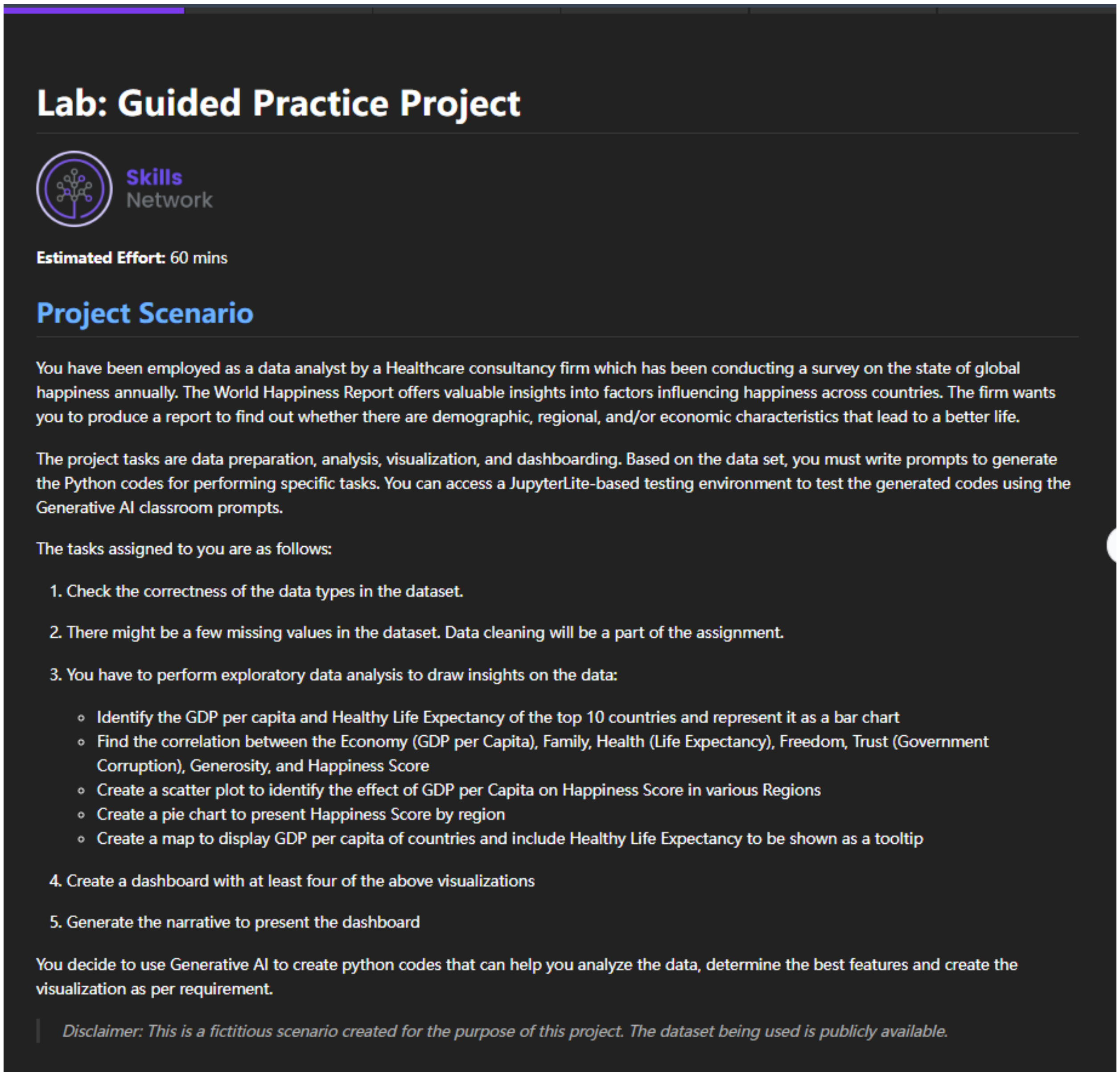


Lab: Guided Practice Project

Skills Network

Estimated Effort: 60 mins

Project Scenario

You have been employed as a data analyst by a Healthcare consultancy firm which has been conducting a survey on the state of global happiness annually. The World Happiness Report offers valuable insights into factors influencing happiness across countries. The firm wants you to produce a report to find out whether there are demographic, regional, and/or economic characteristics that lead to a better life.

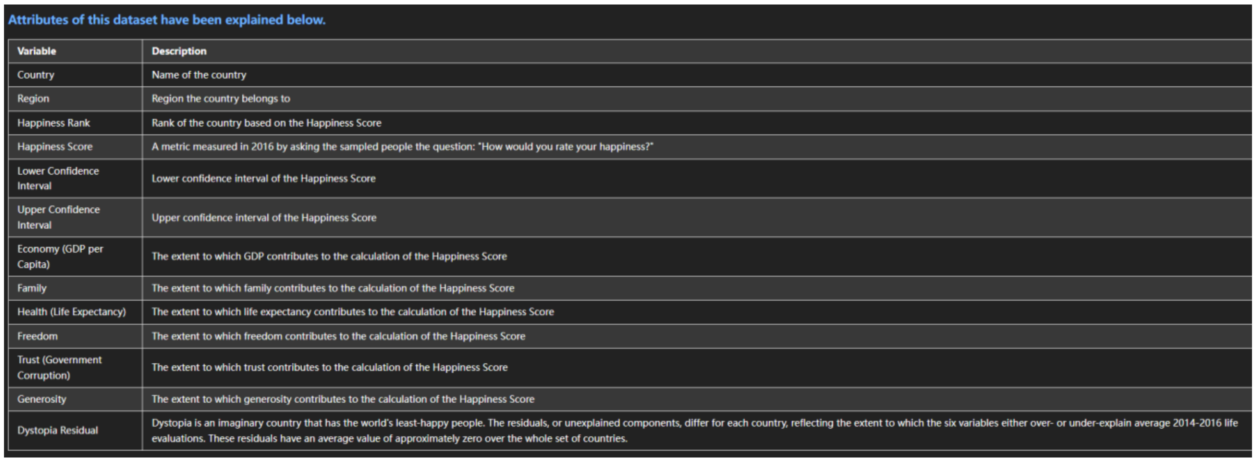
The project tasks are data preparation, analysis, visualization, and dashboarding. Based on the data set, you must write prompts to generate the Python codes for performing specific tasks. You can access a JupyterLite-based testing environment to test the generated codes using the Generative AI classroom prompts.

The tasks assigned to you are as follows:

1. Check the correctness of the data types in the dataset.
2. There might be a few missing values in the dataset. Data cleaning will be a part of the assignment.
3. You have to perform exploratory data analysis to draw insights on the data:
   - Identify the GDP per capita and Healthy Life Expectancy of the top 10 countries and represent it as a bar chart
   - Find the correlation between the Economy (GDP per Capita), Family, Health (Life Expectancy), Freedom, Trust (Government Corruption), Generosity, and Happiness Score
   - Create a scatter plot to identify the effect of GDP per Capita on Happiness Score in various Regions
   - Create a pie chart to present Happiness Score by region
   - Create a map to display GDP per capita of countries and include Healthy Life Expectancy to be shown as a tooltip
4. Create a dashboard with at least four of the above visualizations
5. Generate the narrative to present the dashboard

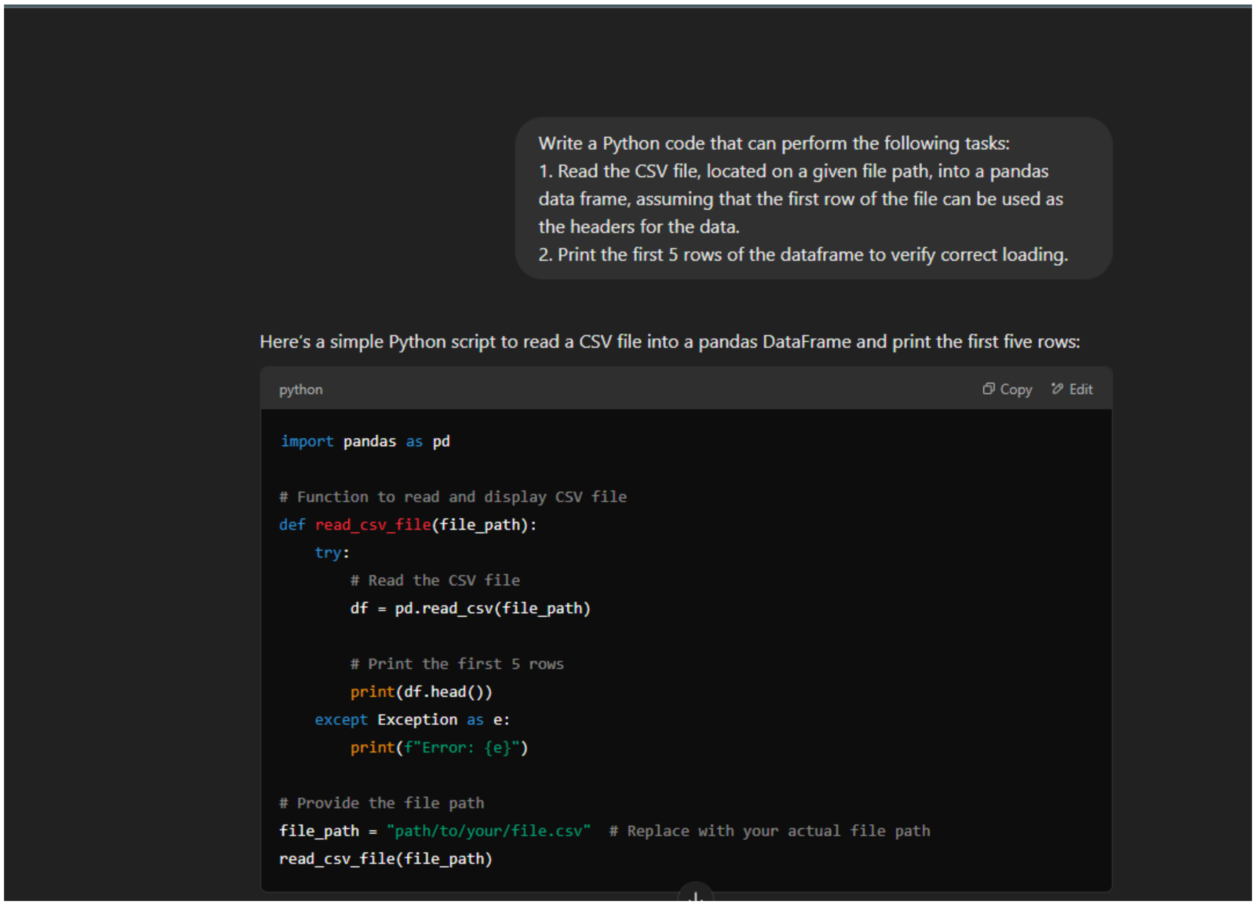
You decide to use Generative AI to create python codes that can help you analyze the data, determine the best features and create the visualization as per requirement.

> *Disclaimer: This is a fictitious scenario created for the purpose of this project. The dataset being used is publicly available.*



Attributes of this dataset have been explained below.

Variable	Description
Country	Name of the country
Region	Region the country belongs to
Happiness Rank	Rank of the country based on the Happiness Score
Happiness Score	A metric measured in 2016 by asking the sampled people the question: "How would you rate your happiness?"
Lower Confidence Interval	Lower confidence interval of the Happiness Score
Upper Confidence Interval	Upper confidence interval of the Happiness Score
Economy (GDP per Capita)	The extent to which GDP contributes to the calculation of the Happiness Score
Family	The extent to which family contributes to the calculation of the Happiness Score
Health (Life Expectancy)	The extent to which life expectancy contributes to the calculation of the Happiness Score
Freedom	The extent to which freedom contributes to the calculation of the Happiness Score
Trust (Government Corruption)	The extent to which trust contributes to the calculation of the Happiness Score
Generosity	The extent to which generosity contributes to the calculation of the Happiness Score
Dystopia Residual	Dystopia is an imaginary country that has the world's least-happy people. The residuals, or unexplained components, differ for each country, reflecting the extent to which the six variables either over- or under-explain average 2014-2016 life evaluations. These residuals have an average value of approximately zero over the whole set of countries.



Write a Python code that can perform the following tasks:
1. Read the CSV file, located on a given file path, into a pandas data frame, assuming that the first row of the file can be used as the headers for the data.
2. Print the first 5 rows of the dataframe to verify correct loading.

