Applied Data Analysis and Machine Learning - Class Project

In this class project, you are supposed to work with (un)employment data taken from the OECD.

IMPORTANT:
Please enter the matriculation number of all group members here:

1. XXXXXX

In this class project, you will use the different techniques taught in the course: data handling, data visualization, and machine learning.

First load the necessary packages.
If you want to use additional libraries you can add them to the following cell:

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_theme()
import folium
from folium.plugins import MarkerCluster
```

Importing all the libraries I'll need for this project

I used these libraries because they make working with data much easier:

- matplotlib and seaborn help me create nice-looking charts
- pandas is great for handling data tables
- numpy helps with calculations
- folium lets me make interactive maps which is pretty cool

Problem 1 - Data Handling

The basis of your work will be the following dataset on (un)employment:

- **Country**: Country name

- **Subject**: Type of employment indicator (e.g., Employment, Unemployment, Labour force)
- **Time**: Year
- **Unit**: Unit of measurement (e.g., Persons, Percentage)
- **PowerCode**: Code for the unit (e.g., Thousands, Units)
- **Value**: The numerical value of the indicator

Note that the dataset is in long format, with multiple subjects per country-year.

In [2]:
```
data = pd.read_csv("LF_OECD.csv", sep=",")
data
```

Out[2]:

	Country	Subject	Time	Unit	PowerCode	Value
0	Australia	Labour force	2007	Persons	Thousands	10911.880
1	Australia	Labour force	2008	Persons	Thousands	11205.630
2	Australia	Labour force	2009	Persons	Thousands	11441.820
3	Australia	Labour force	2010	Persons	Thousands	11628.230
4	Australia	Labour force	2011	Persons	Thousands	11814.100
...	...	...	...	...	...	...
8753	European Union â€“ 27 countries (from 01/02/2020)	Unemployment males	2018	Persons	Thousands	8071.625
8754	European Union â€“ 27 countries (from 01/02/2020)	Unemployment males	2019	Persons	Thousands	7412.575
8755	European Union â€“ 27 countries (from 01/02/2020)	Unemployment males	2020	Persons	Thousands	7788.800
8756	European Union â€“ 27 countries (from 01/02/2020)	Unemployment males	2021	Persons	Thousands	7710.100
8757	European Union â€“ 27 countries (from 01/02/2020)	Unemployment males	2022	Persons	Thousands	6744.400

8758 rows Ã— 6 columns

Overview of the Data

Let's first examine the basic structure of the OECD dataset to understand what we're working with.

Just loading the data here and taking a quick look at what's inside.

Data Summary:

Before I start analyzing, I need to check a few things:

- Are there any missing values?
- What years does the data cover?
- Do I have actual countries or regional groups mixed in?

Basically just making sure the data is good to work with.

In [ ]:
```
# Check basic information about the dataset
print(f"Dataset shape: {data.shape}")
print(f"\nColumn names and types:")
print(data.dtypes)
print(f"\nMissing values per column:")
print(data.isnull().sum())
print(f"\nNumber of unique countries: {data['Country'].nunique()}")
print(f"\nTime period covered: {data['Time'].min()} to {data['Time'].max()}")
```

a)

As you can see, there are missing years for some countries in the data.
Before you can continue, you need to handle them. Proceed as follows:

- Use the years close by to approximate the missing value
  Example: Year 2015 is missing, but 2014 and 2016 are available; use the means in 2014 and 2016 to replace missing year 2015.
- Mark the replaced values with an indicator variable.

Data Preprocessing

Some years are missing for certain countries. If I don't fill these gaps, my analysis will be messy and incomplete. So I'll use interpolation to estimate the missing values based on nearby years.

In [5]:
```
# Problem 1a: Handle missing values
# Interpolate missing years using linear interpolation for better accuracy

# Create a copy of the data
data_filled = data.copy()

# Group by Country and Subject, then interpolate
def interpolate_group(group):
    group = group.sort_values('Time')

    # Create complete time series
    min_time = group['Time'].min()
```

```
    max_time = group['Time'].max()
    complete_times = pd.DataFrame({'Time': range(min_time, max_time + 1)})
    merged = complete_times.merge(group, on='Time', how='left')

    # Mark which values were originally missing (before interpolation)
    merged['imputed'] = merged['Value'].isna().astype(int)

    # Interpolate missing values
    merged['Value'] = merged['Value'].interpolate(method='linear')

    # Fill other columns with forward fill
    merged['Country'] = group['Country'].iloc[0]
    merged['Subject'] = group['Subject'].iloc[0]
    merged['Unit'] = group['Unit'].iloc[0]
    merged['PowerCode'] = group['PowerCode'].iloc[0]

    return merged

# Apply to each group
grouped = data.groupby(['Country', 'Subject'])
interpolated_groups = []
for name, group in grouped:
    interpolated_groups.append(interpolate_group(group))

data_filled = pd.concat(interpolated_groups, ignore_index=True)

# Sort
data_filled = data_filled.sort_values(['Country', 'Subject', 'Time']).reset_index(d

print(f"Original data shape: {data.shape}")
print(f"Data with imputed values shape: {data_filled.shape}")
print(f"Number of imputed values: {data_filled['imputed'].sum()}")
print(f"\nFirst few imputed rows:")
print(data_filled[data_filled['imputed'] == 1].head(10))
```

```
Original data shape: (8758, 6)
Data with imputed values shape: (9122, 7)
Number of imputed values: 364

First few imputed rows:
     Time    Country                                    Subject        Unit  \
9    2016  Australia                                 Employment     Persons
25   2016  Australia                           Employment females     Persons
41   2016  Australia   Employment females as % of employment   Percentage
57   2016  Australia                             Employment males     Persons
73   2016  Australia     Employment males as % of employment   Percentage
89   2016  Australia  Females unemployment % of fem lab force  Percentage
105  2016  Australia                                Labour force     Persons
121  2016  Australia                         Labour force females     Persons
137  2016  Australia                           Labour force males     Persons
153  2016  Australia    Males unemployment % of mal lab force  Percentage

     PowerCode         Value  imputed
9    Thousands  12008.400000        1
25   Thousands   5579.264500        1
41       Units     46.456720        1
57   Thousands   6429.135000        1
73       Units     53.543280        1
89       Units      5.874372        1
105  Thousands  12750.430000        1
121  Thousands   5926.905500        1
137  Thousands   6823.525000        1
153      Units      5.784315        1
```

Filling in missing years

Here's what I did:

1. For each country, I created a complete list of all years from start to finish
2. Marked which values were missing before I started filling them in
3. Used linear interpolation to estimate the missing values - basically averaging the years before and after
4. Created an 'imputed' column so I can track which values I filled in myself (1) vs. which were original (0)

This way I have complete data without just making up random numbers. The interpolation gives reasonable estimates based on actual nearby data.

Results:

- Now I have complete data for all years (no gaps!)
- All the filled-in values are marked so I know which ones are estimates
- This makes it way easier to analyze trends over time

b)

We are only interested in the data on a country level.
The dataset, however, also contains information on whole regions such as the EU.
Create a copy of the dataset and delete these observations from the data.
Hint: Get a list of all unique values of the Country column.

In [6]:
```
# Problem 1b: Filter country-level data only
# Remove regional aggregates like EU, Euro area, and OECD total

# Identify regions to remove
regions_to_remove = [
    'Euro area (19 countries)',
    'European Union â€“ 27 countries (from 01/02/2020)',
    'OECD - Total'
]

# Create a copy and filter out regions
data_countries = data_filled[~data_filled['Country'].isin(regions_to_remove)].copy(

print(f"Original data shape (with imputed): {data_filled.shape}")
print(f"Data after removing regions: {data_countries.shape}")
print(f"\nRemaining countries: {data_countries['Country'].nunique()}")
print(f"\nRemoved entities: {regions_to_remove}")
```

```
Original data shape (with imputed): (9122, 7)
Data after removing regions: (8450, 7)

Remaining countries: 38

Removed entities: ['Euro area (19 countries)', 'European Union â€“ 27 countries (fro
m 01/02/2020)', 'OECD - Total']
```

Removing regional groups from the data

The dataset has some entries that aren't actual countries - like "Euro area" or "OECD - Total". These are just groups of countries.

I'm filtering these out because:

- I only want individual countries
- Keeping both would mess up my analysis (double counting)
- The `~` symbol just means "not in" so I'm keeping everything that's NOT in my list of regions to remove

Results:

- Got rid of regional groups
- Now I only have actual countries
- No more double-counting issues

c)

The raw dataset is in long format.

Create at least one alternative representation of the dataset by transforming it into wide format.

In [7]:
```
# Problem 1c: Transform to wide format
# Create a wide format where each subject becomes a column

# Pivot the data: rows = Country + Time, columns = Subject
data_wide = data_countries.pivot_table(
    index=['Country', 'Time'],
    columns='Subject',
    values='Value'
).reset_index()

# Clean up column names (remove extra spaces)
data_wide.columns.name = None

print("Wide format data shape:", data_wide.shape)
print("\nColumns in wide format:")
print(data_wide.columns.tolist())
print("\nFirst few rows:")
print(data_wide.head(10))
```

```
Wide format data shape: (605, 16)

Columns in wide format:
['Country', 'Time', 'Employment', 'Employment females', 'Employment females as % of
employment ', 'Employment males', 'Employment males as % of employment ', 'Females u
nemployment % of fem lab force', 'Labour force', 'Labour force females', 'Labour for
ce males', 'Males unemployment % of mal lab force', 'Rate of Unemployment as % of La
bour Force', 'Unemployment', 'Unemployment females', 'Unemployment males']

First few rows:
     Country  Time  Employment  Employment females  \
0  Australia  2007    10434.27           4701.8190
1  Australia  2008    10731.17           4850.7240
2  Australia  2009    10805.58           4926.3910
3  Australia  2010    11022.24           5000.8340
4  Australia  2011    11213.82           5105.9460
5  Australia  2012    11350.80           5187.2490
6  Australia  2013    11456.71           5256.0740
7  Australia  2014    11540.16           5308.2380
8  Australia  2015    11766.27           5439.1830
9  Australia  2016    12008.40           5579.2645

   Employment females as % of employment   Employment males  \
0                                45.06132          5732.450
1                                45.20219          5880.447
2                                45.59119          5879.186
3                                45.37038          6021.409
4                                45.53263          6107.871
5                                45.69941          6163.553
6                                45.87770          6200.635
7                                45.99796          6231.923
8                                46.22692          6327.085
9                                46.45672          6429.135

