Olympics Dataset Analysis with Insights

In [99]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("olympics.csv",encoding="ISO-8859-1")
df.head()
```

Out[99]:

	City	Year	Sport	Discipline	Event	Athlete	Gender	Country_Code	Country	Event_gender	Medal
0	Montreal	1976.0	Aquatics	Diving	3m springboard	KÖHLER, Christa	Women	GDR	East Germany	W	Silver
1	Montreal	1976.0	Aquatics	Diving	3m springboard	KOSENKOV, Aleksandr	Men	URS	Soviet Union	M	Bronze
2	Montreal	1976.0	Aquatics	Diving	3m springboard	BOGGS, Philip George	Men	USA	United States	M	Gold
3	Montreal	1976.0	Aquatics	Diving	3m springboard	CAGNOTTO, Giorgio Franco	Men	ITA	Italy	M	Silver
4	Montreal	1976.0	Aquatics	Diving	10m platform	WILSON, Deborah Keplar	Women	USA	United States	W	Bronze

In [101…
```
df.shape
```

Out[101… (15433, 11)

In [103…
```
df.isnull().sum()
```

Out[103…
```
City            117
Year            117
Sport           117
Discipline      117
Event           117
Athlete         117
Gender          117
Country_Code    117
Country         117
Event_gender    117
Medal           117
dtype: int64
```

In [105…
```
num_cols=['Year']
from sklearn.impute import KNNImputer
imputer=KNNImputer(n_neighbors=5)
df[num_cols]=imputer.fit_transform(df[num_cols])
```

In [107…
```
df.isnull().sum()
```

Out[107…
```
City            117
Year              0
Sport           117
Discipline      117
Event           117
Athlete         117
Gender          117
Country_Code    117
Country         117
Event_gender    117
Medal           117
dtype: int64
```

In [109…
```
categorical_cols = ['City', 'Sport', 'Discipline', 'Event', 'Athlete', 'Gender', 'Country_Code', 'Country', 'Event_
df[categorical_cols] = df[categorical_cols].fillna(method='ffill').fillna(method='bfill')
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_10657/3394624300.py:2: FutureWarning: DataFrame.fillna wi
th 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df[categorical_cols] = df[categorical_cols].fillna(method='ffill').fillna(method='bfill')
```

In [111…
```
df.isnull().sum()
```

Out[111…
```
City            0
Year            0
Sport           0
Discipline      0
Event           0
Athlete         0
Gender          0
Country_Code    0
Country         0
Event_gender    0
Medal           0
dtype: int64
```

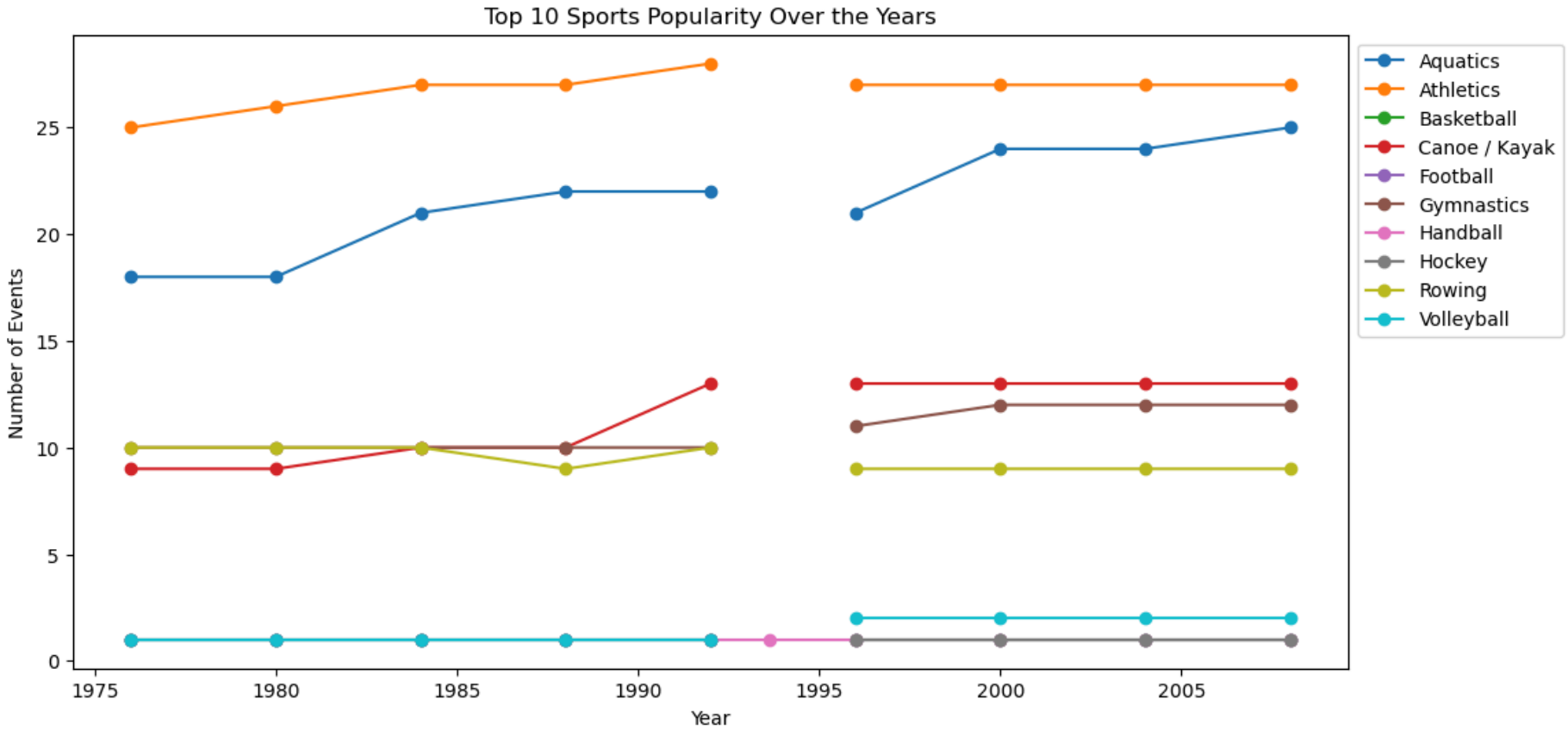
In [113…
```
df.head()
```

Out[113…

	City	Year	Sport	Discipline	Event	Athlete	Gender	Country_Code	Country	Event_gender	Medal
0	Montreal	1976.0	Aquatics	Diving	3m springboard	KÖHLER, Christa	Women	GDR	East Germany	W	Silver
1	Montreal	1976.0	Aquatics	Diving	3m springboard	KOSENKOV, Aleksandr	Men	URS	Soviet Union	M	Bronze
2	Montreal	1976.0	Aquatics	Diving	3m springboard	BOGGS, Philip George	Men	USA	United States	M	Gold
3	Montreal	1976.0	Aquatics	Diving	3m springboard	CAGNOTTO, Giorgio Franco	Men	ITA	Italy	M	Silver
4	Montreal	1976.0	Aquatics	Diving	10m platform	WILSON, Deborah Keplar	Women	USA	United States	W	Bronze

In [115…
```
top_sports = df['Sport'].value_counts().head(10).index
df_top_sports = df[df['Sport'].isin(top_sports)]
sport_trends = df_top_sports.groupby(['Year', 'Sport'])['Event'].nunique().unstack()
sport_trends.plot(figsize=(12, 6), marker='o')
plt.title("Top 10 Sports Popularity Over the Years")
plt.xlabel("Year")
plt.ylabel("Number of Events")
```

