Appendix

In [44]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [45]:
```
# 1. Understand the Problem & Data

# STEP 1: Load the Data
print("--- Loading Data ---")
df = pd.read_csv('/content/AI_index_db.csv')
print("Dataset loaded successfully.")
```

```
--- Loading Data ---
Dataset loaded successfully.
```

In [46]:
```
# 2. Import & Inspect Data

# This gives preview the first few rows
print("--- Checking Data Preview ---")
print(df.head())
print()

# This gives a summary of the dataframe, including the data type of each col
print("\n--- Checking Data Types ---")
df.info()
print()

# Gives the size or dimensions of df
print("--- Checking Data Dimensions ---")
print(df.shape)
print()

# List of numerical columns
num_cols = df.select_dtypes(include=['number']).columns
print("Numerical columns:\n", num_cols)

# List of categorical columns
cat_cols = df.select_dtypes(include=['object']).columns
print("\nCategorical columns:\n", cat_cols)
print()

print(f" The dataset has {len(num_cols)} Numerical columns and {len(cat_cols
```

```
--- Checking Data Preview ---
                   Country  Talent  Infrastructure  Operating Environment
\
0  United States of America  100.00           94.02                   64.56
1                     China   16.51          100.00                   91.57
2            United Kingdom   39.65           71.43                   74.65
3                    Canada   31.28           77.05                   93.94
4                    Israel   35.76           67.58                   82.44

   Research  Development  Government Strategy  Commercial  Total score  \
0    100.00       100.00                77.39      100.00       100.00
1     71.42        79.97                94.87       44.02        62.92
2     36.50        25.03                82.82       18.91        40.93
3     30.67        25.78               100.00       14.88        40.19
4     32.63        27.96                43.91       27.33        39.89

         Region                Cluster  Income group   Political regime
0      Americas          Power players          High  Liberal democracy
1  Asia-Pacific          Power players  Upper middle   Closed autocracy
2        Europe  Traditional champions          High  Liberal democracy
3      Americas  Traditional champions          High  Liberal democracy
4   Middle East           Rising stars          High  Liberal democracy


--- Checking Data Types ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Country               62 non-null     object
 1   Talent                62 non-null     float64
 2   Infrastructure        62 non-null     float64
 3   Operating Environment  62 non-null     float64
 4   Research              62 non-null     float64
 5   Development           62 non-null     float64
 6   Government Strategy   62 non-null     float64
 7   Commercial            62 non-null     float64
 8   Total score           62 non-null     float64
 9   Region                62 non-null     object
 10  Cluster               62 non-null     object
 11  Income group          62 non-null     object
 12  Political regime      62 non-null     object
dtypes: float64(8), object(5)
memory usage: 6.4+ KB

--- Checking Data Dimensions ---
(62, 13)

Numerical columns:
 Index(['Talent', 'Infrastructure', 'Operating Environment', 'Research',
       'Development', 'Government Strategy', 'Commercial', 'Total score'],
      dtype='object')

Categorical columns:
 Index(['Country', 'Region', 'Cluster', 'Income group', 'Political regime'],
```

```
dtype='object')
```

The dataset has 8 Numerical columns and 5 Categorical columns.

In [47]:
```python
# 3. Handle Missing Data

print("--- Checking for Missing Values ---")
# This counts the number of empty (NaN) cells in each column.
missing_values = df.isnull().sum()
print(missing_values)

print("\n--- Checking for Duplicate Rows ---")
# This counts the total number of rows that are exact duplicates of another
duplicate_rows = df.duplicated().sum()
print(f"Number of duplicate rows found: {duplicate_rows}")

# This shows if we have any placeholder missing values like "N/A", "NA", "?"
print("\n--- Checking for Placeholder Missing Values ---")
placeholders = ["N/A", "NA", "?", "None", "-", " "]
for val in placeholders:
    print(f"Count of '{val}':")
    print((df == val).sum())
```

```
--- Checking for Missing Values ---
Country                  0
Talent                   0
Infrastructure           0
Operating Environment    0
Research                 0
Development              0
Government Strategy      0
Commercial               0
Total score              0
Region                   0
Cluster                  0
Income group             0
Political regime         0
dtype: int64

--- Checking for Duplicate Rows ---
Number of duplicate rows found: 0

--- Checking for Placeholder Missing Values ---
Count of 'N/A':
Country                  0
Talent                   0
Infrastructure           0
Operating Environment    0
Research                 0
Development              0
Government Strategy      0
Commercial               0
Total score              0
Region                   0
Cluster                  0
Income group             0
Political regime         0
dtype: int64
Count of 'NA':
Country                  0
Talent                   0
Infrastructure           0
Operating Environment    0
Research                 0
Development              0
Government Strategy      0
Commercial               0
Total score              0
Region                   0
Cluster                  0
Income group             0
Political regime         0
dtype: int64
Count of '?':
Country                  0
Talent                   0
Infrastructure           0
Operating Environment    0
Research                 0
```

