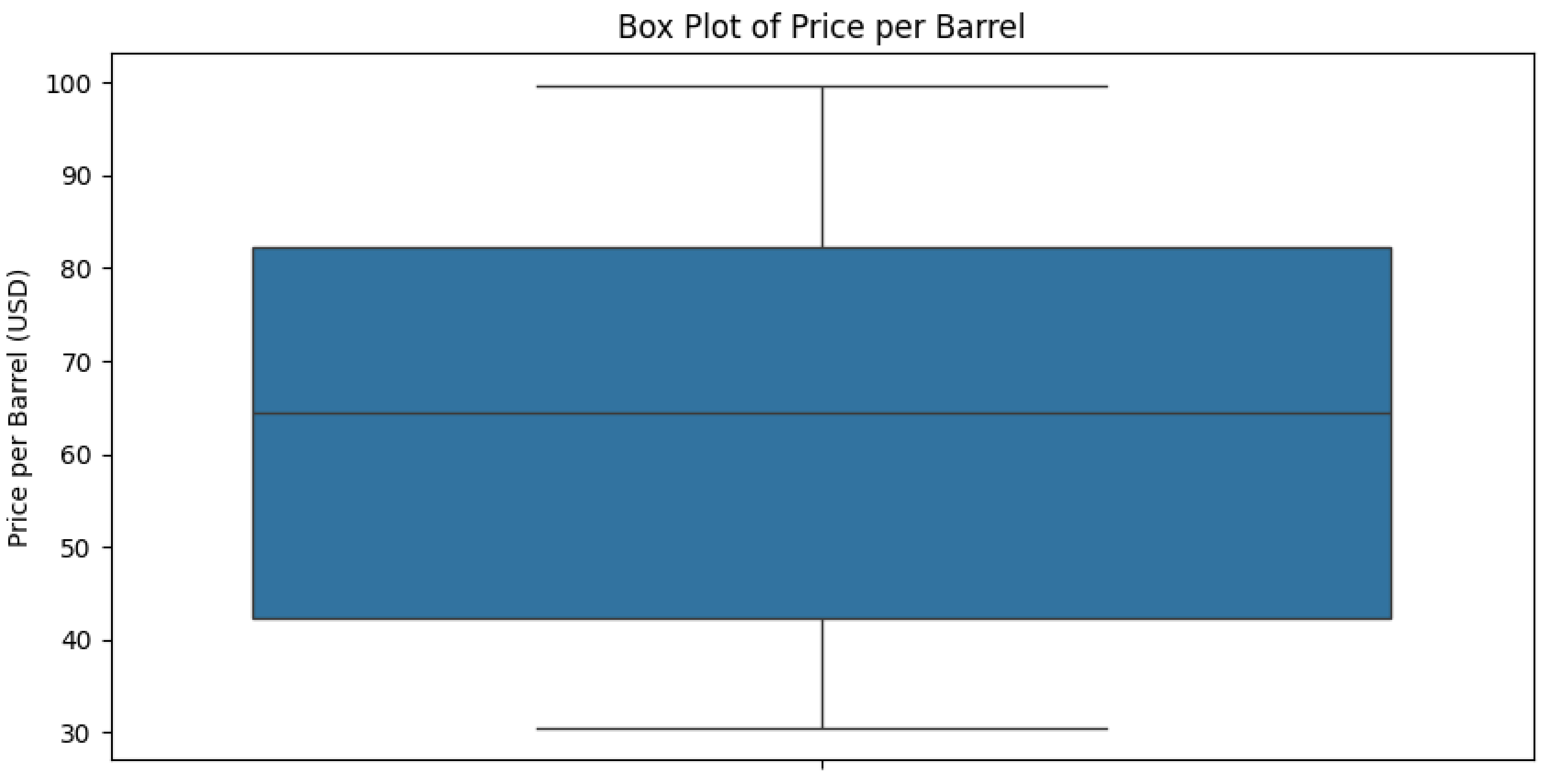
In [5]:

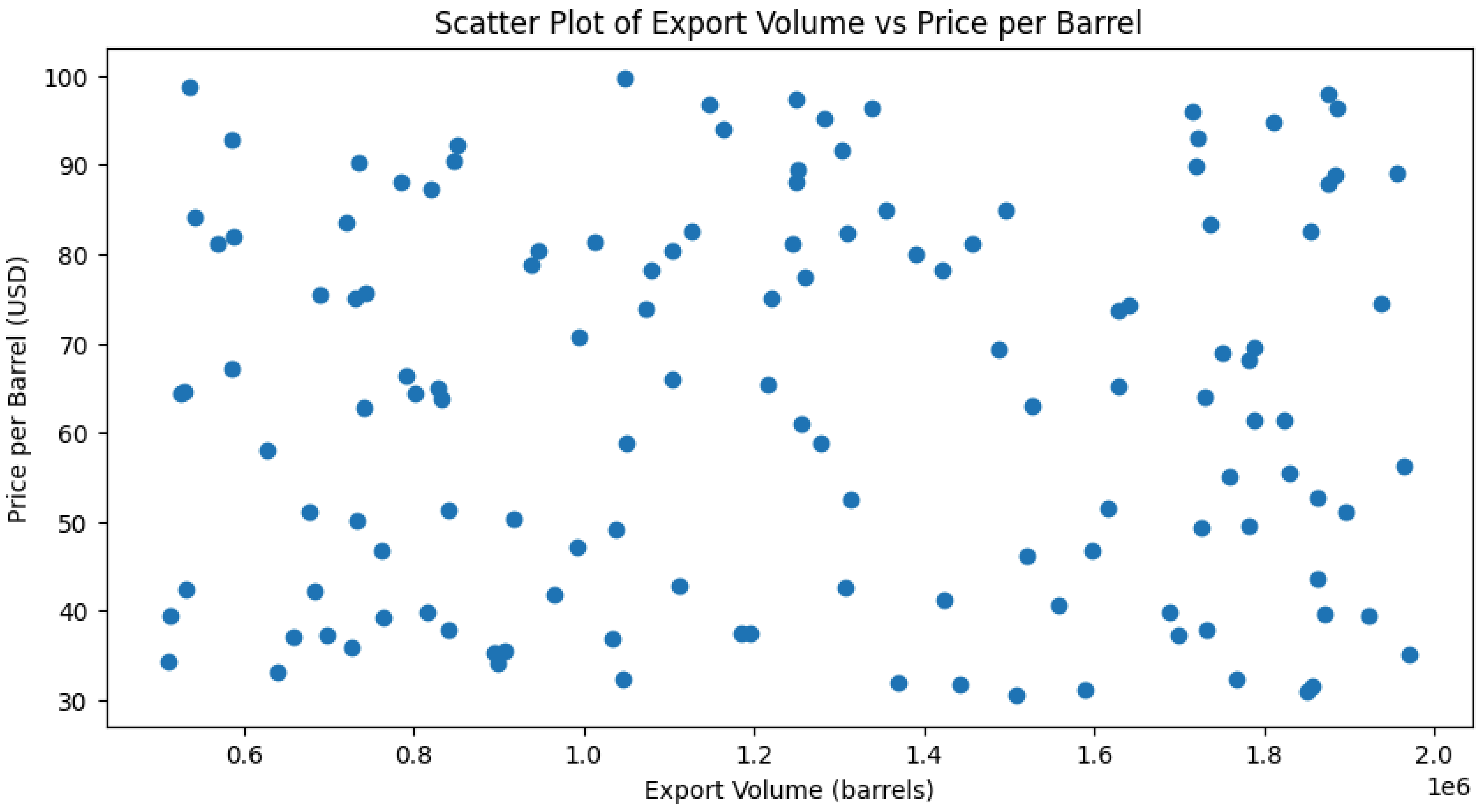
```python
df3 = pd.read_csv('tax_utilization_modified.csv')


fig3 = px.scatter(df3, x="Tax Rate per Year", y="Tax Generated (USD)", size="Tax Generated (USD)", trendline="ols")
fig3.show()
```