```
plt.legend(loc="upper left", bbox_to_anchor=(1,1))
plt.show()
```



Most Popular Sports:

Athletics and Aquatics have consistently had the highest number of events.

Both sports show an increasing trend over the years, indicating their growing importance.

Steady Growth in Certain Sports:

Basketball and Gymnastics have maintained a stable number of events over time.

Canoe/Kayak shows a noticeable increase in events around the early 1990s.

Relatively Stable Sports:

Hockey, Rowing, and Handball have had a constant number of events, with minimal fluctuations.

Emerging Sports:

Volleyball shows a late introduction but maintains steady participation.

Some sports, like Handball, seem to have been introduced in later years.

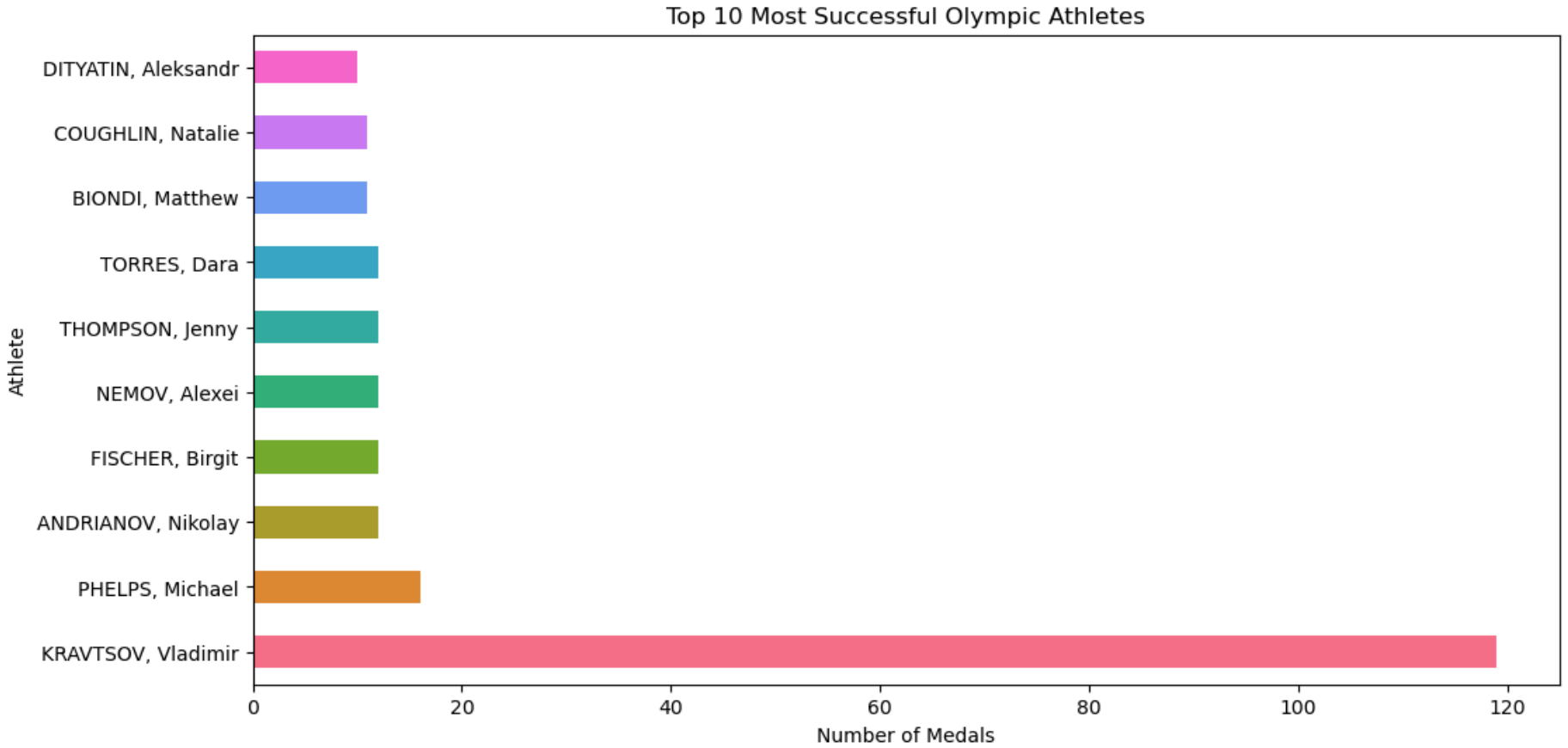
Diversity in Popularity Trends:

Some sports experienced rapid growth (e.g., Aquatics and Athletics), while others remained stable.

This trend suggests changing preferences in sports events over the years, possibly due to evolving audience interests or global sports policies.

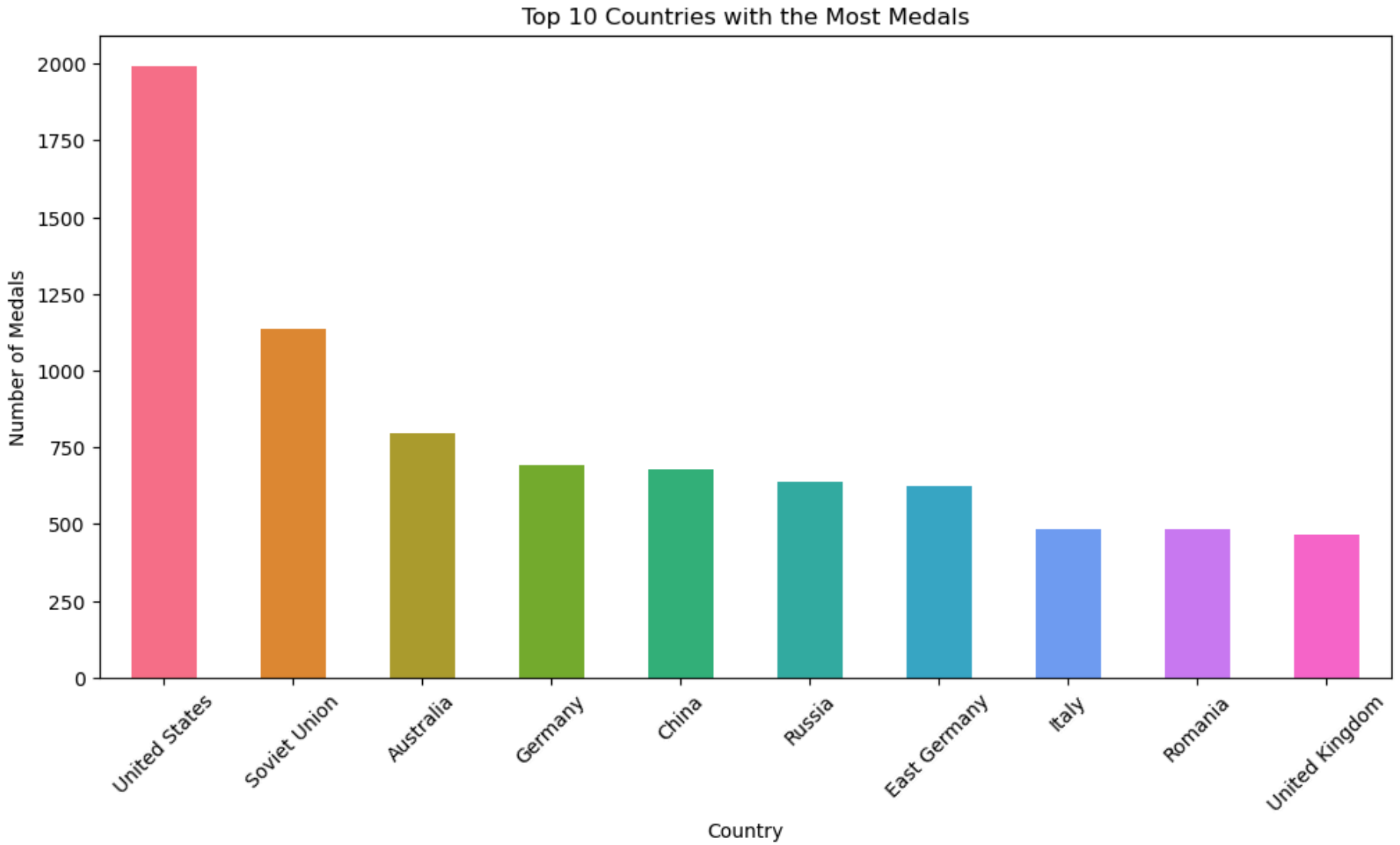
In [117…