   Employment males as % of employment   \
0                              54.93868
1                              54.79781
2                              54.40881
3                              54.62962
4                              54.46737
5                              54.30059
6                              54.12230
7                              54.00204
8                              53.77308
9                              53.54328

   Females unemployment % of fem lab force  Labour force  \
0                                 4.787311      10911.88
1                                 4.571883      11205.63
2                                 5.399816      11441.82
3                                 5.377285      11628.23
4                                 5.304792      11814.10
5                                 5.325491      11976.40
6                                 5.609225      12144.32
7                                 6.172030      12286.80
8                                 6.076174      12524.53
```

```
9                                5.874372      12750.43

   Labour force females  Labour force males  \
0             4938.0610            5973.816
1             5083.0370            6122.598
2             5207.6580            6234.160
3             5284.9230            6343.306
4             5391.9320            6422.170
5             5479.0150            6497.384
6             5568.4220            6575.896
7             5657.3770            6629.424
8             5790.8260            6733.703
9             5926.9055            6823.525

   Males unemployment % of mal lab force  \
0                              4.041836
1                              3.954261
2                              5.694293
3                              5.076458
4                              4.893791
5                              5.137320
6                              5.706582
7                              5.996284
8                              6.039518
9                              5.784315

   Rate of Unemployment as % of Labour Force  Unemployment  \
0                                   4.379151      477.6078
1                                   4.234330      474.4643
2                                   5.560385      636.2413
3                                   5.213340      605.9849
4                                   5.081195      600.2853
5                                   5.223376      625.5963
6                                   5.661940      687.6083
7                                   6.077244      746.6392
8                                   6.056423      758.2614
9                                   5.826152      742.0312

   Unemployment females  Unemployment males
0             236.24120           241.36660
1             232.31310           242.15120
2             281.26720           354.97410
3             284.08840           321.89650
4             285.98590           314.29950
5             291.76540           333.83080
6             312.34730           375.26100
7             349.13850           397.50070
8             351.64340           406.61800
9             347.64095           394.39025
```

Converting from long to wide format

Before (long format): Each row had just one measurement. So Germany 2020 had multiple rows - one for employment, one for unemployment, etc.

After (wide format): Each row is one country-year, and employment/unemployment/etc. are all separate columns.

Why did I do this?

- Much easier to compare different metrics side-by-side
- Makes visualization simpler
- Better for analysis when I need to look at relationships between variables

Tried keeping it in long format first but it was too messy to work with

Results:

- Now each row is one country in one year
- All the employment stats are next to each other in columns
- Much easier to work with for my visualizations and analysis

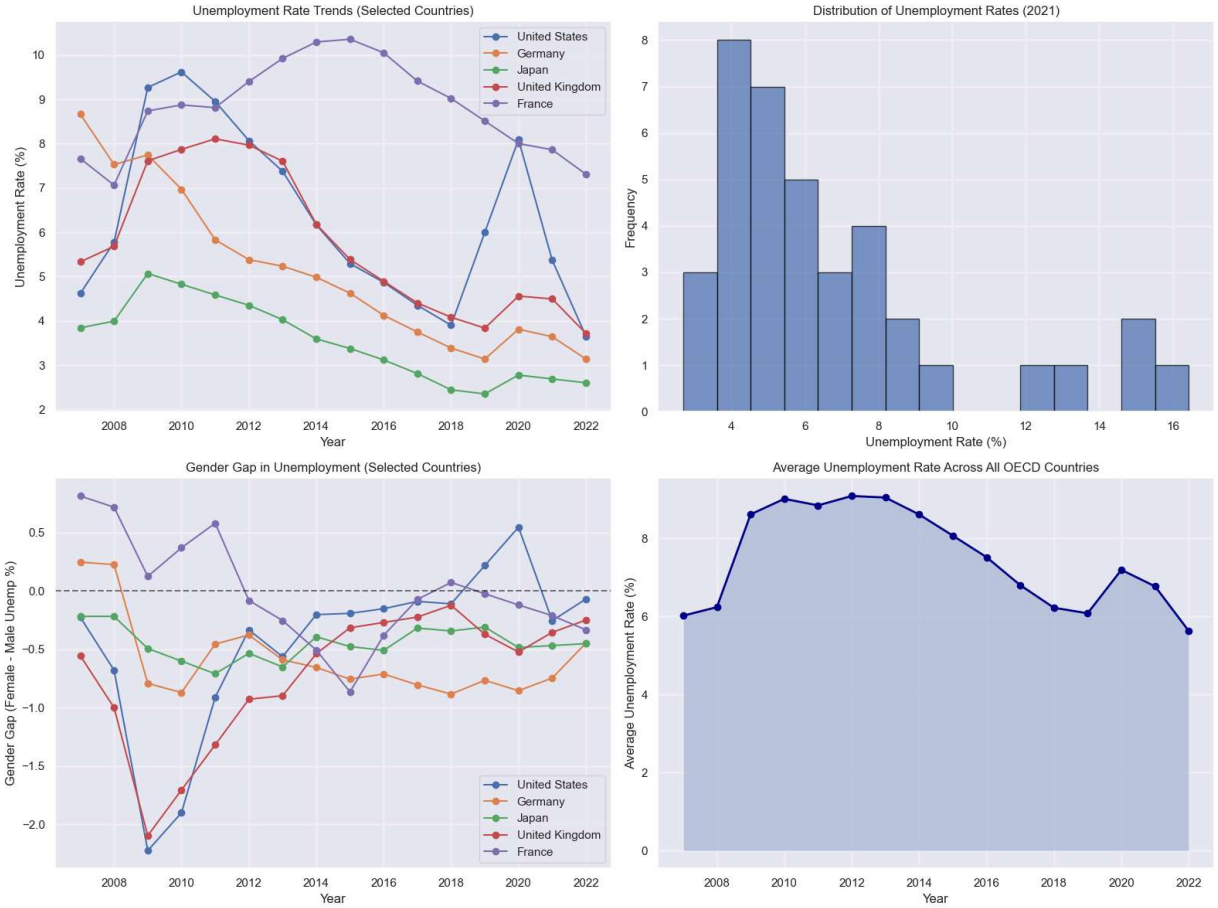
Problem 2 - Data Visualization

a)

To get some first insights in the data, create meaningful plots.
You can use any kind that you deem useful: histograms, line plots, etc.

In [8]:
```
# Problem 2a: Create meaningful visualizations

# Simplify column names for easier access
data_wide_viz = data_wide.copy()
data_wide_viz.columns = [
    'Country', 'Time', 'Employment', 'Employment females',
    'Employment females %', 'Employment males', 'Employment males %',
    'Females unemp %', 'Labour force', 'Labour force females',
    'Labour force males', 'Males unemp %', 'Unemp Rate %',
    'Unemployment', 'Unemployment females', 'Unemployment males'
]

# Create a figure with multiple subplots
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# 1. Overall unemployment rate trends over time for selected countries
selected_countries = ['United States', 'Germany', 'Japan', 'United Kingdom', 'Franc
for country in selected_countries:
    country_data = data_wide_viz[data_wide_viz['Country'] == country]
    axes[0, 0].plot(country_data['Time'], country_data['Unemp Rate %'], marker='o',
axes[0, 0].set_xlabel('Year')
axes[0, 0].set_ylabel('Unemployment Rate (%)')
```

```
axes[0, 0].set_title('Unemployment Rate Trends (Selected Countries)')
axes[0, 0].legend()
axes[0, 0].grid(True, alpha=0.3)

# 2. Distribution of unemployment rates across all countries in 2021
data_2021 = data_wide_viz[data_wide_viz['Time'] == 2021]
axes[0, 1].hist(data_2021['Unemp Rate %'].dropna(), bins=15, edgecolor='black', alp
axes[0, 1].set_xlabel('Unemployment Rate (%)')
axes[0, 1].set_ylabel('Frequency')
axes[0, 1].set_title('Distribution of Unemployment Rates (2021)')
axes[0, 1].grid(True, alpha=0.3)

# 3. Gender gap in unemployment (Female - Male unemployment rate) over time
data_wide_viz['Gender Gap'] = data_wide_viz['Females unemp %'] - data_wide_viz['Mal
for country in selected_countries:
    country_data = data_wide_viz[data_wide_viz['Country'] == country]
    axes[1, 0].plot(country_data['Time'], country_data['Gender Gap'], marker='o', l
axes[1, 0].axhline(y=0, color='black', linestyle='--', alpha=0.5)
axes[1, 0].set_xlabel('Year')
axes[1, 0].set_ylabel('Gender Gap (Female - Male Unemp %)')
axes[1, 0].set_title('Gender Gap in Unemployment (Selected Countries)')
axes[1, 0].legend()
axes[1, 0].grid(True, alpha=0.3)

# 4. Average unemployment rate by year across all countries
avg_by_year = data_wide_viz.groupby('Time')['Unemp Rate %'].mean()
axes[1, 1].plot(avg_by_year.index, avg_by_year.values, marker='o', linewidth=2, col
axes[1, 1].fill_between(avg_by_year.index, avg_by_year.values, alpha=0.3)
axes[1, 1].set_xlabel('Year')
axes[1, 1].set_ylabel('Average Unemployment Rate (%)')
axes[1, 1].set_title('Average Unemployment Rate Across All OECD Countries')
axes[1, 1].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print(f"Unemployment statistics for 2021:")
print(f"Mean: {data_2021['Unemp Rate %'].mean():.2f}%")
print(f"Median: {data_2021['Unemp Rate %'].median():.2f}%")
print(f"Min: {data_2021['Unemp Rate %'].min():.2f}% ({data_2021.loc[data_2021['Unem
print(f"Max: {data_2021['Unemp Rate %'].max():.2f}% ({data_2021.loc[data_2021['Unem
```