In [ ]:

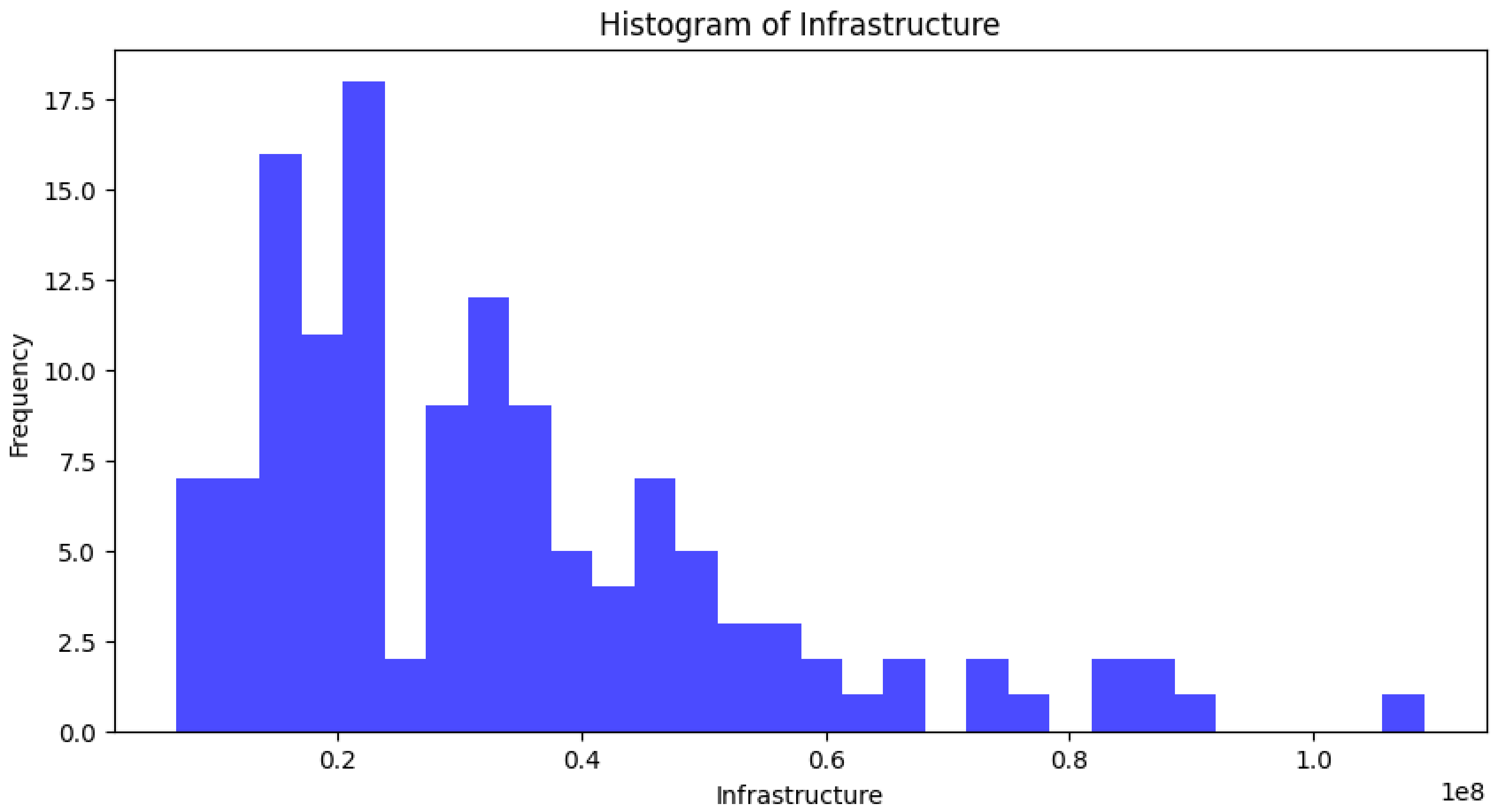
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df1 = pd.read_csv('gcc_oil_export_data.csv')

# Histogram
plt.figure(figsize=(10, 5))
plt.hist(df1['Export Volume (barrels)'], bins=30, color='blue', alpha=0.7)
plt.title('Histogram of Export Volume')
plt.xlabel('Export Volume (barrels)')
plt.ylabel('Frequency')
plt.show()

# Box plot
plt.figure(figsize=(10, 5))
sns.boxplot(df1['Price per Barrel (USD)'])
plt.title('Box Plot of Price per Barrel')
plt.show()

# Scatter plot
plt.figure(figsize=(10, 5))
plt.scatter(df1['Export Volume (barrels)'], df1['Price per Barrel (USD)'])
plt.title('Scatter Plot of Export Volume vs Price per Barrel')
plt.xlabel('Export Volume (barrels)')
plt.ylabel('Price per Barrel (USD)')
plt.show()
```

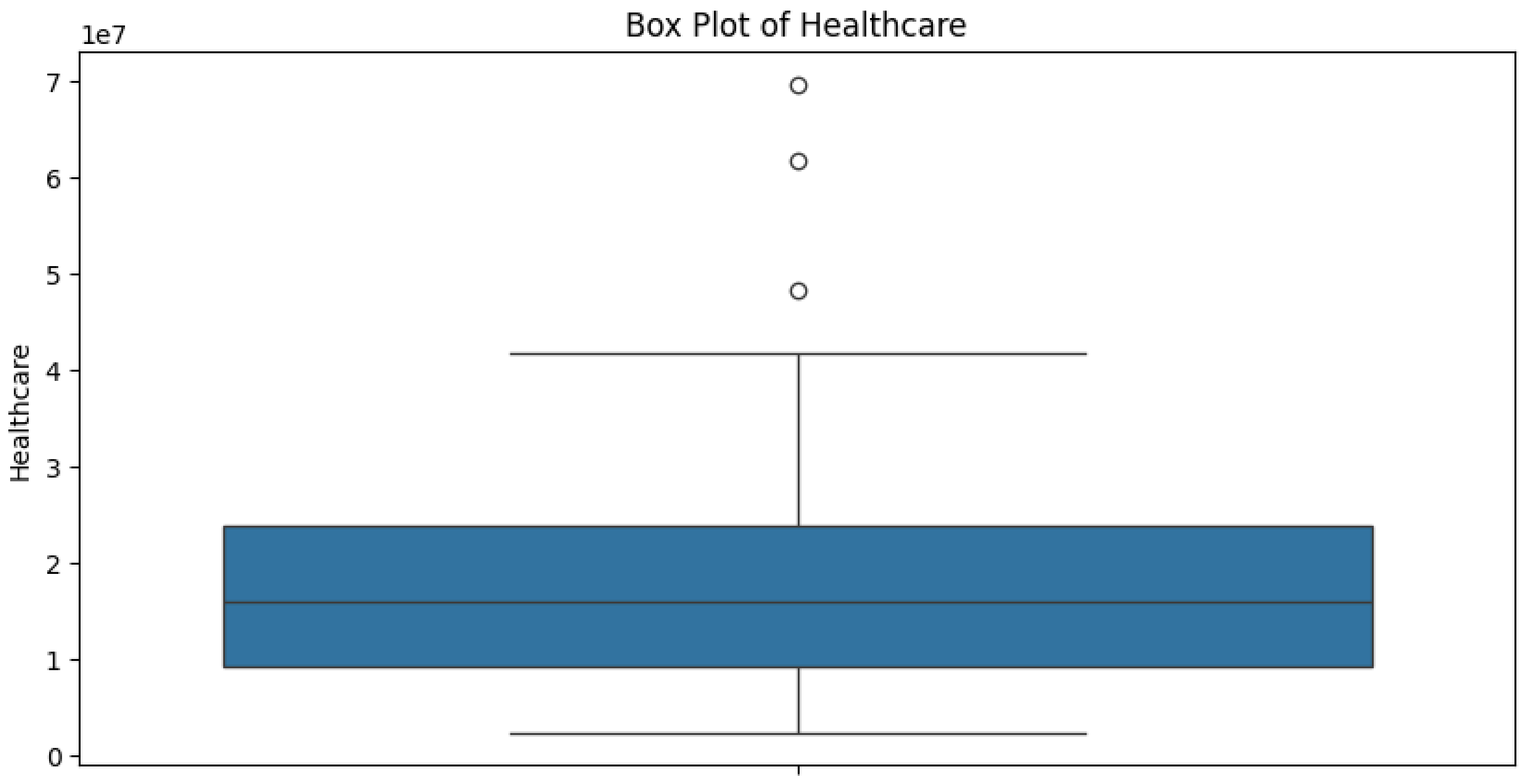


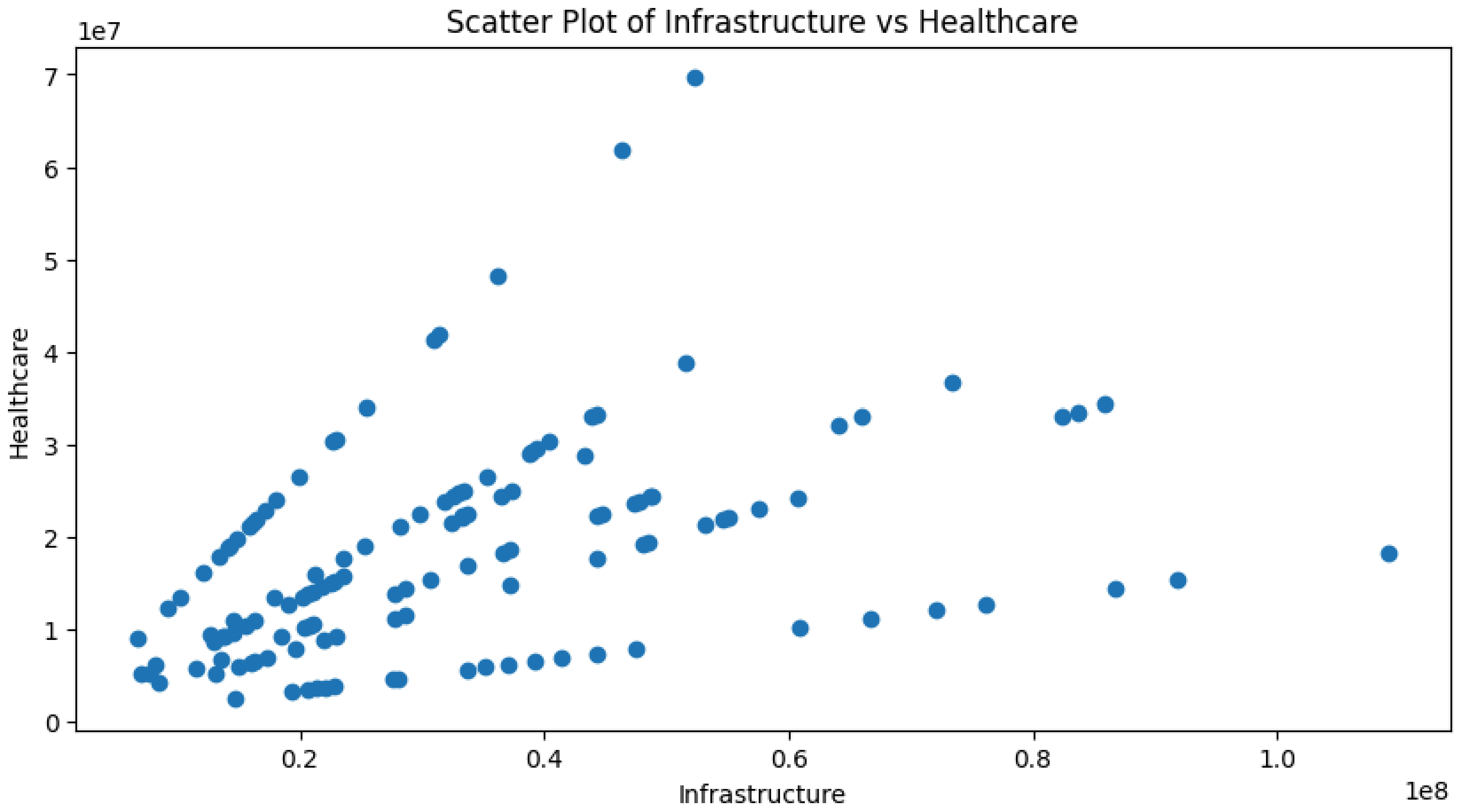




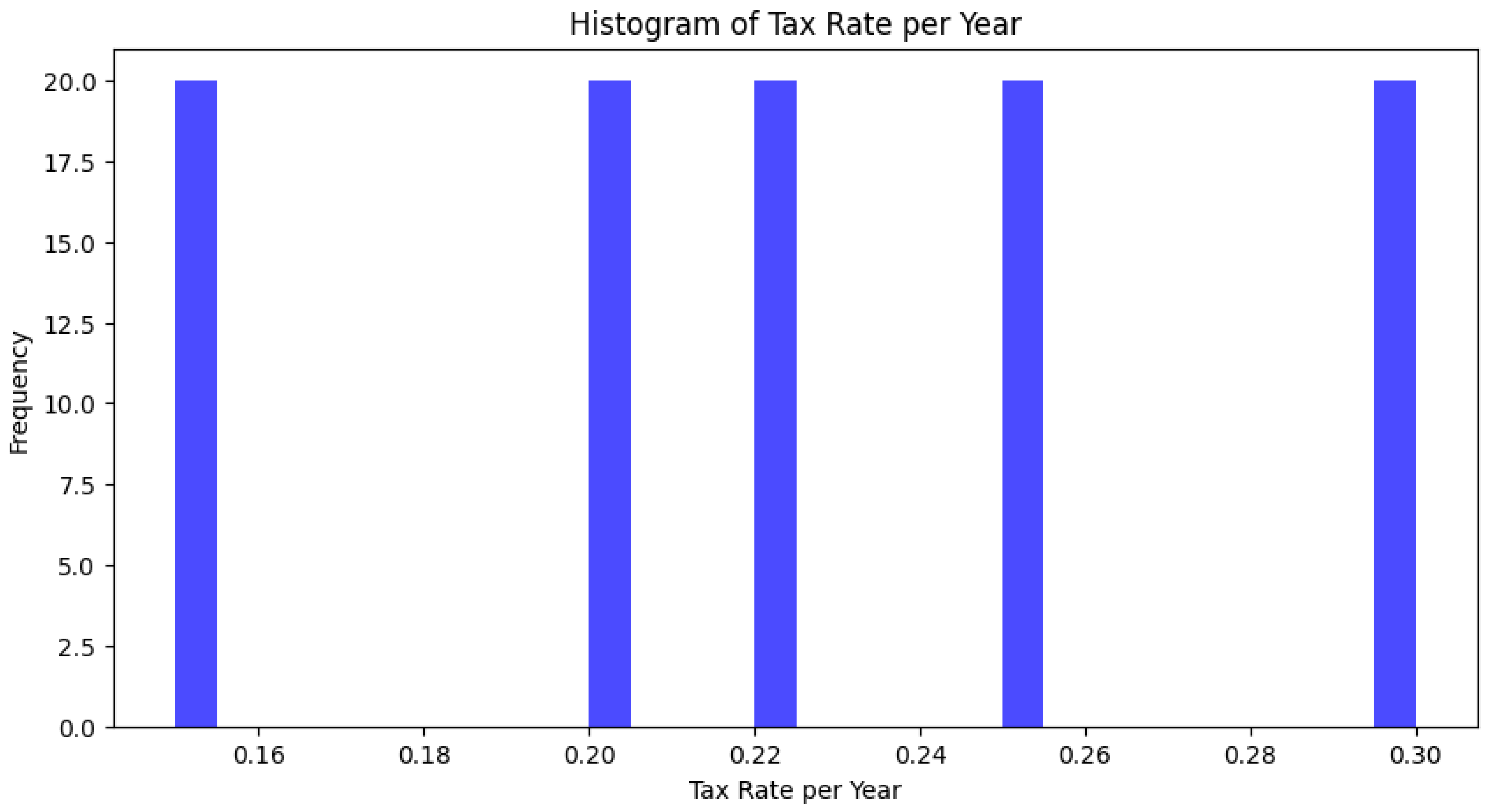
In [ ]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

df2 = pd.read_csv('revenue_utilization.csv')

# Histogram
plt.figure(figsize=(10, 5))
plt.hist(df2['Infrastructure'], bins=30, color='blue', alpha=0.7)
plt.title('Histogram of Infrastructure')
plt.xlabel('Infrastructure')
plt.ylabel('Frequency')
plt.show()

# Box plot
plt.figure(figsize=(10, 5))
sns.boxplot(df2['Healthcare'])
plt.title('Box Plot of Healthcare')
plt.show()

# Scatter plot
plt.figure(figsize=(10, 5))
plt.scatter(df2['Infrastructure'], df2['Healthcare'])
plt.title('Scatter Plot of Infrastructure vs Healthcare')
plt.xlabel('Infrastructure')
plt.ylabel('Healthcare')
plt.show()
```

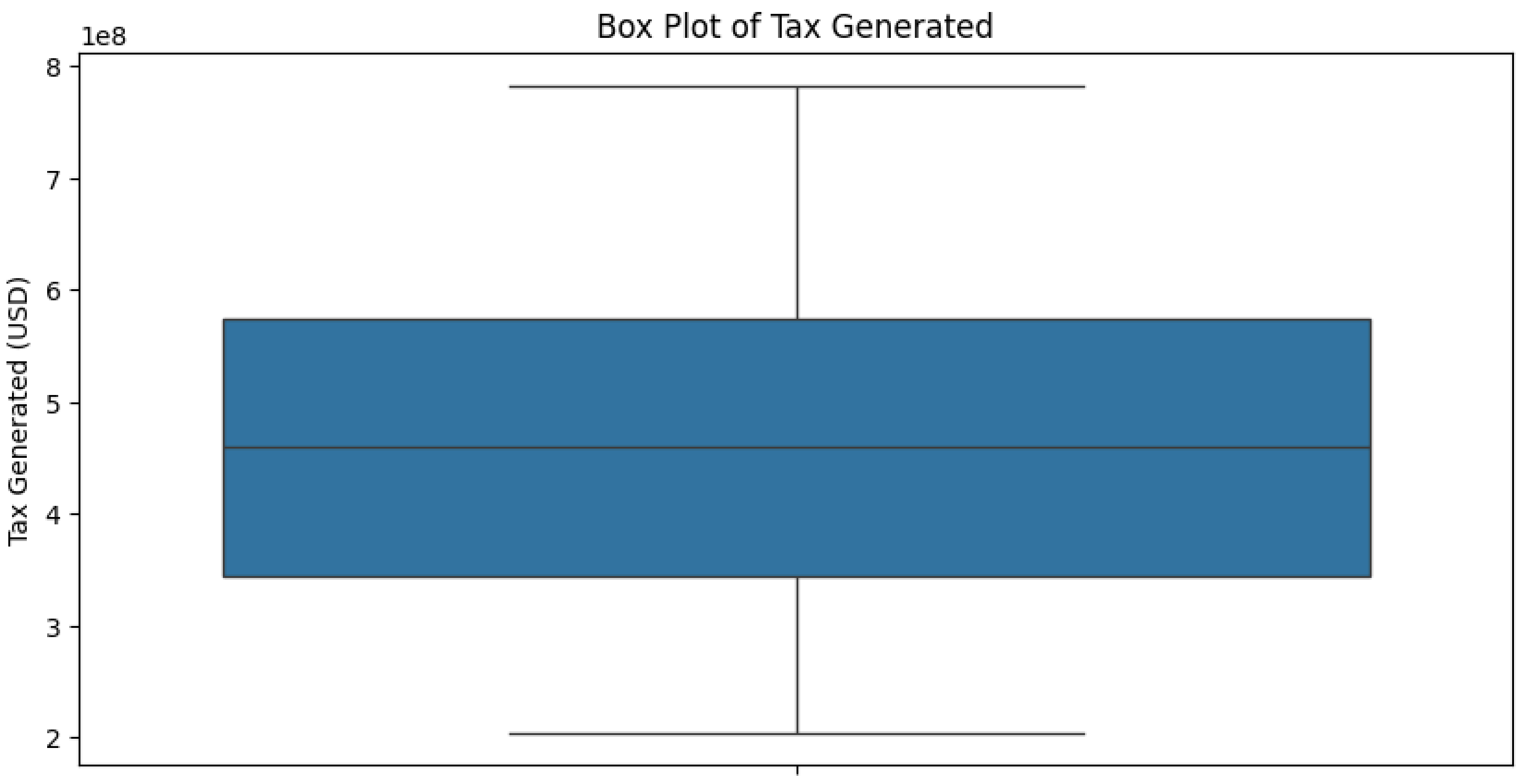


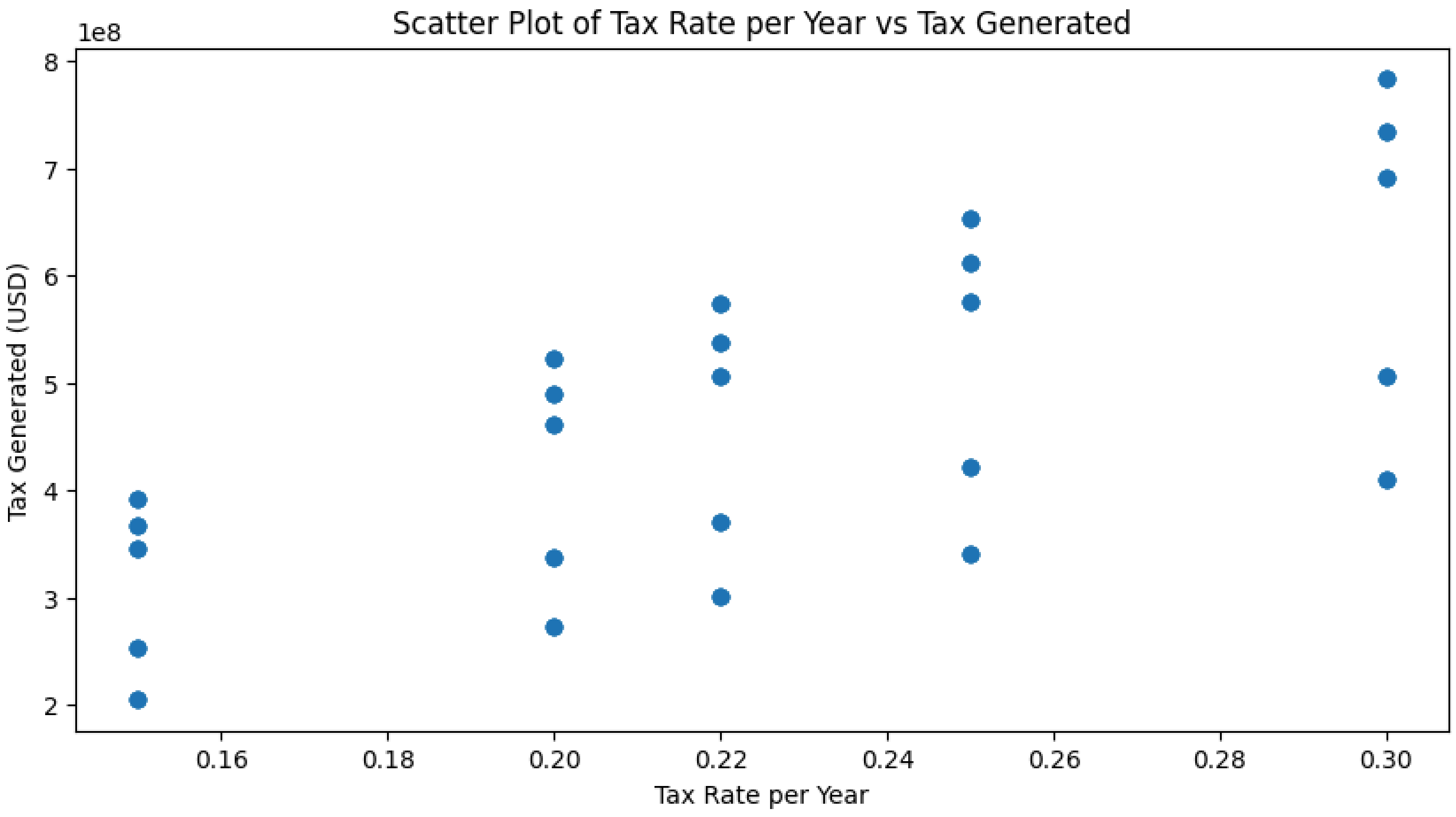




In [ ]:
```
df3 = pd.read_csv('tax_utilization_modified.csv')

# Histogram
plt.figure(figsize=(10, 5))
plt.hist(df3['Tax Rate per Year'], bins=30, color='blue', alpha=0.7)
plt.title('Histogram of Tax Rate per Year')
plt.xlabel('Tax Rate per Year')
plt.ylabel('Frequency')
plt.show()

# Box plot
plt.figure(figsize=(10, 5))
sns.boxplot(df3['Tax Generated (USD)'])
plt.title('Box Plot of Tax Generated')
plt.show()

# Scatter plot
plt.figure(figsize=(10, 5))
plt.scatter(df3['Tax Rate per Year'], df3['Tax Generated (USD)'])
plt.title('Scatter Plot of Tax Rate per Year vs Tax Generated')
plt.xlabel('Tax Rate per Year')
plt.ylabel('Tax Generated (USD)')
plt.show()
```

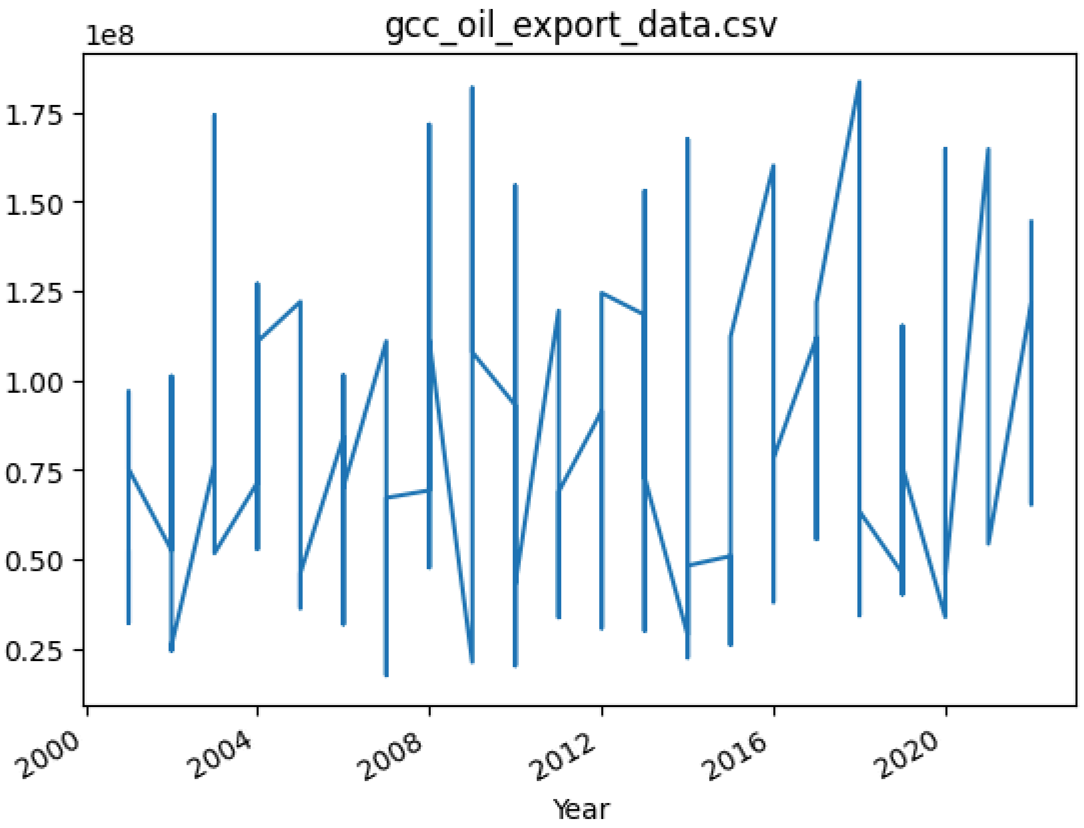






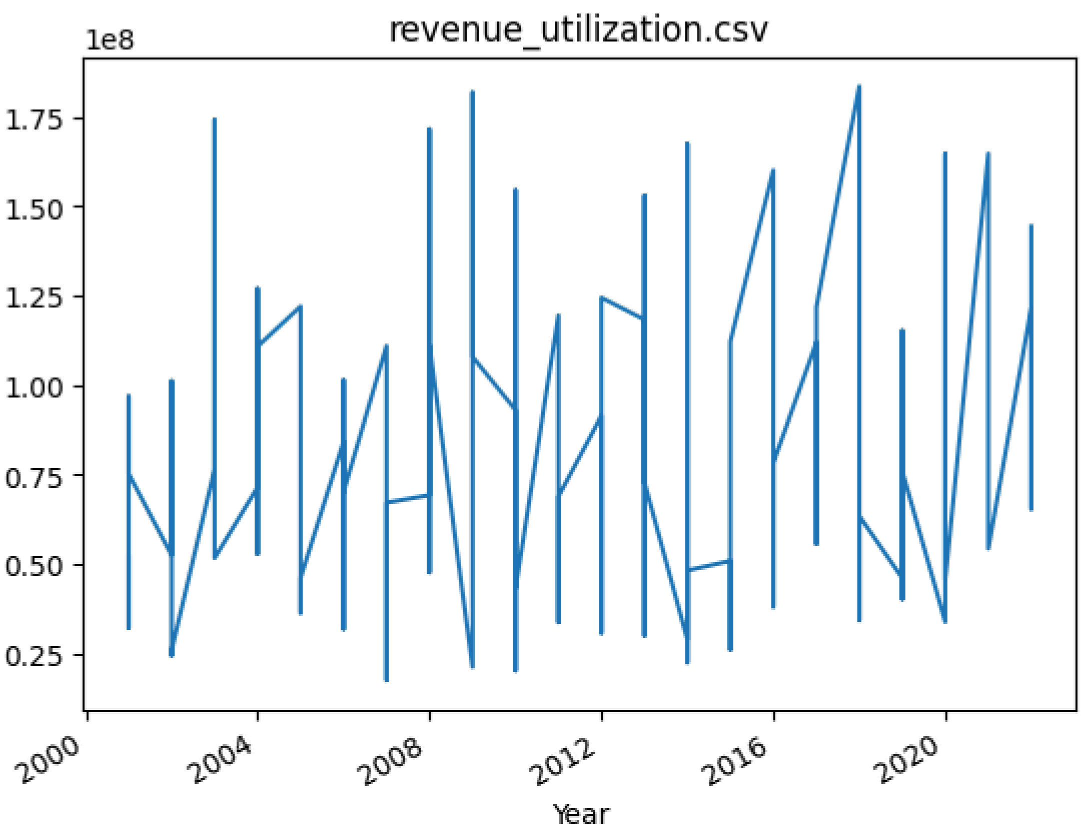
In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt

df1 = pd.read_csv('gcc_oil_export_data.csv')

# Convert the 'Year' column to datetime format
df1['Year'] = pd.to_datetime(df1['Year'], format='%Y')

# Set the 'Year' column as the index
df1.set_index('Year', inplace=True)

# Plot the data
df1['Export Revenue (USD)'].plot()
plt.title('gcc_oil_export_data.csv')
plt.show()
```



In [ ]:

```
df2 = pd.read_csv('revenue_utilization.csv')

# Convert the 'Year' column to datetime format
df2['Year'] = pd.to_datetime(df2['Year'], format='%Y')

# Set the 'Year' column as the index
df2.set_index('Year', inplace=True)

# Plot the data
df2['Export Revenue (USD)'].plot()
plt.title('revenue_utilization.csv')
plt.show()
```



In [4]:

```
import pandas as pd
from sklearn.linear_model import LinearRegression

data = pd.read_csv('gcc_oil_export_data.csv')

X = data['Year'].values.reshape(-1,1)
y = data['Export Revenue (USD)']

# Create a Linear Regression model and fit it to the data
model = LinearRegression()
model.fit(X, y)

# Predict the Export Revenue (USD) for the years 2023, 2024, and 2025
years = pd.DataFrame([2023, 2024, 2025], columns=['Year'])
predictions = model.predict(years)

print("Predicted Export Revenue (USD) for the years 2023, 2024, and 2025:")
for year, prediction in zip(years['Year'], predictions):
    print(f"Year {year}: {prediction}")
```

```
Predicted Export Revenue (USD) for the years 2023, 2024, and 2025:
Year 2023: 87634386.32145023
Year 2024: 88398686.70275545
Year 2025: 89162987.08406067
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted without feature names
  warnings.warn(
```

In [6]:

```python
import pandas as pd
from sklearn.linear_model import LinearRegression

data = pd.read_csv('revenue_utilization.csv')

model = LinearRegression()

years = pd.DataFrame([2023, 2024, 2025], columns=['Year'])

# List of columns to predict
columns_to_predict = ['Infrastructure', 'Healthcare', 'Education', 'Other']

for column in columns_to_predict:
    X = data['Year'].values.reshape(-1,1)
    y = data[column]

    model.fit(X, y)