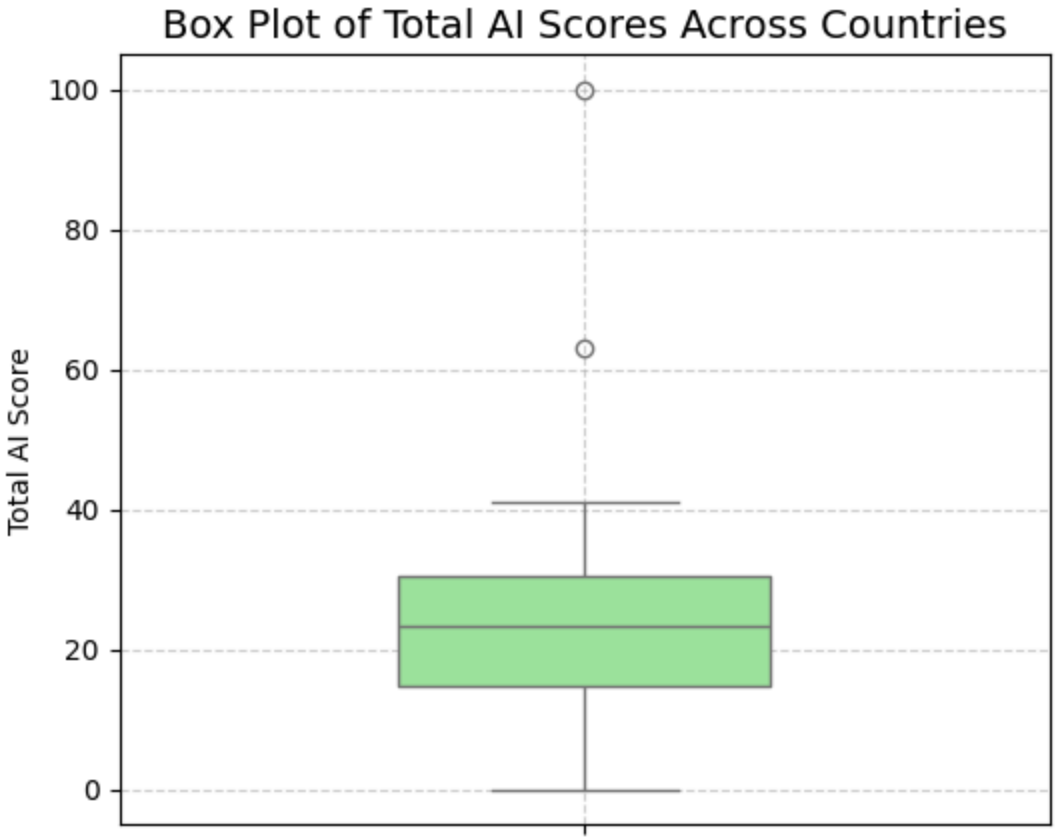
```
Development              0
Government Strategy      0
Commercial               0
Total score              0
Region                   0
Cluster                  0
Income group             0
Political regime         0
dtype: int64
Count of 'None':
Country                  0
Talent                   0
Infrastructure           0
Operating Environment    0
Research                 0
Development              0
Government Strategy      0
Commercial               0
Total score              0
Region                   0
Cluster                  0
Income group             0
Political regime         0
dtype: int64
Count of '-':
Country                  0
Talent                   0
Infrastructure           0
Operating Environment    0
Research                 0
Development              0
Government Strategy      0
Commercial               0
Total score              0
Region                   0
Cluster                  0
Income group             0
Political regime         0
dtype: int64
Count of ' ':
Country                  0
Talent                   0
Infrastructure           0
Operating Environment    0
Research                 0
Development              0
Government Strategy      0
Commercial               0
Total score              0
Region                   0
Cluster                  0
Income group             0
Political regime         0
dtype: int64
```

In [48]:
```
#  4. Explore the data

print("\n--- Getting a Statistical Overview ---")
# This provides stats like mean, min, max, and standard deviation for each n
print(df.describe())
```

```
--- Getting a Statistical Overview ---
          Talent  Infrastructure  Operating Environment    Research  \
count   62.000000       62.000000              62.000000   62.000000
mean    16.803065       63.503710              66.925484   16.610000
std     15.214963       20.217525              20.000424   17.413996
min      0.000000        0.000000               0.000000    0.000000
25%      7.365000       55.857500              58.107500    3.032500
50%     13.445000       65.230000              69.505000   12.930000
75%     24.567500       75.947500              80.500000   25.412500
max    100.000000      100.000000             100.000000  100.000000

       Development  Government Strategy  Commercial  Total score
count    62.000000            62.000000   62.000000    62.000000
mean     14.824677            57.865645    6.171935    23.914677
std      19.419279            26.252448   14.029632    15.123586
min       0.000000             0.000000    0.000000     0.000000
25%       1.202500            41.030000    0.697500    14.805000
50%       9.005000            63.930000    2.585000    23.220000
75%      19.980000            77.952500    5.307500    30.487500
max     100.000000           100.000000  100.000000   100.000000
```

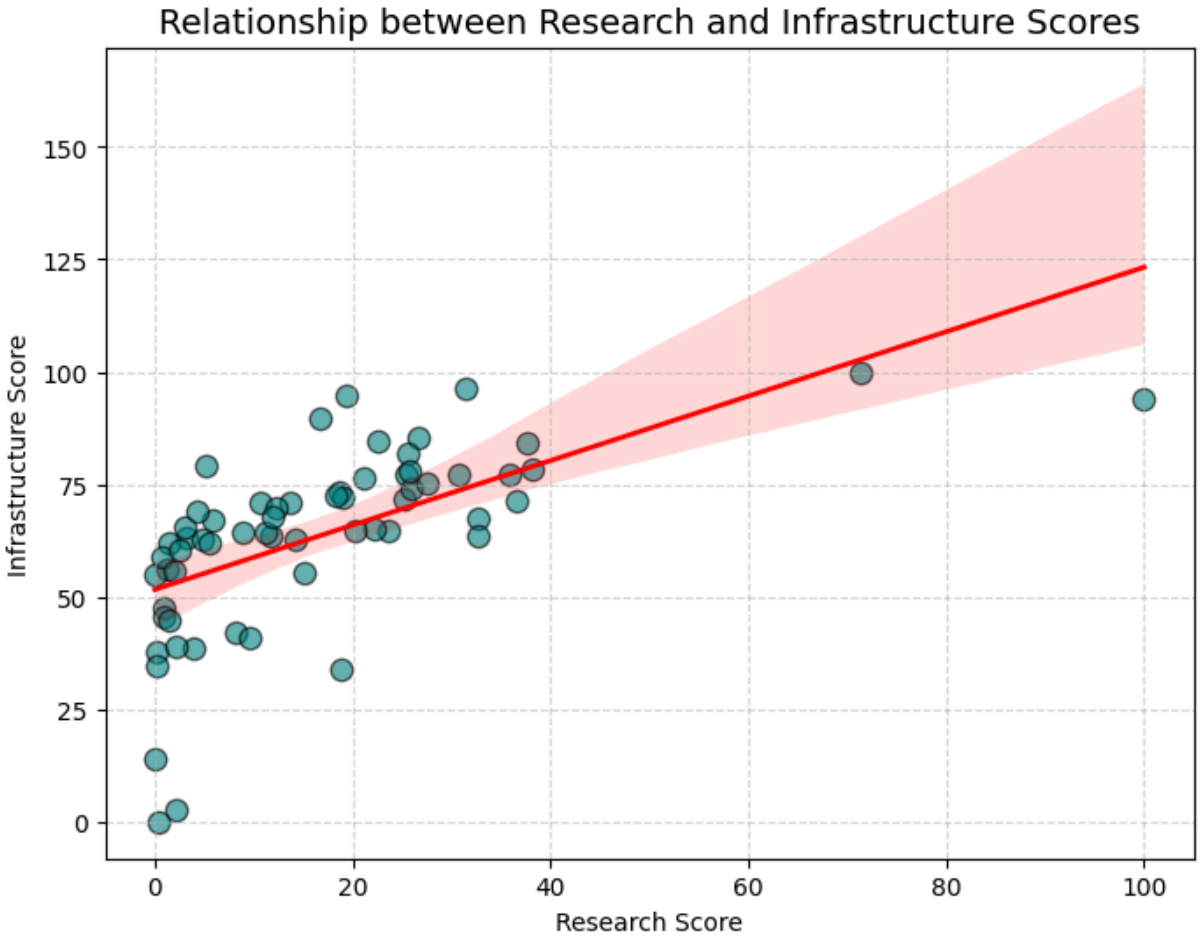
In [49]:
```
# 1. How are total AI scores distributed across countries?

# Sort countries by Total score in descending order
top_countries = df.sort_values(by='Total score', ascending=False)

# Display top 10 countries
print("---Top 10 Countries by Total AI Score---")
print(top_countries[['Country', 'Total score']])


plt.figure(figsize=(6,5))
sns.boxplot(y='Total score', data=df, color='lightgreen', width=0.4)
plt.title('Box Plot of Total AI Scores Across Countries', fontsize=14)
plt.ylabel('Total AI Score')
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

