Milestone3_DSC540

September 22, 2024

Said Moussadeq

17-May-2024

DSC540, Milestone 3

Substance Abuse, Lifestyle Choices, and Their Impact on Mental Health and Suicide Risk

Step 1: Send a GET Request

[46]:
```python
import requests

# URL of the Wikipedia page
url = "https://en.wikipedia.org/wiki/List_of_countries_by_suicide_rate"

# Step 1: Send a GET request to the URL
response = requests.get(url)
```

Step 2: Parse HTML Content

[47]:
```python
from bs4 import BeautifulSoup

# Step 2: Parse the HTML content of the page
soup = BeautifulSoup(response.content, 'html.parser')
```

Step 3: Find the Table

[48]:
```python
# Step 3: Find the table with the specific classes
table = soup.find('table', class_='wikitable sortable sticky-header')
```

Step 4: Extract Headers

[49]:
```python
# Step 4: Extract the headers
if table:
    headers = [header.text.strip() for header in table.find_all('th')]
    print("Original Headers:", headers)
```

```
Original Headers: ['Country', 'Male', 'Female', 'Overall', 'Sources & Year']
```

Step 5: Define New Headers

```
[50]: # Step 5: Define new headers for readability
      new_headers = ['Country', 'Male', 'Female', 'Overall', 'Sources & Year']
```

Step 6: Extract Rows

```
[51]: # Step 6: Extract the rows of the table
      rows = table.find_all('tr')
```

Step 7: Extract Data

```
[52]: # Step 7: Extract data from each row and ensure the correct number of columns
      data = []
      for row in rows[1:]:
          cols = row.find_all('td')
          cols = [col.text.strip() for col in cols]
          if len(cols) == len(headers):
              data.append(cols)
```

Step 8: Create DataFrame

```
[53]: import pandas as pd

      # Step 8: Create a DataFrame
      df = pd.DataFrame(data, columns=new_headers)

      # Display the original DataFrame head as a table
      from IPython.display import display, HTML
      print("Original DataFrame Head:")
      display(df.head())
```

```
Original DataFrame Head:

                       Country   Male Female Overall    Sources & Year
0  Greenland * (Danish Realm)  75.08  25.18   53.34           2019[23]
1                 Lithuania *   32.2    6.7    18.6  2022[24][25][26]
2               South Korea *   35.9   16.2    26.0           2021[27]
3                    Guyana *  41.25  10.20   25.52       2017[28][29]
4                Kazakhstan *  40.68   8.01   23.81           2017[29]
```

Step 9: Convert Numeric Columns

Columns like ‘Male’, ‘Female’, and ‘Overall’ represent numerical suicide rates. If these values are stored as strings (which might happen due to scraping from HTML), converting them to numeric types ensures they can be used in calculations, statistical analyses, and visualizations without issues.

[54]:
```
# Step 9: Convert numeric columns to appropriate matching data types

numeric_columns = ['Male', 'Female', 'Overall']
for col in numeric_columns:
    df[col] = pd.to_numeric(df[col].str.replace(',', ''), errors='coerce')
```

Step 10: Handle Missing Values

Handling missing values ensures that the dataset remains complete and ready for any coming analysis or processing steps

[55]:
```
# Step 10: Handle missing values (fill with 0 or appropriate value)

df.fillna(0, inplace=True)
```

Step 11: Remove Duplicates

Since each country can only be represented once, removing duplicates ensures that each country's suicide rate statistics are unique and accurately represented.

[56]:
```
# Step 11: Identify and remove duplicates

df.drop_duplicates(inplace=True)
```

Step 12: Fix Casing/Inconsistent Values

[57]:
```
# Step 12: Fix casing or inconsistent values

df['Country'] = df['Country'].str.title()
```

Step 13: Standardize "Sources & Year" Column

Standardizing the "Sources & Year" column ensures consistency, improves data quality, simplifies analysis, removes ambiguity, and enhances usability. This makes it easier to filter, group, and aggregate the data by year.

[58]:
```
import re

# Standardize "Sources & Year" column

df['Year'] = df['Sources & Year'].apply(lambda x: re.findall(r'\d{4}', x)[-1]␣
 ↪if re.findall(r'\d{4}', x) else None)
df['Year'] = pd.to_numeric(df['Year'], errors='coerce')
```

Step 14: Remove Special Characters

[59]:
```
# Step 14: Remove special characters
df['Country'] = df['Country'].str.replace(r'[^a-zA-Z\s]', '', regex=True)
```

Step 15: Categorize Data

Categorizing suicide rate data simplifies the analysis and interpretation of data. It also facilitates comparisons between countries, helps recognize patterns and trends, and enhances visualization.