```
# Visualization 4: Most Successful Athletes
top_athletes = df.groupby('Athlete')['Medal'].count().nlargest(10)
colors = sns.color_palette('husl', len(top_athletes))
plt.figure(figsize=(12, 6))
top_athletes.plot(kind='barh', color=colors)
plt.title('Top 10 Most Successful Olympic Athletes')
plt.xlabel('Number of Medals')
plt.ylabel('Athlete')
plt.show()
```



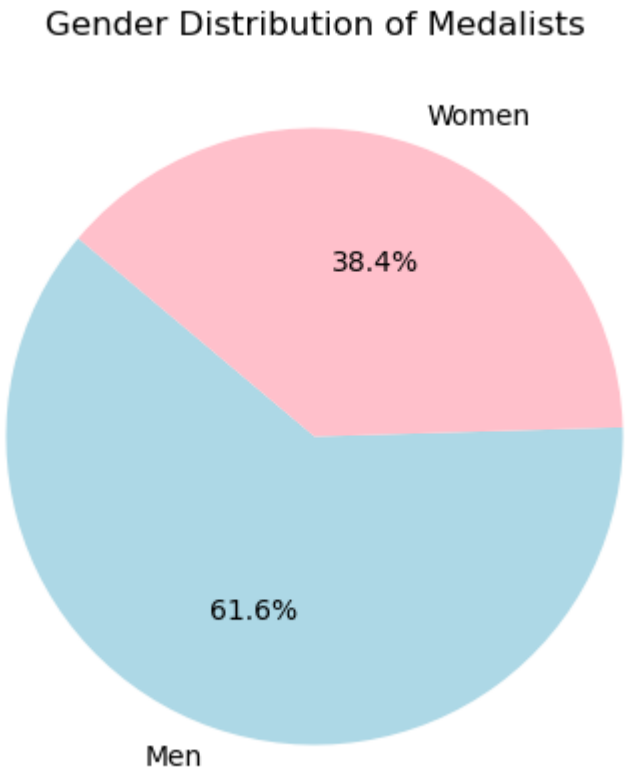
In [119…

