```
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
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score



# Load datasets
tourism_data = pd.read_excel("/content/tourisam.xlsx")
seasonal_variation = pd.read_excel("/content/Seasonal Variation in Tourist Traffic - 2023.xlsx")
ticket_revenue = pd.read_excel("/content/Revenue from Sale of Tickets.xlsx")


def check_missing_values(df, name):
    print(f"\nMissing Values in {name}:")
    print(df.isnull().sum())


# Check for missing values before handling
check_missing_values(ticket_revenue, "Ticket Revenue")
check_missing_values(tourism_data, "Tourism Data")
check_missing_values(seasonal_variation, "Seasonal Variation Data")
```

```
Missing Values in Ticket Revenue:
YEAR                  0
NUMBER OF TOURISTS    0
REVENUE               0
dtype: int64

Missing Values in Tourism Data:
Country of Residence      0
2022(Tourist Arrivals)    0
2023(Tourist Arrivals)    0
Gap(2022-2023)            0
By Air                    0
By sea                    0
Visit for vacation        0
Visit for Busineess       0
Visit for education       0
Female                    0
Male                      0
dtype: int64

Missing Values in Seasonal Variation Data:
Month                0
Number of tourist    0
dtype: int64
```

```
# Handling missing values
for df in [ticket_revenue, tourism_data, seasonal_variation]:
    df.replace(["NO DATA", "NODATA"], np.nan, inplace=True)
    df.fillna(df.median(numeric_only=True), inplace=True)
```

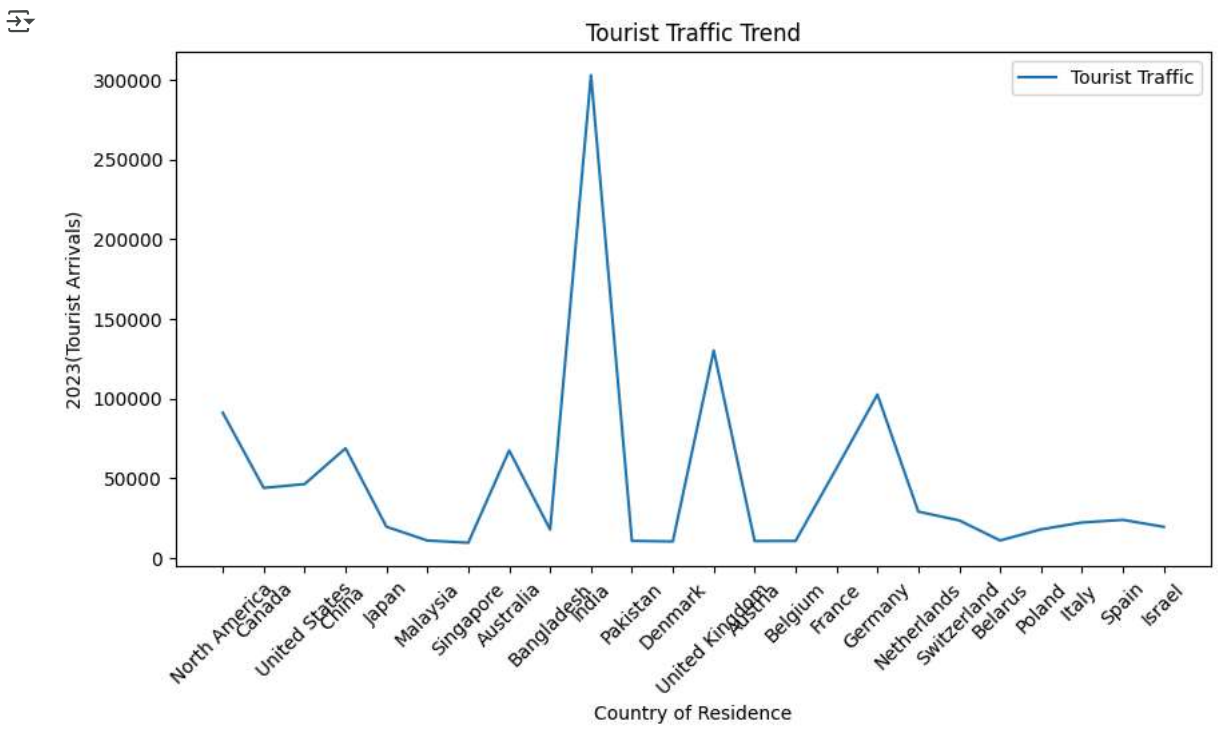
```
<ipython-input-3-a6352413ab8d>:3: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future versior
  df.replace(["NO DATA", "NODATA"], np.nan, inplace=True)
```

```
# Display first few rows
print("Tourism Data:")
print(tourism_data.head())
print("\nSeasonal Variation:")
print(seasonal_variation.head())
print("\nTicket Revenue:")
print(ticket_revenue.head())
```

```
Tourism Data:
  Country of Residence  2022(Tourist Arrivals)  2023(Tourist Arrivals)  \
0        North America                   49409                   91080
1               Canada                   26845                   43944
2        United States                   22230                   46344
3               China                    4715                   68789
4                Japan                    3087                   19583

   Gap(2022-2023)  By Air  By sea  Visit for vacation  Visit for Busineess  \
```

```
0        0.843389   83815    7089              34872              1835
1        0.636953   42814    1107              10355               354
2        1.084750   40328    5841              23944              1461
3       13.589396   68662     127              57230              3794
4        5.343699   17123    2460              13739              1648

   Visit for education  Female   Male
0                   56   43939  47141
1                    7   21331  22613
2                   48   22170  24174
3                   27   38147  30642
4                   12    9566  10017

Seasonal Variation:
      Month  Number of tourist
0   January             102545
1  February             107639
2     March             125495
3     April             105498
4       May              83309

Ticket Revenue:
   YEAR  NUMBER OF TOURISTS  REVENUE
0  2000            155167.0    276.0
1  2001            129201.0    222.0
2  2002            131804.0    242.8
3  2003            212521.0    403.3
4  2004            246380.0    543.1
```

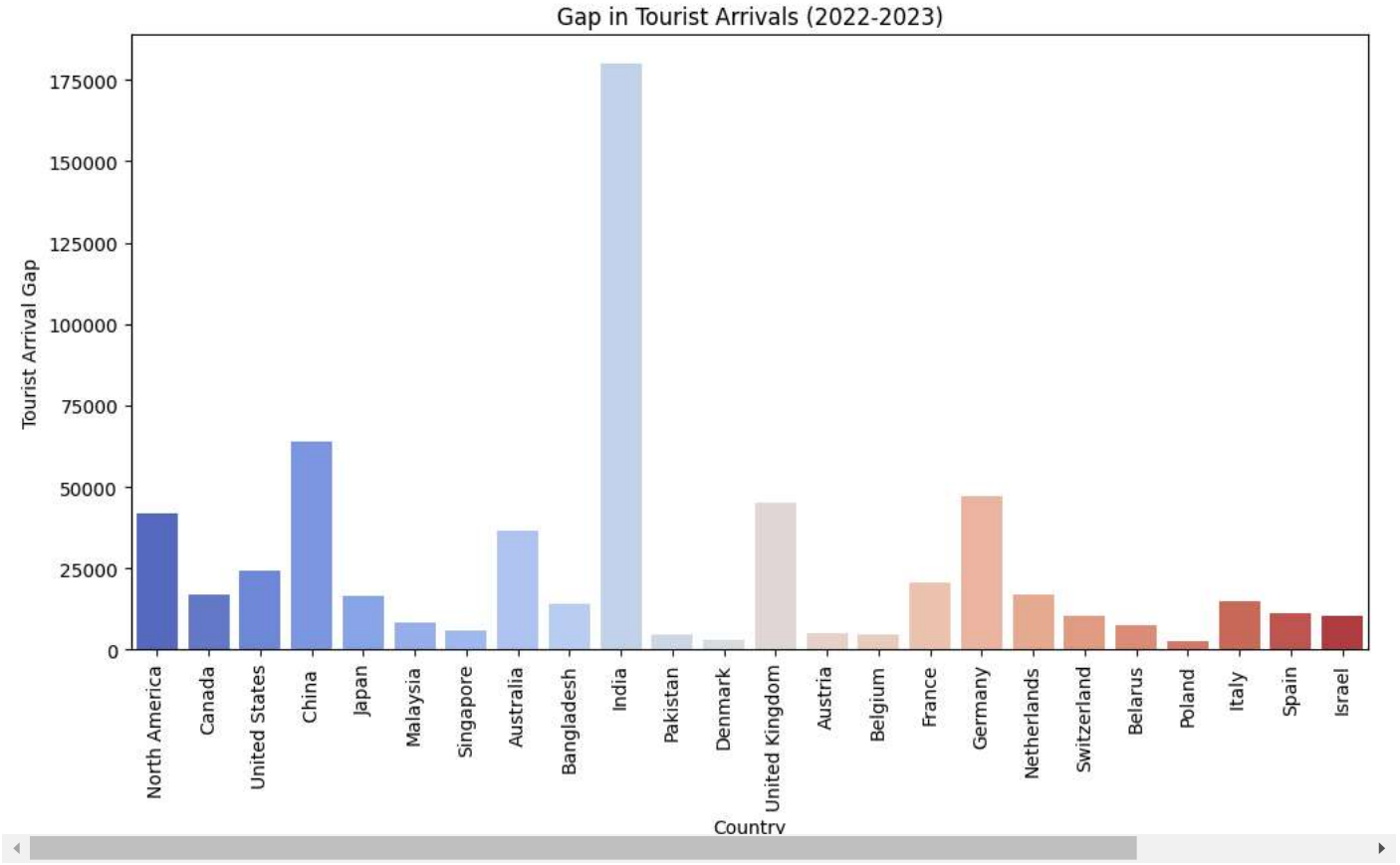
```
plt.figure(figsize=(10,5))
sns.lineplot(data=tourism_data, x='Country of Residence', y='2023(Tourist Arrivals)', label='Tourist Traffic')
plt.xticks(rotation=45)
plt.title("Tourist Traffic Trend")
plt.legend()
plt.show()
```



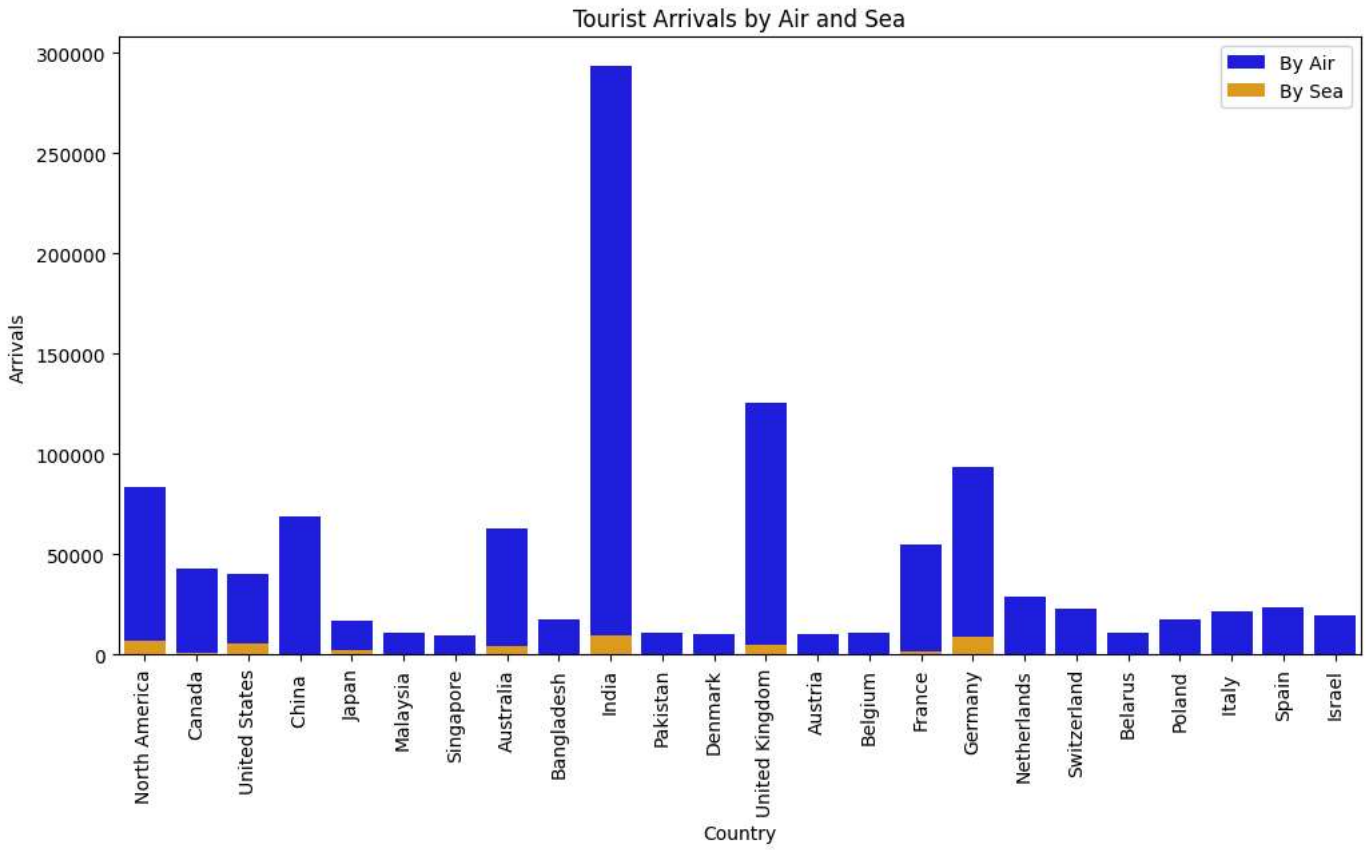

```
# Compute the Gap between 2022 and 2023 arrivals
tourism_data['Gap(2022-2023)'] = tourism_data['2023(Tourist Arrivals)'] - tourism_data['2022(Tourist Arrivals)']
```

```
# Exploratory Data Analysis (EDA)
plt.figure(figsize=(12,6))
sns.barplot(data=tourism_data, x='Country of Residence', y='Gap(2022-2023)', palette='coolwarm')
plt.xticks(rotation=90)
plt.title("Gap in Tourist Arrivals (2022-2023)")
plt.xlabel("Country")
plt.ylabel("Tourist Arrival Gap")
plt.show()
```

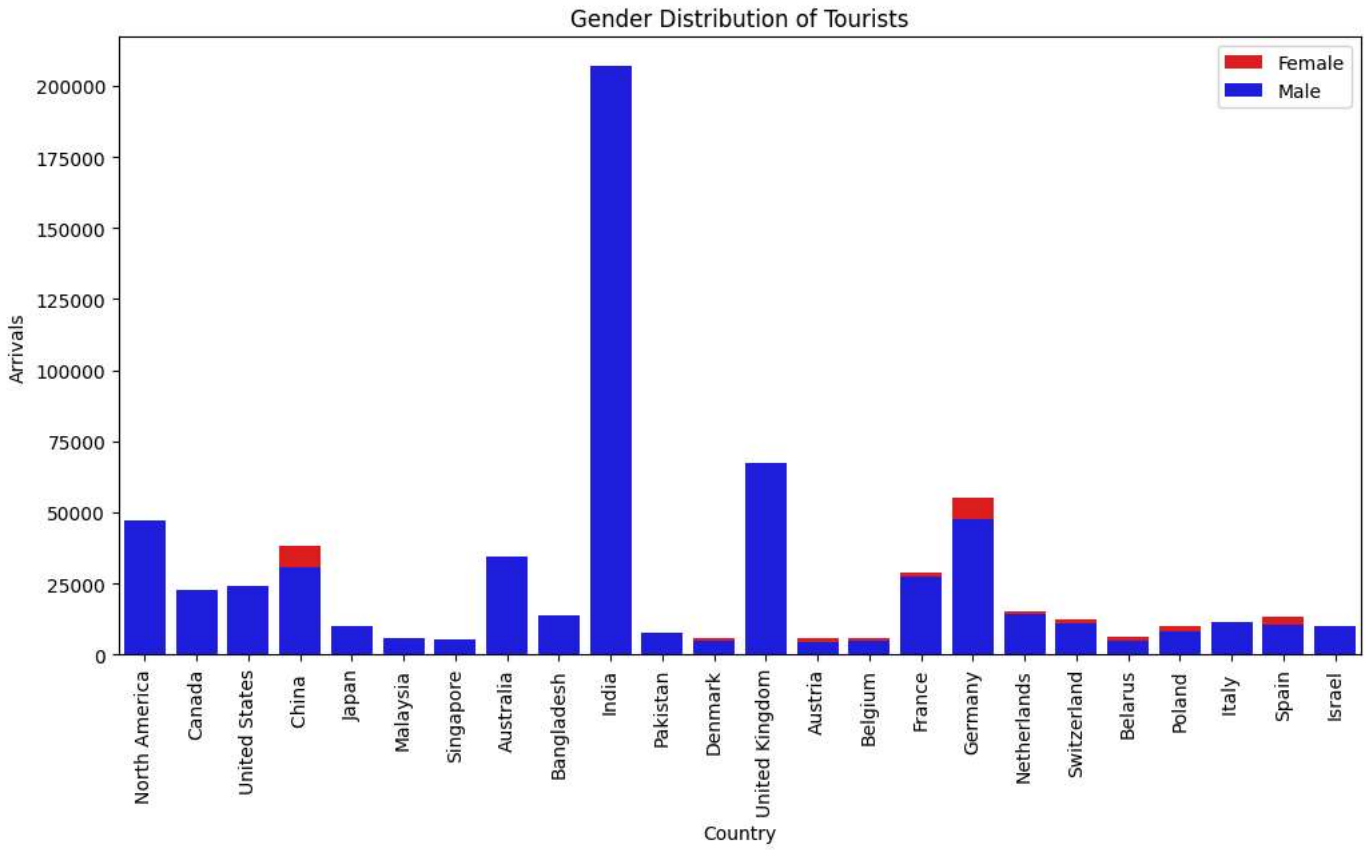
```
<ipython-input-11-30a29b76ff01>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legenc

  sns.barplot(data=tourism_data, x='Country of Residence', y='Gap(2022-2023)', palette='coolwarm')
```




```
plt.figure(figsize=(12,6))
sns.barplot(data=tourism_data, x='Country of Residence', y='By Air', color='blue', label='By Air')
sns.barplot(data=tourism_data, x='Country of Residence', y='By sea', color='orange', label='By Sea')
plt.xticks(rotation=90)
plt.title("Tourist Arrivals by Air and Sea")
plt.xlabel("Country")
plt.ylabel("Arrivals")
plt.legend()
plt.show()
```



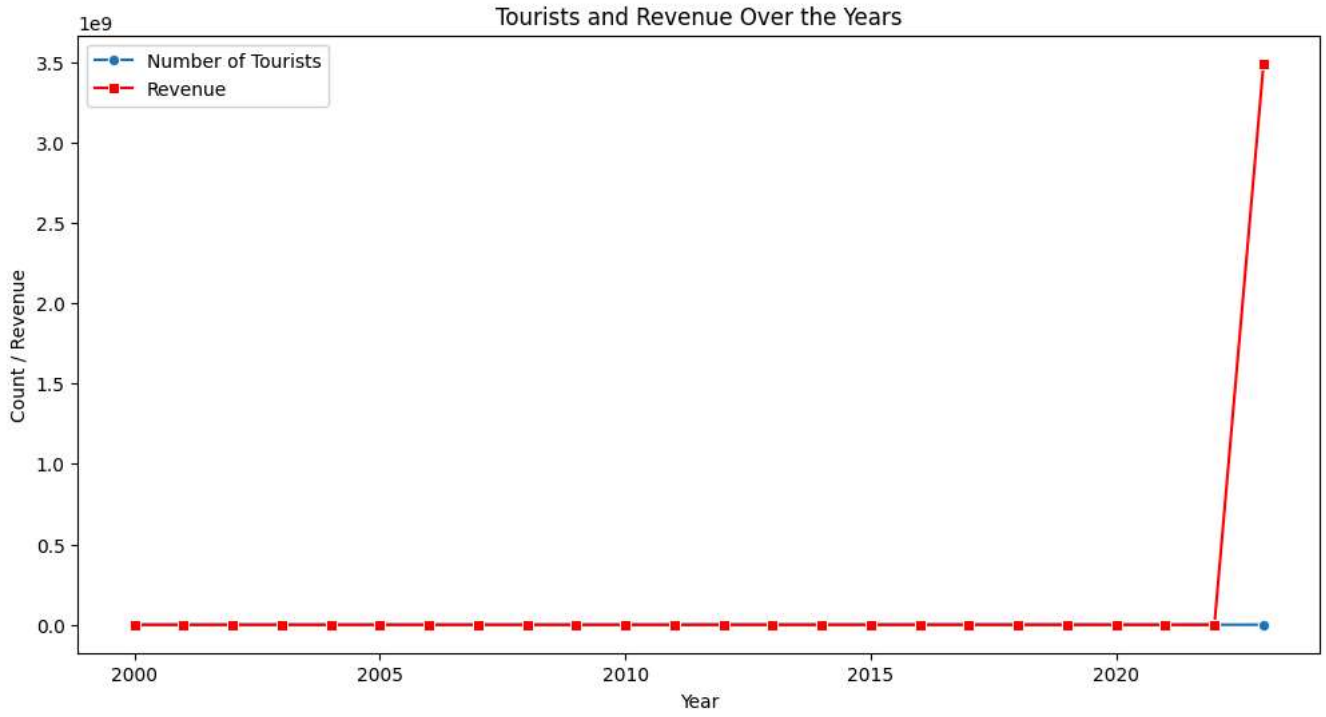

```
plt.figure(figsize=(12,6))
sns.barplot(data=tourism_data, x='Country of Residence', y='Female', color='Red', label='Female')
sns.barplot(data=tourism_data, x='Country of Residence', y='Male', color='blue', label='Male')
plt.xticks(rotation=90)
plt.title("Gender Distribution of Tourists")
plt.xlabel("Country")
plt.ylabel("Arrivals")
plt.legend()
plt.show()
```




```
# Replace variations of missing data indicators with NaN
ticket_revenue.replace(["NO DATA", "NODATA"], np.nan, inplace=True)

# Convert numerical columns to appropriate data types
ticket_revenue['NUMBER OF TOURISTS'] = pd.to_numeric(ticket_revenue['NUMBER OF TOURISTS'], errors='coerce')
ticket_revenue['REVENUE'] = pd.to_numeric(ticket_revenue['REVENUE'], errors='coerce')

# Fill missing values with the median (or choose another strategy)
ticket_revenue.fillna(ticket_revenue.median(), inplace=True)

# Display first few rows
print("Ticket Revenue Data:")
print(ticket_revenue.head())
```

```
Ticket Revenue Data:
   YEAR  NUMBER OF TOURISTS  REVENUE
0  2000            155167.0    276.0
1  2001            129201.0    222.0
2  2002            131804.0    242.8
3  2003            212521.0    403.3
4  2004            246380.0    543.1
<ipython-input-21-f9da923e4500>:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future versic
  ticket_revenue.replace(["NO DATA", "NODATA"], np.nan, inplace=True)
```

```
# Exploratory Data Analysis (EDA)
plt.figure(figsize=(12,6))
sns.lineplot(data=ticket_revenue, x='YEAR', y='NUMBER OF TOURISTS', marker='o', label='Number of Tourists')
sns.lineplot(data=ticket_revenue, x='YEAR', y='REVENUE', marker='s', label='Revenue', color='red')
plt.title("Tourists and Revenue Over the Years")
plt.xlabel("Year")
plt.ylabel("Count / Revenue")
plt.legend()
plt.show()
```




```
# Machine Learning - Predicting Revenue Based on Number of Tourists
X = ticket_revenue[['NUMBER OF TOURISTS']]
y = ticket_revenue['REVENUE']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Use Random Forest Regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Model Evaluation
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MAE: {mae}")
print(f"MSE: {mse}")
print(f"R2 Score: {r2}")

# Plot Actual vs Predicted Revenue
plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred, alpha=0.7, color='blue', label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', color='red', label='Ideal Fit')
plt.xlabel("Actual Revenue")
plt.ylabel("Predicted Revenue")
plt.title("Actual vs Predicted Revenue")
plt.legend()
plt.show()
```

```
MAE: 271.71290000000016
MSE: 148355.3909390502
R2 Score: 0.9446036030312182
```