[60]:
```
# Step 15: Categorize data
# Categorizing the 'Overall' column to group countries by suicide rates
df['Overall Category'] = pd.cut(df['Overall'], bins=[-float('inf'), -1, 0, 10,␣
  ↪20, 30, float('inf')],
                                labels=['Invalid', 'Zero', 'Low', 'Moderate',␣
  ↪'High', 'Very High'])
```

Step 16: Drop "Sources & Year" Column

[61]:
```
# Step 16: Drop the original "Sources & Year" column as we now have a separate␣
  ↪"Year" column
df.drop(columns=['Sources & Year'], inplace=True)
```

Display Cleaned DataFrame

[62]:
```
# Display the cleaned DataFrame head
print("Cleaned DataFrame Head:")
display(df.head())
```

```
Cleaned DataFrame Head:

                   Country   Male  Female  Overall  Year Overall Category
0  Greenland Danish Realm  75.08   25.18    53.34  2019        Very High
1               Lithuania  32.20    6.70    18.60  2022         Moderate
2             South Korea  35.90   16.20    26.00  2021             High
3                  Guyana  41.25   10.20    25.52  2017             High
4              Kazakhstan  40.68    8.01    23.81  2017             High
```

Ethical Implications of Data Wrangling:

Dealing with a sensitive topic that has to do with the suicide rates across different countries requires a lot of accuracy, privacy and well-representation. The changes made to the data involved converting numeric columns, handling missing values, removing duplicates, standardizing and cleaning country names, extracting years, and categorizing data. There are no specific legal or regulatory guidelines for this public data sourced from the World Health Organization (WHO) to Wikipedia page, but it is important to ensure data integrity and avoid misrepresentation. The transformations doesn't introduce risks of inaccuracies. Assumptions made include filling missing values with 0 and interpreting special characters as irrelevant for country names. The data was sourced from a publicly available WHO page, generally considered credible.

```
[77]: import sqlite3
      from sqlalchemy import create_engine
```

```
[78]: print("Step 17: Loading data into SQLite database")
      engine = create_engine('sqlite:///suicide_rates.db', echo=True)
      sqlite_connection = engine.connect()

      sqlite_table = "SuicideRates"
      df.to_sql(sqlite_table, sqlite_connection, if_exists='fail', index=False)

      sqlite_connection.close()

      print("Data loaded into SQLite database successfully.")
```

```
Step 17: Loading data into SQLite database
2024-05-30 19:28:31,843 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-05-30 19:28:31,846 INFO sqlalchemy.engine.Engine PRAGMA
main.table_info("SuicideRates")
2024-05-30 19:28:31,849 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-05-30 19:28:31,851 INFO sqlalchemy.engine.Engine PRAGMA
temp.table_info("SuicideRates")
2024-05-30 19:28:31,852 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-05-30 19:28:31,857 INFO sqlalchemy.engine.Engine
CREATE TABLE "SuicideRates" (
        "Country" TEXT,
        "Male" FLOAT,
        "Female" FLOAT,
        "Overall" FLOAT,
        "Year" BIGINT,
        "Overall Category" TEXT
)


2024-05-30 19:28:31,858 INFO sqlalchemy.engine.Engine [no key 0.00142s] ()
2024-05-30 19:28:31,875 INFO sqlalchemy.engine.Engine INSERT INTO "SuicideRates"
("Country", "Male", "Female", "Overall", "Year", "Overall Category") VALUES (?,
?, ?, ?, ?, ?)
2024-05-30 19:28:31,875 INFO sqlalchemy.engine.Engine [generated in 0.00267s]
[('Greenland\u202f Danish Realm', 75.08, 25.18, 53.34, 2019, 'Very High'),
('Lithuania\u202f', 32.2, 6.7, 18.6, 2022, 'Moderate'), ('South Korea\u202f',
35.9, 16.2, 26.0, 2021, 'High'), ('Guyana\u202f', 41.25, 10.2, 25.52, 2017,
'High'), ('Kazakhstan\u202f', 40.68, 8.01, 23.81, 2017, 'High'), ('Slovenia',
0.0, 0.0, 18.09, 2016, 'Moderate'), ('Sri Lanka\u202f', 0.0, 0.0, 14.6, 2018,
'Moderate'), ('Hungary', 0.0, 0.0, 17.98, 2016, 'Moderate')  … displaying 10
of 112 total bound parameter sets …  ('Haiti', 0.0, 0.0, 0.0, 2003, 'Zero'),
('Nepal\u202f', 0.0, 0.0, 0.0, 2003, 'Zero')]
2024-05-30 19:28:31,875 INFO sqlalchemy.engine.Engine SELECT name FROM
sqlite_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER
```

```
BY name
2024-05-30 19:28:31,875 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-05-30 19:28:31,875 INFO sqlalchemy.engine.Engine COMMIT
Data loaded into SQLite database successfully.
```

[ ]: