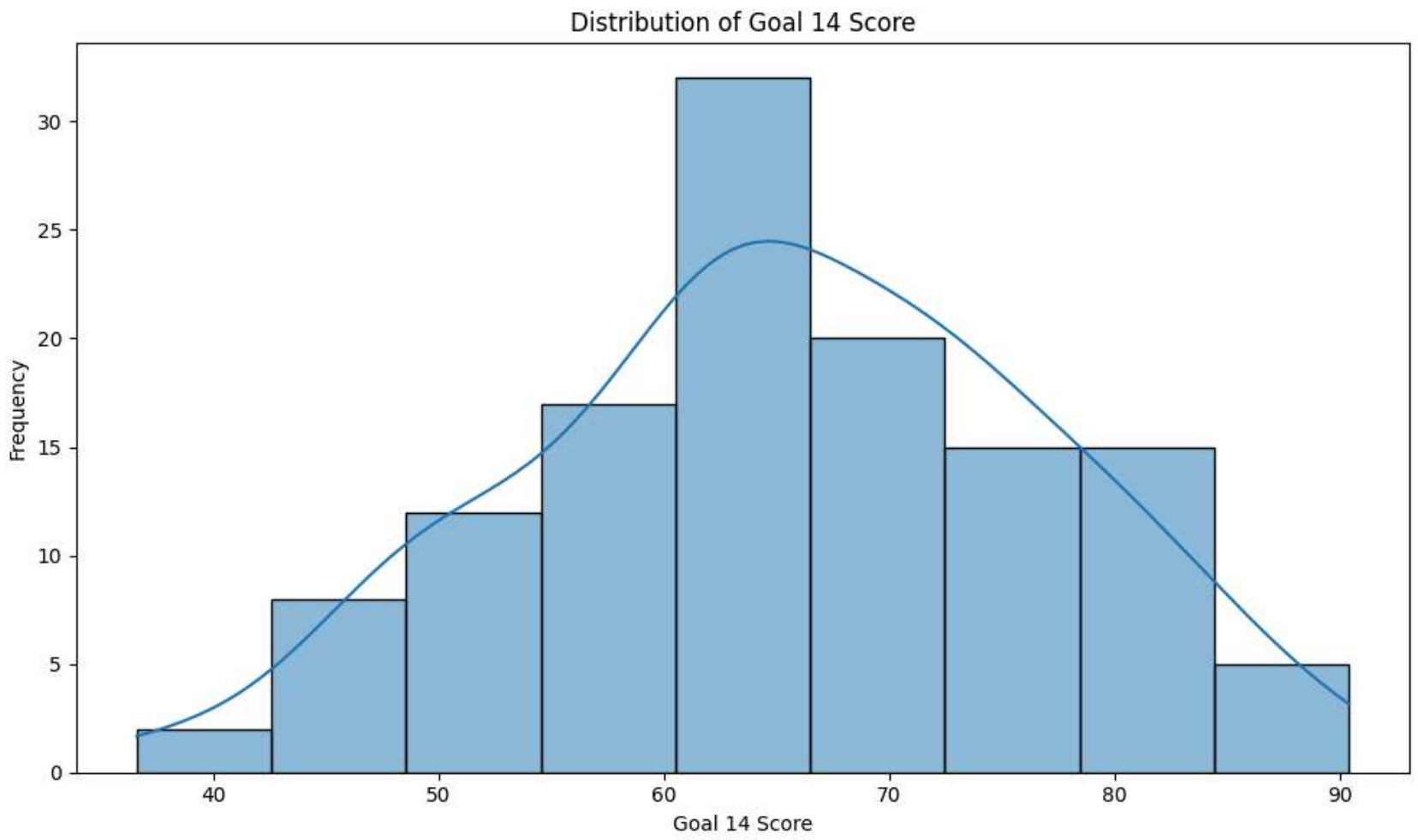
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

df = pd.read_csv("/content/sustainable_development_report_2023.csv")
print(df.head())
```

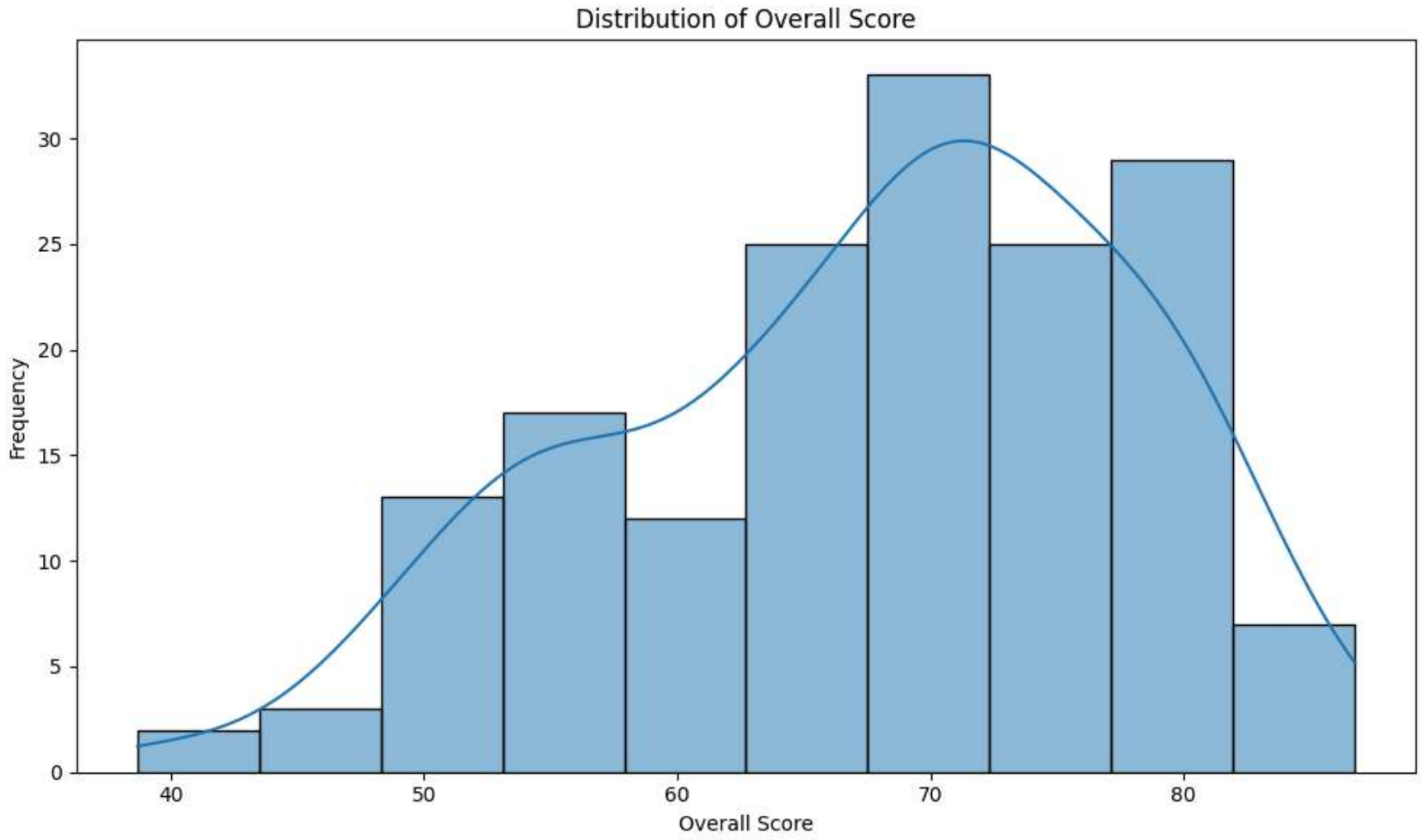
```
  country_code  country region  overall_score  goal_1_score  goal_2_score  \
0          FIN  Finland   OECD      86.760595       99.5750     60.886750
1          SWE   Sweden   OECD      85.981397       98.8885     63.074125
2          DNK  Denmark   OECD      85.683637       99.2155     71.025250
3          DEU  Germany   OECD      83.358447       99.5105     72.366000
4          AUT  Austria   OECD      82.280189       99.4510     73.067500

   goal_3_score  goal_4_score  goal_5_score  goal_6_score  ...  goal_8_score  \
0     95.386385     97.169333      92.11125       94.3276  ...     86.789000
1     96.904000     99.761667      91.44025       95.0576  ...     84.966429
2     95.398500     99.339667      86.99800       90.7316  ...     87.562429
3     93.039357     97.162667      81.92025       88.4434  ...     86.967286
4     92.468000     97.914333      84.57925       92.1636  ...     83.274143

   goal_9_score  goal_10_score  goal_11_score  goal_12_score  goal_13_score  \
0     95.995714        98.4685      91.233750      60.059571      68.793667
1     97.586286        94.9650      90.389250      56.830571      70.031000
2     96.984857        98.1560      93.038500      44.571714      60.780667
3     95.788429        88.1470      90.096500      55.412857      64.002000
4     96.982143        94.6345      92.473667      49.623286      57.332000

   goal_14_score  goal_15_score  goal_16_score  goal_17_score
0      87.928000        85.0700      92.521091       75.60100
1      69.348667        80.1882      88.508455       85.77025
2      76.303333        92.7924      93.844909       82.14800
3      73.996000        79.2318      89.457545       84.39025
4            NaN        73.5836      87.911455       71.13025

[5 rows x 21 columns]
```

```
# Visualize the distribution of goal_14_score using Seaborn
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="goal_14_score", kde=True)
plt.title("Distribution of Goal 14 Score")
plt.xlabel("Goal 14 Score")
plt.ylabel("Frequency")
```

```
plt.tight_layout()
plt.show()
```




```
# Visualize the distribution of overall_score using Seaborn
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="overall_score", kde=True)
plt.title("Distribution of Overall Score")
plt.xlabel("Overall Score")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```




```
# Check for missing values in the dataset
print("\nMissing values per column:")
print(df.isnull().sum())
```

```
Missing values per column:
country_code      0
country           0
region            0
overall_score     0
goal_1_score     15
goal_2_score      0
goal_3_score      0
goal_4_score      0
```

```
goal_5_score      0
goal_6_score      0
goal_7_score      0
goal_8_score      0
goal_9_score      0
goal_10_score    17
goal_11_score     0
goal_12_score     0
goal_13_score     0
goal_14_score    40
goal_15_score     0
goal_16_score     0
goal_17_score     0
dtype: int64
```

```
# Show summary statistics for df
print("\nSummary Statistics for the DataFrame:")
display(df.describe())
```

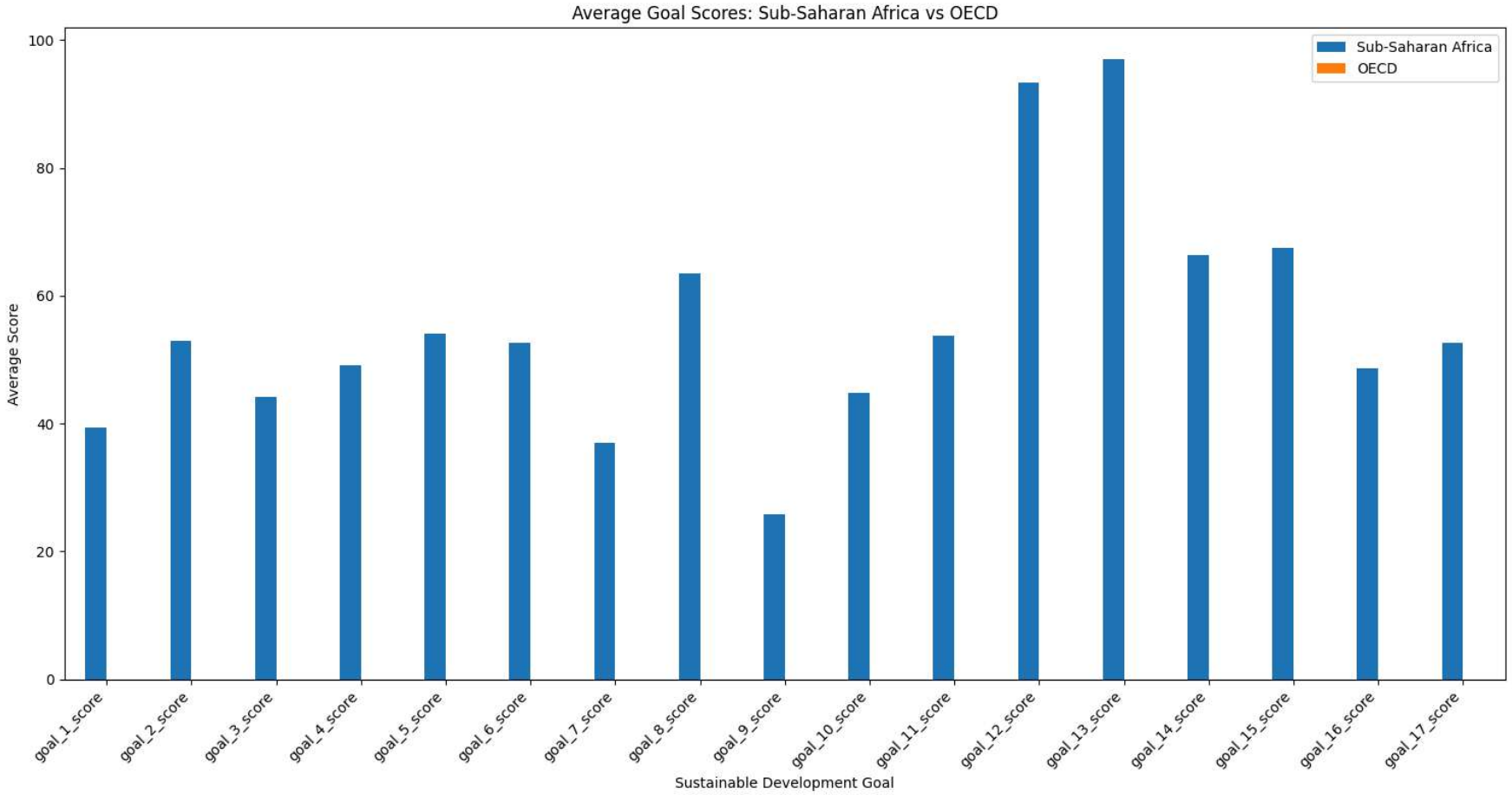
Summary Statistics for the DataFrame:

	overall_score	goal_1_score	goal_2_score	goal_3_score	goal_4_score	goal_5_score	goal_6_score	goal_7_score	goal_8_score	go
count	166.000000	151.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	
mean	67.549197	75.234401	59.799100	69.694078	76.512968	63.285420	66.710744	61.413598	71.952935	
std	10.295499	31.169948	10.620853	20.354575	23.181919	16.399691	14.091641	20.364351	10.592308	
min	38.676086	0.000000	19.805800	12.952714	1.232250	13.054750	32.600000	8.697000	39.535000	
25%	60.547488	55.779250	54.007188	51.860089	61.417938	51.046250	55.237250	47.521312	66.426857	
50%	69.376528	93.300500	61.027500	75.437629	84.772875	65.869875	67.878000	68.612750	73.157643	
75%	74.947511	98.950750	67.264335	85.524428	95.644063	76.137000	76.044200	74.364000	79.626036	
max	86.760595	100.000000	83.401125	97.115143	99.761667	94.021667	95.057600	99.550750	93.382750	

```
# Get the OECD average scores
oecd_average_scores = average_score_by_region[average_score_by_region["region"] == "OECD"]

# Transpose the Sub-Saharan Africa average scores for easier comparison
sub_saharan_africa_avg_transposed = average_scores_sub_saharan_africa.to_frame(name="Sub-Saharan Africa")

# Select only the goal columns from the transposed Sub-Saharan Africa data
sub_saharan_africa_avg_goals = sub_saharan_africa_avg_transposed[sub_saharan_africa_avg_transposed.index.str.startswith("goal_")]
```

```
# Transpose the OECD average scores for easier comparison
oecd_average_scores_transposed = oecd_average_scores.set_index("region").T

# Select only the goal columns from the transposed OECD data
oecd_average_goals = oecd_average_scores_transposed[oecd_average_scores_transposed.index.str.startswith("goal_")]

# Combine the two transposed DataFrames
comparison_df = pd.concat([sub_saharan_africa_avg_goals, oecd_average_goals], axis=1)

# Plotting the comparison
plt.figure(figsize=(15, 8))
comparison_df.plot(kind='bar', figsize=(15, 8))
plt.title('Average Goal Scores: Sub-Saharan Africa vs OECD')
plt.xlabel('Sustainable Development Goal')
plt.ylabel('Average Score')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
<Figure size 1500x800 with 0 Axes>
```



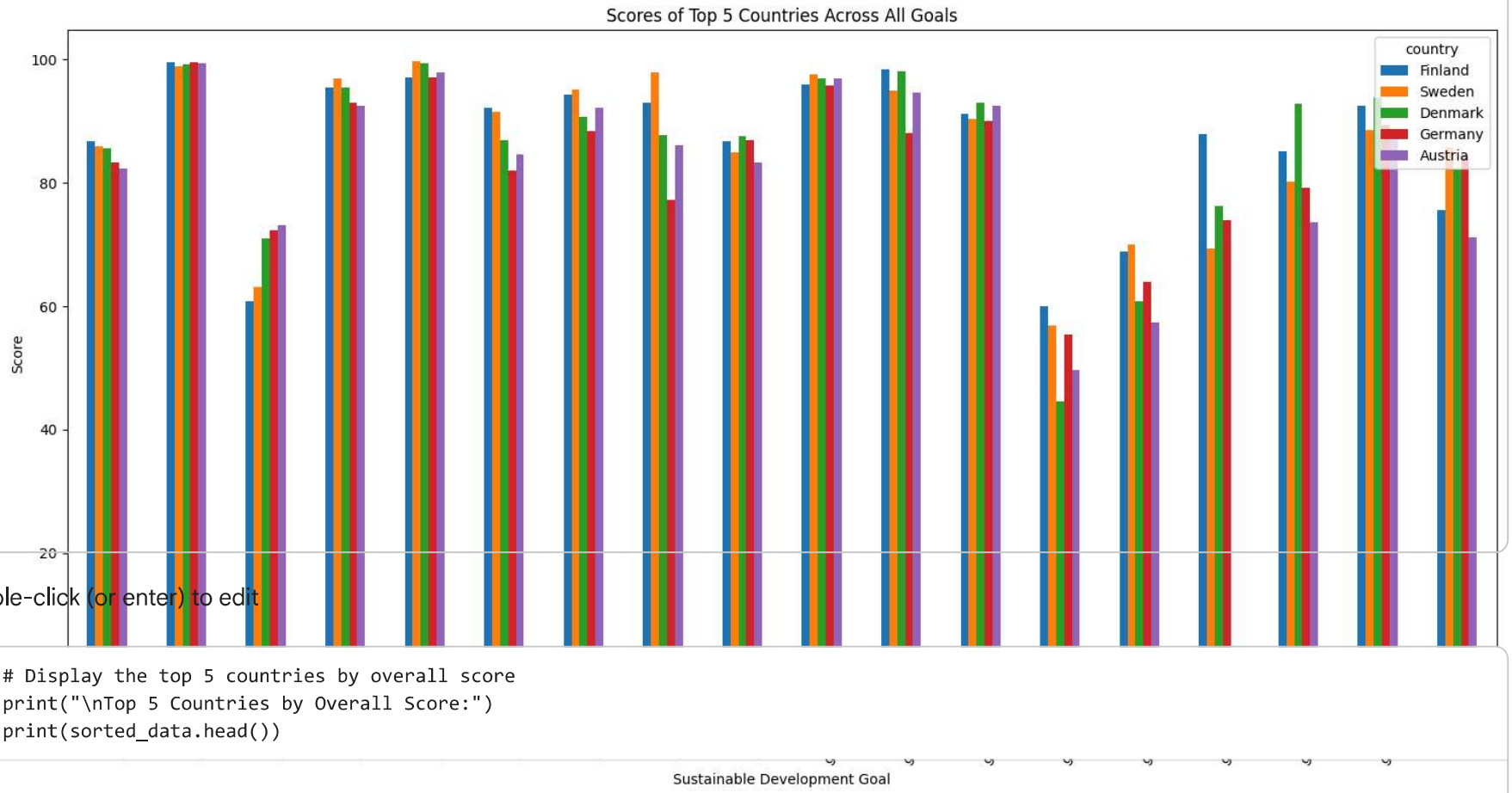

```
# Filter data for the Sub-Saharan Africa region
sub_saharan_africa_data = df[df["region"] == "Sub-Saharan Africa"]

# Calculate the average scores for each goal in the Sub-Saharan Africa region
average_scores_sub_saharan_africa = sub_saharan_africa_data.mean(numeric_only=True)

print("\nAverage Scores for Sub-Saharan Africa Region Across All Goals:")
print(average_scores_sub_saharan_africa)
```

```
Average Scores for Sub-Saharan Africa Region Across All Goals:
overall_score    55.461529
goal_1_score     39.445256
```

```
goal_2_score     52.988240
goal_3_score     44.102043
goal_4_score     49.195380
goal_5_score     54.055420
goal_6_score     52.640587
goal_7_score     36.977539
goal_8_score     63.414371
goal_9_score     25.760039
goal_10_score    44.866956
goal_11_score    53.809474
goal_12_score    93.407992
goal_13_score    97.047867
goal_14_score    66.334599
goal_15_score    67.490306
goal_16_score    48.723877
goal_17_score    52.586052
dtype: float64
```

```
# Select relevant columns for the top 5 countries (excluding country code, country, and region)
top_5_countries_scores = sorted_data.head(5).drop(columns=['country_code', 'country', 'region'])

# Transpose the DataFrame for easier plotting
top_5_countries_scores_transposed = top_5_countries_scores.T

# Rename the columns with country names
top_5_countries_scores_transposed.columns = sorted_data.head(5)['country']

# Plotting the scores
plt.figure(figsize=(15, 8))
top_5_countries_scores_transposed.plot(kind='bar', figsize=(15, 8))
plt.title('Scores of Top 5 Countries Across All Goals')
plt.xlabel('Sustainable Development Goal')
plt.ylabel('Score')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
<Figure size 1500x800 with 0 Axes>
```



