Due Date: Monday 11/25, 11:59 PM

We will explore some aspects of the pandas package along with SQL. We will get experience with querying databases, filling missing values, joining tables and generating scatterplots. By completing Homework 9, you should take away...

- Practice connecting to a sqlite database to query tables for data
- Gain experience with the merge operation in pandas for joining tables by common entries.

These skills are helpful for exploratory data analysis.

Submission Instructions

For this assignment and future assignments (Homework 10,11) you will submit a copy to Gradescope. Follow these steps

1. Download as HTML (`File->Download As->HTML(.html)`).
2. Open the HTML in the browser. Print to .pdf
3. Upload to Gradescope. Tag your answers.

Note that

- Please map your answers to our questions. Otherwise you may lose points. Please see the rubric below.
- You should break long lines of code into multiple lines. Otherwise your code will extend out of view from the cell. Consider using `\` followed by a new line.
- For each textual response, please include relevant code that informed your response. For each plotting question, please include the code used to generate the plot.
- You should not display large output cells such as all rows of a table. Instead convert the input cell from Code to Markdown back to Code to remove the output cell.

Collaboration Policy

Data science is a collaborative activity. While you may talk with others about the homework, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** at the top of your solution.

Rubric

Question	Points
Gradescope	2
Question 0	1
Question 1	3
Question 2	2
Question 3	2
Question 4	1
Question 5	2
Question 6	2
Total	15

For this homework you will need the following database `demographic.db` available alongside the notebook. Please import the following packages

In [ ]:

```
import sqlalchemy
from IPython.display import Image
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
plt.rcParams['figure.figsize'] = (5, 5)
plt.rcParams['figure.dpi'] = 150
pd.options.display.max_rows = 20
pd.options.display.max_columns = 15
pd.set_option('precision', 2)
```

Question 0: (1 point)

Use the function `create_engine` in the `sqlalchemy` package to make a connection called `sqlite_engine` to the database `demographic.db` . You can specify the connection details with the string `sqlite_uri` below...

In [ ]:

```
sqlite_uri = "sqlite:///demographic.db"

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```
# TEST
inspector = sqlalchemy.inspect(sqlite_engine)
assert inspector.get_table_names() == ['Country', 'GDP']
```

In [ ]:

```

```

Use the `read_sql` function in the pandas package to load the `GDP` table in `demographic.db` into a DataFame called `df_gdp` . Recall that `*` is a wildcard in SQL meaning all columns for the `SELECT` command.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```
# TEST
assert len(df_gdp) == 260
```

Question 1: (3 point)

a. Use the method `set_index` for DataFrames to make `GDP per capita` the index.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```

```

b. Use the method `replace` for DataFrames to replace the empty string with `np.NaN` .

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```

```

c. Use the method `astype('float32')` for DataFrames to cast all strings to float.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```
# TEST
assert df_gdp.values.dtype == 'float32'
```

Transpose the table to have year for index and country for column.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```
# TEST
assert df_gdp.shape == (213, 260)
```

In [ ]:

```
```

Question 2: (2.0 points)

Plot *year* versus *GDP* for the three countries with the largest sum of GDP over the whole range of years.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

Question 3: (2.0 points)

Print the name of all countries that contain no missing values in the record of GDP. These columns should not contain `np.NaN` .

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

Plot GDP over time for these countries.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

Question 4: (1.0 points)

Use the `read_sql` function in the pandas package to load the `Country` table in `demographic.db` into a DataFame called `df_country` . Recall that `*` is a wildcard in SQL meaning all columns for the `SELECT` command.

In [ ]:

```
df_country = pd.read_sql("SELECT * FROM Country", sqlite_engine)
```

Use the method `set_index` for DataFrames to make `Country (en)` the index.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```
```

Question 5: (2.0 points)

Create a DataFrame called `df_average` from `df_gdp` by averaging the columns.

- Use the `fillna(0)` method of DataFrames to replace `np.NaN` with 0.
- Use the `mean()` method of DataFrames to average over the columns.

Note that `df_average` should contain one column called `Average GDP`.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

Create a DataFrame from `df_country` just containing `Life Expectancy` and `Population` called `df_country_temp`.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```
# TESTS
assert df_country_temp.columns[0] == 'Life expectancy'
assert df_country_temp.columns[1] == 'Population'
```

Use the `merge` function in pandas to inner join `df_average` and `df_country_temp` on their indices. Call the resulting DataFrame `df_merged`.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```

```

Question 6: (2.0 points)

Use the `sort_values` method for DataFrames to sort `df_merged` by the value in the column `Average GDP`. The order should be descending.

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```
# TEST
assert df_merged['Average GDP'].is_monotonic_decreasing
```

Use the method `astype('float32')` for DataFrames to cast string to float in `df_merged`

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```

We want to plot *Population* against *Life expectancy*...

- Take the first 100 rows in `df_merged` corresponding to the highest 100 Average GDP
- Apply the logarithm function to the column `Population` before generating the plot
- Normalize the "Average GDP" (e.g. dividing by the max) and scale the plotted points according to the normalized "Average GDP" (the size of the points should be scaled proportional to the normalized data).

Your plot should be similar to the following one:

In [ ]:

```
Image('dfscatter.png')
```

In [ ]:

```
# YOUR CODE HERE
raise NotImplementedError()
```