Here's a simple Python script to read a CSV file into a pandas DataFrame and print the first five rows:

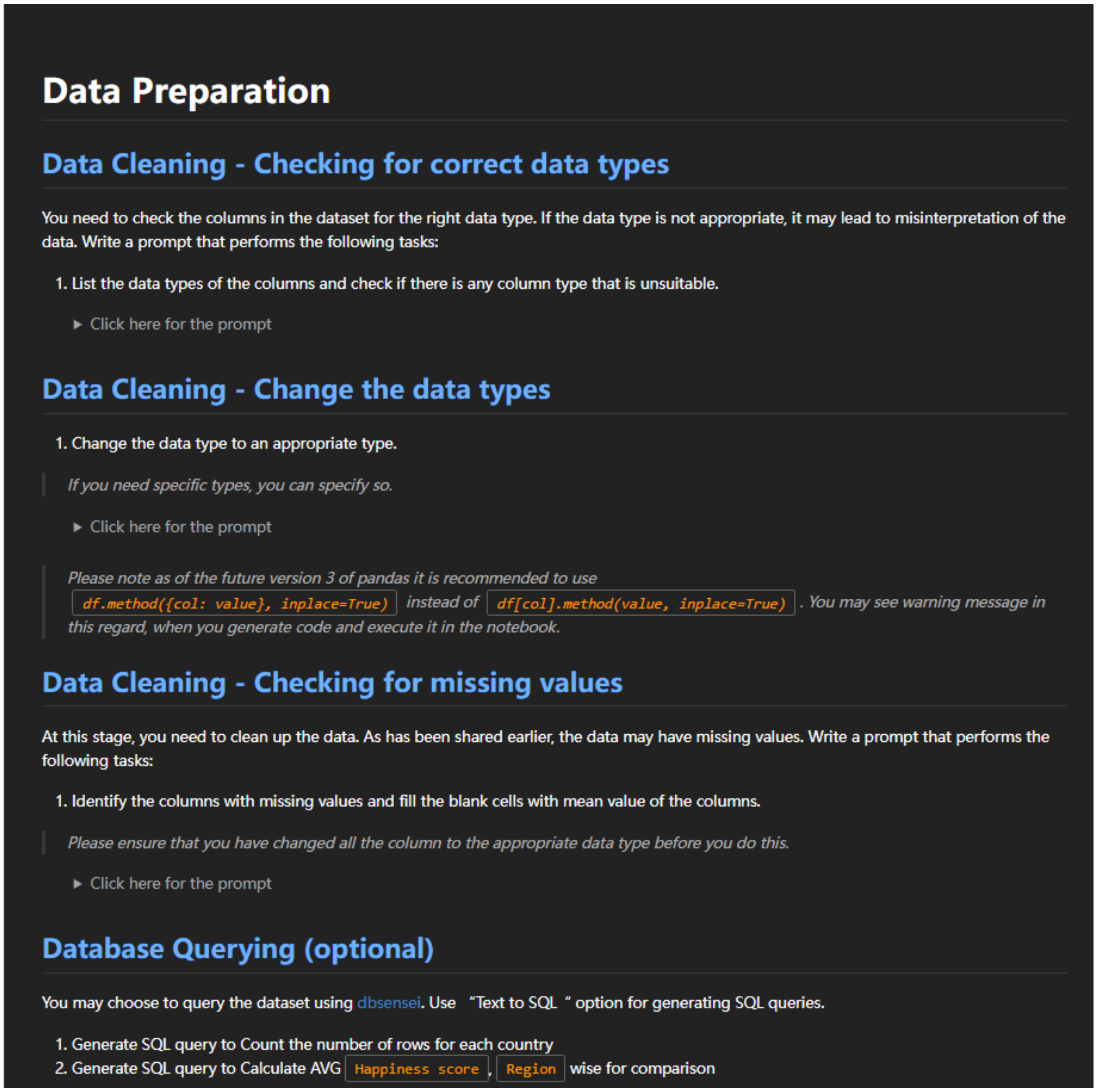
```python
import pandas as pd

# Function to read and display CSV file
def read_csv_file(file_path):
    try:
        # Read the CSV file
        df = pd.read_csv(file_path)

        # Print the first 5 rows
        print(df.head())
    except Exception as e:
        print(f"Error: {e}")

# Provide the file path
file_path = "path/to/your/file.csv"  # Replace with your actual file path
read_csv_file(file_path)
```

Test Environment

[4]:

```
# Keep appending the code generated to this cell, or add more cells below this to execute in parts
import pandas as pd

# Function to read and display CSV file
def read_csv_file(file_path):
    try:
        # Read the CSV file
        df = pd.read_csv(file_path)

        # Print the first 5 rows
        print(df.head())
    except Exception as e:
        print(f"Error: {e}")

# Provide the file path
file_path = URL
read_csv_file(file_path)
```

```
       Country          Region  Happiness Rank  Happiness Score  \
0      Denmark  Western Europe               1            7.526
1  Switzerland  Western Europe               2            7.509
2      Iceland  Western Europe               3            7.501
3       Norway  Western Europe               4            7.498
4      Finland  Western Europe               5            7.413

   Lower Confidence Interval Upper Confidence Interval  \
0                      7.460                     7.592
1                      7.428                      7.59
2                      7.333                     7.669
3                      7.421                     7.575
4                      7.351                     7.475

  Economy (GDP per Capita)   Family Health (Life Expectancy)  Freedom  \
0                 1.44178  1.16374                  0.79504  0.57941
1                 1.52733  1.14524                  0.86303  0.58557
2                 1.42666  1.18326                  0.86733  0.56624
3                 1.57744  1.12690                  0.79579  0.59609
4                 1.40598  1.13464                  0.81091  0.57104

   Trust (Government Corruption)  Generosity  Dystopia Residual
0                        0.44453     0.36171            2.73939
1                        0.41203     0.28083            2.69463
2                        0.14975     0.47678            2.83137
3                        0.35776     0.37895            2.66465
4                        0.41004     0.25492            2.82596
```



Data Preparation

Data Cleaning - Checking for correct data types

You need to check the columns in the dataset for the right data type. If the data type is not appropriate, it may lead to misinterpretation of the data. Write a prompt that performs the following tasks:

1. List the data types of the columns and check if there is any column type that is unsuitable.

▸ Click here for the prompt

Data Cleaning - Change the data types

1. Change the data type to an appropriate type.

> *If you need specific types, you can specify so.*

▸ Click here for the prompt

> *Please note as of the future version 3 of pandas it is recommended to use `df.method({col: value}, inplace=True)` instead of `df[col].method(value, inplace=True)`. You may see warning message in this regard, when you generate code and execute it in the notebook.*

Data Cleaning - Checking for missing values

At this stage, you need to clean up the data. As has been shared earlier, the data may have missing values. Write a prompt that performs the following tasks:

1. Identify the columns with missing values and fill the blank cells with mean value of the columns.

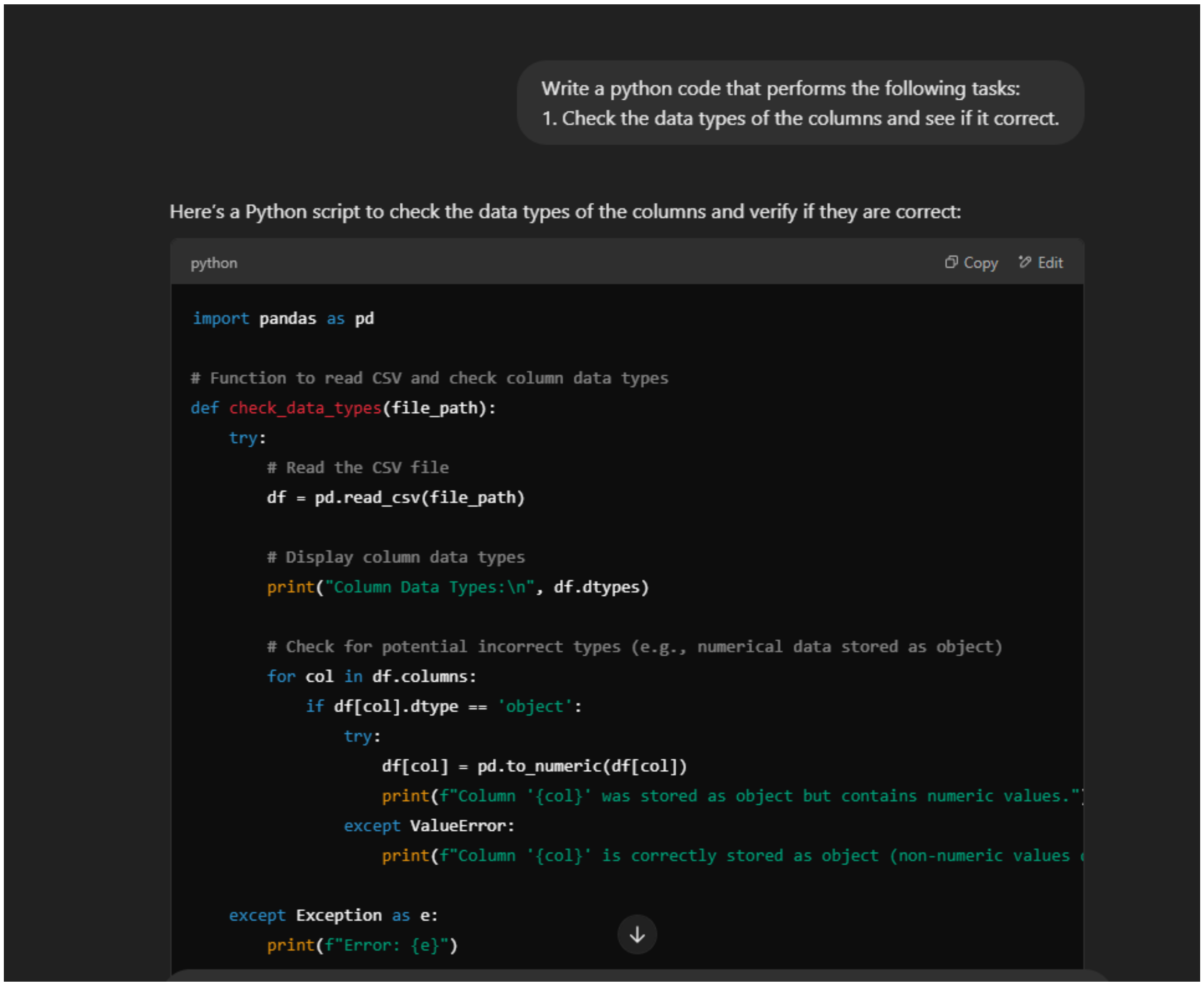
> *Please ensure that you have changed all the column to the appropriate data type before you do this.*

▸ Click here for the prompt

Database Querying (optional)

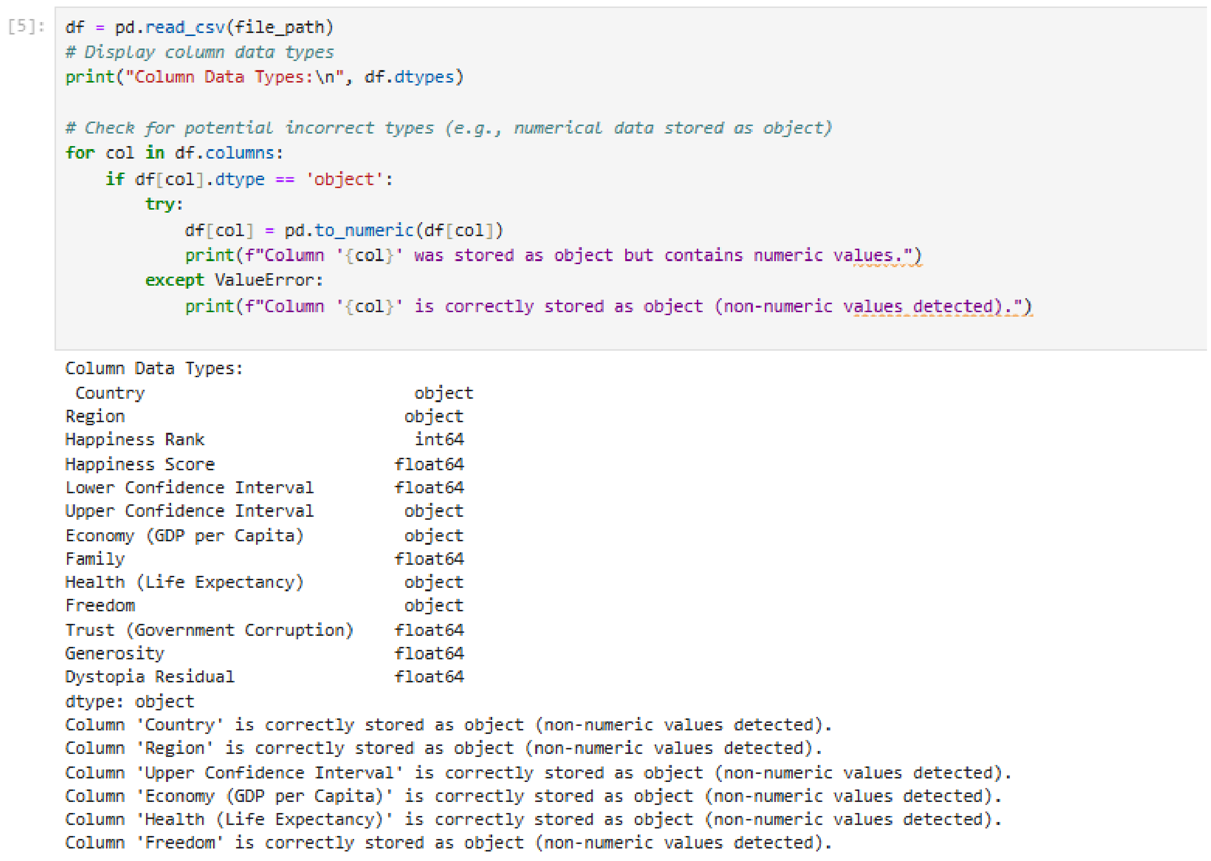
You may choose to query the dataset using dbsensei. Use "Text to SQL " option for generating SQL queries.