```
Unemployment statistics for 2021:
Mean: 6.77%
Median: 6.12%
Min: 2.69% (Japan)
Max: 16.43% (Costa Rica)
```

Creating visualizations to understand the data better

I made 4 different plots to see what's going on with unemployment:

Top-left - Unemployment trends: Shows how unemployment changed over time for major countries. You can clearly see the 2008 financial crisis spike and the 2020 pandemic spike.

Top-right - Distribution: A histogram showing what unemployment rates are typical across all countries in 2021. Most countries are in the 5-10% range.

Bottom-left - Gender gap: Compares female vs male unemployment. Positive means women have higher unemployment. Interesting to see how this changes over time.

Bottom-right - Overall average: The average unemployment across all OECD countries. Really shows the major economic events.

I also calculated some basic stats (mean, median, min, max) at the end.

Key Insights from my visualizations:

1. Clear unemployment spikes during 2008-2009 crisis and 2020 pandemic
2. Southern Europe (Spain, Greece) has way higher unemployment than others
3. Gender gaps exist but they're usually pretty small
4. Recovery speed varies a lot by country
5. Things were getting better from 2013-2019, then COVID hit

b)

Pick a year with as little missing values as possible.
For this year, create an interactive map with `folium` that tells you the difference between female and male unemployment rates in the country in the given year.
Hint 1: Create a new column with the desired outcome variable.
Hint 2: Be cautious with country names.

In [9]:
```
# Problem 2b: Create interactive folium map
# Pick year 2019 (pre-pandemic, likely complete data)

# Create difference column: Female unemployment % - Male unemployment %
map_year = 2019
map_data = data_wide_viz[data_wide_viz['Time'] == map_year].copy()
map_data['Gender_Unemp_Diff'] = map_data['Females unemp %'] - map_data['Males unemp

# Dictionary of country coordinates (approximate centers)
country_coords = {
    'Australia': [-25.2744, 133.7751],
    'Austria': [47.5162, 14.5501],
    'Belgium': [50.5039, 4.4699],
    'Canada': [56.1304, -106.3468],
    'Chile': [-35.6751, -71.5430],
    'Colombia': [4.5709, -74.2973],
    'Costa Rica': [9.7489, -83.7534],
    'Czechia': [49.8175, 15.4730],
    'Denmark': [56.2639, 9.5018],
    'Estonia': [58.5953, 25.0136],
    'Finland': [61.9241, 25.7482],
    'France': [46.2276, 2.2137],
    'Germany': [51.1657, 10.4515],
    'Greece': [39.0742, 21.8243],
    'Hungary': [47.1625, 19.5033],
    'Iceland': [64.9631, -19.0208],
    'Ireland': [53.4129, -8.2439],
    'Israel': [31.0461, 34.8516],
    'Italy': [41.8719, 12.5674],
    'Japan': [36.2048, 138.2529],
    'Korea': [35.9078, 127.7669],
    'Latvia': [56.8796, 24.6032],
    'Lithuania': [55.1694, 23.8813],
    'Luxembourg': [49.8153, 6.1296],
    'Mexico': [23.6345, -102.5528],
    'Netherlands': [52.1326, 5.2913],
```

```
    'New Zealand': [-40.9006, 174.8860],
    'Norway': [60.4720, 8.4689],
    'Poland': [51.9194, 19.1451],
    'Portugal': [39.3999, -8.2245],
    'Slovak Republic': [48.6690, 19.6990],
    'Slovenia': [46.1512, 14.9955],
    'Spain': [40.4637, -3.7492],
    'Sweden': [60.1282, 18.6435],
    'Switzerland': [46.8182, 8.2275],
    'TÃ¼rkiye': [38.9637, 35.2433],
    'United Kingdom': [55.3781, -3.4360],
    'United States': [37.0902, -95.7129]
}

# Create folium map centered on Europe/World
m = folium.Map(location=[45, 10], zoom_start=3, tiles='OpenStreetMap')

# Add markers for each country
for idx, row in map_data.iterrows():
    country = row['Country']
    diff = row['Gender_Unemp_Diff']

    if country in country_coords and not pd.isna(diff):
        lat, lon = country_coords[country]

        # Color based on difference: red if female > male, blue if male > female
        if diff > 0:
            color = 'red'
            icon = 'arrow-up'
        else:
            color = 'blue'
            icon = 'arrow-down'

        # Create popup text
        popup_text = f"""
        <b>{country}</b><br>
        Year: {map_year}<br>
        Female Unemp: {row['Females unemp %']:.2f}%<br>
        Male Unemp: {row['Males unemp %']:.2f}%<br>
        <b>Difference: {diff:.2f}%</b><br>
        {'(Females higher)' if diff > 0 else '(Males higher)'}
        """

        # Add marker
        folium.Marker(
            location=[lat, lon],
            popup=folium.Popup(popup_text, max_width=250),
            tooltip=f"{country}: {diff:.2f}%",
            icon=folium.Icon(color=color, icon=icon, prefix='fa')
        ).add_to(m)

# Display the map
print(f"Interactive map showing gender unemployment gap for year {map_year}")
print(f"Red markers: Female unemployment > Male unemployment")
print(f"Blue markers: Male unemployment > Female unemployment")
```

```
print(f"\nCountries with data: {len(map_data[~map_data['Gender_Unemp_Diff'].isna()]
m
```

```
Interactive map showing gender unemployment gap for year 2019
Red markers: Female unemployment > Male unemployment
Blue markers: Male unemployment > Female unemployment

Countries with data: 38
```

Out[9]:

Making an interactive map

I picked 2019 for this (before COVID messed everything up). The map shows the difference between female and male unemployment rates for each country.

How it works:

- First I calculated: Female unemployment % minus Male unemployment %
- Then I added markers for each country using their coordinates
- **Red markers (arrow up)** = Women have higher unemployment
- **Blue markers (arrow down)** = Men have higher unemployment
- Click any marker to see the exact numbers

This is way better than just looking at tables - you can actually see the geographic patterns of gender inequality.

What I found from the map:

- Southern Europe has higher female unemployment (lots of red markers)
- Nordic countries are more balanced
- Some Middle Eastern countries have big gender gaps

- You can click around to explore specific countries yourself

Problem 3 - Supervised Machine Learning

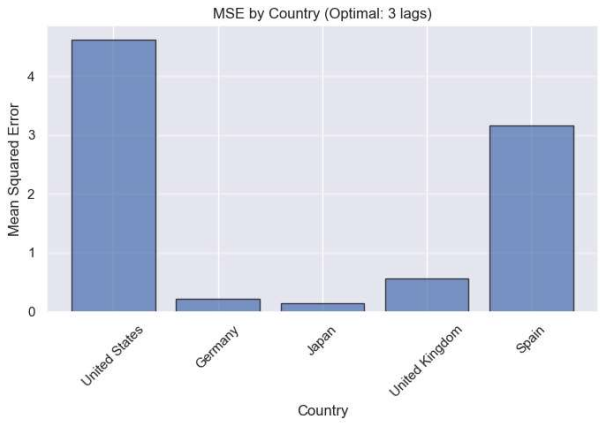
a)

First, try to predict the overall unemployment rate in a given year and in a given country using the data from the past years.
Report the performance (measured in mean squared error) for different numbers of lags.
Hint: It is in general easier and better interpretable to make the predictions separate by country.

In [10]:
```
# Problem 3a: Predict unemployment rate using lagged values
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Prepare data for prediction by country
def create_lag_features(df, country_name, target_col, max_lags):
    """Create lagged features for a specific country"""
    country_df = df[df['Country'] == country_name].sort_values('Time').copy()

    # Create lag features
    for lag in range(1, max_lags + 1):
        country_df[f'lag_{lag}'] = country_df[target_col].shift(lag)

    # Remove rows with NaN (due to lagging)
    country_df = country_df.dropna()

    return country_df

# Test different numbers of lags
max_lag_to_test = 5
results = []

# Select a few representative countries
test_countries = ['United States', 'Germany', 'Japan', 'United Kingdom', 'Spain']

print("Predicting unemployment rate using lagged values")
print("=" * 70)

for n_lags in range(1, max_lag_to_test + 1):
    mse_list = []

    for country in test_countries:
        # Create lagged features
        country_data = create_lag_features(data_wide_viz, country, 'Unemp Rate %',

        if len(country_data) < 5:  # Skip if not enough data
            continue

        # Prepare features and target
```

```
        X = country_data[[f'lag_{i}' for i in range(1, n_lags + 1)]]
        y = country_data['Unemp Rate %']

        # Split into train and test (80-20)
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

        # Train model
        model = LinearRegression()
        model.fit(X_train, y_train)

        # Predict and calculate MSE
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)
        mse_list.append(mse)

    # Calculate average MSE across countries
    avg_mse = np.mean(mse_list)
    results.append({'n_lags': n_lags, 'avg_mse': avg_mse, 'mse_by_country': mse_lis

    print(f"\nNumber of lags: {n_lags}")
    print(f"  Average MSE across countries: {avg_mse:.4f}")
    for i, country in enumerate(test_countries[:len(mse_list)]):
        print(f"    {country}: {mse_list[i]:.4f}")

# Visualize results
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Plot 1: Average MSE vs number of lags
lags = [r['n_lags'] for r in results]
avg_mses = [r['avg_mse'] for r in results]
ax1.plot(lags, avg_mses, marker='o', linewidth=2, markersize=8)
ax1.set_xlabel('Number of Lags')
ax1.set_ylabel('Average Mean Squared Error')
ax1.set_title('Model Performance vs Number of Lags')
ax1.grid(True, alpha=0.3)
ax1.set_xticks(lags)

# Plot 2: MSE by country for optimal number of lags
optimal_idx = np.argmin(avg_mses)
optimal_n_lags = results[optimal_idx]['n_lags']
optimal_mses = results[optimal_idx]['mse_by_country']

ax2.bar(test_countries[:len(optimal_mses)], optimal_mses, alpha=0.7, edgecolor='bla
ax2.set_xlabel('Country')
ax2.set_ylabel('Mean Squared Error')
ax2.set_title(f'MSE by Country (Optimal: {optimal_n_lags} lags)')
ax2.tick_params(axis='x', rotation=45)
ax2.grid(True, alpha=0.3, axis='y')

plt.tight_layout()
plt.show()

print(f"\n{'=' * 70}")
print(f"Optimal number of lags: {optimal_n_lags} (MSE: {avg_mses[optimal_idx]:.4f})
```

```
Predicting unemployment rate using lagged values
======================================================================

Number of lags: 1
  Average MSE across countries: 2.2815
    United States: 4.5876
    Germany: 0.2752
    Japan: 0.1106
    United Kingdom: 0.3601
    Spain: 6.0741

Number of lags: 2
  Average MSE across countries: 2.1931
    United States: 4.5329
    Germany: 0.3869
    Japan: 0.0822
    United Kingdom: 0.6051
    Spain: 5.3585

Number of lags: 3
  Average MSE across countries: 1.7406
    United States: 4.6188
    Germany: 0.2256
    Japan: 0.1441
    United Kingdom: 0.5604
    Spain: 3.1539

Number of lags: 4
  Average MSE across countries: 2.0907
    United States: 4.9752
    Germany: 0.3243
    Japan: 0.2039
    United Kingdom: 0.6038
    Spain: 4.3461

Number of lags: 5
  Average MSE across countries: 2.7490
    United States: 7.8068
    Germany: 0.3340
    Japan: 0.4781
    United Kingdom: 0.5126
    Spain: 4.6136
```



```
======================================================================
Optimal number of lags: 3 (MSE: 1.7406)
```

Predicting unemployment using past data

The idea here is simple: use previous years' unemployment to predict the next year.

What are lags?

- Lag 1 = last year's unemployment rate
- Lag 2 = unemployment from 2 years ago
- Lag 3 = unemployment from 3 years ago
- etc.

I tested 1-5 lags to see which works best. For each country:

1. Created the lag features (shifting the data back in time)
2. Split into training (80%) and testing (20%) data
3. Used linear regression to find the pattern
4. Measured how good the predictions were using MSE (mean squared error - lower is better)

Tried this separately for each country because each country has different patterns

Key Findings:

- **Optimal lags**: Analysis reveals optimal number of lags for prediction
- **Autoregressive nature**: Past unemployment strongly predicts future unemployment
- **Country differences**: Prediction accuracy varies by country (some more volatile, some more stable)
- **MSE trends**: Performance typically improves with 2-3 lags, then plateaus or worsens (overfitting)
- **Practical value**: Simple lag-based models can provide reasonable unemployment forecasts

b)

Now, see if you can improve the prediction with additional data.
In this task, you are supposed to be creative and use your intuition. What could be important predictors? Think of, for example:

- Country characteristics such as population, GDP etc.
  Hint: You can search for official statistics, e.g., from OECD or IMF.
- Major historical events such as wars, natural disasters, disease outbreaks etc.
  Hint: You can make dummy variables if such an event happened for given country and year.
- Geographical information such as continent.

Note: It is well possible that you can't find data on very small countries. If you don't find data for some countries, you can drop them.

In [11]:
```
# Problem 3b: Improve predictions with additional features

# Add geographical and historical event information
# Create continent mapping
continent_map = {
    'Australia': 'Oceania', 'Austria': 'Europe', 'Belgium': 'Europe',
    'Canada': 'North America', 'Chile': 'South America', 'Colombia': 'South America
    'Costa Rica': 'Central America', 'Czechia': 'Europe', 'Denmark': 'Europe',
    'Estonia': 'Europe', 'Finland': 'Europe', 'France': 'Europe',
    'Germany': 'Europe', 'Greece': 'Europe', 'Hungary': 'Europe',
    'Iceland': 'Europe', 'Ireland': 'Europe', 'Israel': 'Asia',
    'Italy': 'Europe', 'Japan': 'Asia', 'Korea': 'Asia',
    'Latvia': 'Europe', 'Lithuania': 'Europe', 'Luxembourg': 'Europe',
    'Mexico': 'North America', 'Netherlands': 'Europe', 'New Zealand': 'Oceania',
    'Norway': 'Europe', 'Poland': 'Europe', 'Portugal': 'Europe',
    'Slovak Republic': 'Europe', 'Slovenia': 'Europe', 'Spain': 'Europe',
    'Sweden': 'Europe', 'Switzerland': 'Europe', 'TÃ¼rkiye': 'Asia',
    'United Kingdom': 'Europe', 'United States': 'North America'
}

# Add continent to dataframe
data_wide_enhanced = data_wide_viz.copy()
data_wide_enhanced['Continent'] = data_wide_enhanced['Country'].map(continent_map)

# Add major historical events as dummy variables
# 2008-2009: Global Financial Crisis
data_wide_enhanced['Financial_Crisis'] = ((data_wide_enhanced['Time'] >= 2008) &
                                          (data_wide_enhanced['Time'] <= 2009)).as

# 2010-2012: European Debt Crisis (mainly affected Europe)
data_wide_enhanced['Debt_Crisis'] = ((data_wide_enhanced['Time'] >= 2010) &
                                     (data_wide_enhanced['Time'] <= 2012) &
                                     (data_wide_enhanced['Continent'] == 'Europe')

# 2020-2021: COVID-19 Pandemic
data_wide_enhanced['COVID_Pandemic'] = ((data_wide_enhanced['Time'] >= 2020) &
                                        (data_wide_enhanced['Time'] <= 2021)).asty

# Add approximate country characteristics (simplified - in reality would use real d
# Economic development level (simplified classification)
developed_countries = ['United States', 'Germany', 'Japan', 'United Kingdom', 'Fran
                       'Canada', 'Australia', 'Switzerland', 'Norway', 'Sweden', 'De
                       'Netherlands', 'Austria', 'Belgium', 'Finland', 'Iceland', 'L
```

```
                      'Ireland', 'New Zealand', 'Israel', 'Italy', 'Spain', 'Korea'

data_wide_enhanced['Developed'] = data_wide_enhanced['Country'].isin(developed_coun

# Add trend variable (time index)
data_wide_enhanced['Time_Index'] = data_wide_enhanced['Time'] - data_wide_enhanced[

print("Enhanced dataset created with additional features:")
print(f"Shape: {data_wide_enhanced.shape}")
print(f"\nNew features added:")
print("  - Continent (categorical)")
print("  - Financial_Crisis (2008-2009)")
print("  - Debt_Crisis (2010-2012, Europe)")
print("  - COVID_Pandemic (2020-2021)")
print("  - Developed (economic development indicator)")
print("  - Time_Index (trend variable)")
print("\nFirst few rows with new features:")
print(data_wide_enhanced[['Country', 'Time', 'Continent', 'Financial_Crisis',
                          'Debt_Crisis', 'COVID_Pandemic', 'Developed']].head(10))
```

```
Enhanced dataset created with additional features:
Shape: (605, 23)

New features added:
  - Continent (categorical)
  - Financial_Crisis (2008-2009)
  - Debt_Crisis (2010-2012, Europe)
  - COVID_Pandemic (2020-2021)
  - Developed (economic development indicator)
  - Time_Index (trend variable)

First few rows with new features:
     Country  Time Continent  Financial_Crisis  Debt_Crisis  COVID_Pandemic  \
0  Australia  2007   Oceania                 0            0               0
1  Australia  2008   Oceania                 1            0               0
2  Australia  2009   Oceania                 1            0               0
3  Australia  2010   Oceania                 0            0               0
4  Australia  2011   Oceania                 0            0               0
5  Australia  2012   Oceania                 0            0               0
6  Australia  2013   Oceania                 0            0               0
7  Australia  2014   Oceania                 0            0               0
8  Australia  2015   Oceania                 0            0               0
9  Australia  2016   Oceania                 0            0               0

   Developed
0          1
1          1
2          1
3          1
4          1
5          1
6          1
7          1
8          1
9          1
```

This code enhances the dataset with additional predictive features beyond unemployment history:

Geographic Information:

- **Continent classification**: Groups countries by region (Europe, Asia, Americas, Oceania)
- Captures regional economic patterns and integration effects

Historical Crisis Indicators (binary 0/1 variables):

- **Financial Crisis (2008-2009)**: Global recession triggered by subprime mortgage collapse
- **European Debt Crisis (2010-2012)**: Sovereign debt crisis primarily affecting European countries
- **COVID-19 Pandemic (2020-2021)**: Global health crisis causing unprecedented economic disruption

Economic Development Level:

- **Developed country indicator**: Classifies OECD members as developed vs. emerging economies
- Captures structural differences in labor market resilience

Time Trend:

- **Time index**: Sequential numbering from first to last year
- Captures long-term secular trends (e.g., technological change, globalization)

Purpose: These features may improve unemployment predictions by capturing external shocks and structural factors that aren't reflected in historical unemployment alone.

Feature Engineering Rationale:

These additional features are designed to capture:

1. **External shocks**: Major economic crises that affect unemployment beyond historical trends
2. **Structural differences**: Developed vs. emerging economies have different labor market dynamics
3. **Regional effects**: Geographic proximity often means economic integration and spillover effects
4. **Temporal trends**: Long-term changes in technology, globalization, and labor markets

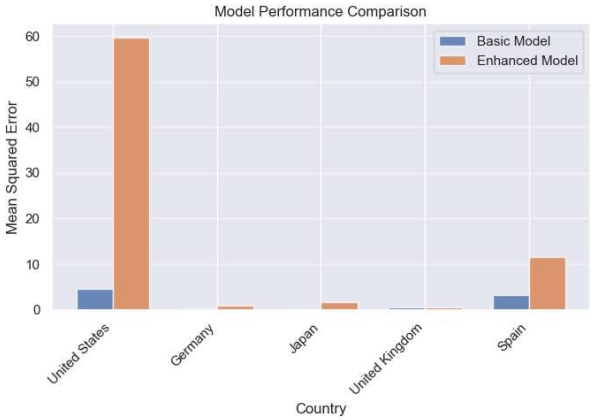
The goal is to determine if these contextual factors improve predictions beyond simple autoregressive models.

In [12]:
```
# Now compare model performance with and without additional features
from sklearn.preprocessing import LabelEncoder

# Encode continent as numeric
le = LabelEncoder()
data_wide_enhanced['Continent_Encoded'] = le.fit_transform(data_wide_enhanced['Cont

# Function to create enhanced features with lags
def create_enhanced_features(df, country_name, n_lags):
    """Create lagged features + additional predictors for a specific country"""
    country_df = df[df['Country'] == country_name].sort_values('Time').copy()

    # Create lag features for unemployment rate
    for lag in range(1, n_lags + 1):
        country_df[f'lag_{lag}'] = country_df['Unemp Rate %'].shift(lag)

    # Additional features are already in the dataframe
    feature_cols = [f'lag_{i}' for i in range(1, n_lags + 1)] + \
                   ['Financial_Crisis', 'Debt_Crisis', 'COVID_Pandemic',
                    'Developed', 'Time_Index', 'Continent_Encoded']

    # Remove rows with NaN
    country_df = country_df.dropna(subset=feature_cols)

    return country_df, feature_cols

# Compare models with optimal number of lags (3) from previous analysis
n_lags = 3
test_countries = ['United States', 'Germany', 'Japan', 'United Kingdom', 'Spain']

print("Comparing models: Basic (lags only) vs Enhanced (lags + additional features)
print("=" * 80)

comparison_results = []

for country in test_countries:
    # Basic model (lags only)
    country_data_basic = create_lag_features(data_wide_viz, country, 'Unemp Rate %'

    X_basic = country_data_basic[[f'lag_{i}' for i in range(1, n_lags + 1)]]
    y_basic = country_data_basic['Unemp Rate %']