    # Predict the column for the years 2023, 2024, and 2025
    predictions = model.predict(years)

    print(f"Predicted {column} for the years 2023, 2024, and 2025:")
    for year, prediction in zip(years['Year'], predictions):
        print(f"Year {year}: {prediction}")
```

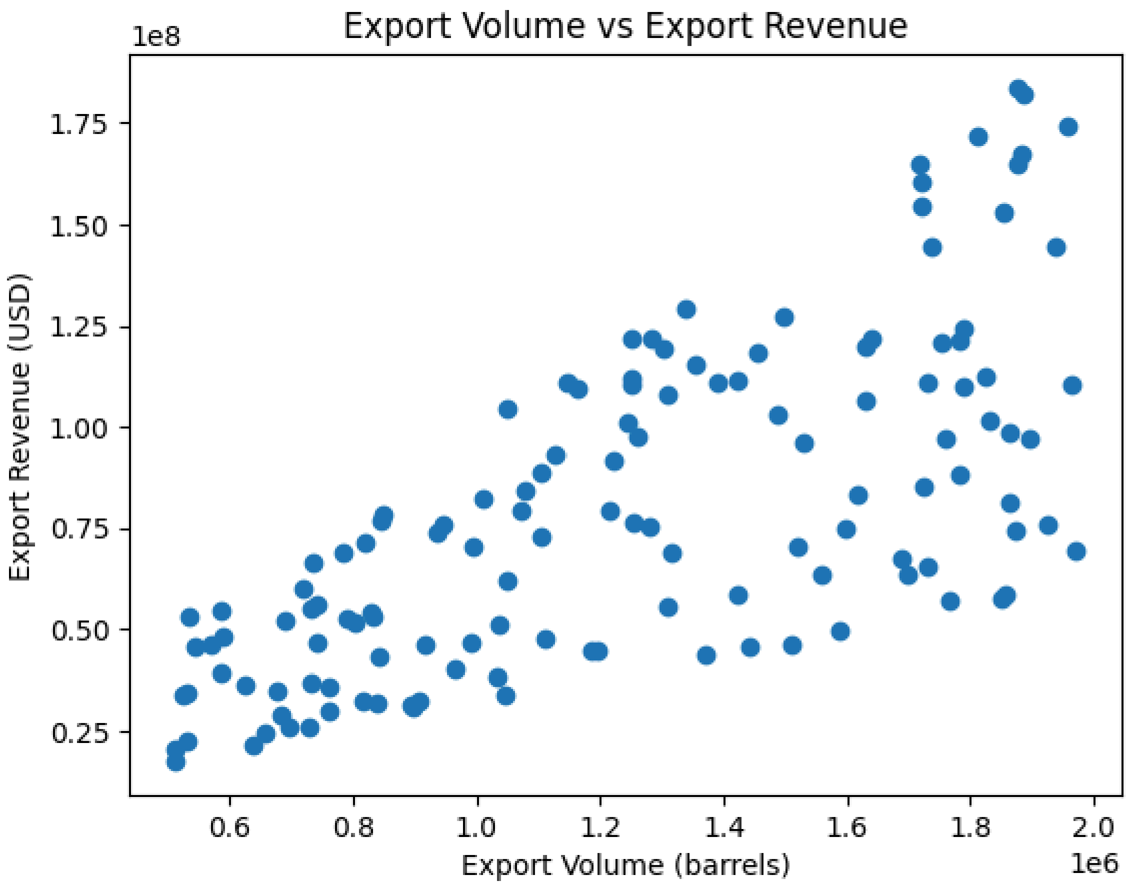
```
Predicted Infrastructure for the years 2023, 2024, and 2025:
Year 2023: 36235102.169761896
Year 2024: 36515822.656440735
Year 2025: 36796543.14311969
Predicted Healthcare for the years 2023, 2024, and 2025:
Year 2023: 20108806.492099524
Year 2024: 20285769.450674713
Year 2025: 20462732.4092499
Predicted Education for the years 2023, 2024, and 2025:
Year 2023: 16872565.83989179
Year 2024: 17069483.41818279
Year 2025: 17266400.99647379
Predicted Other for the years 2023, 2024, and 2025:
Year 2023: 14417911.821212143
Year 2024: 14527611.179090917
Year 2025: 14637310.536969721
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted without feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted without feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted without feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted without feature names
  warnings.warn(
```

In [ ]:

In [ ]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error

gcc_oil_export_data = pd.read_csv("gcc_oil_export_data.csv")
revenue_utilization = pd.read_csv("revenue_utilization.csv")

merged_data = pd.merge(gcc_oil_export_data, revenue_utilization, on=["Country", "Year"])

merged_data.dropna(inplace=True)

# Exploratory Data Analysis (EDA)
# Visualize the relationship between variables
plt.scatter(merged_data["Export Volume (barrels)"], merged_data["Export Revenue (USD)_x"])
plt.xlabel("Export Volume (barrels)")
plt.ylabel("Export Revenue (USD)")
plt.title("Export Volume vs Export Revenue")
plt.show()

# Statistical Analysis
# Calculate correlation coefficient between Export Volume and Export Revenue
correlation = merged_data["Export Volume (barrels)"].corr(merged_data["Export Revenue (USD)_x"])
print("Correlation between Export Volume and Export Revenue:", correlation)

# Forecasting (Example using ARIMA model for simplicity)
# Prepare data for time series analysis
time_series_data = merged_data.groupby("Year")["Export Volume (barrels)"].sum().values

# Split data into train and test sets
train_size = int(len(time_series_data) * 0.8)
train_data, test_data = time_series_data[:train_size], time_series_data[train_size:]

# Define ARIMA model
model = ARIMA(train_data, order=(5,1,0))
model_fit = model.fit()

# Make predictions
predictions = model_fit.forecast(steps=len(test_data))[0]

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(test_data, predictions))
print("Root Mean Squared Error (RMSE) of ARIMA model:", rmse)

plt.plot(test_data, label='Actual')
plt.plot(predictions, color='red', label='Predicted')
plt.xlabel("Year")
plt.ylabel("Export Volume (barrels)")
plt.title("Actual vs Predicted Export Volume")
plt.legend()
plt.show()
```




```
Correlation between Export Volume and Export Revenue: 0.7038808170270694
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-4817c5986c29> in <cell line: 47>()
     45 
     46 # Calculate RMSE
---> 47 rmse = np.sqrt(mean_squared_error(test_data, predictions))
     48 print("Root Mean Squared Error (RMSE) of ARIMA model:", rmse)
     49 

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py in mean_squared_error(y_true, y_pred, sample_weight, multioutput, squared)
    440     0.825...
    441     """
--> 442     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    443         y_true, y_pred, multioutput
    444     )

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py in _check_reg_targets(y_true, y_pred, multioutput, dtype)
     98         correct keyword.
     99     """
--> 100     check_consistent_length(y_true, y_pred)
    101     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
    102     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in check_consistent_length(*arrays)
    392     """
    393 
--> 394     lengths = [_num_samples(X) for X in arrays if X is not None]
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in <listcomp>(.0)
    392     """
    393 
--> 394     lengths = [_num_samples(X) for X in arrays if X is not None]
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _num_samples(x)
    333     if hasattr(x, "shape") and x.shape is not None:
    334         if len(x.shape) == 0:
--> 335             raise TypeError(
    336                 "Singleton array %r cannot be considered a valid collection." % x
    337             )

TypeError: Singleton array 7834611.449160476 cannot be considered a valid collection.
```