1. Generate SQL query to Count the number of rows for each country
2. Generate SQL query to Calculate AVG `Happiness score`, `Region` wise for comparison


Write a python code that performs the following tasks:
1. Check the data types of the columns and see if it correct.

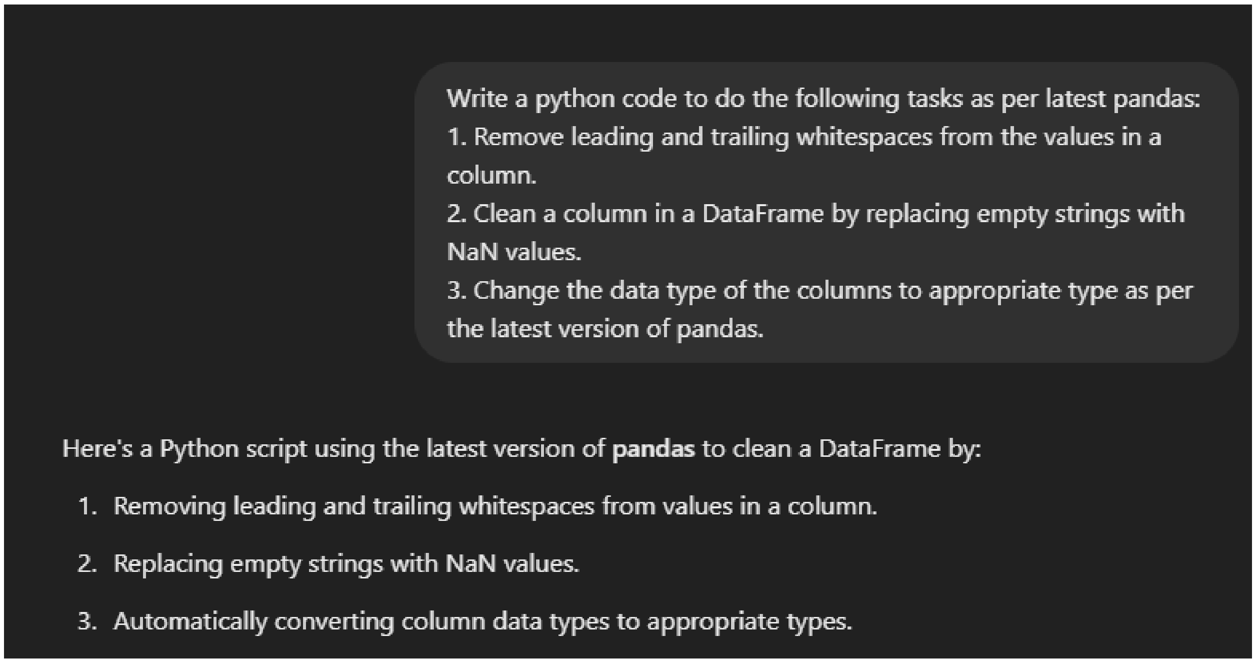
Here's a Python script to check the data types of the columns and verify if they are correct:

```python
import pandas as pd

# Function to read CSV and check column data types
def check_data_types(file_path):
    try:
        # Read the CSV file
        df = pd.read_csv(file_path)

        # Display column data types
        print("Column Data Types:\n", df.dtypes)

        # Check for potential incorrect types (e.g., numerical data stored as object)
        for col in df.columns:
            if df[col].dtype == 'object':
                try:
                    df[col] = pd.to_numeric(df[col])
                    print(f"Column '{col}' was stored as object but contains numeric values.")
                except ValueError:
                    print(f"Column '{col}' is correctly stored as object (non-numeric values 

    except Exception as e:
        print(f"Error: {e}")
```



```
[5]: df = pd.read_csv(file_path)
     # Display column data types
     print("Column Data Types:\n", df.dtypes)

     # Check for potential incorrect types (e.g., numerical data stored as object)
     for col in df.columns:
         if df[col].dtype == 'object':
             try:
                 df[col] = pd.to_numeric(df[col])
                 print(f"Column '{col}' was stored as object but contains numeric values.")
             except ValueError:
                 print(f"Column '{col}' is correctly stored as object (non-numeric values detected).")
```

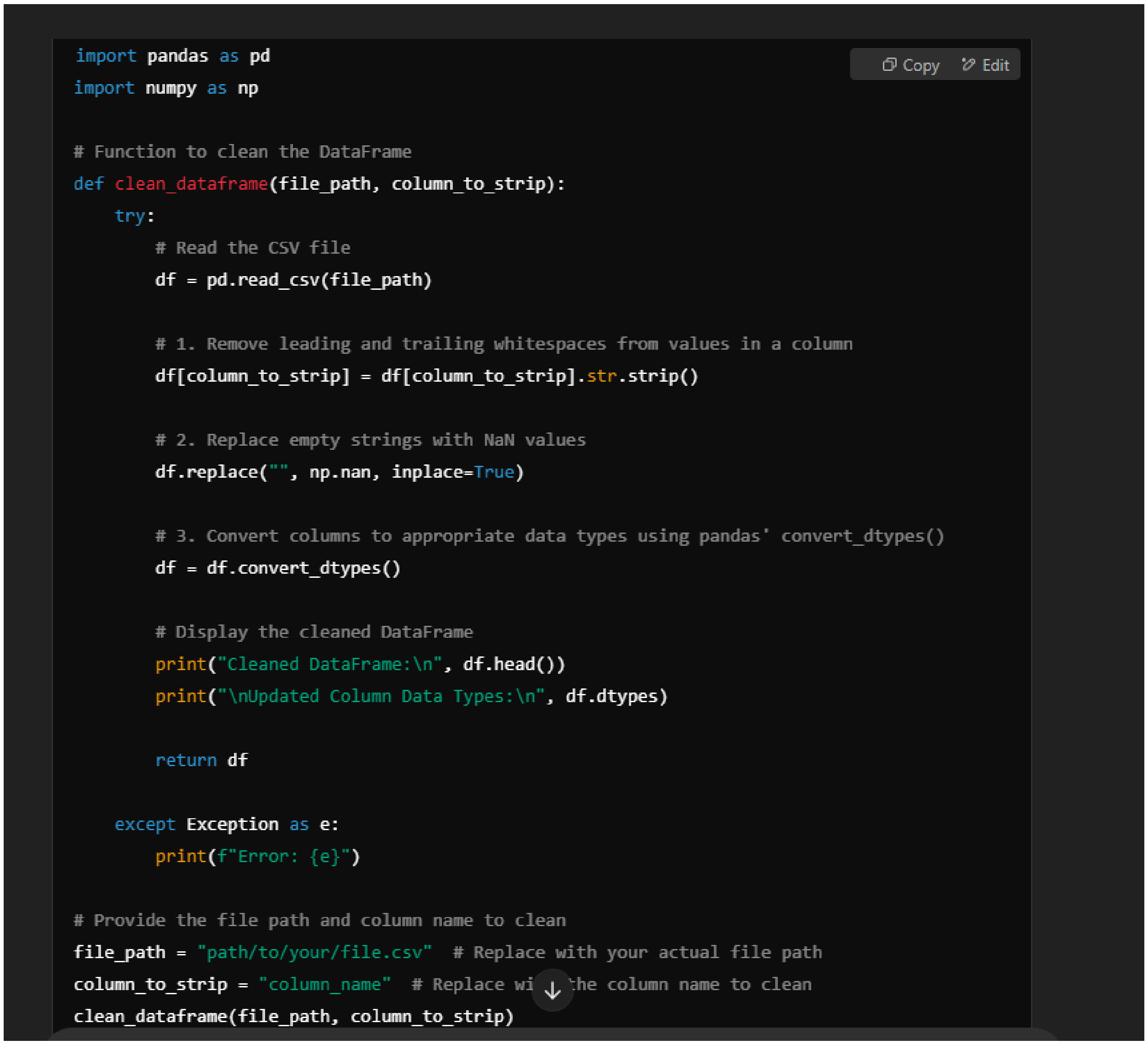
```
Column Data Types:
 Country                          object
Region                           object
Happiness Rank                    int64
Happiness Score                 float64
Lower Confidence Interval       float64
Upper Confidence Interval        object
Economy (GDP per Capita)         object
Family                          float64
Health (Life Expectancy)         object
Freedom                          object
Trust (Government Corruption)   float64
Generosity                      float64
Dystopia Residual               float64
dtype: object
Column 'Country' is correctly stored as object (non-numeric values detected).
Column 'Region' is correctly stored as object (non-numeric values detected).
Column 'Upper Confidence Interval' is correctly stored as object (non-numeric values detected).
Column 'Economy (GDP per Capita)' is correctly stored as object (non-numeric values detected).
Column 'Health (Life Expectancy)' is correctly stored as object (non-numeric values detected).
Column 'Freedom' is correctly stored as object (non-numeric values detected).
```



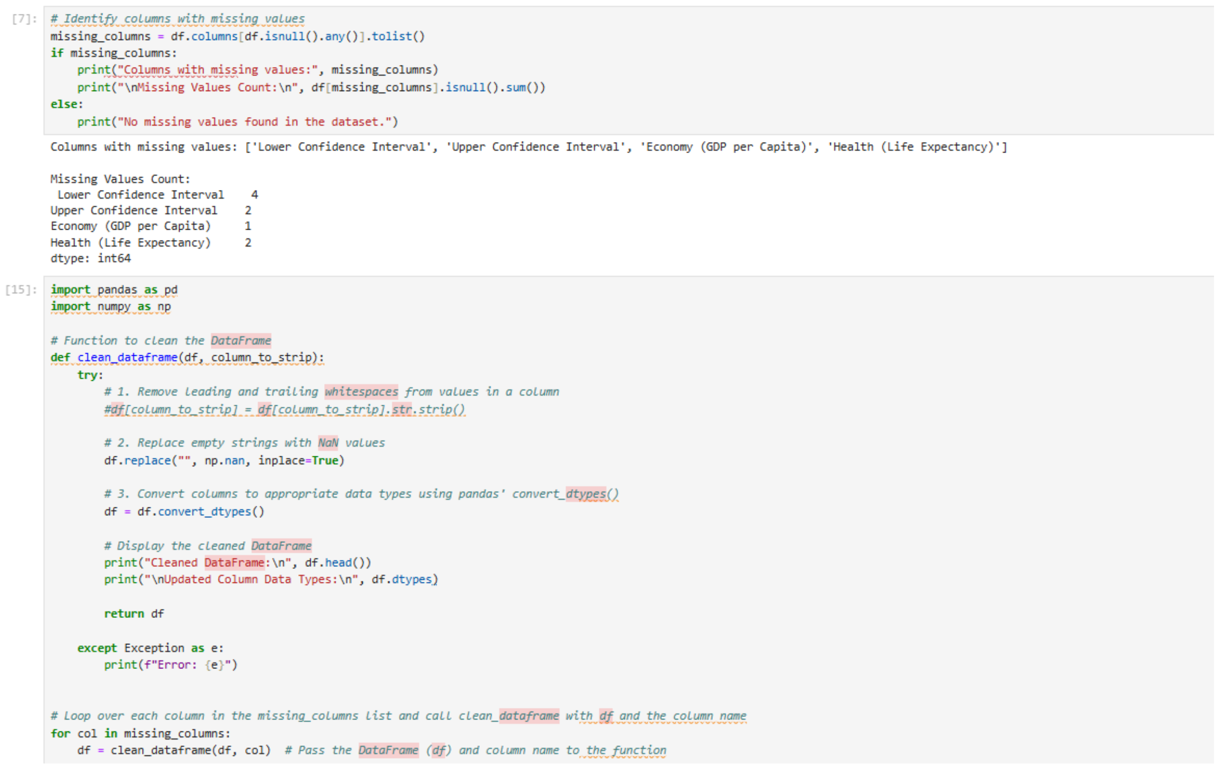
Write a python code to do the following tasks as per latest pandas:
1. Remove leading and trailing whitespaces from the values in a column.
2. Clean a column in a DataFrame by replacing empty strings with NaN values.
3. Change the data type of the columns to appropriate type as per the latest version of pandas.

Here's a Python script using the latest version of **pandas** to clean a DataFrame by:

1. Removing leading and trailing whitespaces from values in a column.
2. Replacing empty strings with NaN values.
3. Automatically converting column data types to appropriate types.



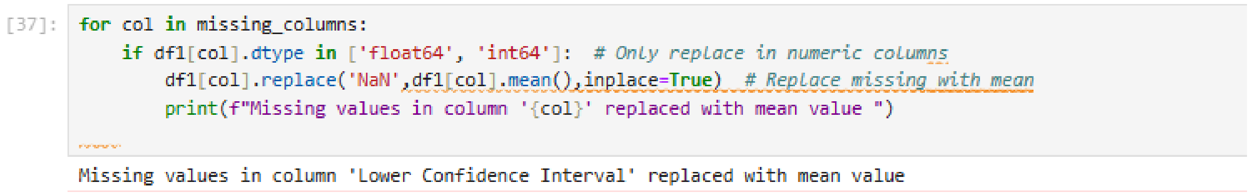
```python
import pandas as pd
import numpy as np

# Function to clean the DataFrame
def clean_dataframe(file_path, column_to_strip):
    try:
        # Read the CSV file
        df = pd.read_csv(file_path)

        # 1. Remove leading and trailing whitespaces from values in a column
        df[column_to_strip] = df[column_to_strip].str.strip()

        # 2. Replace empty strings with NaN values
        df.replace("", np.nan, inplace=True)

        # 3. Convert columns to appropriate data types using pandas' convert_dtypes()
        df = df.convert_dtypes()

        # Display the cleaned DataFrame
        print("Cleaned DataFrame:\n", df.head())
        print("\nUpdated Column Data Types:\n", df.dtypes)

        return df

    except Exception as e:
        print(f"Error: {e}")

# Provide the file path and column name to clean
file_path = "path/to/your/file.csv"  # Replace with your actual file path
column_to_strip = "column_name"  # Replace wi the column name to clean
clean_dataframe(file_path, column_to_strip)
```



```
[7]: # Identify columns with missing values
     missing_columns = df.columns[df.isnull().any()].tolist()
     if missing_columns:
         print("Columns with missing values:", missing_columns)
         print("\nMissing Values Count:\n", df[missing_columns].isnull().sum())
     else:
         print("No missing values found in the dataset.")
```

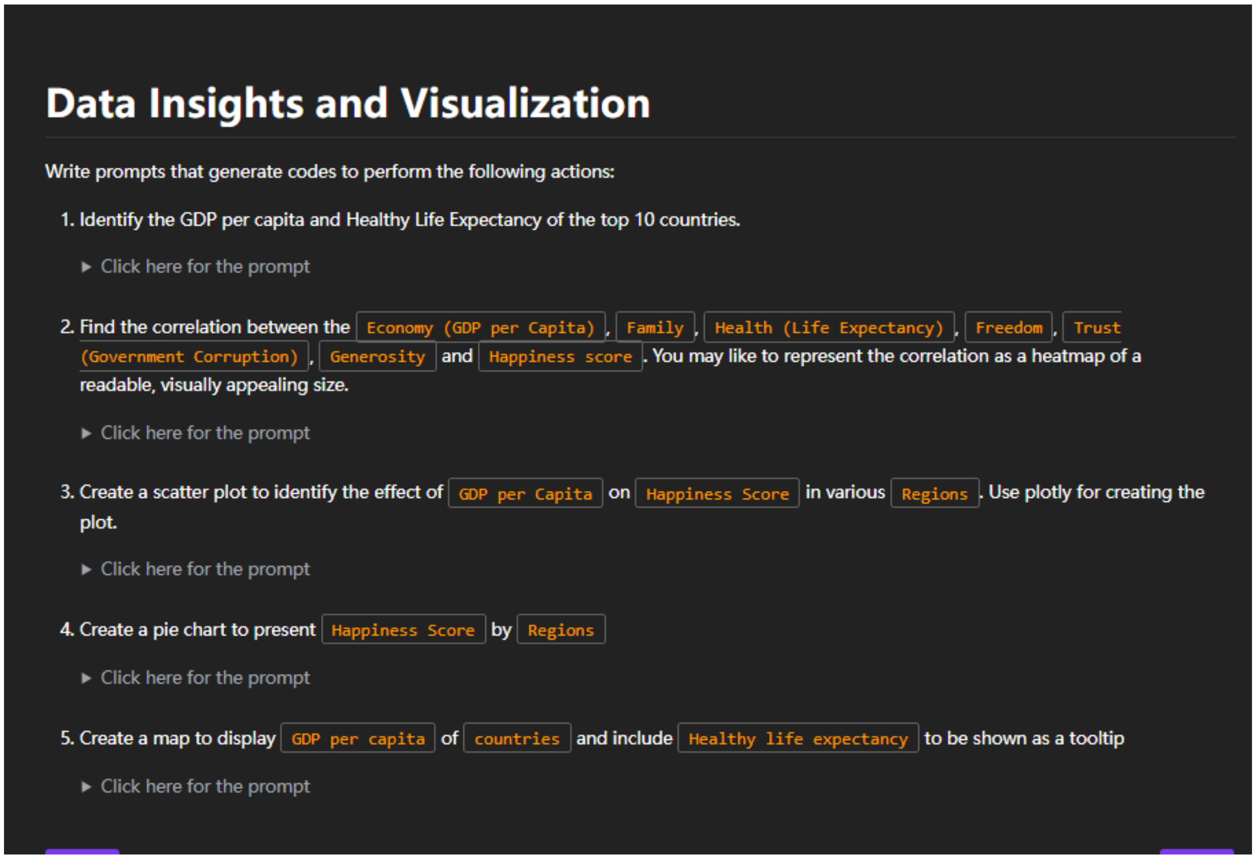
```
Columns with missing values: ['Lower Confidence Interval', 'Upper Confidence Interval', 'Economy (GDP per Capita)', 'Health (Life Expectancy)']

Missing Values Count:
 Lower Confidence Interval    4
Upper Confidence Interval    2
Economy (GDP per Capita)     1
Health (Life Expectancy)     2
dtype: int64
```

```
[15]: import pandas as pd
      import numpy as np

      # Function to clean the DataFrame
      def clean_dataframe(df, column_to_strip):
          try:
              # 1. Remove leading and trailing whitespaces from values in a column
              #df[column_to_strip] = df[column_to_strip].str.strip()

              # 2. Replace empty strings with NaN values
              df.replace("", np.nan, inplace=True)

              # 3. Convert columns to appropriate data types using pandas' convert_dtypes()
              df = df.convert_dtypes()

              # Display the cleaned DataFrame
              print("Cleaned DataFrame:\n", df.head())
              print("\nUpdated Column Data Types:\n", df.dtypes)

              return df

          except Exception as e:
              print(f"Error: {e}")


      # Loop over each column in the missing_columns list and call clean_dataframe with df and the column name
      for col in missing_columns:
          df = clean_dataframe(df, col)  # Pass the DataFrame (df) and column name to the function
```



```
[37]: for col in missing_columns:
          if df1[col].dtype in ['float64', 'int64']:  # Only replace in numeric columns
              df1[col].replace('NaN',df1[col].mean(),inplace=True)  # Replace missing with mean
              print(f"Missing values in column '{col}' replaced with mean value ")
```

```
Missing values in column 'Lower Confidence Interval' replaced with mean value
```



Data Insights and Visualization

Write prompts that generate codes to perform the following actions:

1. Identify the GDP per capita and Healthy Life Expectancy of the top 10 countries.

   ▶ Click here for the prompt

2. Find the correlation between the `Economy (GDP per Capita)`, `Family`, `Health (Life Expectancy)`, `Freedom`, `Trust (Government Corruption)`, `Generosity` and `Happiness score`. You may like to represent the correlation as a heatmap of a readable, visually appealing size.

   ▶ Click here for the prompt

3. Create a scatter plot to identify the effect of `GDP per Capita` on `Happiness Score` in various `Regions`. Use plotly for creating the plot.

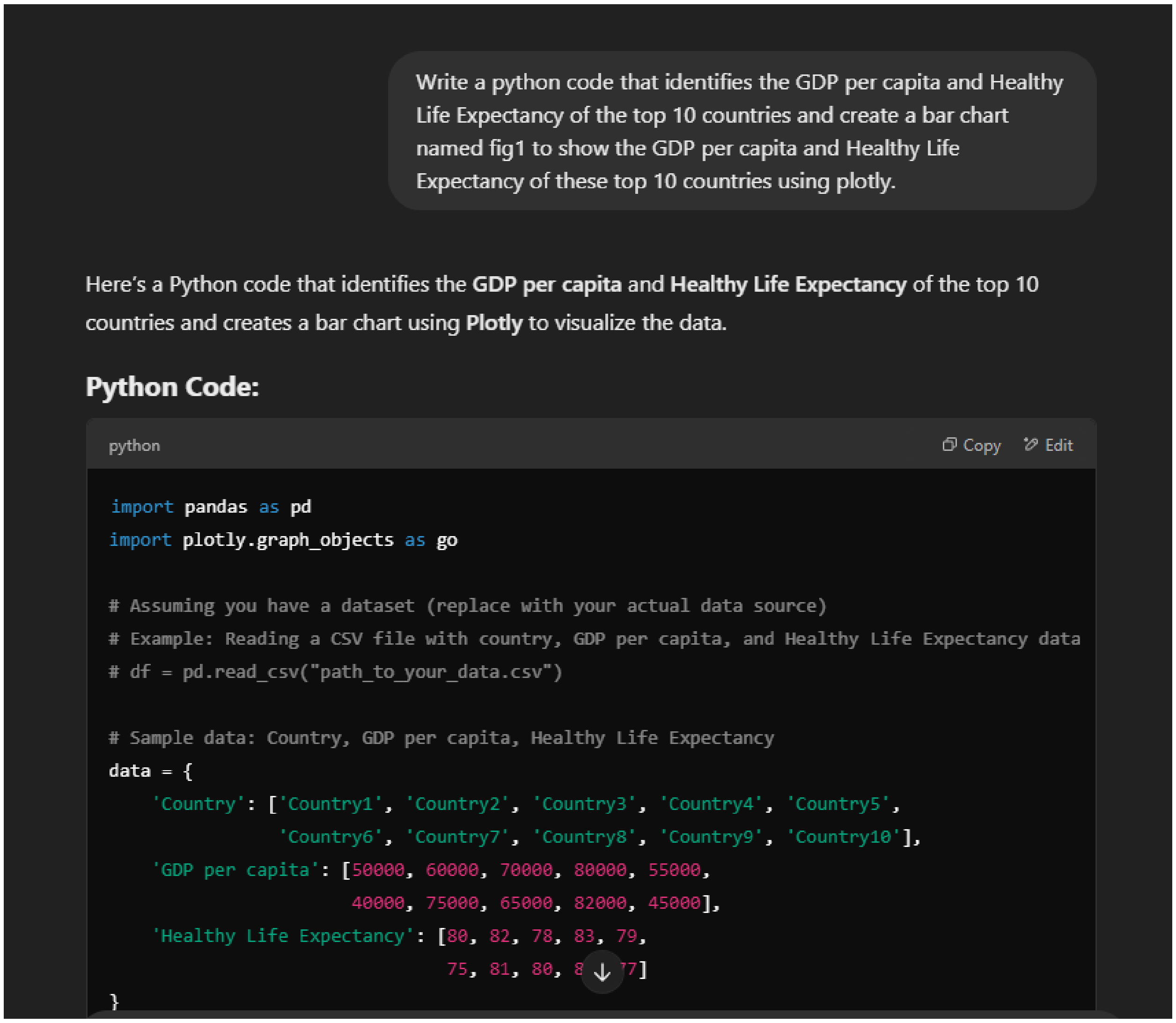
   ▶ Click here for the prompt

4. Create a pie chart to present `Happiness Score` by `Regions`

   ▶ Click here for the prompt

5. Create a map to display `GDP per capita` of `countries` and include `Healthy life expectancy` to be shown as a tooltip

   ▶ Click here for the prompt

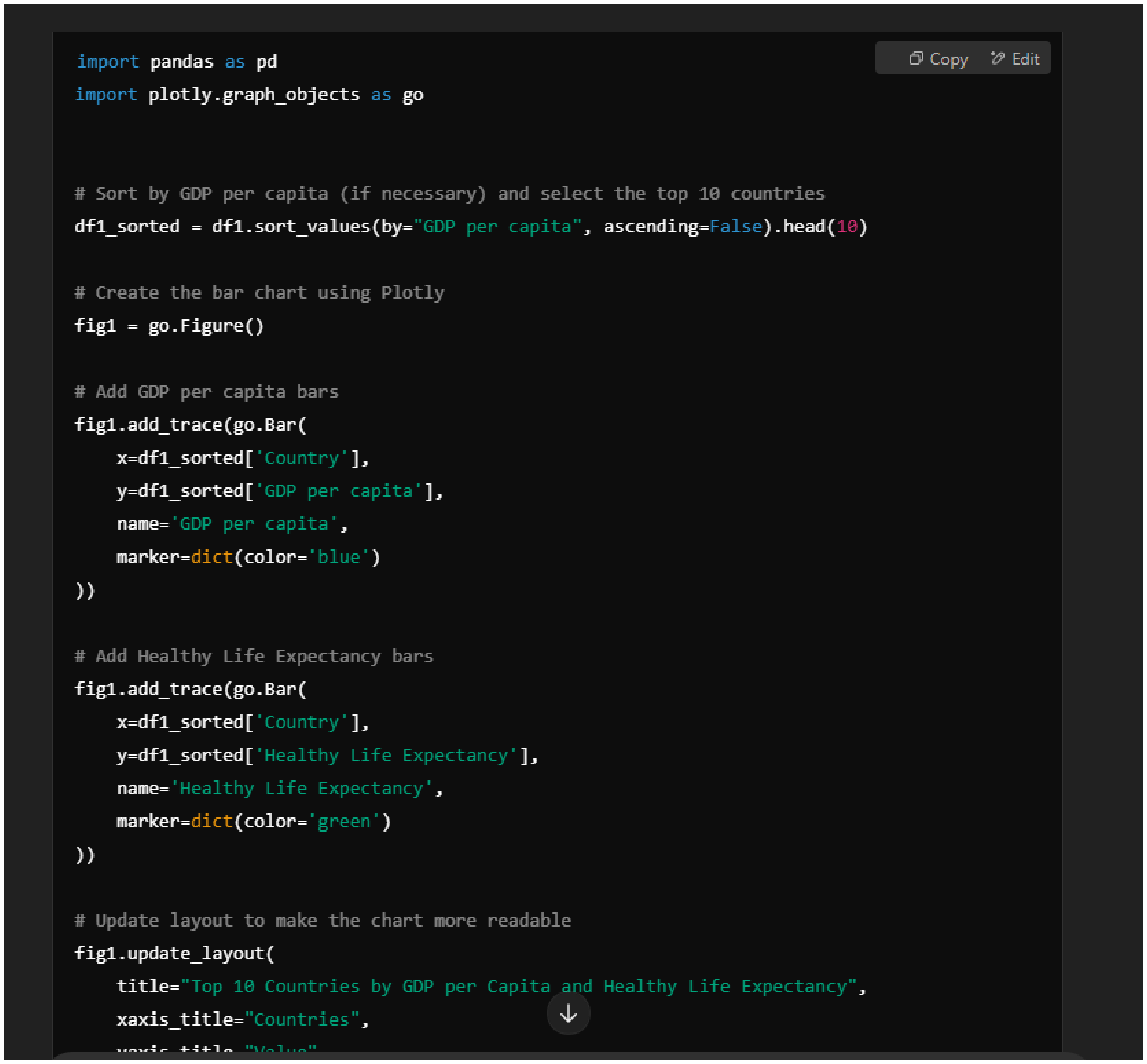


Write a python code that identifies the GDP per capita and Healthy Life Expectancy of the top 10 countries and create a bar chart named fig1 to show the GDP per capita and Healthy Life Expectancy of these top 10 countries using plotly.

Here's a Python code that identifies the **GDP per capita** and **Healthy Life Expectancy** of the top 10 countries and creates a bar chart using **Plotly** to visualize the data.

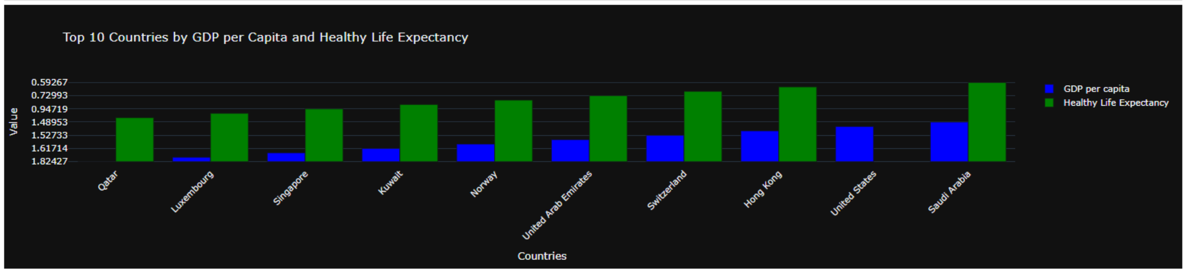
Python Code:

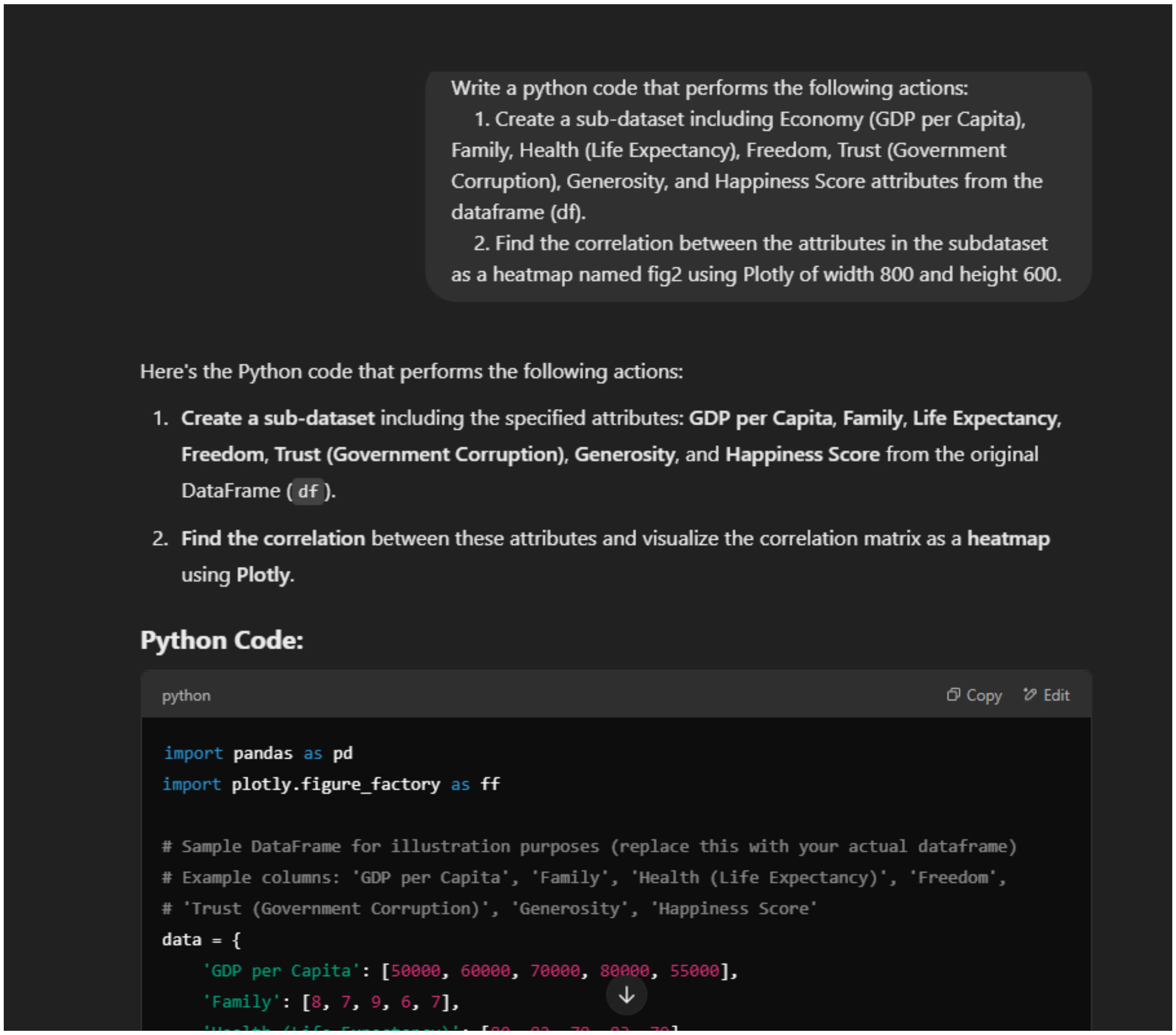
```python
import pandas as pd
import plotly.graph_objects as go

# Assuming you have a dataset (replace with your actual data source)
# Example: Reading a CSV file with country, GDP per capita, and Healthy Life Expectancy data
# df = pd.read_csv("path_to_your_data.csv")

# Sample data: Country, GDP per capita, Healthy Life Expectancy
data = {
    'Country': ['Country1', 'Country2', 'Country3', 'Country4', 'Country5',
                'Country6', 'Country7', 'Country8', 'Country9', 'Country10'],
    'GDP per capita': [50000, 60000, 70000, 80000, 55000,
                       40000, 75000, 65000, 82000, 45000],
    'Healthy Life Expectancy': [80, 82, 78, 83, 79,
                                75, 81, 80, 8[illegible]7]
}
```



```python
import pandas as pd
import plotly.graph_objects as go


# Sort by GDP per capita (if necessary) and select the top 10 countries
df1_sorted = df1.sort_values(by="GDP per capita", ascending=False).head(10)

# Create the bar chart using Plotly
fig1 = go.Figure()

# Add GDP per capita bars
fig1.add_trace(go.Bar(
    x=df1_sorted['Country'],
    y=df1_sorted['GDP per capita'],
    name='GDP per capita',
    marker=dict(color='blue')
))

# Add Healthy Life Expectancy bars
fig1.add_trace(go.Bar(
    x=df1_sorted['Country'],
    y=df1_sorted['Healthy Life Expectancy'],
    name='Healthy Life Expectancy',
    marker=dict(color='green')
))

# Update layout to make the chart more readable
fig1.update_layout(
    title="Top 10 Countries by GDP per Capita and Healthy Life Expectancy",
    xaxis_title="Countries",
    yaxis_title="Value"
```

```
import pandas as pd
import plotly.graph_objects as go


# Sort by GDP per capita (if necessary) and select the top 10 countries
df1_sorted = df1.sort_values(by="Economy (GDP per Capita)", ascending=False).head(10)

# Create the bar chart using Plotly
fig1 = go.Figure()

# Add GDP per capita bars
fig1.add_trace(go.Bar(
    x=df1_sorted['Country'],
    y=df1_sorted['Economy (GDP per Capita)'],
    name='GDP per capita',
    marker=dict(color='blue')
))

# Add Healthy Life Expectancy bars
fig1.add_trace(go.Bar(
    x=df1_sorted['Country'],
    y=df1_sorted['Health (Life Expectancy)'],
    name='Healthy Life Expectancy',
    marker=dict(color='green')
))

# Update layout to make the chart more readable
fig1.update_layout(
    title="Top 10 Countries by GDP per Capita and Healthy Life Expectancy",
    xaxis_title="Countries",
    yaxis_title="Value",
    barmode='group',  # Group bars together
    xaxis_tickangle=-45,  # Rotate country names for better readability
    template='plotly_dark'  # Use a dark theme for the plot
)

# Show the figure
fig1.show()
```





Write a python code that performs the following actions:
1. Create a sub-dataset including Economy (GDP per Capita), Family, Health (Life Expectancy), Freedom, Trust (Government Corruption), Generosity, and Happiness Score attributes from the dataframe (df).
2. Find the correlation between the attributes in the subdataset as a heatmap named fig2 using Plotly of width 800 and height 600.

Here's the Python code that performs the following actions:

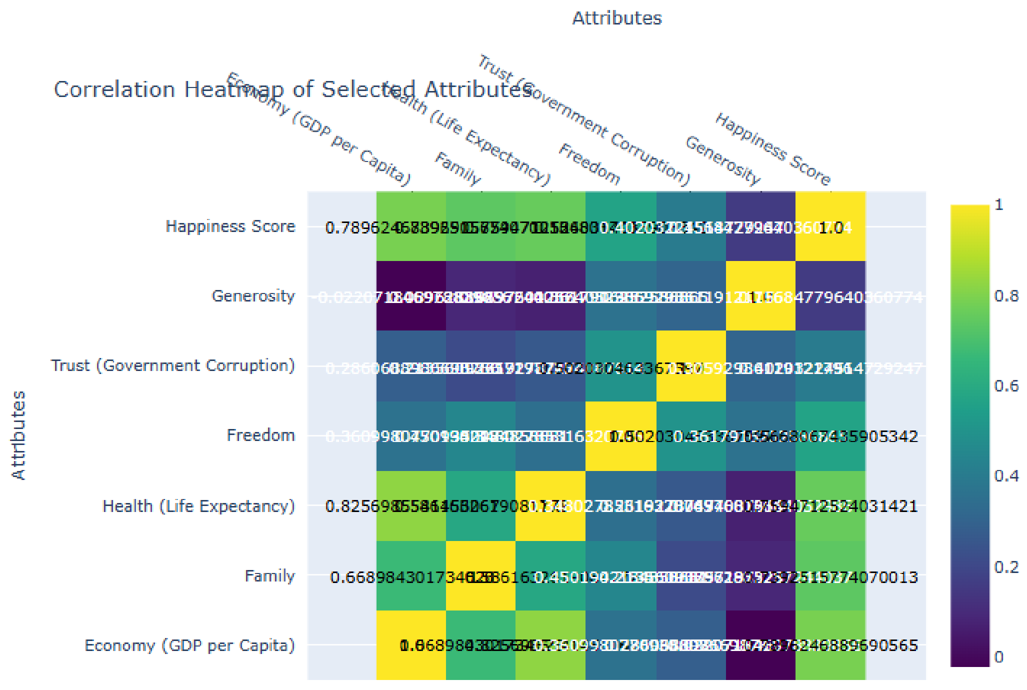
1. **Create a sub-dataset** including the specified attributes: **GDP per Capita**, **Family**, **Life Expectancy**, **Freedom**, **Trust (Government Corruption)**, **Generosity**, and **Happiness Score** from the original DataFrame (`df`).
2. **Find the correlation** between these attributes and visualize the correlation matrix as a **heatmap** using **Plotly**.