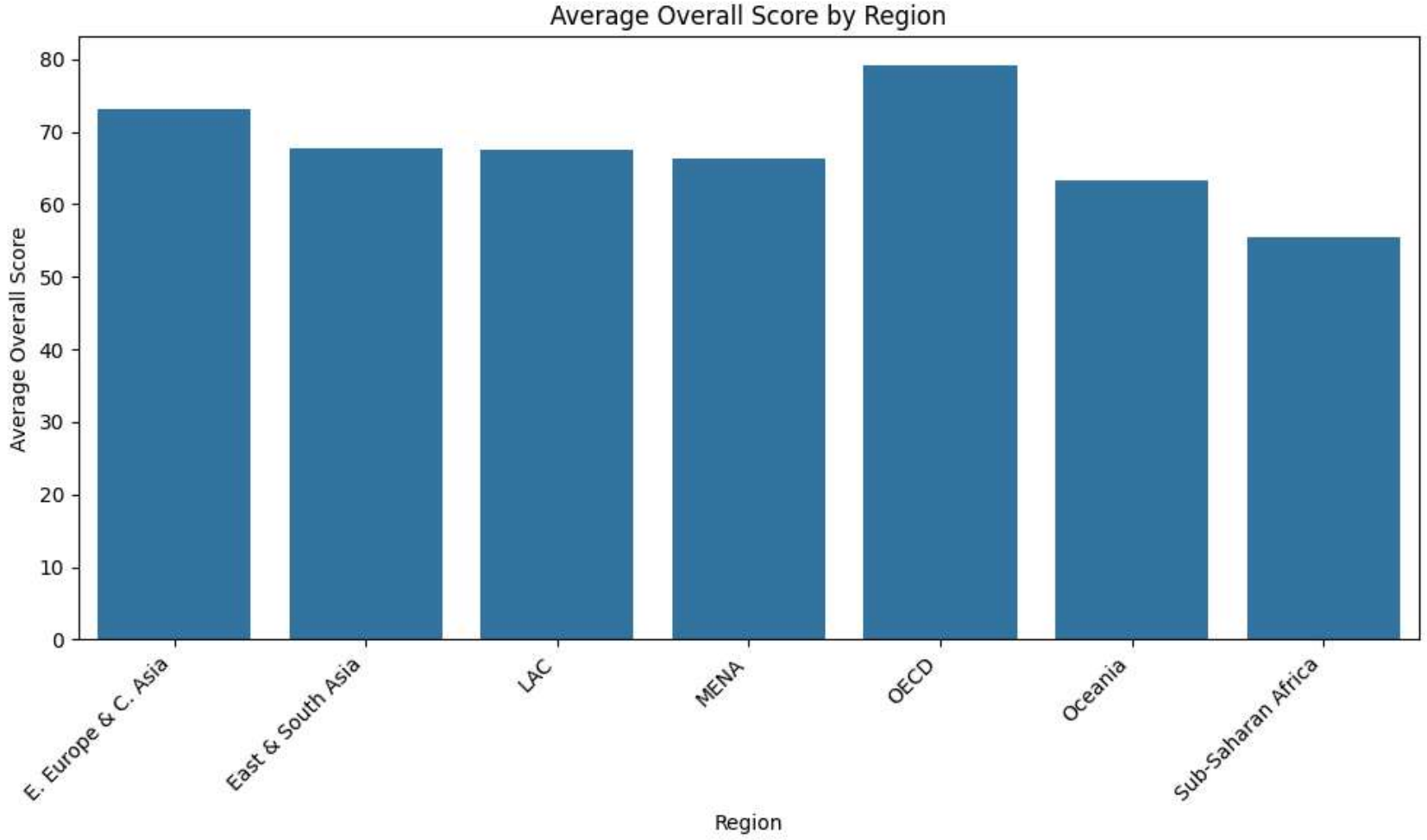
Double-click (or enter) to edit

```
# Display the top 5 countries by overall score
print("\nTop 5 Countries by Overall Score:")
print(sorted_data.head())
```

```
Top 5 Countries by Overall Score:
  country_code  country region  overall_score  goal_1_score  goal_2_score  \
0          FIN  Finland   OECD      86.760595       99.5750     60.886750
1          SWE   Sweden   OECD      85.981397       98.8885     63.074125
2          DNK  Denmark   OECD      85.683637       99.2155     71.025250
3          DEU  Germany   OECD      83.358447       99.5105     72.366000
4          AUT  Austria   OECD      82.280189       99.4510     73.067500

   goal_3_score  goal_4_score  goal_5_score  goal_6_score  ...  goal_8_score  \
0     95.386385     97.169333      92.11125       94.3276  ...     86.789000
1     96.904000     99.761667      91.44025       95.0576  ...     84.966429
2     95.398500     99.339667      86.99800       90.7316  ...     87.562429
3     93.039357     97.162667      81.92025       88.4434  ...     86.967286
4     92.468000     97.914333      84.57925       92.1636  ...     83.274143

   goal_9_score  goal_10_score  goal_11_score  goal_12_score  goal_13_score  \
0     95.995714         98.4685      91.233750      60.059571      68.793667
```

```
1     97.586286        94.9650      90.389250      56.830571      70.031000
2     96.984857        98.1560      93.038500      44.571714      60.780667
3     95.788429        88.1470      90.096500      55.412857      64.002000
4     96.982143        94.6345      92.473667      49.623286      57.332000

   goal_14_score  goal_15_score  goal_16_score  goal_17_score
0      87.928000        85.0700      92.521091       75.60100
1      69.348667        80.1882      88.508455       85.77025
2      76.303333        92.7924      93.844909       82.14800
3      73.996000        79.2318      89.457545       84.39025
4            NaN        73.5836      87.911455       71.13025

[5 rows x 21 columns]
```

```
# Plot average overall score by region
plt.figure(figsize=(10, 6))
sns.barplot(x="region", y="overall_score", data=average_score_by_region)
plt.title("Average Overall Score by Region")
plt.xlabel("Region")
plt.ylabel("Average Overall Score")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```




```
# Analyze average 'overall_score' by region
average_score_by_region = df.groupby("region")["overall_score"].mean().reset_index()
print("\nAverage Overall Score by Region:")
print(average_score_by_region)
```

```
Average Overall Score by Region:
               region  overall_score
0  E. Europe & C. Asia      73.149119
1    East & South Asia      67.655676
2                  LAC      67.598713
3                 MENA      66.283838
4                 OECD      79.150447
```