```
---Top 10 Countries by Total AI Score---
                     Country  Total score
0   United States of America       100.00
1                      China        62.92
2             United Kingdom        40.93
3                     Canada        40.19
4                     Israel        39.89
..                       ...          ...
57                Sri Lanka         6.62
58                    Egypt         4.83
59                    Kenya         2.30
60                  Nigeria         1.38
61                 Pakistan         0.00

[62 rows x 2 columns]
```



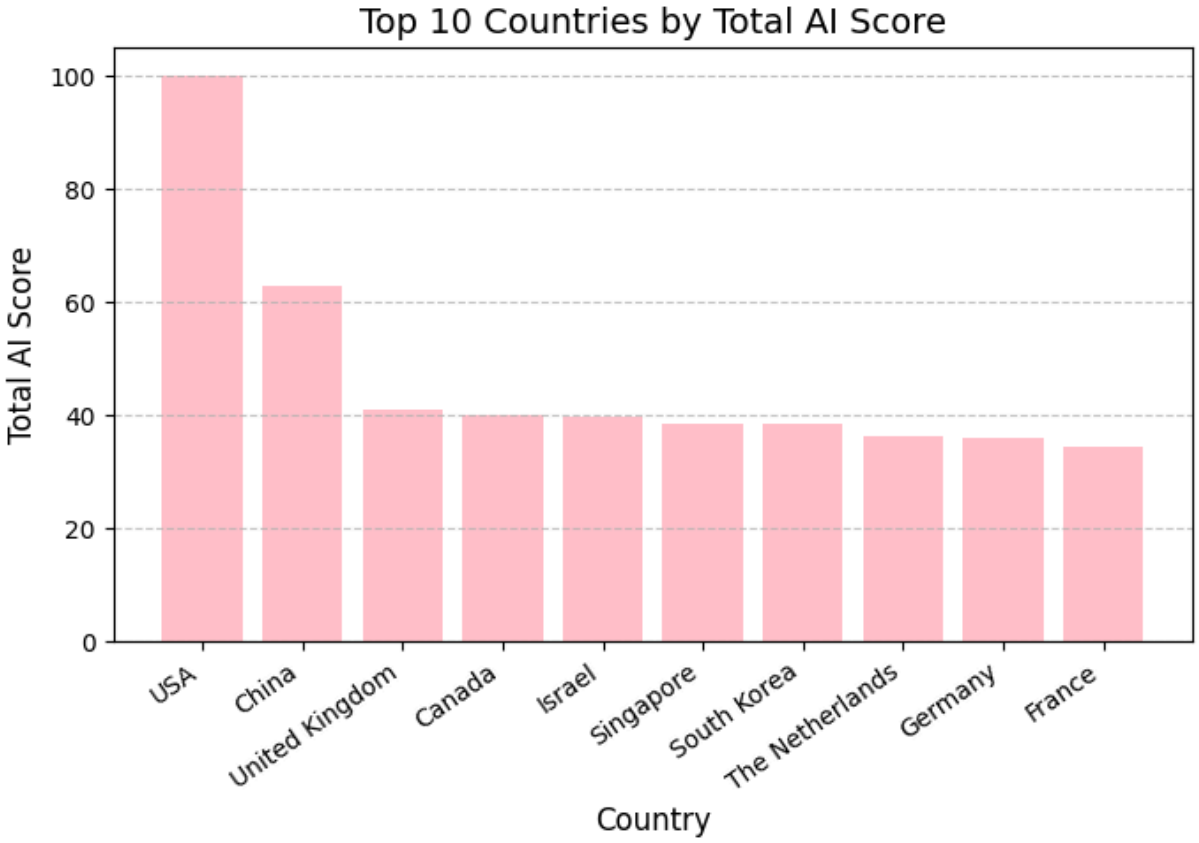
In [50]:
```
# 2.What is the relationship between Research and Infrastructure scores?

plt.figure(figsize=(8,6))

# Scatter + Regression Line together
sns.regplot(
    x='Research',
    y='Infrastructure',
    data=df,
    scatter_kws={'color': 'teal', 'alpha': 0.6, 's': 80, 'edgecolor': 'black
    line_kws={'color': 'red', 'lw': 2}
)

plt.title('Relationship between Research and Infrastructure Scores', fontsiz
```

```
plt.xlabel('Research Score')
plt.ylabel('Infrastructure Score')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```