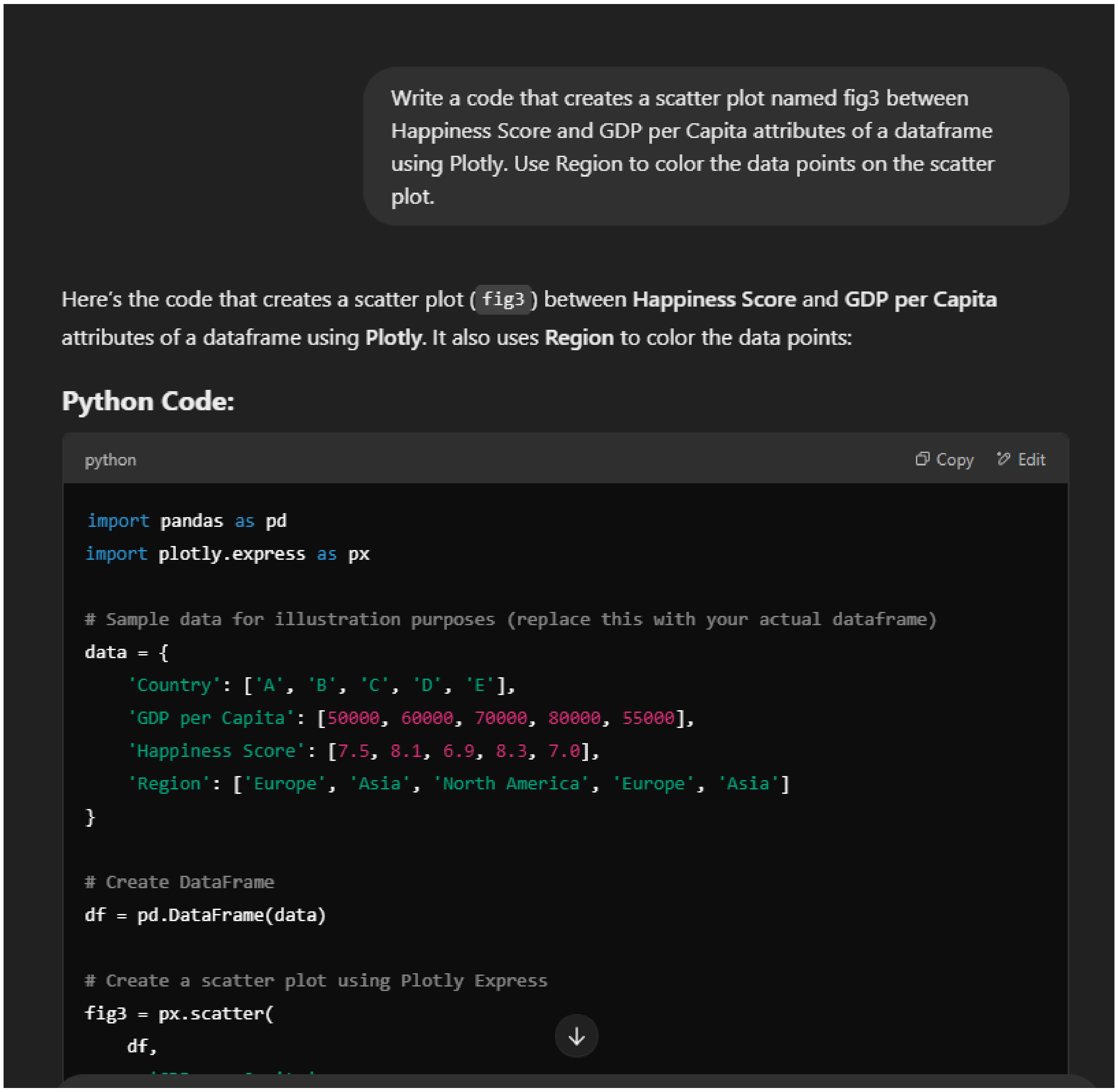
Python Code:

```python
import pandas as pd
import plotly.figure_factory as ff

# Sample DataFrame for illustration purposes (replace this with your actual dataframe)
# Example columns: 'GDP per Capita', 'Family', 'Health (Life Expectancy)', 'Freedom',
# 'Trust (Government Corruption)', 'Generosity', 'Happiness Score'
data = {
    'GDP per Capita': [50000, 60000, 70000, 80000, 55000],
    'Family': [8, 7, 9, 6, 7],
```

[45]:

```python
import pandas as pd
import plotly.figure_factory as ff

# Create sub-dataset with specified columns
sub_dataset = df1[['Economy (GDP per Capita)', 'Family', 'Health (Life Expectancy)',
                   'Freedom', 'Trust (Government Corruption)', 'Generosity', 'Happiness Score']]

# Step 1: Clean the data by converting to numeric and handling errors (non-numeric values)
# This will coerce any non-numeric values to NaN (Not a Number)
sub_dataset = sub_dataset.apply(pd.to_numeric, errors='coerce')

# Step 2: Fill missing values (NaN) if necessary
# You can fill NaN values with the column's mean (or another strategy such as median)
sub_dataset.fillna(sub_dataset.mean(), inplace=True)

# Step 3: Calculate the correlation matrix
correlation_matrix = sub_dataset.corr()

# Step 4: Create a heatmap using Plotly
fig2 = ff.create_annotated_heatmap(
    z=correlation_matrix.values,
    x=correlation_matrix.columns.tolist(),
    y=correlation_matrix.index.tolist(),
    colorscale='Viridis',  # Color scale for the heatmap
    showscale=True
)

# Step 5: Update layout for the figure
fig2.update_layout(
    title="Correlation Heatmap of Selected Attributes",
    xaxis_title="Attributes",
    yaxis_title="Attributes",
    width=800,  # Width of the heatmap
    height=600  # Height of the heatmap
)

# Show the figure
fig2.show()
```

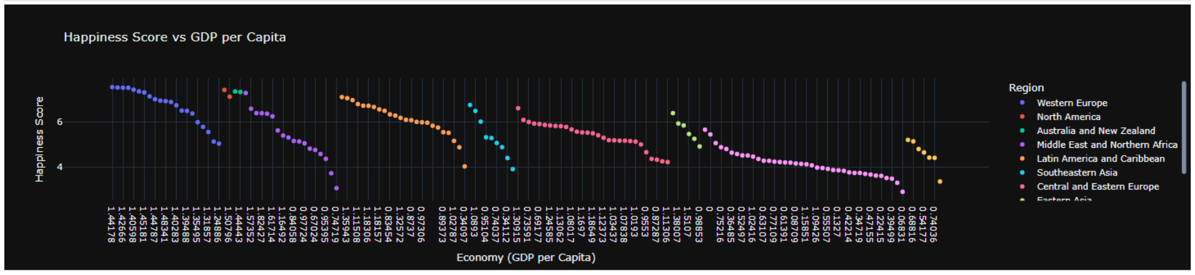

Correlation Heatmap of Selected Attributes
Attributes
Economy (GDP per Capita)
Family
Health (Life Expectancy)
Freedom
Trust (Government Corruption)
Generosity
Happiness Score
Attributes
Happiness Score
Generosity
Trust (Government Corruption)
Freedom
Health (Life Expectancy)
Family
Economy (GDP per Capita)
1
0.8
0.6
0.4
0.2
0

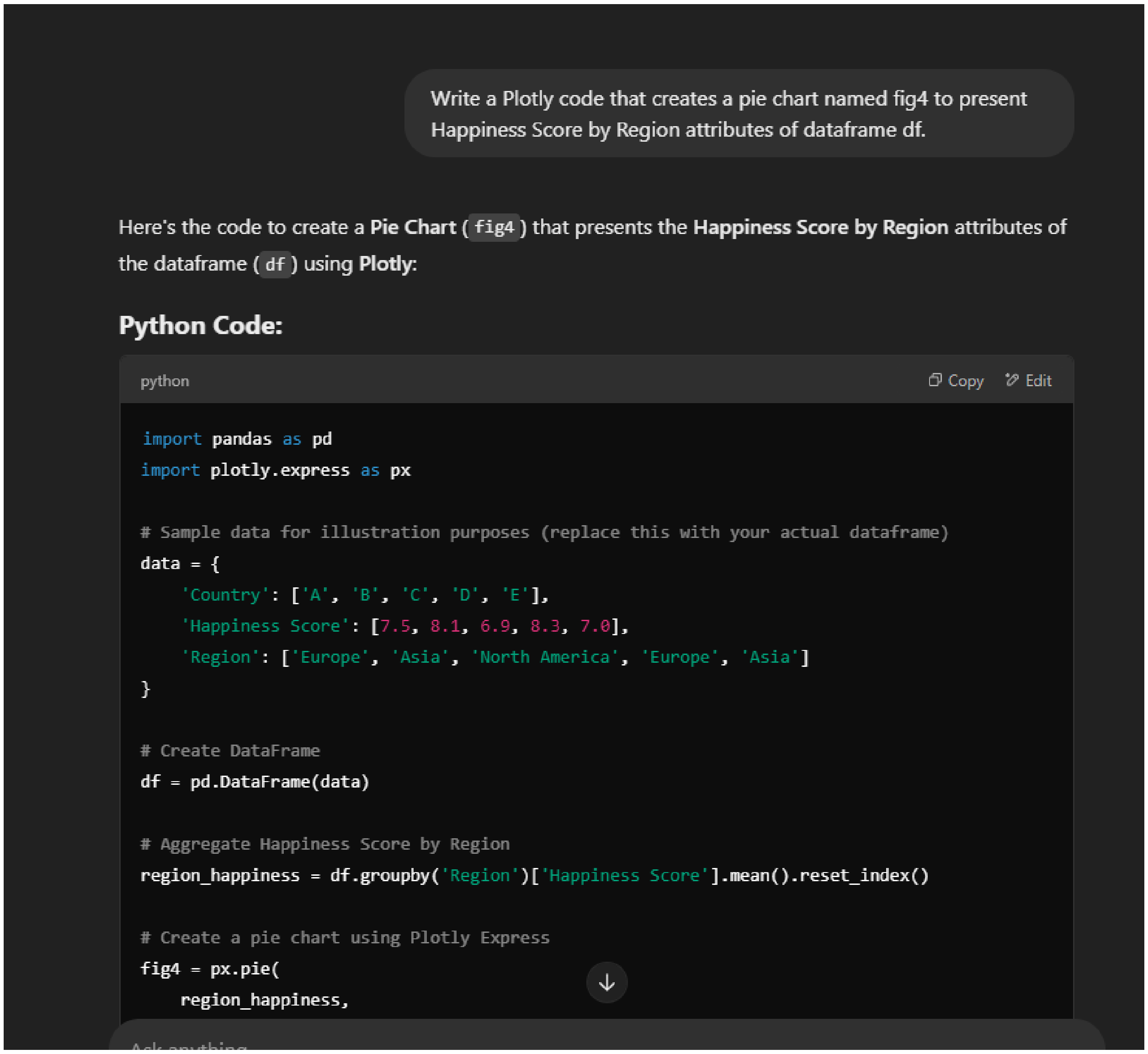

Write a code that creates a scatter plot named fig3 between Happiness Score and GDP per Capita attributes of a dataframe using Plotly. Use Region to color the data points on the scatter plot.