```
country_medals = df['Country'].value_counts().nlargest(10)
colors = sns.color_palette('husl', len(country_medals))  # Generate a palette of unique colors
plt.figure(figsize=(12, 6))
country_medals.plot(kind='bar', color=colors)
plt.title('Top 10 Countries with the Most Medals')
plt.xlabel('Country')
plt.ylabel('Number of Medals')
plt.xticks(rotation=45)
plt.show()
```



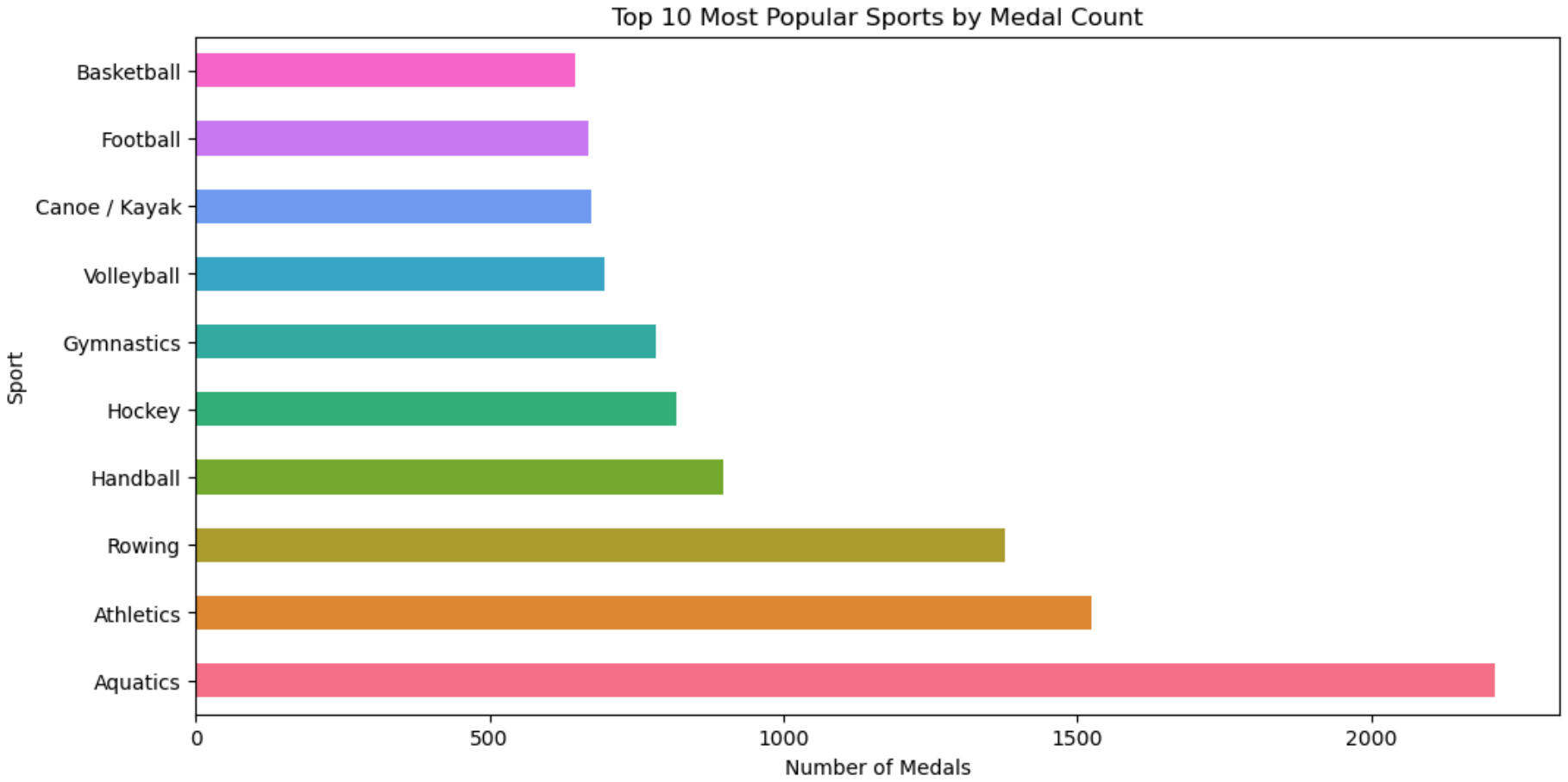
In [121…
```
# Visualization 3: Gender Distribution of Medalists
gender_counts = df['Gender'].value_counts()
plt.figure(figsize=(5, 5))
gender_counts.plot(kind='pie', autopct='%1.1f%%', startangle=140, colors=['lightblue', 'pink'])
plt.title('Gender Distribution of Medalists')
```

```
plt.ylabel('')
plt.show()
```

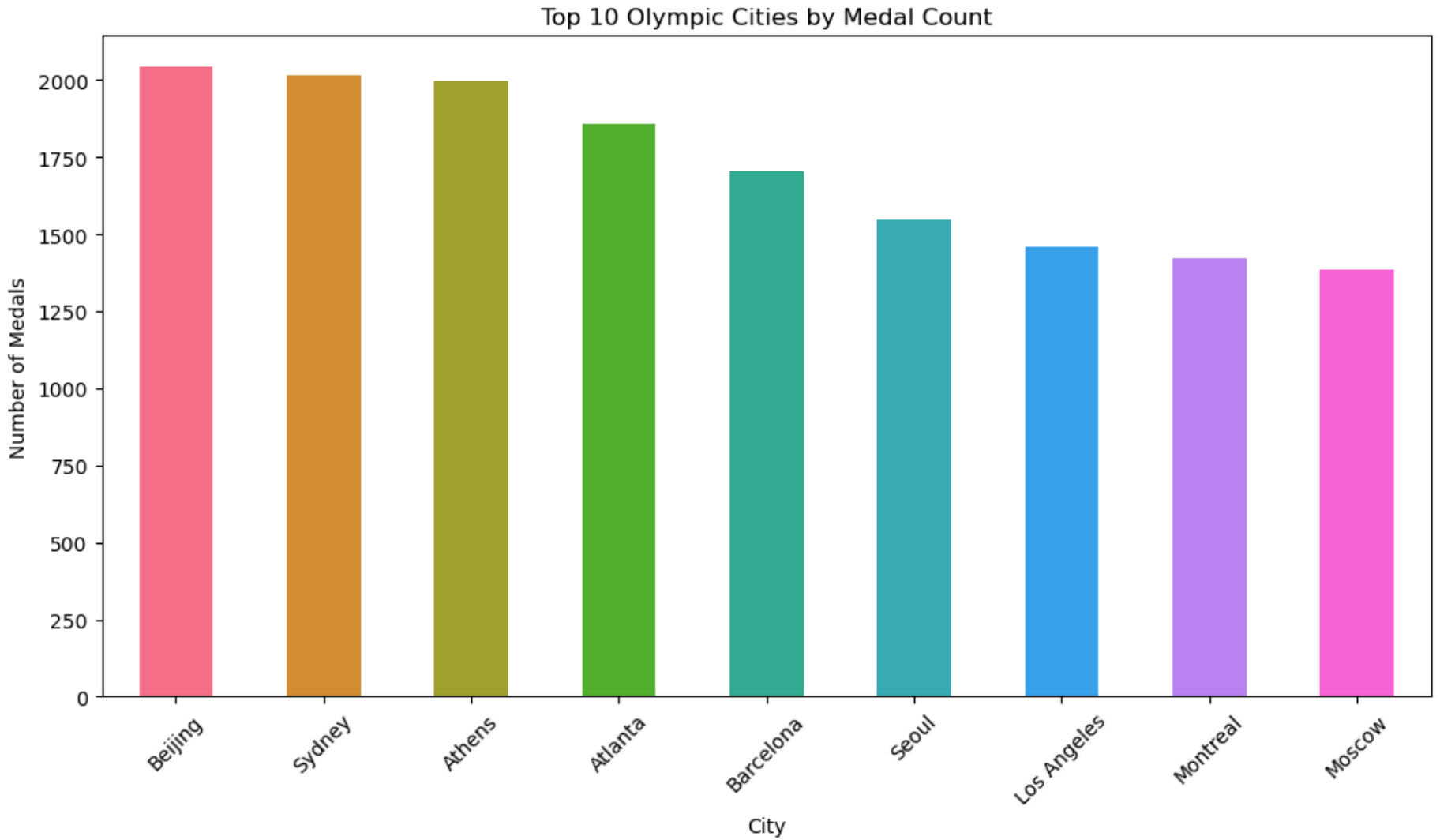


In [123…

```
# Visualization 4: Most Popular Sports by Medal Count
sport_medals = df['Sport'].value_counts().nlargest(10)
colors = sns.color_palette('husl', len(sport_medals))
plt.figure(figsize=(12, 6))
sport_medals.plot(kind='barh', color=colors)
plt.title('Top 10 Most Popular Sports by Medal Count')
plt.xlabel('Number of Medals')
plt.ylabel('Sport')
plt.show()
```



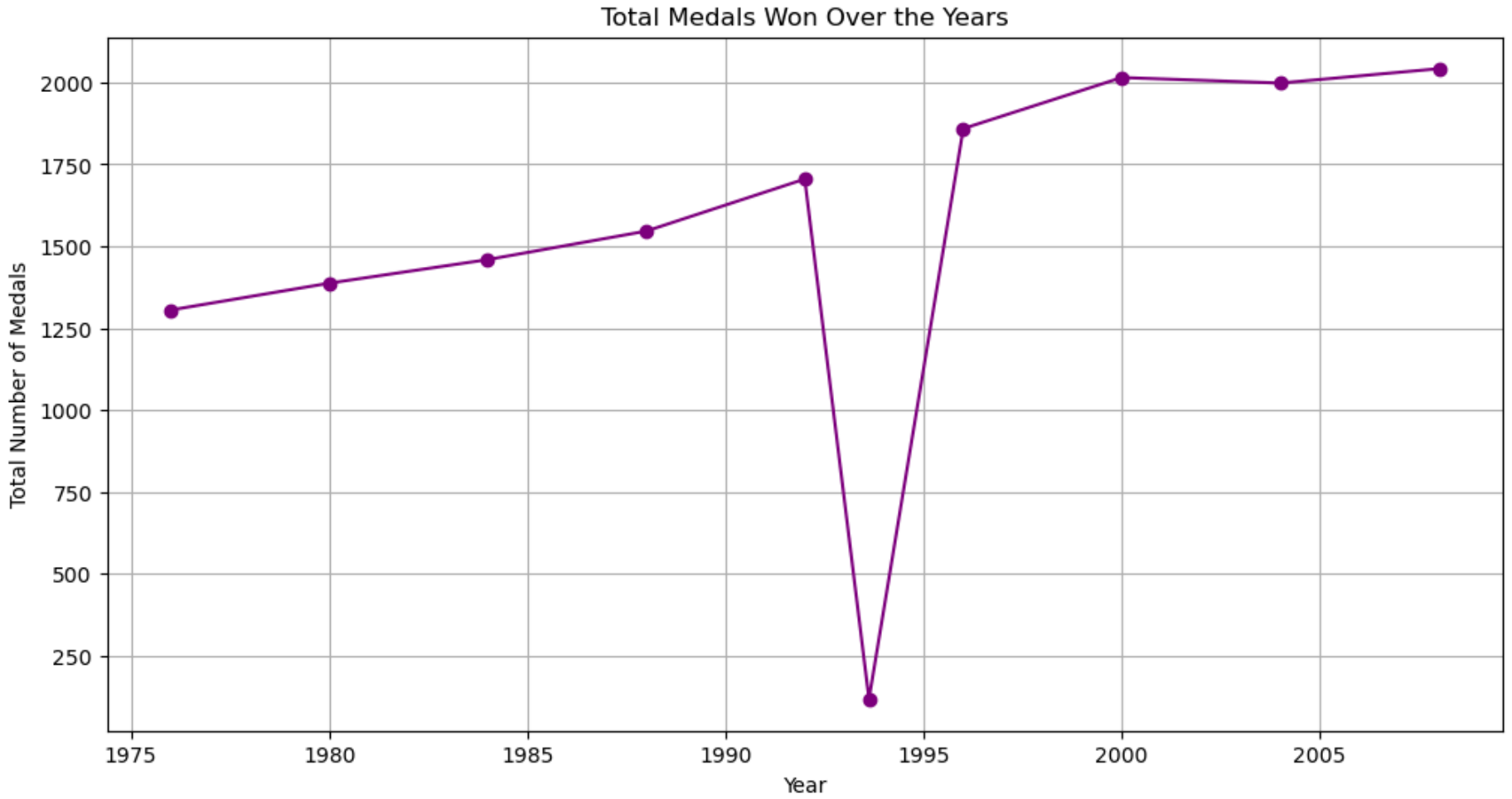
In [125…
```
# Visualization 5: Medal Distribution Across Olympic Cities
city_medals = df['City'].value_counts().nlargest(10)
colors = sns.color_palette('husl', len(city_medals))
plt.figure(figsize=(12, 6))
city_medals.plot(kind='bar', color=colors)
plt.title('Top 10 Olympic Cities by Medal Count')
plt.xlabel('City')
plt.ylabel('Number of Medals')
plt.xticks(rotation=45)
plt.show()
```



In [127…

```
# Visualization 3: Total Medals Won Over the Years
medals_over_years = df.groupby('Year')['Medal'].count()
plt.figure(figsize=(12, 6))
medals_over_years.plot(kind='line', marker='o', linestyle='-', color='purple')
plt.title('Total Medals Won Over the Years')
plt.xlabel('Year')
plt.ylabel('Total Number of Medals')
```

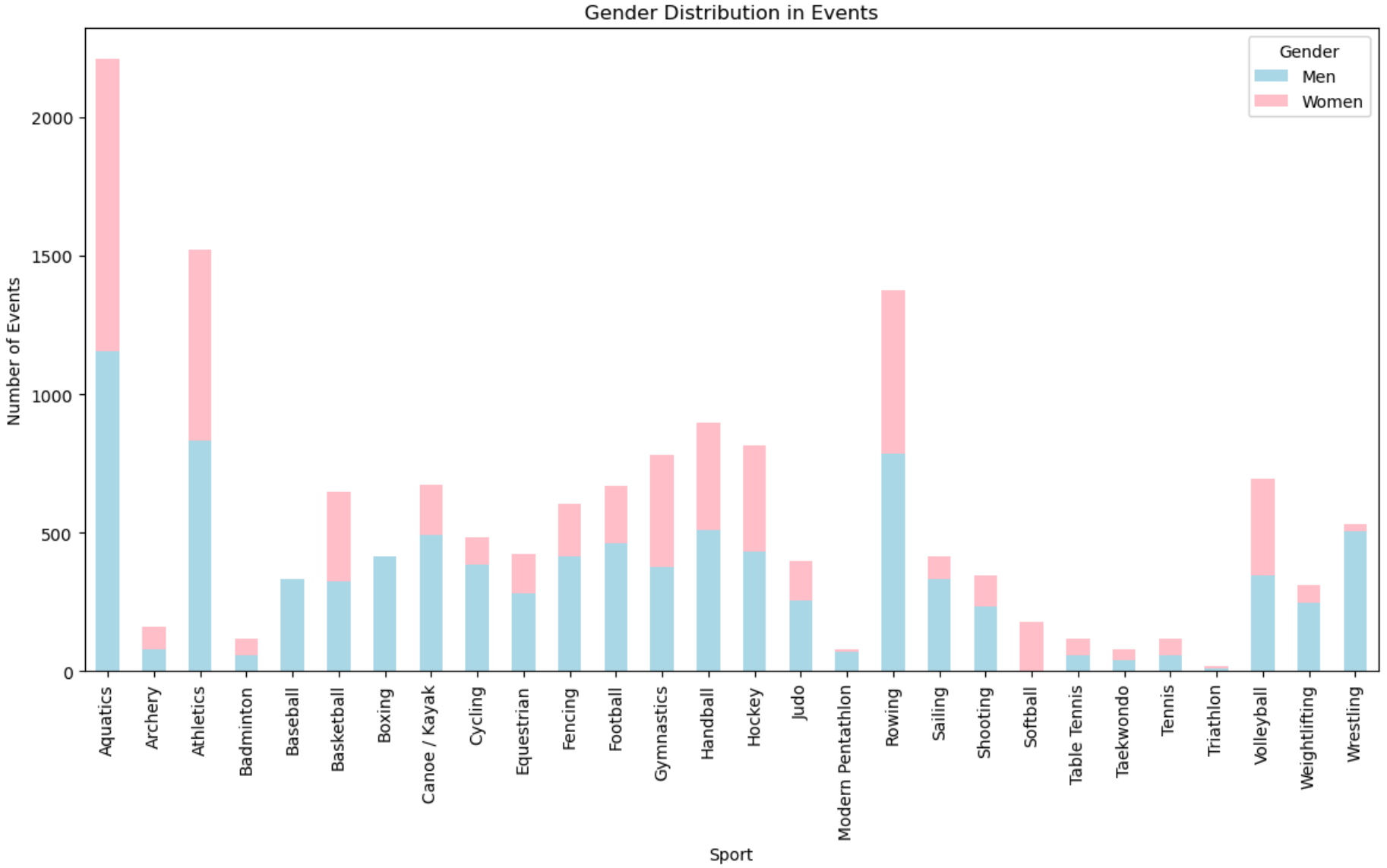
```
plt.grid()
plt.show()
```



In [129…

```
# Visualization 7: Gender Distribution in Events
gender_event_counts = df.groupby(['Sport', 'Gender']).size().unstack()
gender_event_counts.plot(kind='bar', stacked=True, figsize=(14, 7), color=['lightblue', 'pink'])
plt.title('Gender Distribution in Events')
plt.xlabel('Sport')
plt.ylabel('Number of Events')
plt.xticks(rotation=90)
```