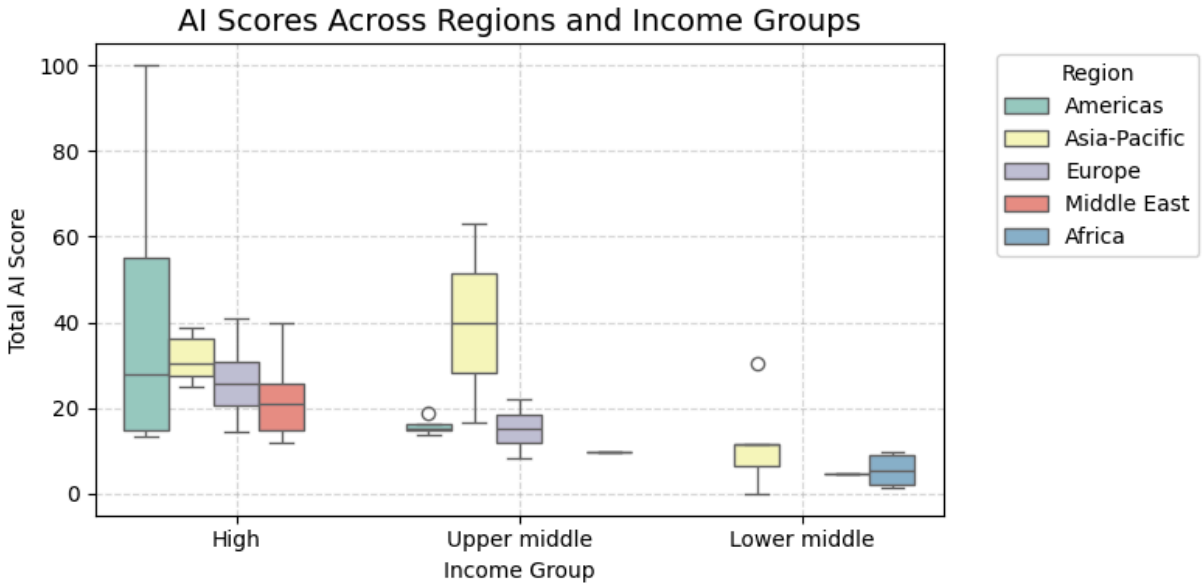
In [51]:
```
# 3. Which countries have the top 10 total AI scores?

# Replace long country name for readability
df['Country'] = df['Country'].replace({'United States of America': 'USA'})

# Sort countries by Total score in descending order
top_countries = df.sort_values(by='Total score', ascending=False)

# Display top 10 countries
print("Top 10 Countries by Total AI Score:")
print(top_countries[['Country', 'Total score']].head(10))

# Plot top 10 countries
plt.figure(figsize=(7, 5))
plt.bar(top_countries['Country'][:10], top_countries['Total score'][:10], co
plt.title('Top 10 Countries by Total AI Score', fontsize=14)
plt.xlabel('Country', fontsize=12)
plt.ylabel('Total AI Score', fontsize=12)
plt.xticks(rotation=35, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
Top 10 Countries by Total AI Score:
           Country  Total score
0              USA       100.00
1            China        62.92
2   United Kingdom        40.93
3           Canada        40.19
4           Israel        39.89
5        Singapore        38.67
6      South Korea        38.60
7  The Netherlands        36.35
8          Germany        36.04
9           France        34.42
```

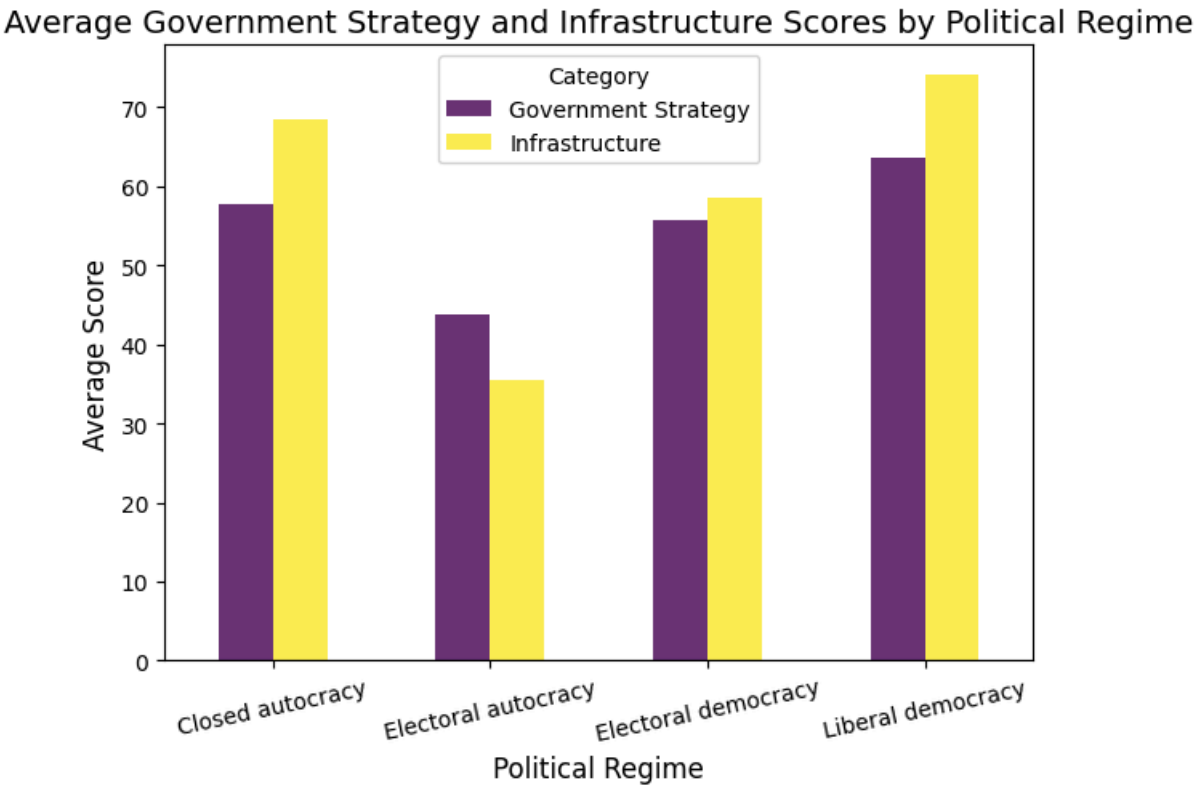


In [52]:
```
# 4.Are there noticeable differences in AI scores across regions over differ

plt.figure(figsize=(8,4))
sns.boxplot(
    x='Income group',
    y='Total score',
    hue='Region',
    data=df,
    palette='Set3'
)

plt.title('AI Scores Across Regions and Income Groups', fontsize=14)
plt.xlabel('Income Group')
plt.ylabel('Total AI Score')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, linestyle='--', alpha=0.5)
```