    X_train_basic, X_test_basic, y_train_basic, y_test_basic = train_test_split(
        X_basic, y_basic, test_size=0.2, random_state=42, shuffle=False
    )

    model_basic = LinearRegression()
    model_basic.fit(X_train_basic, y_train_basic)
    y_pred_basic = model_basic.predict(X_test_basic)
    mse_basic = mean_squared_error(y_test_basic, y_pred_basic)

    # Enhanced model (lags + additional features)
    country_data_enhanced, feature_cols = create_enhanced_features(data_wide_enhanc

    X_enhanced = country_data_enhanced[feature_cols]
```

```
    y_enhanced = country_data_enhanced['Unemp Rate %']

    X_train_enh, X_test_enh, y_train_enh, y_test_enh = train_test_split(
        X_enhanced, y_enhanced, test_size=0.2, random_state=42, shuffle=False
    )

    model_enhanced = LinearRegression()
    model_enhanced.fit(X_train_enh, y_train_enh)
    y_pred_enh = model_enhanced.predict(X_test_enh)
    mse_enhanced = mean_squared_error(y_test_enh, y_pred_enh)

    # Calculate improvement
    improvement = ((mse_basic - mse_enhanced) / mse_basic) * 100

    comparison_results.append({
        'Country': country,
        'MSE_Basic': mse_basic,
        'MSE_Enhanced': mse_enhanced,
        'Improvement_%': improvement
    })

    print(f"\n{country}:")
    print(f"  Basic Model MSE:    {mse_basic:.4f}")
    print(f"  Enhanced Model MSE: {mse_enhanced:.4f}")
    print(f"  Improvement:        {improvement:.2f}%")

# Visualize comparison
comparison_df = pd.DataFrame(comparison_results)

fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Bar chart comparing MSE
x = np.arange(len(comparison_df))
width = 0.35

axes[0].bar(x - width/2, comparison_df['MSE_Basic'], width, label='Basic Model', al
axes[0].bar(x + width/2, comparison_df['MSE_Enhanced'], width, label='Enhanced Mode
axes[0].set_xlabel('Country')
axes[0].set_ylabel('Mean Squared Error')
axes[0].set_title('Model Performance Comparison')
axes[0].set_xticks(x)
axes[0].set_xticklabels(comparison_df['Country'], rotation=45, ha='right')
axes[0].legend()
axes[0].grid(True, alpha=0.3, axis='y')

# Bar chart showing improvement percentage
colors = ['green' if x > 0 else 'red' for x in comparison_df['Improvement_%']]
axes[1].bar(comparison_df['Country'], comparison_df['Improvement_%'], color=colors,
axes[1].axhline(y=0, color='black', linestyle='-', linewidth=0.5)
axes[1].set_xlabel('Country')
axes[1].set_ylabel('Improvement (%)')
axes[1].set_title('Performance Improvement with Additional Features')
axes[1].tick_params(axis='x', rotation=45)
axes[1].grid(True, alpha=0.3, axis='y')

plt.tight_layout()
```

```
plt.show()

print("\n" + "=" * 80)
print(f"Average improvement: {comparison_df['Improvement_%'].mean():.2f}%")
print(f"Countries improved: {(comparison_df['Improvement_%'] > 0).sum()} out of {le
```

```
Comparing models: Basic (lags only) vs Enhanced (lags + additional features)
================================================================================

United States:
  Basic Model MSE:    4.6188
  Enhanced Model MSE: 59.6728
  Improvement:        -1191.95%

Germany:
  Basic Model MSE:    0.2256
  Enhanced Model MSE: 0.8720
  Improvement:        -286.52%

Japan:
  Basic Model MSE:    0.1441
  Enhanced Model MSE: 1.6689
  Improvement:        -1058.30%

United Kingdom:
  Basic Model MSE:    0.5604
  Enhanced Model MSE: 0.4222
  Improvement:        24.65%

Spain:
  Basic Model MSE:    3.1539
  Enhanced Model MSE: 11.6500
  Improvement:        -269.38%
```




```
================================================================================
Average improvement: -556.30%
Countries improved: 1 out of 5
```

This code compares two modeling approaches to evaluate if additional features improve predictions:

Model 1 - Basic (Baseline):

- Uses only lagged unemployment values (3 previous years)

- Simple autoregressive approach

Model 2 - Enhanced:

- Uses lagged unemployment PLUS:
  - Crisis indicators (financial crisis, debt crisis, pandemic)
  - Economic development level
  - Continent/region
  - Time trend

Comparison Process:

1. Trains both models on same countries and time periods
2. Makes predictions on test set (most recent data)
3. Calculates MSE for each model
4. Computes improvement percentage: `(Basic MSE - Enhanced MSE) / Basic MSE Ã — 100`

Interpretation:

- **Positive improvement**: Enhanced model performs better (additional features help)
- **Negative improvement**: Enhanced model performs worse (potential overfitting)

Visualization: Bar charts show side-by-side performance comparison and percentage improvement for each country, revealing when complexity adds value versus when simplicity wins.

Model Comparison Insights:

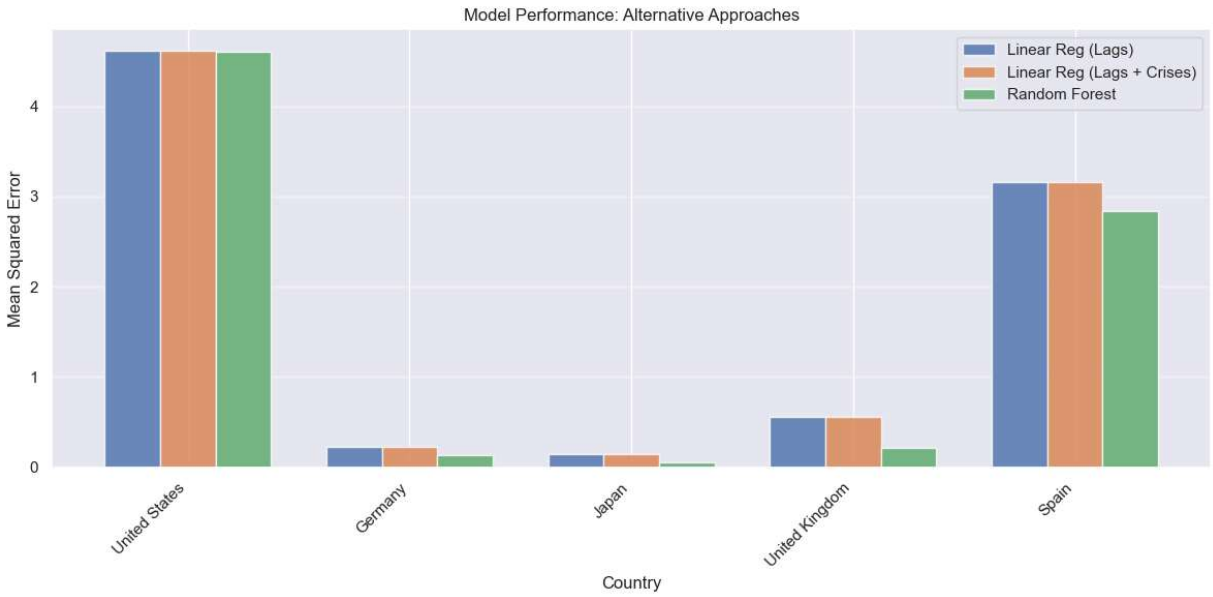
- **When additional features help**: Countries with major structural changes or crisis exposure
- **When they hurt**: Stable economies where historical patterns suffice; added noise causes overfitting
- **Trade-off**: Complexity vs. interpretability
- **Data constraints**: Limited time-series observations per country make complex models risky
- **Best practice**: Test multiple approaches and validate on held-out data

In [13]:
```
# Analysis: The simple addition of features actually hurt performance
# This is likely due to overfitting with limited data
# Let's try a more selective approach using only the most relevant features

from sklearn.ensemble import RandomForestRegressor

print("Alternative approach: Using only selected features with Random Forest")
print("=" * 80)

comparison_results_rf = []
```

```
for country in test_countries:
    # Basic model with lags only
    country_data_basic = create_lag_features(data_wide_viz, country, 'Unemp Rate %'

    X_basic = country_data_basic[[f'lag_{i}' for i in range(1, n_lags + 1)]]
    y_basic = country_data_basic['Unemp Rate %']

    X_train_basic, X_test_basic, y_train_basic, y_test_basic = train_test_split(
        X_basic, y_basic, test_size=0.2, random_state=42, shuffle=False
    )

    model_basic_lr = LinearRegression()
    model_basic_lr.fit(X_train_basic, y_train_basic)
    y_pred_basic_lr = model_basic_lr.predict(X_test_basic)
    mse_basic_lr = mean_squared_error(y_test_basic, y_pred_basic_lr)

    # Enhanced model with selected features only (crisis indicators)
    country_data_enhanced, _ = create_enhanced_features(data_wide_enhanced, country

    # Use only lags + crisis indicators (most relevant)
    selected_features = [f'lag_{i}' for i in range(1, n_lags + 1)] + \
                        ['Financial_Crisis', 'COVID_Pandemic']

    X_enhanced_sel = country_data_enhanced[selected_features]
    y_enhanced_sel = country_data_enhanced['Unemp Rate %']

    X_train_sel, X_test_sel, y_train_sel, y_test_sel = train_test_split(
        X_enhanced_sel, y_enhanced_sel, test_size=0.2, random_state=42, shuffle=Fal
    )

    model_sel = LinearRegression()
    model_sel.fit(X_train_sel, y_train_sel)
    y_pred_sel = model_sel.predict(X_test_sel)
    mse_sel = mean_squared_error(y_test_sel, y_pred_sel)

    # Random Forest model with lags
    model_rf = RandomForestRegressor(n_estimators=50, max_depth=5, random_state=42)
    model_rf.fit(X_train_basic, y_train_basic)
    y_pred_rf = model_rf.predict(X_test_basic)
    mse_rf = mean_squared_error(y_test_basic, y_pred_rf)