Here's the code that creates a scatter plot (`fig3`) between **Happiness Score** and **GDP per Capita** attributes of a dataframe using **Plotly**. It also uses **Region** to color the data points:

Python Code:

```python
import pandas as pd
import plotly.express as px

# Sample data for illustration purposes (replace this with your actual dataframe)
data = {
    'Country': ['A', 'B', 'C', 'D', 'E'],
    'GDP per Capita': [50000, 60000, 70000, 80000, 55000],
    'Happiness Score': [7.5, 8.1, 6.9, 8.3, 7.0],
    'Region': ['Europe', 'Asia', 'North America', 'Europe', 'Asia']
}

# Create DataFrame
df = pd.DataFrame(data)

# Create a scatter plot using Plotly Express
fig3 = px.scatter(
    df,
[46]: import pandas as pd
import plotly.express as px

# Create a scatter plot using Plotly Express
fig3 = px.scatter(
    df1,
    x='Economy (GDP per Capita)',
    y='Happiness Score',
    color='Region',  # Color the points based on Region
    title="Happiness Score vs GDP per Capita",
    labels={"GDP per Capita": "GDP per Capita", "Happiness Score": "Happiness Score"},
    template='plotly_dark'  # Use a dark theme for the plot
)

# Show the figure
fig3.show()
```





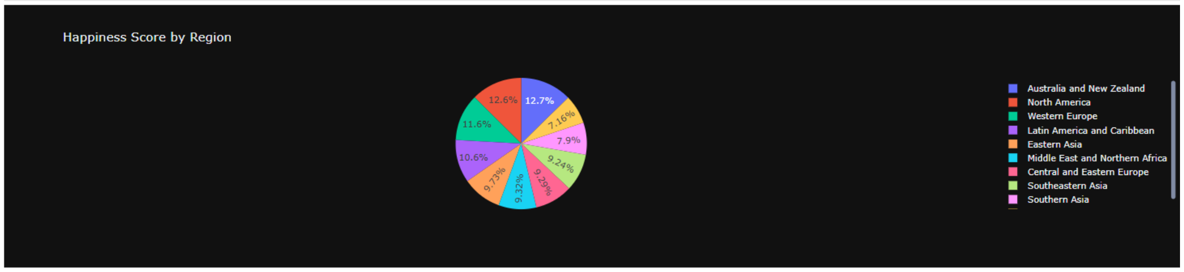
Write a Plotly code that creates a pie chart named fig4 to present Happiness Score by Region attributes of dataframe df.

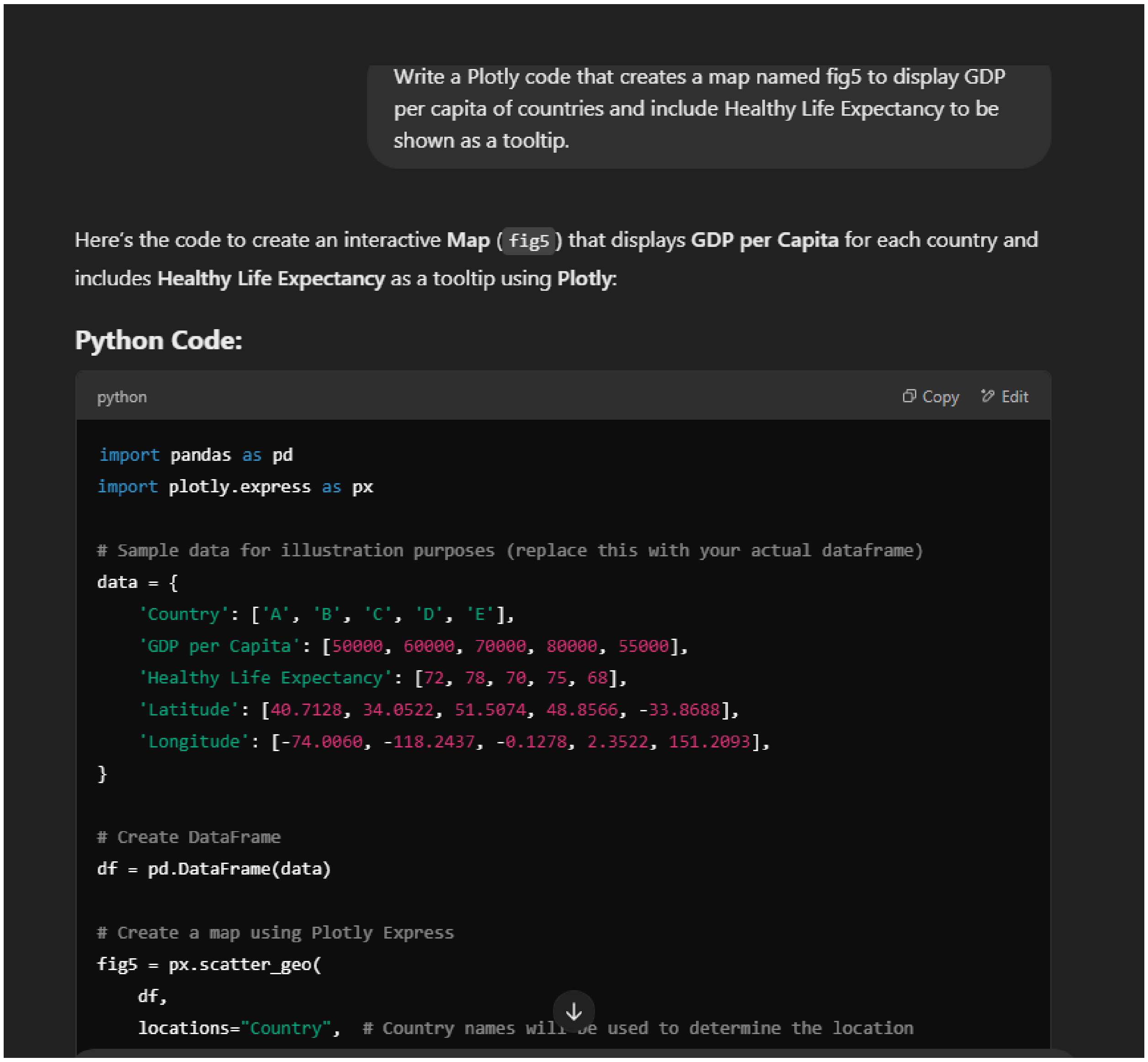
Here's the code to create a **Pie Chart** (`fig4`) that presents the **Happiness Score by Region** attributes of the dataframe (`df`) using **Plotly**:

Python Code:

```python
import pandas as pd
import plotly.express as px

# Sample data for illustration purposes (replace this with your actual dataframe)
data = {
    'Country': ['A', 'B', 'C', 'D', 'E'],
    'Happiness Score': [7.5, 8.1, 6.9, 8.3, 7.0],
    'Region': ['Europe', 'Asia', 'North America', 'Europe', 'Asia']
}

# Create DataFrame
df = pd.DataFrame(data)

# Aggregate Happiness Score by Region
region_happiness = df.groupby('Region')['Happiness Score'].mean().reset_index()

# Create a pie chart using Plotly Express
fig4 = px.pie(
    region_happiness,
```

[47]:
```python
import pandas as pd
import plotly.express as px

# Aggregate Happiness Score by Region
region_happiness = df1.groupby('Region')['Happiness Score'].mean().reset_index()

# Create a pie chart using Plotly Express
fig4 = px.pie(
    region_happiness,
    names='Region',
    values='Happiness Score',
    title="Happiness Score by Region",
    template='plotly_dark'  # Use a dark theme for the plot
)

# Show the figure
fig4.show()
```





Write a Plotly code that creates a map named fig5 to display GDP per capita of countries and include Healthy Life Expectancy to be shown as a tooltip.

Here's the code to create an interactive **Map** (`fig5`) that displays **GDP per Capita** for each country and includes **Healthy Life Expectancy** as a tooltip using **Plotly**:

Python Code:

python

```python
import pandas as pd
import plotly.express as px

# Sample data for illustration purposes (replace this with your actual dataframe)
data = {
    'Country': ['A', 'B', 'C', 'D', 'E'],
    'GDP per Capita': [50000, 60000, 70000, 80000, 55000],
    'Healthy Life Expectancy': [72, 78, 70, 75, 68],
    'Latitude': [40.7128, 34.0522, 51.5074, 48.8566, -33.8688],
    'Longitude': [-74.0060, -118.2437, -0.1278, 2.3522, 151.2093],
}

# Create DataFrame
df = pd.DataFrame(data)

# Create a map using Plotly Express
fig5 = px.scatter_geo(
    df,
    locations="Country",  # Country names will be used to determine the location
```

[50]:
```python
import pandas as pd
import plotly.express as px

# Ensure the 'Economy (GDP per Capita)' column is numeric
df1['Economy (GDP per Capita)'] = pd.to_numeric(df1['Economy (GDP per Capita)'], errors='coerce')
df1['Health (Life Expectancy)'] = pd.to_numeric(df1['Health (Life Expectancy)'], errors='coerce')

# Fill NaN values in 'Economy (GDP per Capita)' with 0 or any other appropriate value
df1['Economy (GDP per Capita)'].fillna(0, inplace=True)

# Create a map using Plotly Express
fig5 = px.scatter_geo(
    df1,
    locations="Country",  # Country names will be used to determine the location
    size="Economy (GDP per Capita)",  # Size of the points will be based on GDP per Capita
    hover_name="Country",  # Hover label will show the country name
    hover_data=["Health (Life Expectancy)"],  # Tooltip will display Healthy Life Expectancy
    title="GDP per Capita and Healthy Life Expectancy by Country",
    projection="natural earth",  # Map projection style
    template='plotly_dark'  # Use a dark theme for the map
)

# Show the map
fig5.show()
```

Dashboarding and Storytelling

Write prompts that generate codes to write at least four of the graph plots generated in the previous steps into a HTML page.

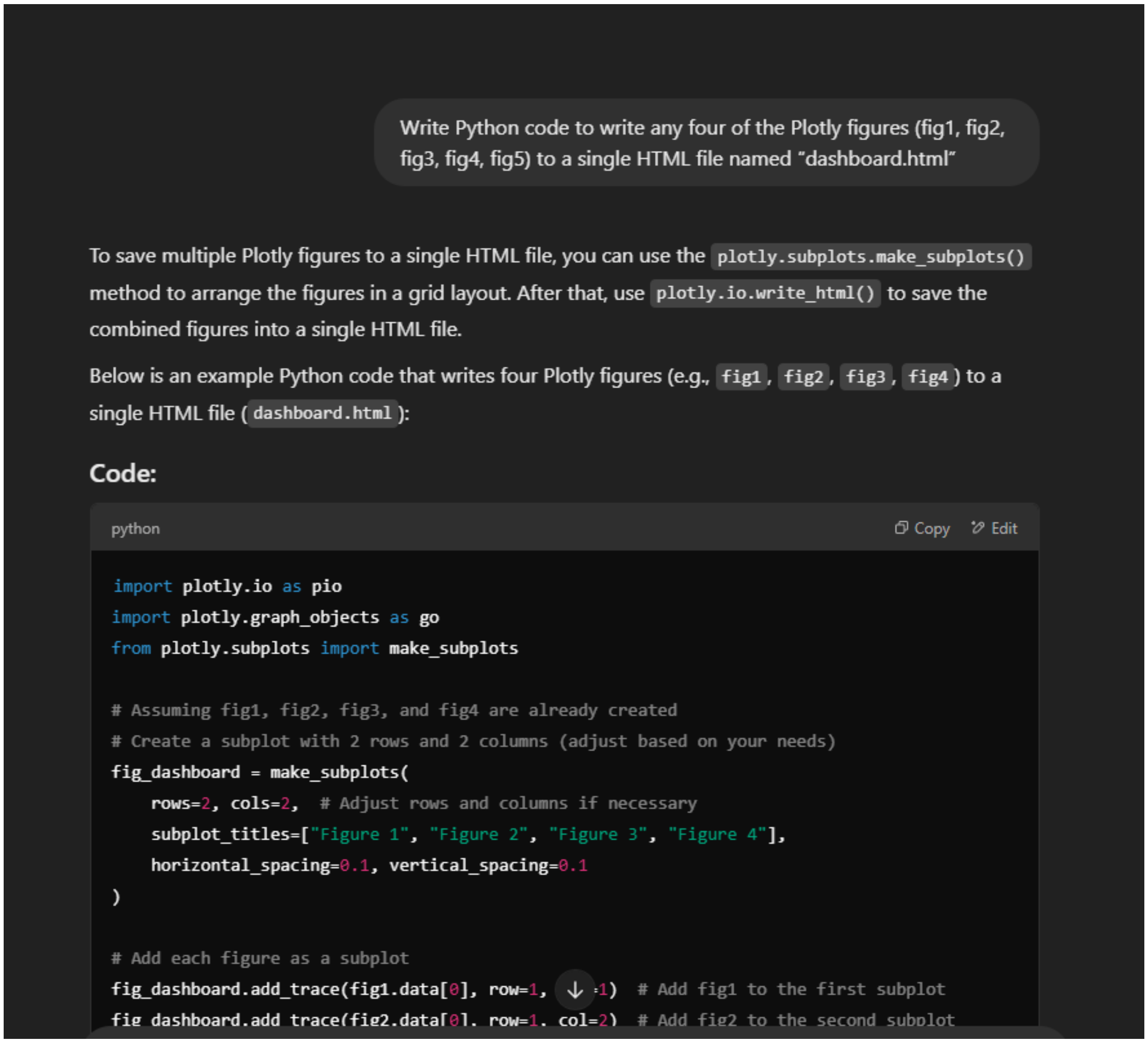
▶ Click here for the prompt

> *NOTE: Please ensure that the code has the plotly codes generated for creating various visualizations in the previous section.*

2. Generate the narrative to present the dashboard

▶ Click here for the prompt

You can use this narrative to showcase your dashboard in a business meeting!



Write Python code to write any four of the Plotly figures (fig1, fig2, fig3, fig4, fig5) to a single HTML file named "dashboard.html"

To save multiple Plotly figures to a single HTML file, you can use the `plotly.subplots.make_subplots()` method to arrange the figures in a grid layout. After that, use `plotly.io.write_html()` to save the combined figures into a single HTML file.

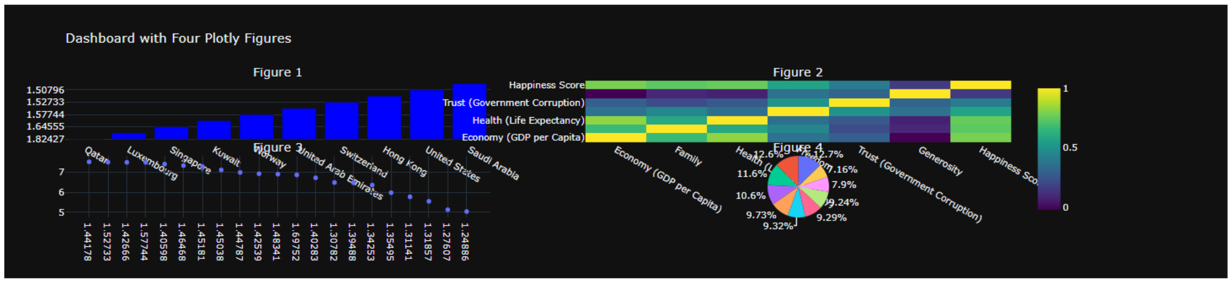
Below is an example Python code that writes four Plotly figures (e.g., `fig1`, `fig2`, `fig3`, `fig4`) to a single HTML file (`dashboard.html`):

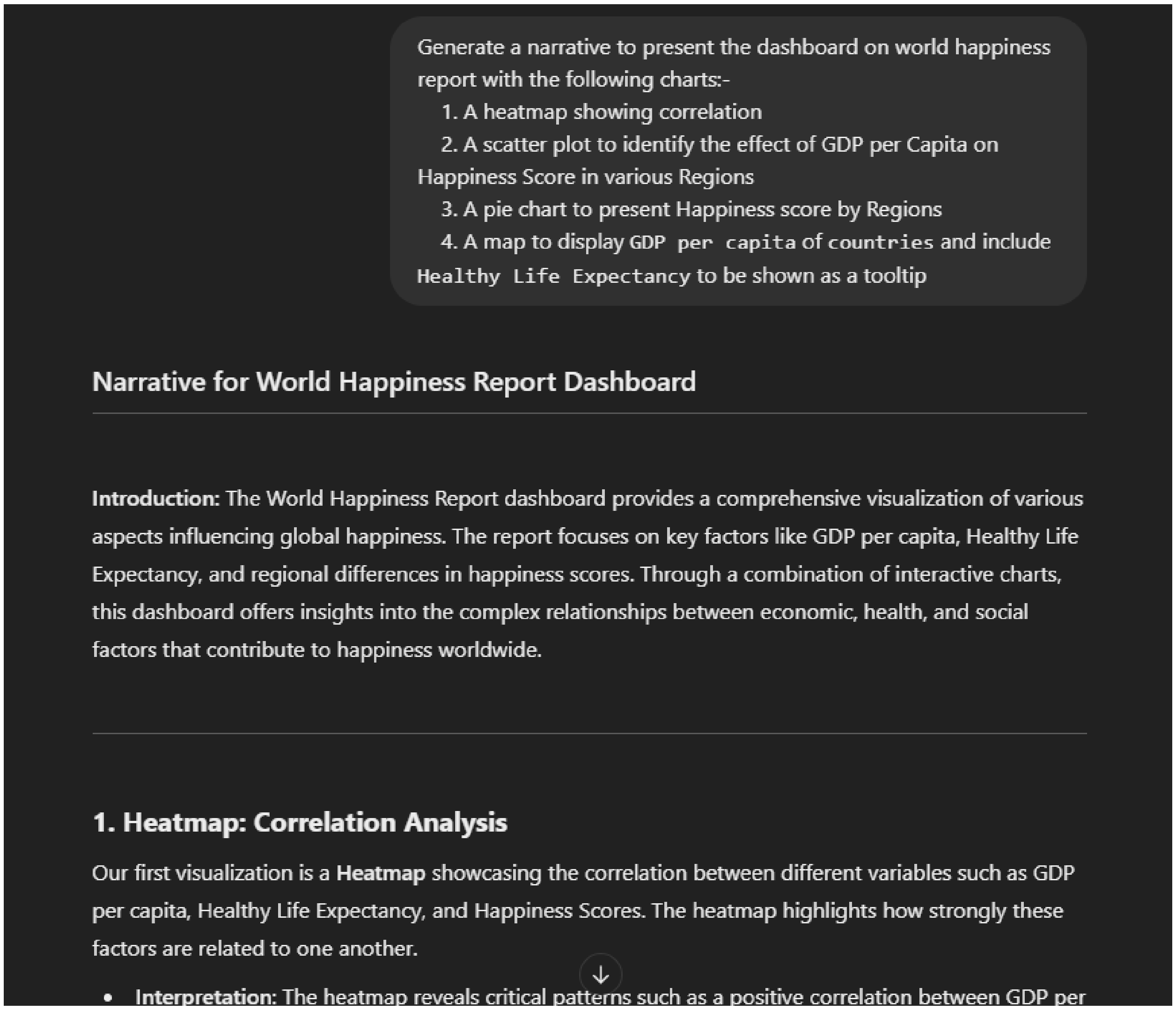
Code:

python

```python
import plotly.io as pio
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Assuming fig1, fig2, fig3, and fig4 are already created
# Create a subplot with 2 rows and 2 columns (adjust based on your needs)
fig_dashboard = make_subplots(
    rows=2, cols=2,  # Adjust rows and columns if necessary
    subplot_titles=["Figure 1", "Figure 2", "Figure 3", "Figure 4"],
    horizontal_spacing=0.1, vertical_spacing=0.1
)

# Add each figure as a subplot
fig_dashboard.add_trace(fig1.data[0], row=1, [illegible]=1)  # Add fig1 to the first subplot
fig_dashboard.add_trace(fig2.data[0], row=1, col=2)  # Add fig2 to the second subplot
```

```
[52]: import plotly.io as pio
      import plotly.graph_objects as go
      from plotly.subplots import make_subplots

      # Create a subplot with 2 rows and 2 columns.
      # Specify the subplot types: "xy" for scatter plots and "domain" for pie charts
      fig_dashboard = make_subplots(
          rows=2, cols=2,  # Adjust rows and columns if necessary
          subplot_titles=["Figure 1", "Figure 2", "Figure 3", "Figure 4"],
          horizontal_spacing=0.1, vertical_spacing=0.1,
          specs=[[{"type": "xy"}, {"type": "xy"}],  # First row: scatter/xy plots
                 [{"type": "xy"}, {"type": "domain"}]]  # Second row: scatter/xy and pie chart
      )

      # Add each figure as a subplot
      fig_dashboard.add_trace(fig1.data[0], row=1, col=1)  # Add fig1 to the first subplot
      fig_dashboard.add_trace(fig2.data[0], row=1, col=2)  # Add fig2 to the second subplot
      fig_dashboard.add_trace(fig3.data[0], row=2, col=1)  # Add fig3 to the third subplot
      fig_dashboard.add_trace(fig4.data[0], row=2, col=2)  # Add fig4 (pie chart) to the fourth subplot

      # Update layout for the dashboard
      fig_dashboard.update_layout(
          title_text="Dashboard with Four Plotly Figures",
          template='plotly_dark',  # Optional: Choose your template
          showlegend=False
      )

      # Write the dashboard to an HTML file
      pio.write_html(fig_dashboard, file='dashboard.html')

      # Optionally, you can view the dashboard in a browser
      fig_dashboard.show()
```





Generate a narrative to present the dashboard on world happiness report with the following charts:-

1. A heatmap showing correlation
2. A scatter plot to identify the effect of GDP per Capita on Happiness Score in various Regions
3. A pie chart to present Happiness score by Regions
4. A map to display `GDP per capita` of `countries` and include `Healthy Life Expectancy` to be shown as a tooltip

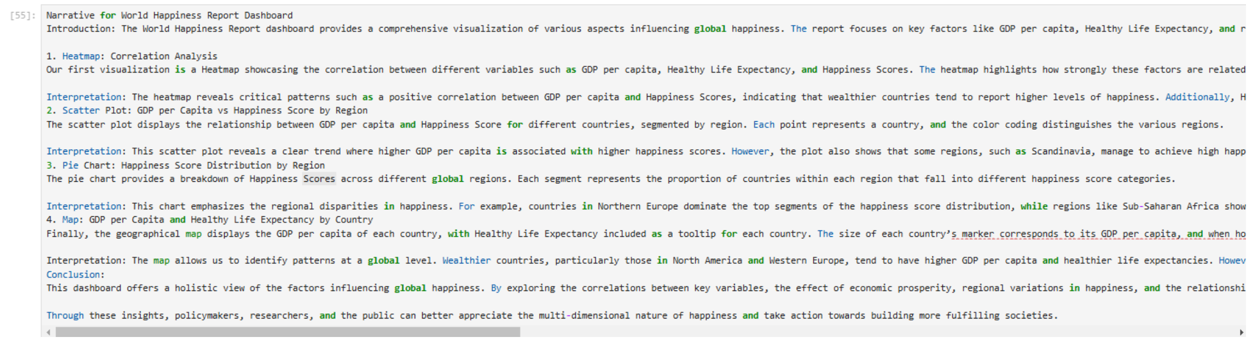
Narrative for World Happiness Report Dashboard

Introduction: The World Happiness Report dashboard provides a comprehensive visualization of various aspects influencing global happiness. The report focuses on key factors like GDP per capita, Healthy Life Expectancy, and regional differences in happiness scores. Through a combination of interactive charts, this dashboard offers insights into the complex relationships between economic, health, and social factors that contribute to happiness worldwide.

1. Heatmap: Correlation Analysis

Our first visualization is a **Heatmap** showcasing the correlation between different variables such as GDP per capita, Healthy Life Expectancy, and Happiness Scores. The heatmap highlights how strongly these factors are related to one another.

- **Interpretation:** The heatmap reveals critical patterns such as a positive correlation between GDP per



```
[55]: Narrative for World Happiness Report Dashboard
Introduction: The World Happiness Report dashboard provides a comprehensive visualization of various aspects influencing global happiness. The report focuses on key factors like GDP per capita, Healthy Life Expectancy, and r

1. Heatmap: Correlation Analysis
Our first visualization is a Heatmap showcasing the correlation between different variables such as GDP per capita, Healthy Life Expectancy, and Happiness Scores. The heatmap highlights how strongly these factors are related

Interpretation: The heatmap reveals critical patterns such as a positive correlation between GDP per capita and Happiness Scores, indicating that wealthier countries tend to report higher levels of happiness. Additionally, H
2. Scatter Plot: GDP per Capita vs Happiness Score by Region
The scatter plot displays the relationship between GDP per capita and Happiness Score for different countries, segmented by region. Each point represents a country, and the color coding distinguishes the various regions.

Interpretation: This scatter plot reveals a clear trend where higher GDP per capita is associated with higher happiness scores. However, the plot also shows that some regions, such as Scandinavia, manage to achieve high happ
3. Pie Chart: Happiness Score Distribution by Region
The pie chart provides a breakdown of Happiness Scores across different global regions. Each segment represents the proportion of countries within each region that fall into different happiness score categories.

Interpretation: This chart emphasizes the regional disparities in happiness. For example, countries in Northern Europe dominate the top segments of the happiness score distribution, while regions like Sub-Saharan Africa show
4. Map: GDP per Capita and Healthy Life Expectancy by Country
Finally, the geographical map displays the GDP per capita of each country, with Healthy Life Expectancy included as a tooltip for each country. The size of each country’s marker corresponds to its GDP per capita, and when ho

Interpretation: The map allows us to identify patterns at a global level. Wealthier countries, particularly those in North America and Western Europe, tend to have higher GDP per capita and healthier life expectancies. Howev
Conclusion:
This dashboard offers a holistic view of the factors influencing global happiness. By exploring the correlations between key variables, the effect of economic prosperity, regional variations in happiness, and the relationshi

Through these insights, policymakers, researchers, and the public can better appreciate the multi-dimensional nature of happiness and take action towards building more fulfilling societies.
```