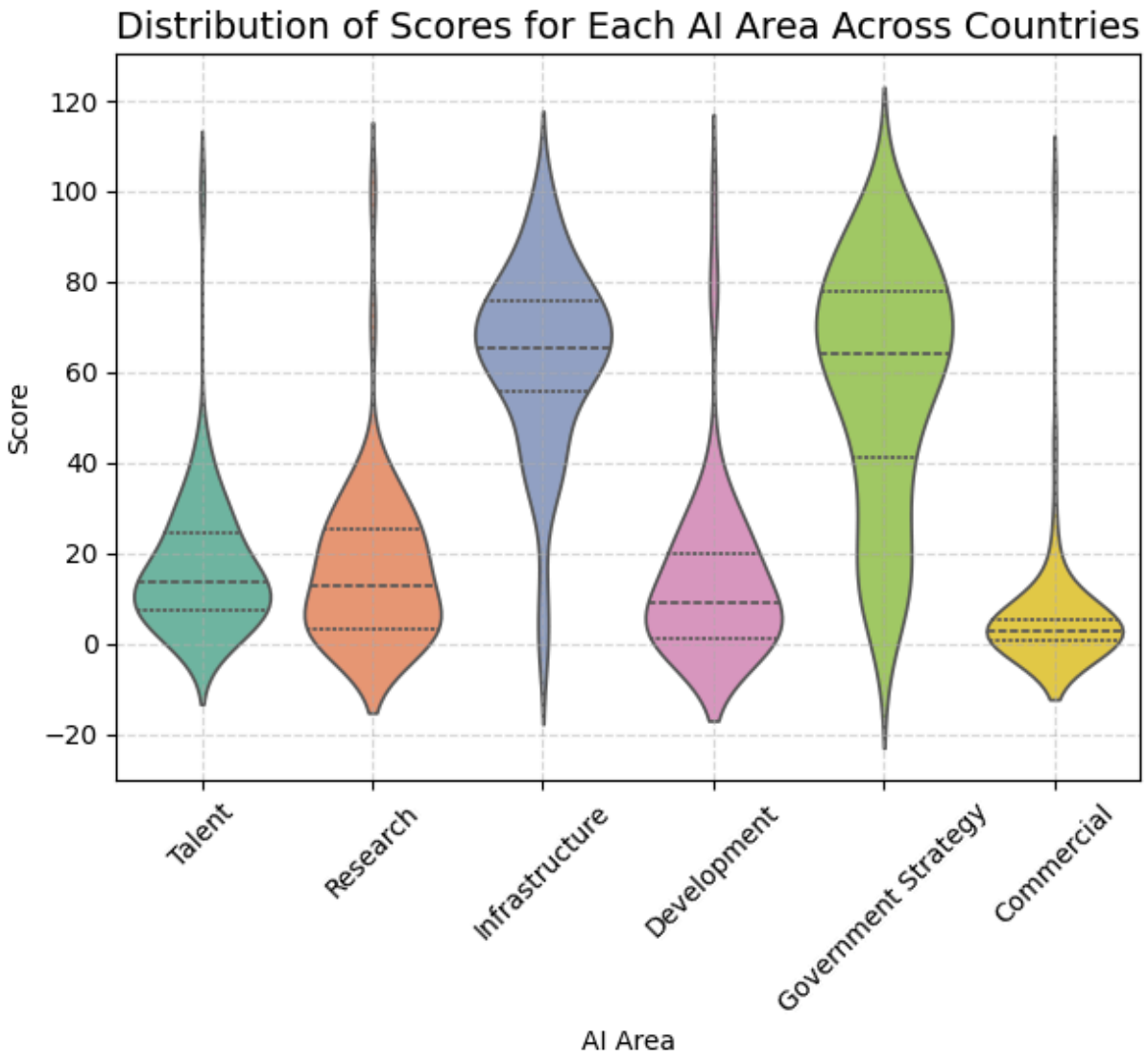
```
plt.tight_layout()
plt.show()
```



In [53]:
```
# 5. How do different political regimes differ in their average government s

# Group and get averages
avg_scores = df.groupby('Political regime')[['Government Strategy', 'Infrast

# Plot directly
avg_scores.plot(kind='bar', figsize=(7,5), colormap='viridis', alpha = 0.8)

plt.title('Average Government Strategy and Infrastructure Scores by Politica
plt.xlabel('Political Regime', fontsize=12)
plt.ylabel('Average Score', fontsize=12)
plt.legend(title='Category')
plt.xticks(rotation=12)
# plt.tight_layout()
plt.show()
```



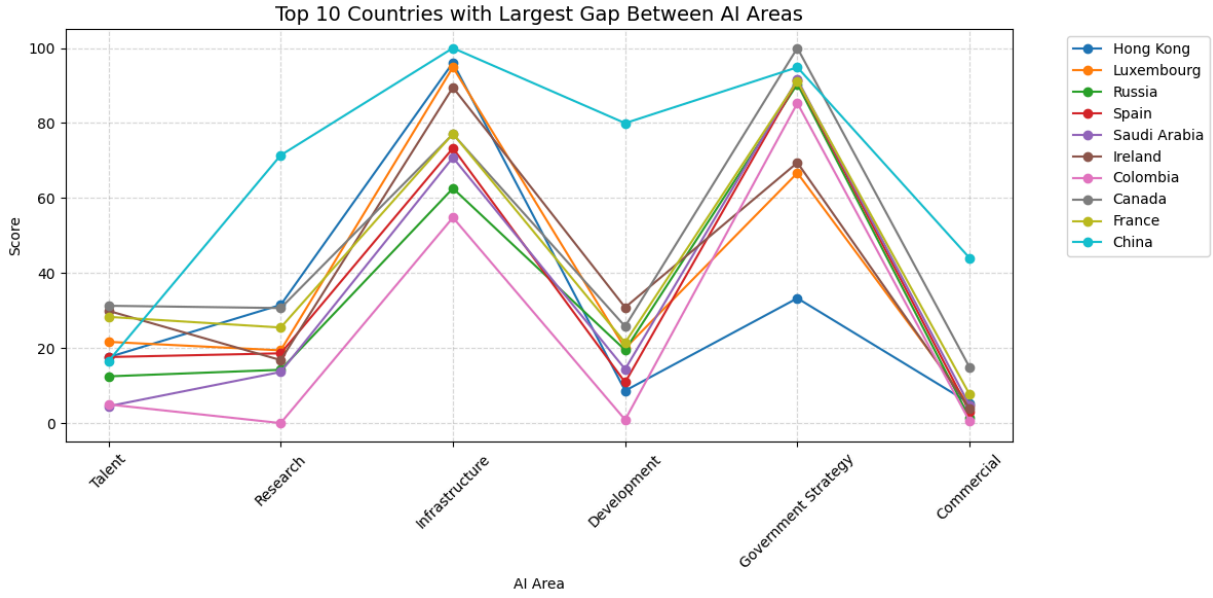
In [54]:
```
# 6. What is the distribution of scores for each AI area (Talent, Research,

# Select only AI-related score columns
ai_areas = ['Talent', 'Research', 'Infrastructure', 'Development',
            'Government Strategy', 'Commercial']

plt.figure(figsize=(7,5))
sns.violinplot(data=df[ai_areas], palette='Set2', inner='quartile')
plt.title('Distribution of Scores for Each AI Area Across Countries', fontsi
plt.xlabel('AI Area')
plt.ylabel('Score')
plt.grid(True, linestyle='--', alpha=0.5)
plt.xticks(rotation=45)
plt.show()
```



In [55]:
```
# 7. Which countries or regions show the largest gap between their highest a

# Select AI-related score columns
ai_areas = ['Talent', 'Research', 'Infrastructure', 'Development',
            'Government Strategy', 'Commercial']

# Calculate min, max, and range for each country
df['Max Score'] = df[ai_areas].max(axis=1)
df['Min Score'] = df[ai_areas].min(axis=1)
df['Score Range'] = df['Max Score'] - df['Min Score']
print("---Countries or regions show the largest gap between their highest ar
print()

# Sort countries by range (largest first)
top_gaps = df.sort_values('Score Range', ascending=False).head(10)
print(top_gaps[['Country', 'Max Score', 'Min Score', 'Score Range']])

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

for i, row in top_gaps.iterrows():
    plt.plot(ai_areas, row[ai_areas], marker='o', label=row['Country'])
```

```
plt.title('Top 10 Countries with Largest Gap Between AI Areas', fontsize=14)
plt.xlabel('AI Area')
plt.ylabel('Score')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, linestyle='--', alpha=0.5)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
---Countries or regions show the largest gap between their highest and lowes
t area scores---

         Country  Max Score  Min Score  Score Range
19     Hong Kong      96.11       5.30        90.81
14    Luxembourg      94.88       4.68        90.20
31        Russia      90.40       1.38        89.02
20         Spain      91.28       3.08        88.20
25  Saudi Arabia      91.63       4.49        87.14
11       Ireland      89.50       3.94        85.56
48      Colombia      85.29       0.00        85.29
3         Canada     100.00      14.88        85.12
9         France      91.20       7.65        83.55
1          China     100.00      16.51        83.49
```



In [56]:

```
# 5. Transform Data

# To line cleans and standardizes all your column names
df.columns = df.columns.str.lower().str.replace(' ', '_')
df.head()
print("---Column names after standardizing---")
print(df.columns)
print()

# To check for consistencies - returns an array of all the distinct (unique)
print("---Unique values in income group---")
print(df['income_group'].unique())
print()
print("---Unique values in cluster---")
```

```
print(df['cluster'].unique())
print()
print("---Unique values in political regime---")
print(df['political_regime'].unique())
```

```
---Column names after standardizing---
Index(['country', 'talent', 'infrastructure', 'operating_environment',
       'research', 'development', 'government_strategy', 'commercial',
       'total_score', 'region', 'cluster', 'income_group', 'political_regim
e',
       'max_score', 'min_score', 'score_range'],
      dtype='object')

---Unique values in income group---
['High' 'Upper middle' 'Lower middle']

---Unique values in cluster---
['Power players' 'Traditional champions' 'Rising stars' 'Waking up'
 'Nascent']

---Unique values in political regime---
['Liberal democracy' 'Closed autocracy' 'Electoral democracy'
 'Electoral autocracy']
```

In [57]:
```
# Encode categorical variables

# Display all unique values present in the 'region' column
print("---All unique values present in the 'region' column---")
print(df.region.unique())
# Create a dictionary to map each region name to a numerical value
region_mapping = {
    'Americas': 0,
    'Asia-Pacific': 1,
    'Europe': 2,
    'Middle East': 3,
    'Africa': 4
}
# Replace region names in the 'region' column with their corresponding numer
df['region'] = df['region'].map(region_mapping)
print()

# Display all unique values present in the 'cluster' column
print("---All unique values present in the 'cluster' column---")
print(df.cluster.unique())
# Create a dictionary to map each cluster name to a numerical value
cluster_mapping = {
    'Power players': 0,
    'Traditional champions': 1,
    'Rising stars': 2,
    'Waking up': 3,
    'Nascent': 4
}
# Replace region names in the 'cluster' column with their corresponding nume
df['cluster'] = df['cluster'].map(cluster_mapping)
print()
```

```
# Display all unique values present in the 'income_group' column
print("---All unique values present in the 'income_group' column---")
print(df.income_group.unique())
# Create a dictionary to map each income_group name to a numerical value
income_mapping = {
    'High': 0,
    'Upper middle': 1,
    'Lower middle': 2,
}
# Replace region names in the 'income_group' column with their corresponding
df['income_group'] = df['income_group'].map(income_mapping)
print()

# Display all unique values present in the 'political_regime' column
print("---All unique values present in the 'political_regime' column---")
print(df.political_regime.unique())
# Create a dictionary to map each political_regime name to a numerical value
political_regime_mapping = {
    'Liberal democracy': 0,
    'Closed autocracy': 1,
    'Electoral democracy': 2,
    'Electoral autocracy': 3
}
# Replace region names in the 'political_regime' column with their correspon
df['political_regime'] = df['political_regime'].map(political_regime_mapping
print()

print("---Replaced columns with their corresponding numeric codes---")
print(df.head())
```

```
---All unique values present in the 'region' column---
['Americas' 'Asia-Pacific' 'Europe' 'Middle East' 'Africa']

---All unique values present in the 'cluster' column---
['Power players' 'Traditional champions' 'Rising stars' 'Waking up'
 'Nascent']

---All unique values present in the 'income_group' column---
['High' 'Upper middle' 'Lower middle']

---All unique values present in the 'political_regime' column---
['Liberal democracy' 'Closed autocracy' 'Electoral democracy'
 'Electoral autocracy']

---Replaced columns with their corresponding numeric codes---
          country  talent  infrastructure  operating_environment  research
\
0             USA  100.00           94.02                  64.56    100.00
1           China   16.51          100.00                  91.57     71.42
2  United Kingdom   39.65           71.43                  74.65     36.50
3          Canada   31.28           77.05                  93.94     30.67
4          Israel   35.76           67.58                  82.44     32.63

   development  government_strategy  commercial  total_score  region  cluste
r  \
0       100.00                77.39      100.00       100.00       0
0
1        79.97                94.87       44.02        62.92       1
0
2        25.03                82.82       18.91        40.93       2
1
3        25.78               100.00       14.88        40.19       0
1
4        27.96                43.91       27.33        39.89       3
2

   income_group  political_regime  max_score  min_score  score_range
0             0                 0     100.00      77.39        22.61
1             1                 1     100.00      16.51        83.49
2             0                 0      82.82      18.91        63.91
3             0                 0     100.00      14.88        85.12
4             0                 0      67.58      27.33        40.25
```

In [58]:
```
# Create new variables from existing ones

#  Example: balance between Research and Development
df['RnD_ratio'] = df['research'] / (df['development'] + 1e-6)

# Example: average technical capability
df['tech_strength'] = (df['talent'] + df['infrastructure'] + df['research'])

# Example: overall policy strength
df['policy_effectiveness'] = (df['government_strategy'] + df['operating_envi

print("---After adding three new columns - RnD_ratio, tech_strength, policy_
```

```
print(df.head())
```

```
---After adding three new columns - RnD_ratio, tech_strength, policy_effecti
veness---
          country  talent  infrastructure  operating_environment  research
\
0             USA  100.00           94.02                  64.56    100.00
1           China   16.51          100.00                  91.57     71.42
2  United Kingdom   39.65           71.43                  74.65     36.50
3          Canada   31.28           77.05                  93.94     30.67
4          Israel   35.76           67.58                  82.44     32.63

   development  government_strategy  commercial  total_score  region  cluste
r  \
0       100.00                77.39      100.00       100.00       0
0
1        79.97                94.87       44.02        62.92       1
0
2        25.03                82.82       18.91        40.93       2
1
3        25.78               100.00       14.88        40.19       0
1
4        27.96                43.91       27.33        39.89       3
2

   income_group  political_regime  max_score  min_score  score_range  \
0             0                 0     100.00      77.39        22.61
1             1                 1     100.00      16.51        83.49
2             0                 0      82.82      18.91        63.91
3             0                 0     100.00      14.88        85.12
4             0                 0      67.58      27.33        40.25

   RnD_ratio  tech_strength  policy_effectiveness
0   1.000000      98.006667                70.975
1   0.893085      62.643333                93.220
2   1.458250      49.193333                78.735
3   1.189682      46.333333                96.970
4   1.167024      45.323333                63.175
```

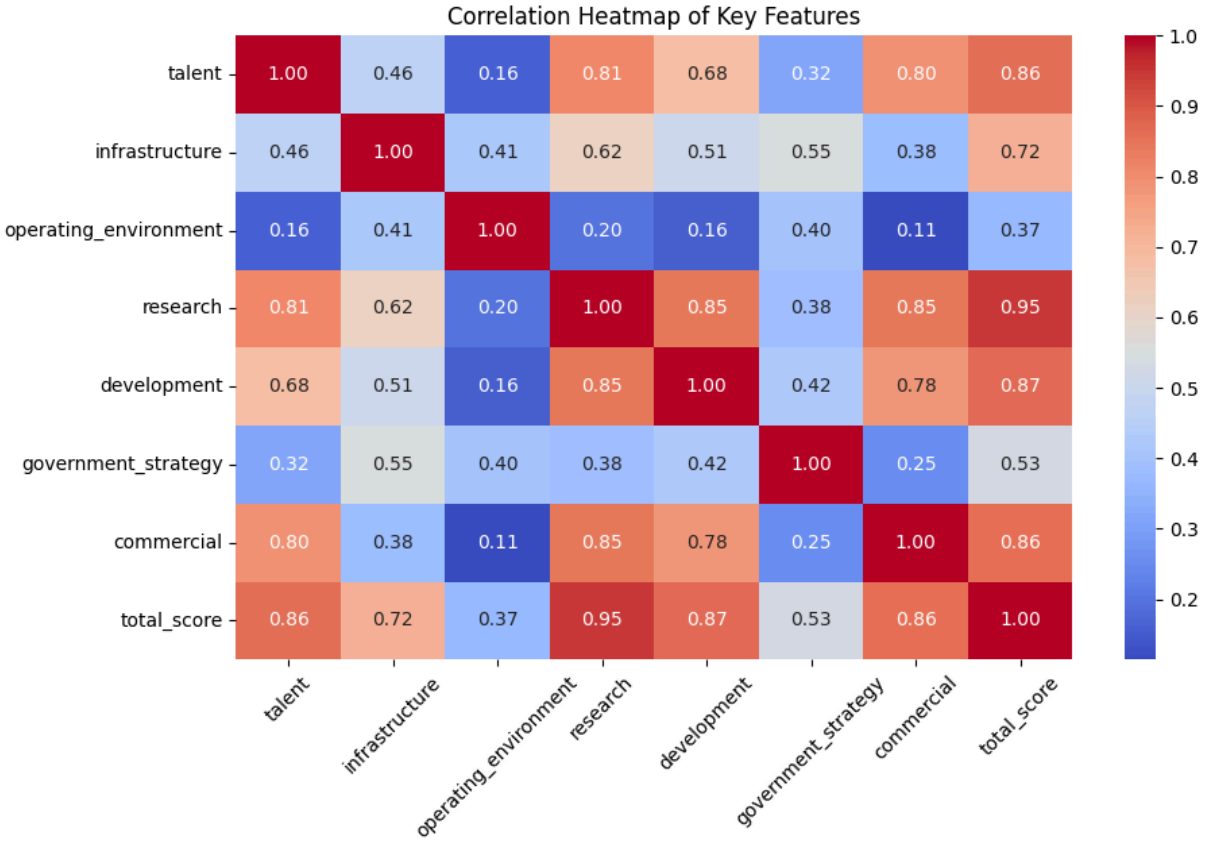
In [59]:
```
# Aggregate/group data based on specific variables

# Average AI score by Region
region_avg = df.groupby('region')[['total_score']].mean().reset_index()

# Average AI score by Region and Income Group
region_income_avg = df.groupby(['region', 'income_group'])[['total_score']].

# Average AI score by Political Regime
political_avg = df.groupby('political_regime')[['total_score']].mean().reset

# Display results
print("---Average Total AI Score by Region---")
print(region_avg, "\n")

print("---Average Total AI Score by Region and Income Group---")
```

```
print(region_income_avg, "\n")

print("---Average Total AI Score by Political Regime---")
print(political_avg)

# RnD_ratio representing the balance between research and development
# tech_strength as an average of talent, infrastructure, and research
# policy_effectiveness as the mean of government strategy and operating envi
```

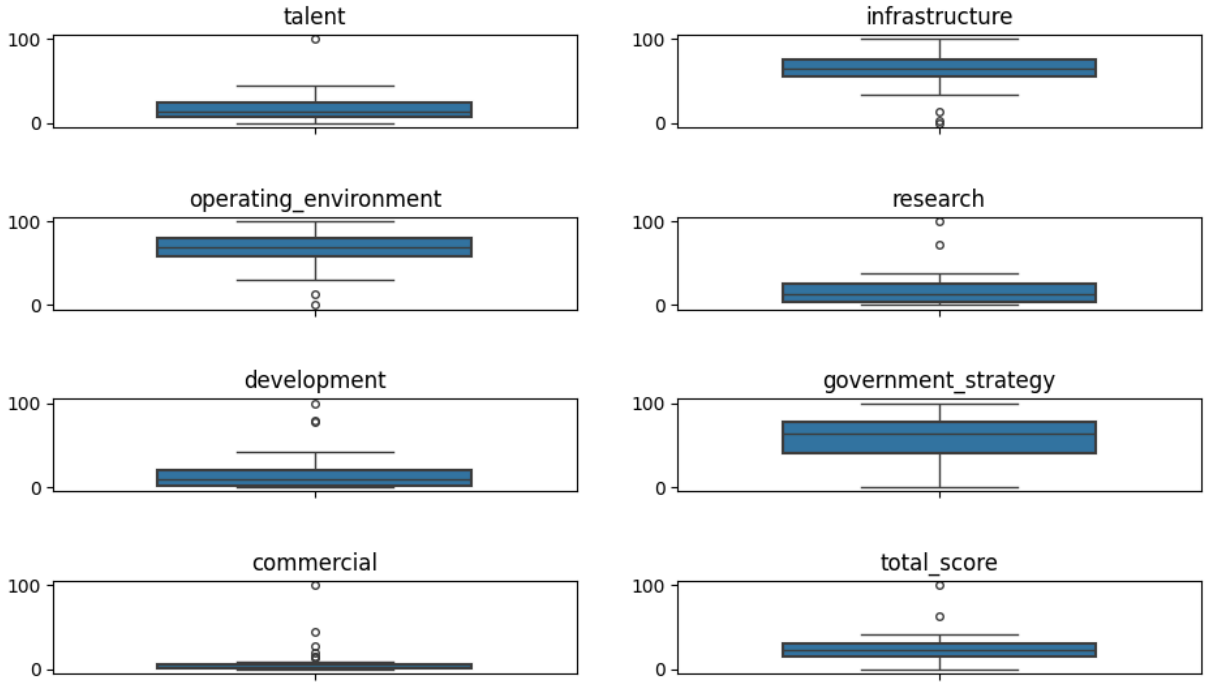
```
---Average Total AI Score by Region---
   region  total_score
0       0    29.031250
1       1    25.792143
2       2    25.493103
3       3    19.656667
4       4     6.426000

---Average Total AI Score by Region and Income Group---
    region  income_group  total_score
0        0             0    42.197500
1        0             1    15.865000
2        1             0    31.634286
3        1             1    39.790000
4        1             2    12.014000
5        2             0    26.252593
6        2             1    15.240000
7        3             0    22.622000
8        3             2     4.830000
9        4             1     9.710000
10       4             2     5.605000

---Average Total AI Score by Political Regime---
   political_regime  total_score
0                 0    31.948519
1                 1    22.375714
2                 2    18.447500
3                 3    11.815000
```

In [60]:
```
# 6) Visualize Correlations

plt.figure(figsize=(10, 6))

numerical_cols = ['talent', 'infrastructure', 'operating_environment', 'rese
       'development', 'government_strategy', 'commercial', 'total_score']

correlation_matrix = df[numerical_cols].corr() #positive corelation (if one
print("---Strong Positive Correlations (≥ 0.7)---\n")
print(correlation_matrix[correlation_matrix >= 0.7])
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f") #blu
plt.title('Correlation Heatmap of Key Features')
plt.xticks(rotation=45)
plt.show()
```

```
---Strong Positive Correlations (≥ 0.7)---

                         talent  infrastructure  operating_environment  \
talent                 1.000000             NaN                    NaN
infrastructure              NaN        1.000000                    NaN
operating_environment       NaN             NaN                    1.0
research               0.810255             NaN                    NaN
development                 NaN             NaN                    NaN
government_strategy         NaN             NaN                    NaN
commercial             0.795071             NaN                    NaN
total_score            0.861969        0.716481                    NaN

                        research  development  government_strategy  commercia
l  \
talent                 0.810255          NaN                  NaN    0.79507
1
infrastructure              NaN          NaN                  NaN         Na
N
operating_environment       NaN          NaN                  NaN         Na
N
research               1.000000     0.845912                  NaN    0.84735
3
development            0.845912     1.000000                  NaN    0.77592
9
government_strategy         NaN          NaN                  1.0         Na
N
commercial             0.847353     0.775929                  NaN    1.00000
0
total_score            0.945877     0.866337                  NaN    0.85798
5

                       total_score
talent                    0.861969
infrastructure            0.716481
operating_environment          NaN
research                  0.945877
development               0.866337
government_strategy            NaN
commercial                0.857985
total_score               1.000000
```



In [61]:
```
# 7.  Handle Outliers

# Select numerical columns and drop the last 6
numeric_df = df[['talent', 'infrastructure',  'operating_environment', 'rese

# Create subplots with larger boxes
n_cols = 2
n_rows = (len(numeric_df.columns) + 1) // n_cols

plt.figure(figsize=(10, 6))  # increased overall figure size
# plt.title('Box Plots of Numerical Columns with Outliers Removed', fontsize
print()

for i, col in enumerate(numeric_df.columns, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.boxplot(y=numeric_df[col], width=0.6, fliersize=4, boxprops=dict(lir
    plt.title(col, fontsize=12)
    plt.ylabel("")  # optional: hide y-axis label for a cleaner look

plt.tight_layout(pad=3.0)  # more spacing between subplots
plt.show()
```


talent
infrastructure
operating_environment
research
development
government_strategy
commercial
total_score
100
0