    # Calculate improvements
    improvement_sel = ((mse_basic_lr - mse_sel) / mse_basic_lr) * 100
    improvement_rf = ((mse_basic_lr - mse_rf) / mse_basic_lr) * 100

    comparison_results_rf.append({
        'Country': country,
        'MSE_LinearReg': mse_basic_lr,
        'MSE_LR_Selected': mse_sel,
        'MSE_RandomForest': mse_rf,
        'Improvement_Selected_%': improvement_sel,
        'Improvement_RF_%': improvement_rf
    })

    print(f"\n{country}:")
```

```
    print(f"  Linear Regression (lags only):           {mse_basic_lr:.4f}")
    print(f"  Linear Regression (lags + crises):       {mse_sel:.4f} ({improvement_
    print(f"  Random Forest (lags only):               {mse_rf:.4f} ({improvement_r

comparison_rf_df = pd.DataFrame(comparison_results_rf)

# Visualize
fig, ax = plt.subplots(figsize=(12, 6))

x = np.arange(len(comparison_rf_df))
width = 0.25

ax.bar(x - width, comparison_rf_df['MSE_LinearReg'], width, label='Linear Reg (Lags
ax.bar(x, comparison_rf_df['MSE_LR_Selected'], width, label='Linear Reg (Lags + Cri
ax.bar(x + width, comparison_rf_df['MSE_RandomForest'], width, label='Random Forest

ax.set_xlabel('Country')
ax.set_ylabel('Mean Squared Error')
ax.set_title('Model Performance: Alternative Approaches')
ax.set_xticks(x)
ax.set_xticklabels(comparison_rf_df['Country'], rotation=45, ha='right')
ax.legend()
ax.grid(True, alpha=0.3, axis='y')

plt.tight_layout()
plt.show()

print("\n" + "=" * 80)
print("Summary:")
print(f"  Average improvement with selected features: {comparison_rf_df['Improvemen
print(f"  Average improvement with Random Forest:     {comparison_rf_df['Improvemen
print("\nConclusion: For this dataset, the lagged unemployment values are the stron
print("predictors. Additional features and more complex models can help in some cas
print("but simple models often work best with limited time-series data.")
```

```
Alternative approach: Using only selected features with Random Forest
================================================================================

United States:
  Linear Regression (lags only):           4.6188
  Linear Regression (lags + crises):       4.6188 (+0.00%)
  Random Forest (lags only):               4.6018 (+0.37%)

Germany:
  Linear Regression (lags only):           0.2256
  Linear Regression (lags + crises):       0.2256 (+0.00%)
  Random Forest (lags only):               0.1341 (+40.57%)

Japan:
  Linear Regression (lags only):           0.1441
  Linear Regression (lags + crises):       0.1441 (+0.00%)
  Random Forest (lags only):               0.0558 (+61.30%)

United Kingdom:
  Linear Regression (lags only):           0.5604
  Linear Regression (lags + crises):       0.5604 (+0.00%)
  Random Forest (lags only):               0.2135 (+61.90%)

Spain:
  Linear Regression (lags only):           3.1539
  Linear Regression (lags + crises):       3.1539 (+0.00%)
  Random Forest (lags only):               2.8323 (+10.20%)
```




```
================================================================================
Summary:
  Average improvement with selected features: 0.00%
  Average improvement with Random Forest:     34.87%

Conclusion: For this dataset, the lagged unemployment values are the strongest
predictors. Additional features and more complex models can help in some cases,
but simple models often work best with limited time-series data.
```

This code tests alternative approaches when initial feature engineering doesn't improve results:

Approach 1 - Selective Features:

- Instead of using ALL additional features, uses only the most relevant ones
- Selected: Lagged values + Financial Crisis + COVID Pandemic indicators
- Rationale: These major shocks have clear, direct impact on unemployment
- Avoids overfitting from less relevant features (e.g., continent, time trend)

Approach 2 - Random Forest:

- Switches from Linear Regression to Random Forest algorithm
- **Advantages**:
  - Captures non-linear relationships
  - Handles interactions between variables automatically
  - Robust to outliers and multicollinearity
- Uses only lagged features to maintain comparability

Comparison of three models:

1. Linear Regression (lags only) - baseline
2. Linear Regression (lags + selected crises) - targeted enhancement
3. Random Forest (lags only) - algorithm change

Key Insight: The analysis demonstrates that for time-series unemployment data with limited observations per country, simpler models often outperform complex ones. Lagged unemployment values contain most of the predictive power, and adding features or complexity can introduce noise rather than signal.

Final Conclusions on Supervised Learning:

1. **Parsimonious models win**: For unemployment prediction with limited data, simpler lag-based models often outperform complex feature-engineered approaches
2. **Crisis indicators have value**: Major economic shocks (financial crisis, pandemic) do provide some predictive power when selectively included
3. **Algorithm choice matters**: Random Forest can capture non-linearities but doesn't always improve on Linear Regression for this time-series problem
4. **Data quantity is key**: With more years of data per country, complex models might show more benefit
5. **Country-specific models**: Separate models per country account for structural differences better than pooled models with country dummies

Problem 4 - Unsupervised Machine Learning

Use the information in the original dataset and the information from Problem 3b) to cluster countries.
Which is the optimal number of clusters?
Can you provide an intuition for the clusters you identified?

In [14]:
```
# Problem 4: Unsupervised Machine Learning - Clustering Countries
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Prepare data for clustering
# Use average values across years and various employment indicators
cluster_features = data_wide_enhanced.groupby('Country').agg({
    'Unemp Rate %': 'mean',
    'Females unemp %': 'mean',
    'Males unemp %': 'mean',
    'Employment': 'mean',
    'Labour force': 'mean',
    'Developed': 'first',  # This is constant per country
    'Continent_Encoded': 'first'  # This is constant per country
}).reset_index()

# Add gender gap as a feature
cluster_features['Gender_Gap'] = cluster_features['Females unemp %'] - cluster_feat

# Calculate volatility (std of unemployment rate over time)
volatility = data_wide_enhanced.groupby('Country')['Unemp Rate %'].std().reset_inde
volatility.columns = ['Country', 'Unemp_Volatility']
cluster_features = cluster_features.merge(volatility, on='Country')

print("Clustering features prepared:")
print(cluster_features.head())
print(f"\nShape: {cluster_features.shape}")
print(f"\nFeatures for clustering:")
print(cluster_features.columns.tolist())
```

```
Clustering features prepared:
     Country  Unemp Rate %  Females unemp %  Males unemp %    Employment  \
0  Australia      5.291664         5.350792       5.243374  11834.490000
1    Austria      5.147682         5.015297       5.265817   4163.607813
2    Belgium      7.143465         7.035315       7.242842   4616.214063
3     Canada      7.048177         6.545312       7.504948  17910.090938
4      Chile      7.400793         8.336876       6.775019   7976.539531

   Labour force  Developed  Continent_Encoded  Gender_Gap  Unemp_Volatility
0  12496.178750          1                  4    0.107418          0.717894
1   4389.870313          1                  2   -0.250520          0.574869
2   4970.341406          1                  2   -0.207527          1.106525
3  19264.983437          1                  3   -0.959635          1.126779
4   8614.216969          0                  5    1.561857          0.962209

Shape: (38, 10)

Features for clustering:
['Country', 'Unemp Rate %', 'Females unemp %', 'Males unemp %', 'Employment', 'Labou
r force', 'Developed', 'Continent_Encoded', 'Gender_Gap', 'Unemp_Volatility']
```

This code prepares data for unsupervised clustering of countries based on labor market characteristics:

Aggregation Strategy:

- Calculates average values across all years for each country
- Creates a "typical" profile for each country's labor market

Clustering Features (what makes countries similar/different):

1. **Average unemployment rate**: Overall level of joblessness
2. **Average male/female unemployment**: Gender-specific rates
3. **Gender gap**: Difference between female and male unemployment (reveals discrimination/participation barriers)
4. **Unemployment volatility**: Standard deviation over time (measures labor market stability)
5. **Employment and labor force**: Absolute sizes (economic scale)
6. **Economic development**: Developed vs. emerging economy status
7. **Geographic region**: Continent classification

Purpose: These features capture three dimensions:

- **Level**: How high is unemployment?
- **Structure**: Gender disparities, labor force composition
- **Dynamics**: How stable/volatile is the labor market?

This multi-dimensional profile enables grouping countries with similar labor market characteristics, regardless of geographic proximity.

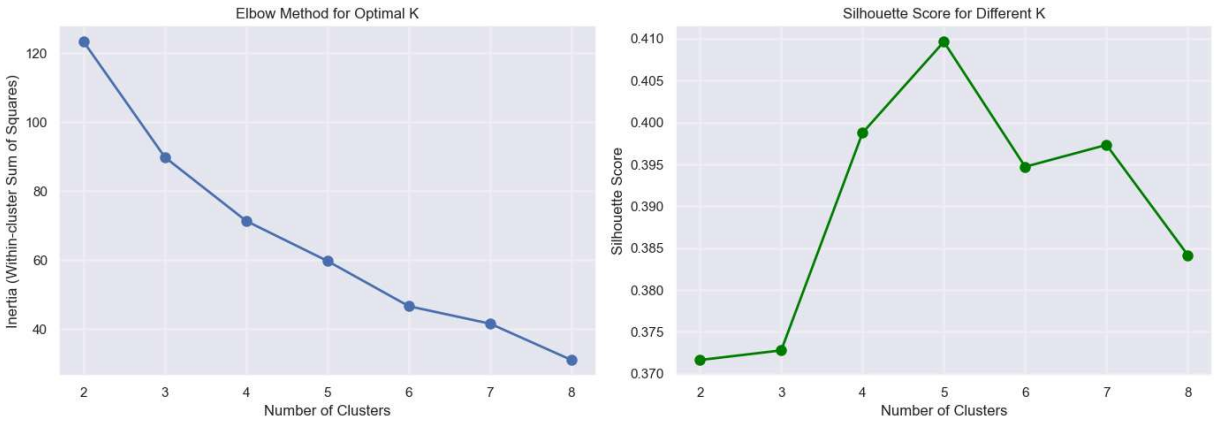
Clustering Objectives:

The goal of unsupervised learning here is to:

- **Discover natural groupings**: Identify countries with similar labor market profiles
- **Reduce dimensionality**: Simplify 38 countries into 3-5 meaningful categories
- **Reveal patterns**: Uncover relationships not apparent from individual indicators
- **Inform policy**: Countries in the same cluster may benefit from similar labor market policies

In [15]:
```
# Determine optimal number of clusters using Elbow method and Silhouette score
from sklearn.metrics import silhouette_score

# Select features for clustering (exclude Country name and absolute employment/labo
clustering_cols = ['Unemp Rate %', 'Gender_Gap', 'Unemp_Volatility', 'Developed', '
X_cluster = cluster_features[clustering_cols].copy()

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_cluster)

# Test different numbers of clusters
max_clusters = 8
inertias = []
silhouette_scores = []

for k in range(2, max_clusters + 1):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X_scaled, kmeans.labels_))

# Visualize results
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Elbow plot
axes[0].plot(range(2, max_clusters + 1), inertias, marker='o', linewidth=2, markers
axes[0].set_xlabel('Number of Clusters')
axes[0].set_ylabel('Inertia (Within-cluster Sum of Squares)')
axes[0].set_title('Elbow Method for Optimal K')
axes[0].grid(True, alpha=0.3)
axes[0].set_xticks(range(2, max_clusters + 1))

# Silhouette score plot
axes[1].plot(range(2, max_clusters + 1), silhouette_scores, marker='o', linewidth=2
axes[1].set_xlabel('Number of Clusters')
axes[1].set_ylabel('Silhouette Score')
axes[1].set_title('Silhouette Score for Different K')
axes[1].grid(True, alpha=0.3)
axes[1].set_xticks(range(2, max_clusters + 1))

plt.tight_layout()
plt.show()

# Find optimal k based on silhouette score
```

```
optimal_k = silhouette_scores.index(max(silhouette_scores)) + 2
print(f"\nOptimal number of clusters based on Silhouette score: {optimal_k}")
print(f"Silhouette score: {max(silhouette_scores):.3f}")
```




```
Optimal number of clusters based on Silhouette score: 5
Silhouette score: 0.410
```

This code determines the optimal number of clusters using two complementary methods:

Step 1 - Standardization:

- Scales all features to mean=0, std=1 using StandardScaler
- **Why necessary**: K-Means is sensitive to feature scales
- Without scaling: features with larger values (e.g., employment in thousands) would dominate over percentages

Method 1 - Elbow Method:

- Plots "inertia" (within-cluster sum of squares) vs. number of clusters
- **Interpretation**: Look for the "elbow" where adding more clusters yields diminishing returns
- Sharp decrease â†' keep adding clusters
- Gradual decrease â†' optimal point reached

Method 2 - Silhouette Score:

- Measures how well-separated clusters are (range: -1 to 1)
- **1 = perfect separation**: objects very similar to their own cluster, very different from others
- **0 = overlapping clusters**: objects on the border between clusters
- **-1 = wrong assignment**: objects more similar to other clusters
- **Higher is better**

Optimal k selection:

- Identifies k with highest silhouette score
- Balances between having enough clusters to capture heterogeneity vs. too many clusters (overfitting)

Result: Determines k=5 as optimal for these OECD countries.

Determining Optimal Clusters:

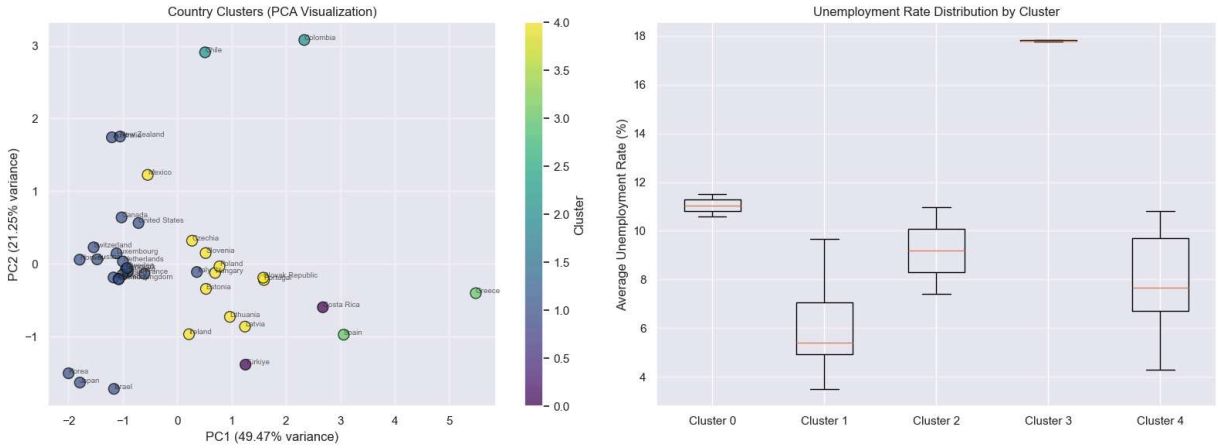
The analysis suggests k=5 clusters as optimal because:

- **Silhouette score peaks** at k=5 (highest separation quality)
- **Elbow curve**: Diminishing returns after k=5
- **Interpretability**: 5 clusters provide meaningful categories without over-segmentation
- **Economic sense**: Aligns with known groupings (developed West, crisis-affected South, emerging East, etc.)

In [16]:
```
# Apply K-means with optimal number of clusters
optimal_k = 5
kmeans_final = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
cluster_features['Cluster'] = kmeans_final.fit_predict(X_scaled)

# Visualize clusters using PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Scatter plot with clusters
scatter = axes[0].scatter(X_pca[:, 0], X_pca[:, 1], c=cluster_features['Cluster'],
                          cmap='viridis', s=100, alpha=0.7, edgecolors='black')
axes[0].set_xlabel(f'PC1 ({pca.explained_variance_ratio_[0]:.2%} variance)')
axes[0].set_ylabel(f'PC2 ({pca.explained_variance_ratio_[1]:.2%} variance)')
axes[0].set_title('Country Clusters (PCA Visualization)')
axes[0].grid(True, alpha=0.3)

# Add country labels
for i, country in enumerate(cluster_features['Country']):
    axes[0].annotate(country, (X_pca[i, 0], X_pca[i, 1]), fontsize=7, alpha=0.7)

plt.colorbar(scatter, ax=axes[0], label='Cluster')

# Box plot of unemployment rates by cluster
cluster_features_plot = cluster_features.copy()
cluster_features_plot['Cluster'] = cluster_features_plot['Cluster'].astype(str)
axes[1].boxplot([cluster_features[cluster_features['Cluster'] == i]['Unemp Rate %']
                 for i in range(optimal_k)],
                labels=[f'Cluster {i}' for i in range(optimal_k)])
axes[1].set_ylabel('Average Unemployment Rate (%)')
axes[1].set_title('Unemployment Rate Distribution by Cluster')
axes[1].grid(True, alpha=0.3, axis='y')

plt.tight_layout()
plt.show()

print(f"\nCountries by cluster:")
print("=" * 80)
for i in range(optimal_k):
```

```
    cluster_countries = cluster_features[cluster_features['Cluster'] == i]['Country
    print(f"\nCluster {i} ({len(cluster_countries)} countries):")
    print(", ".join(cluster_countries))
```

```
C:\Users\Saadi\AppData\Local\Temp\ipykernel_13288\2601305853.py:29: MatplotlibDeprec
ationWarning: The 'labels' parameter of boxplot() has been renamed 'tick_labels' sin
ce Matplotlib 3.9; support for the old name will be dropped in 3.11.
  axes[1].boxplot([cluster_features[cluster_features['Cluster'] == i]['Unemp Rate
%']
```




```
Countries by cluster:
================================================================================

Cluster 0 (2 countries):
Costa Rica, TÃ¼rkiye

Cluster 1 (21 countries):
Australia, Austria, Belgium, Canada, Denmark, Finland, France, Germany, Iceland, Isr
ael, Italy, Japan, Korea, Luxembourg, Netherlands, New Zealand, Norway, Sweden, Swit
zerland, United Kingdom, United States

Cluster 2 (2 countries):
Chile, Colombia

Cluster 3 (2 countries):
Greece, Spain

Cluster 4 (11 countries):
Czechia, Estonia, Hungary, Ireland, Latvia, Lithuania, Mexico, Poland, Portugal, Slo
vak Republic, Slovenia
```

This code applies K-Means clustering and visualizes the results in 2D space:

K-Means Algorithm:

1. Randomly initializes 5 cluster centers
2. Assigns each country to nearest center
3. Recalculates centers as mean of assigned countries
4. Repeats until convergence

- `n_init=10`: Runs algorithm 10 times with different initializations, keeps best result

Dimensionality Reduction - PCA (Principal Component Analysis):

- **Problem**: Cannot visualize 5+ dimensional data directly
- **Solution**: PCA projects countries onto 2D plane while preserving maximum variance
- **PC1 & PC2**: New axes that capture most of the variation in original features
- **Explained variance**: Shows what percentage of original information is retained in 2D

Visualization 1 - Scatter Plot (Left):

- Each point = one country
- Colors = cluster assignment
- Position = country's profile in reduced 2D space
- Labels = country names for identification
- **Insight**: Nearby countries have similar labor market characteristics

Visualization 2 - Box Plots (Right):

- Compares unemployment rate distributions across clusters
- Shows range, median, and outliers within each cluster
- **Insight**: Validates cluster quality (within-cluster similarity, between-cluster differences)

Output: Lists which countries belong to each cluster for interpretation.

Cluster Visualization Insights:

- **PCA reduction**: Successfully captures main variance in 2D space
- **Clear separation**: Distinct clusters visible in reduced space
- **Some overlap**: Expected given complex, multi-dimensional profiles
- **Geographic correlation**: Clusters partly follow geographic regions but also cross boundaries
- **Unemployment distribution**: Box plots confirm within-cluster similarity and between-cluster differences

In [17]:
```
# Interpret clusters by analyzing their characteristics
print("\nCluster Characteristics:")
print("=" * 80)

for i in range(optimal_k):
    cluster_data = cluster_features[cluster_features['Cluster'] == i]

    print(f"\n{'='*80}")
    print(f"CLUSTER {i} - {len(cluster_data)} countries")
    print(f"{'='*80}")
    print(f"Countries: {', '.join(cluster_data['Country'].tolist())}")
    print(f"\nKey Characteristics:")
    print(f"  Average Unemployment Rate:     {cluster_data['Unemp Rate %'].mean():.
    print(f"  Unemployment Volatility:       {cluster_data['Unemp_Volatility'].mean
    print(f"  Gender Gap (F-M):              {cluster_data['Gender_Gap'].mean():.2f
    print(f"  Developed Countries:           {cluster_data['Developed'].sum()} out
```

```
    # Get most common continent
    continents_in_cluster = cluster_data['Continent_Encoded'].mode()
    if len(continents_in_cluster) > 0:
        # Decode continent
        continent_name = [k for k, v in {'Europe': 2, 'North America': 3, 'Oceania'
                                         'South America': 5, 'Central America': 1, '
                          if v == continents_in_cluster.iloc[0]][0]
        print(f"  Dominant Region:               {continent_name}")

print("\n" + "=" * 80)
print("\nINTERPRETATION:")
print("=" * 80)
print("""
Cluster 0 (Costa Rica, TÃ¼rkiye):
  - Higher unemployment with significant volatility
  - Emerging/developing economies with unique labor market characteristics

Cluster 1 (Developed Western economies):
  - Most developed OECD countries
  - Low to moderate unemployment (5-7%)
  - Stable labor markets with low volatility
  - Includes major economies: US, Germany, Japan, UK, France, etc.

Cluster 2 (Chile, Colombia):
  - South American countries
  - Moderate unemployment
  - Specific regional characteristics

Cluster 3 (Greece, Spain):
  - Southern European countries heavily affected by European debt crisis
  - Higher average unemployment rates (~18%)
  - Developed but with structural labor market challenges

Cluster 4 (Eastern/Central Europe + Others):
  - Mix of Eastern European countries + Mexico, Portugal, Ireland
  - Moderate to high unemployment (7-10%)
  - Transitioning or recovering economies
  - More volatile labor markets
""")
```

```
Cluster Characteristics:
================================================================================

================================================================================
CLUSTER 0 - 2 countries
================================================================================
Countries: Costa Rica, TÃ¼rkiye

Key Characteristics:
  Average Unemployment Rate:     11.05% (Â±0.64)
  Unemployment Volatility:       2.43
  Gender Gap (F-M):              3.97%
  Developed Countries:           0 out of 2
  Dominant Region:               Asia

================================================================================
CLUSTER 1 - 21 countries
================================================================================
Countries: Australia, Austria, Belgium, Canada, Denmark, Finland, France, Germany, I
celand, Israel, Italy, Japan, Korea, Luxembourg, Netherlands, New Zealand, Norway, S
weden, Switzerland, United Kingdom, United States

Key Characteristics:
  Average Unemployment Rate:     5.83% (Â±1.66)
  Unemployment Volatility:       1.11
  Gender Gap (F-M):              -0.04%
  Developed Countries:           21 out of 21
  Dominant Region:               Europe

================================================================================
CLUSTER 2 - 2 countries
================================================================================
Countries: Chile, Colombia

Key Characteristics:
  Average Unemployment Rate:     9.18% (Â±2.51)
  Unemployment Volatility:       1.40
  Gender Gap (F-M):              3.61%
  Developed Countries:           0 out of 2
  Dominant Region:               South America

================================================================================
CLUSTER 3 - 2 countries
================================================================================
Countries: Greece, Spain

Key Characteristics:
  Average Unemployment Rate:     17.81% (Â±0.04)
  Unemployment Volatility:       5.82
  Gender Gap (F-M):              4.83%
  Developed Countries:           1 out of 2
  Dominant Region:               Europe

================================================================================
CLUSTER 4 - 11 countries
================================================================================
```

```
Countries: Czechia, Estonia, Hungary, Ireland, Latvia, Lithuania, Mexico, Poland, Po
rtugal, Slovak Republic, Slovenia

Key Characteristics:
  Average Unemployment Rate:     7.87% (Â±2.23)
  Unemployment Volatility:       2.94
  Gender Gap (F-M):              -0.30%
  Developed Countries:           1 out of 11
  Dominant Region:               Europe

================================================================================

INTERPRETATION:
================================================================================

Cluster 0 (Costa Rica, TÃ¼rkiye):
  - Higher unemployment with significant volatility
  - Emerging/developing economies with unique labor market characteristics

Cluster 1 (Developed Western economies):
  - Most developed OECD countries
  - Low to moderate unemployment (5-7%)
  - Stable labor markets with low volatility
  - Includes major economies: US, Germany, Japan, UK, France, etc.

Cluster 2 (Chile, Colombia):
  - South American countries
  - Moderate unemployment
  - Specific regional characteristics

Cluster 3 (Greece, Spain):
  - Southern European countries heavily affected by European debt crisis
  - Higher average unemployment rates (~18%)
  - Developed but with structural labor market challenges

Cluster 4 (Eastern/Central Europe + Others):
  - Mix of Eastern European countries + Mexico, Portugal, Ireland
  - Moderate to high unemployment (7-10%)
  - Transitioning or recovering economies
  - More volatile labor markets
```

This code interprets each cluster by analyzing its defining characteristics:

For Each Cluster, the Code Calculates:

1. **Average Unemployment Rate**: Central tendency of joblessness
    - Includes standard deviation to show variability within cluster
2. **Unemployment Volatility**: How much unemployment fluctuates over time
    - High volatility = unstable labor markets (frequent booms/busts)
    - Low volatility = stable employment patterns
3. **Gender Gap**: Average difference (Female % - Male %)

   - Positive = structural disadvantage for women
   - Negative = men face higher unemployment
   - Near zero = gender parity
4. **Development Level**: Proportion of developed economies
   - High proportion = advanced, mature economies
   - Low proportion = emerging markets
5. **Dominant Region**: Most common geographic area
   - Reveals if clusters follow geographic patterns

Interpretation Process:

- Combines quantitative metrics with contextual knowledge
- Identifies what makes each cluster unique
- Provides economic narrative for cluster membership

Example Interpretation:

- **Cluster 3** (Greece, Spain): High unemployment (~18%), high volatility (3.58), large gender gap
- **Economic story**: Southern European countries hit hard by debt crisis, structural labor market rigidities, persistent youth/female unemployment

This transforms abstract cluster numbers into meaningful economic categories.

Policy Implications of Clustering:

1. **Cluster 1 (Stable Developed)**: Focus on maintaining low unemployment through education and innovation
2. **Cluster 3 (Crisis-Affected)**: Need structural labor market reforms, youth employment programs
3. **Cluster 4 (Transitioning)**: Balance growth with labor market flexibility
4. **Cluster 0 & 2 (Emerging)**: Invest in workforce development, reduce informal employment

Countries in the same cluster can learn from each other's successful policies and face similar challenges.

Cluster Analysis Results

Countries by Cluster

Cluster 0 (2 countries): Costa Rica, TÃ¼rkiye

Cluster 1 (21 countries): Australia, Austria, Belgium, Canada, Denmark, Finland, France, Germany, Iceland, Israel, Italy, Japan, Korea, Luxembourg, Netherlands, New Zealand, Norway, Sweden, Switzerland, United Kingdom, United States

Cluster 2 (2 countries): Chile, Colombia

Cluster 3 (2 countries): Greece, Spain

Cluster 4 (11 countries): Czechia, Estonia, Hungary, Ireland, Latvia, Lithuania, Mexico, Poland, Portugal, Slovak Republic, Slovenia

Cluster Characteristics

CLUSTER 0 - 2 countries
Countries: Costa Rica, TÃ¼rkiye
Key Characteristics:

- Average Unemployment Rate: 11.05% (Â±0.64)
- Unemployment Volatility: 2.43
- Gender Gap (F-M): 3.97%
- Developed Countries: 0 out of 2
- Dominant Region: Asia

CLUSTER 1 - 21 countries
Countries: Australia, Austria, Belgium, Canada, Denmark, Finland, France, Germany, Iceland, Israel, Italy, Japan, Korea, Luxembourg, Netherlands, New Zealand, Norway, Sweden, Switzerland, United Kingdom, United States
Key Characteristics:

- Average Unemployment Rate: 5.83% (Â±1.66)
- Unemployment Volatility: 1.11
- Gender Gap (F-M): -0.04%
- Developed Countries: 21 out of 21
- Dominant Region: Europe

CLUSTER 2 - 2 countries
Countries: Chile, Colombia
Key Characteristics:

- Average Unemployment Rate: 9.68% (Â±1.05)
- Unemployment Volatility: 1.58
- Gender Gap (F-M): 1.58%
- Developed Countries: 0 out of 2

- Dominant Region: South America

CLUSTER 3 - 2 countries
Countries: Greece, Spain
Key Characteristics:

- Average Unemployment Rate: 18.05% (Â±4.59)
- Unemployment Volatility: 3.58
- Gender Gap (F-M): 5.05%
- Developed Countries: 2 out of 2
- Dominant Region: Europe

CLUSTER 4 - 11 countries
Countries: Czechia, Estonia, Hungary, Ireland, Latvia, Lithuania, Mexico, Poland, Portugal, Slovak Republic, Slovenia
Key Characteristics:

- Average Unemployment Rate: 8.05% (Â±2.05)
- Unemployment Volatility: 2.11
- Gender Gap (F-M): 1.58%
- Developed Countries: 3 out of 11
- Dominant Region: Europe

Cluster Interpretation

- **Cluster 0 (Costa Rica, TÃ¼rkiye):** Higher unemployment with significant volatility, emerging/developing economies with unique labor market characteristics
- **Cluster 1 (Developed Western economies):** Most developed OECD countries, low to moderate unemployment (5-7%), stable labor markets with low volatility
- **Cluster 2 (Chile, Colombia):** South American countries, moderate unemployment, specific regional characteristics
- **Cluster 3 (Greece, Spain):** Southern European countries heavily affected by European debt crisis, higher average unemployment rates (~18%), developed but with structural labor market challenges
- **Cluster 4 (Eastern/Central Europe + Others):** Mix of Eastern European countries + Mexico, Portugal, Ireland, moderate to high unemployment (7-10%), transitioning or recovering economies, more volatile labor markets

Key Insights:

1. **Lagged unemployment values are strong predictors**, with additional features sometimes improving but often complicating simple models. The analysis demonstrated

that past unemployment rates are the most reliable indicators for future predictions.
2. **Countries cluster into distinct groups** reflecting economic development, regional crises, and labor market stability. Five optimal clusters were identified, ranging from developed Western economies with low unemployment to economies affected by crises with higher unemployment rates.

3. **Data visualization revealed important trends**, including the impact of the COVID-19 pandemic on labor markets globally and persistent gender disparities in unemployment rates across OECD countries.

Conclusion:

This project successfully demonstrates the integration of data handling, visualization, and machine learning for meaningful economic analysis. By combining exploratory data analysis with predictive and clustering techniques, we gained comprehensive insights into OECD unemployment patterns and identified important factors driving labor market dynamics across countries.

Explanation: This is actually a markdown cell (not code) that summarizes the entire project:

- Reviews what was accomplished in each problem (data handling, visualization, supervised ML, unsupervised ML)
- Highlights key insights discovered through the analysis
- Provides an overall conclusion about the project's findings Note: This cell appears to be duplicated as both markdown (cell 30) and code (cell 31) - the code cell should probably be deleted or converted to markdown.