```
plt.legend(title='Gender')
plt.show()
```

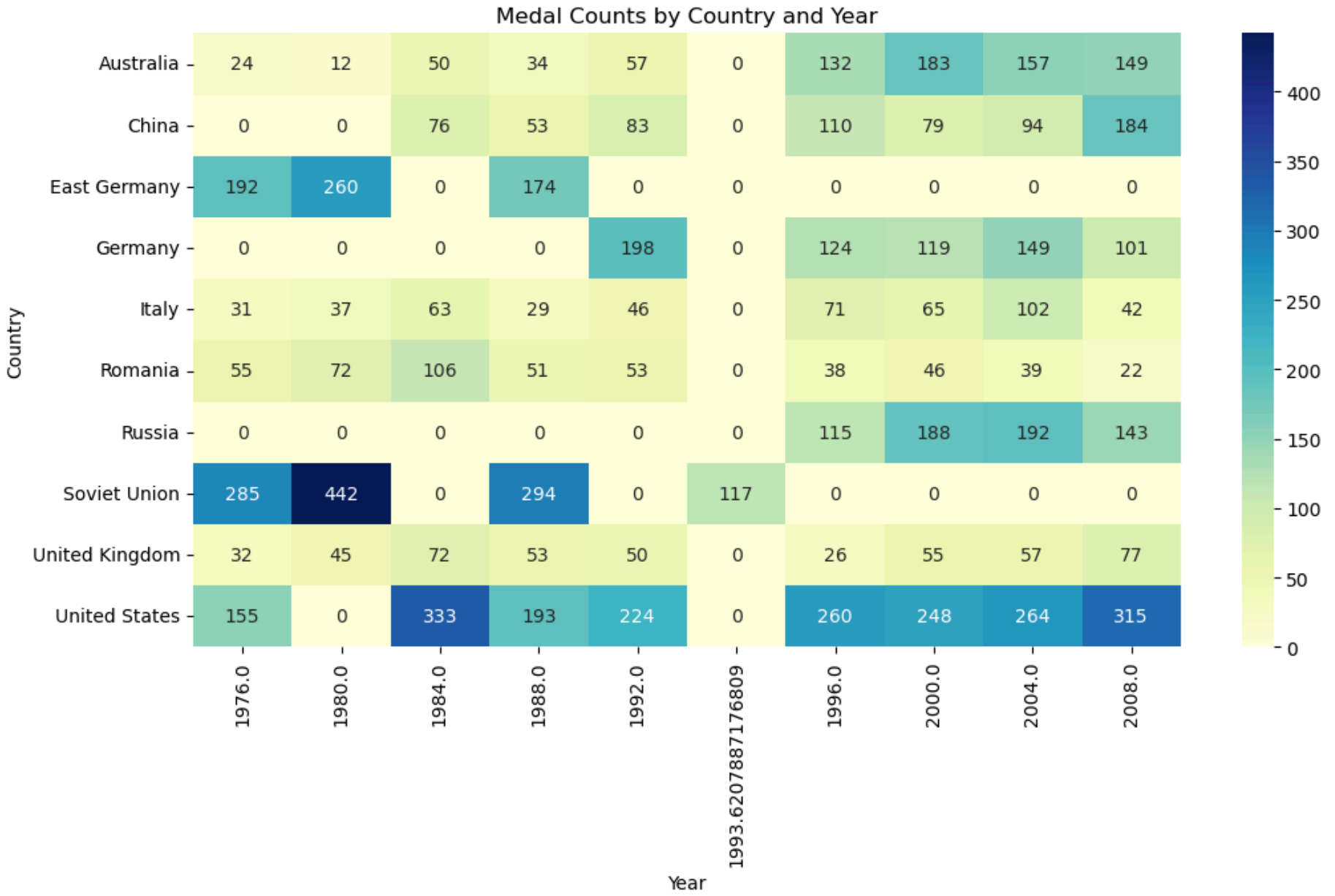


In [131…

```
# Visualization 8: Heatmap of Medal Counts by Country and Year
top_countries = df['Country'].value_counts().nlargest(10).index
filtered_df = df[df['Country'].isin(top_countries)]
```

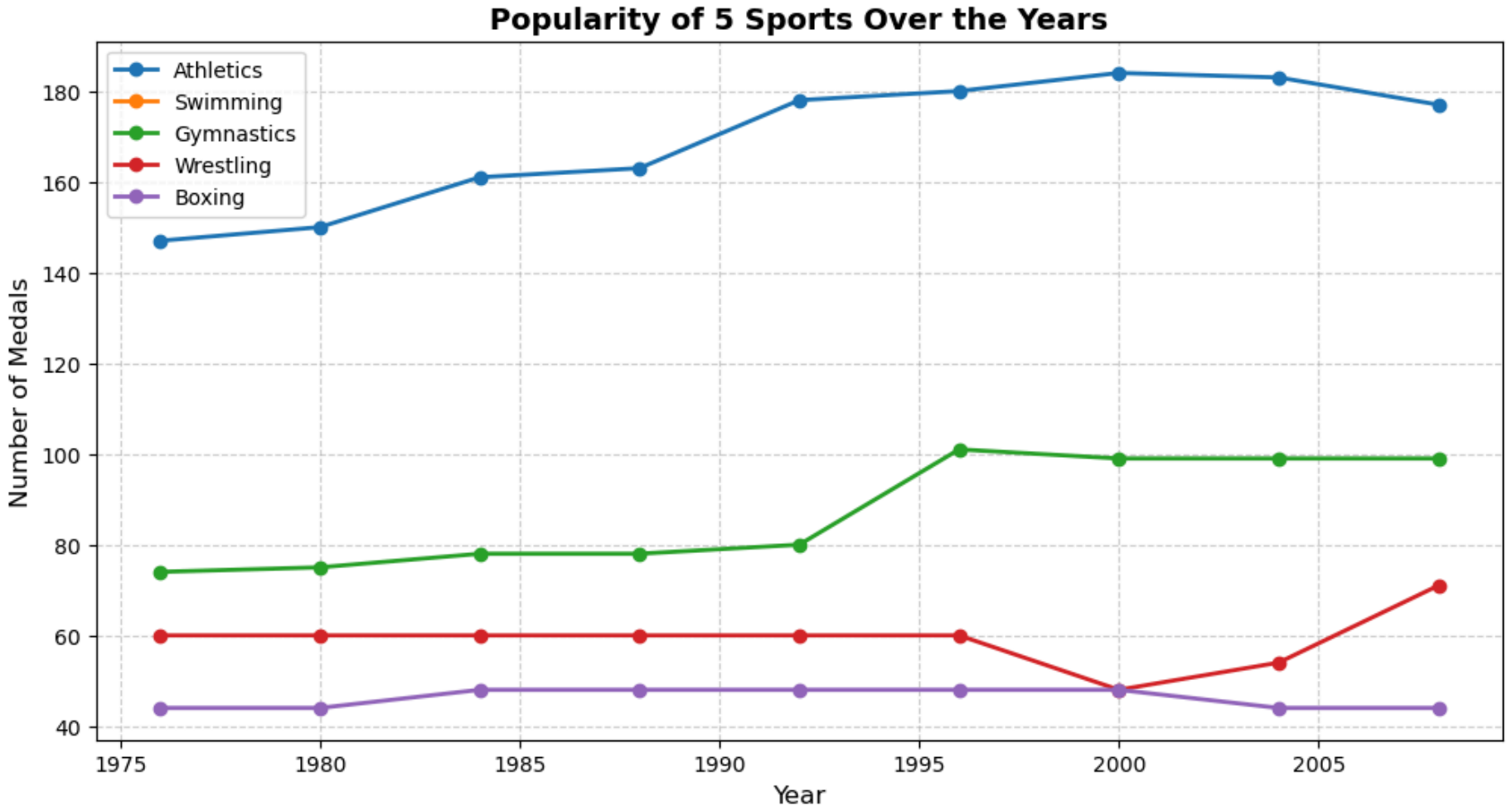
```
pivot_table = filtered_df.pivot_table(index='Country', columns='Year', values='Medal', aggfunc='count', fill_value=

plt.figure(figsize=(12, 6))
sns.heatmap(pivot_table, cmap='YlGnBu', annot=True, fmt='d')
plt.title('Medal Counts by Country and Year')
plt.xlabel('Year')
plt.ylabel('Country')
plt.show()
```



In [133…
```
# Visualization 8: Popularity of 5 Sports Over the Years
sports = ['Athletics', 'Swimming', 'Gymnastics', 'Wrestling', 'Boxing']  # Change sports if needed
plt.figure(figsize=(12, 6))
```

```
colors = sns.color_palette('tab10', len(sports))  # Generate distinct colors
for sport, color in zip(sports, colors):
    sport_trend = df[df['Sport'] == sport].groupby('Year')['Medal'].count()
    plt.plot(sport_trend, marker='o', linestyle='-', label=sport, color=color, linewidth=2)
plt.title('Popularity of 5 Sports Over the Years', fontsize=14, fontweight='bold')
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of Medals', fontsize=12)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```



Applying Predictive Analytics

In [135…
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

In [137…
```
le = LabelEncoder()
df['Country_Code']=le.fit_transform(df['Country_Code'])
df['Sport']=le.fit_transform(df['Sport'])
df['Gender']=le.fit_transform(df['Gender'])
df['Event_gender']=le.fit_transform(df['Event_gender'])
df['Medal']=df['Medal'].map({'Gold': 1,
'Silver': 2,
'Bronze': 3, np.nan: 0})
# Features and target
X = df[['Country_Code','Event_gender','Sport','Gender']]
y = df['Medal']
```

In [139…
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=10)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[139… ((10803, 4), (4630, 4), (10803,), (4630,))

In [141…
```
lr=LogisticRegression()
lr.fit(X_train,y_train)
```

Out[141…
```
LogisticRegression()
```

In [143…
```
y_pred=lr.predict(X_test)
```

In [153…
```
cm=confusion_matrix(y_test,y_pred)
cr=classification_report(y_test,y_pred)
score=accuracy_score(y_test,y_pred)
print('The Confusion Matrix is :',cm)
```

```
print('    ')
print('The classification report is :',cr)
print('    ')
print('The accuracy score is :',score)
```

```
The Confusion Matrix is : [[710  37 775]
 [632  20 877]
 [617  21 941]]

The classification report is :               precision    recall  f1-score   support

           1       0.36      0.47      0.41      1522
           2       0.26      0.01      0.02      1529
           3       0.36      0.60      0.45      1579

    accuracy                           0.36      4630
   macro avg       0.33      0.36      0.29      4630
weighted avg       0.33      0.36      0.30      4630


The accuracy score is : 0.36090712742980563
```

In [155…

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

# Define hyperparameter grid
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],  # Regularization strength
    'solver': ['liblinear', 'lbfgs', 'saga'],  # Different solvers
    'max_iter': [100, 500, 1000]  # Number of iterations
}

# Initialize Logistic Regression model
lr = LogisticRegression()

# Grid Search with cross-validation
grid_search = GridSearchCV(lr, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best parameters and model
```

```
print("Best Parameters:", grid_search.best_params_)
best_lr = grid_search.best_estimator_
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
  n_iter_i = _check_optimize_result(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was r
eached which means the coef_ did not converge
  warnings.warn(
Best Parameters: {'C': 0.01, 'max_iter': 500, 'solver': 'saga'}
```

In [157…

```
best_lr.fit(X_train,y_train)
```

Out[157…

```
▾ LogisticRegression
LogisticRegression(C=0.01, max_iter=500, solver='saga')
```

In [161…

```
y_pred_new=best_lr.predict(X_test)
```

In [165…

```
cm_new=confusion_matrix(y_test,y_pred_new)
cr_new=classification_report(y_test,y_pred_new)
score_new=accuracy_score(y_test,y_pred_new)
print('The Confusion Matrix is :',cm_new)
print('   ')
print('The classification report is :',cr_new)
print('    ')
print('The accuracy score is :',score_new)
```

```
The Confusion Matrix is : [[710  26 786]
 [633  15 881]
 [615  20 944]]

The classification report is :               precision    recall  f1-score   support

           1       0.36      0.47      0.41      1522
           2       0.25      0.01      0.02      1529
           3       0.36      0.60      0.45      1579

    accuracy                           0.36      4630
   macro avg       0.32      0.36      0.29      4630
weighted avg       0.32      0.36      0.29      4630


The accuracy score is : 0.36047516198704105
```

In [ ]: