Problem Set 3: Pandas, Data Visualization

In this problem set, we will continue to explore Pandas. We will focus on data manipulation - appending, joining, and sorting. And generating new columns of data. Next, we will introduce **Matplotlib** which is a popular graphing library in Python. We will learn how to make simple graphs.

In [2]:

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

Often we need to add data to an existing DataFrame. Let us look at how we can append data in Pandas. Let us first define two new Dataframes.

In [3]:

```
df_upper=pd.DataFrame([[1,3], [0,2]], columns=['L', 'C'])
df_lower=pd.DataFrame([[-1, 0], [5, 4]], columns=['C', 'R'])
```

Q1. Suppose we wish to append df_lower to df_upper. Use the Pandas **concat()** method to stack df_lower below df_upper. Explore what happens to the row index, and data values.

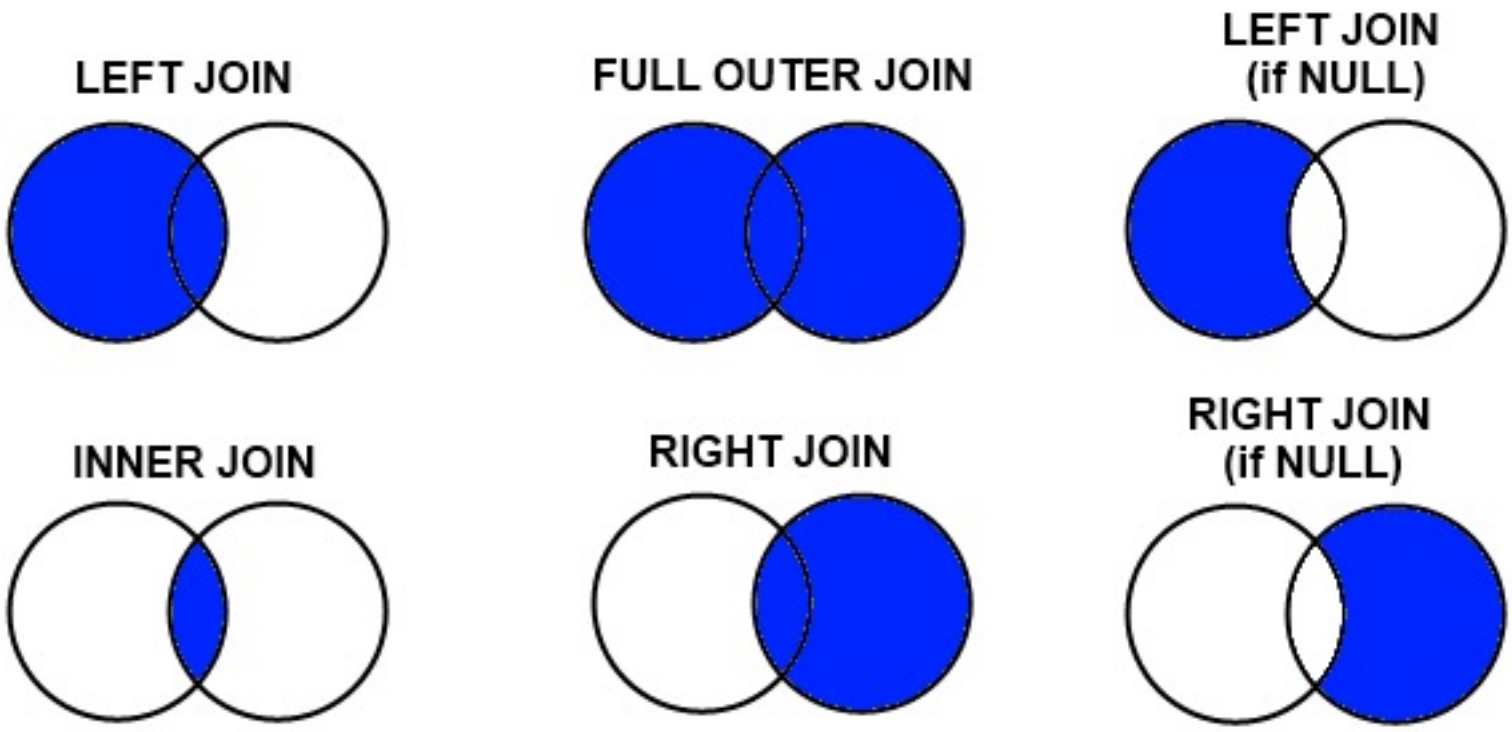
Q2. Use the **append()** method to stack df_lower below df_upper.

Q3. Suppose we wish to instead to append df_lower to the right of df_upper i.e. stack the dataframes by column. Use the **concat()** method to achieve this.

Q4. By default, .concat generates what is called an *outer join* i.e. the union of the dataframes. Suppose instead, we want the intersecting set i.e. only keep the overlapping (common to both dataframes) indexes or columns. Use the **concat()** method with the *inner join* option to stack the two dataframes by row (lower below the upper.)

Q5. Use the **concat()** method with the *inner join* option to combine the two dataframes by columns (lower to the right the upper.) Predict the output before verifying.

We will now look at **merging** dataframes. This involves combining two DataFrames so that the rows are aligned based on common values of specified columns. For instance, one dataframe may contain employee personal information such as employee id, name, age, and gender. Another may contain payroll information such as employee id, gross and net salary, and deductions. Suppose we wish to measure the gender gap in salary. We would want to merge the two dataframes using the values in employee id so that we have gender and salary in the same combined dataframe. The following graphic, sourced from *Stackoverflow* is a helpful overview of Merge/Join types as used in Pandas, R, SQL, and other data-orientated languages and libraries. Here, left refers to the first listed DataFrame and right to the second. The shaded area represent the resulting dataframe.



Define two new dataframes.

In [47]:

```
df_left=pd.DataFrame([['London', 'Yes'], ['Paris', 'Yes'], ['Frankfurt', 'No']],
```

In [48]:

```
df_right=pd.DataFrame([['London', 'UK'], ['Paris', 'France'], ['Madrid', 'Spain']]
```

In [49]:

```
df_left
```

Out[49]:

	City	Capital?
0	London	Yes
1	Paris	Yes
2	Frankfurt	No

In [50]:

```
df_right
```

Out[50]:

	City	Country
0	London	UK
1	Paris	France
2	Madrid	Spain

Q6. Use the **merge** method to merge the two dataframes on city values.

Q7. Merge the two dataframes on city value so that the result has the full contents of df_left and only the matching rows in df_right i.e. do a left join.

Q8. Merge the two dataframes on city value so that the result has the full contents of df_right and only the matching rows in df_left i.e. do a right join.

Q9. Finally, merge the two dataframes to include all data (matching and non-matching), and track which row comes from where. Hint: Use *how='outer'* and *indicator=True*

Now let us explore sorting in Pandas. One method is *sort_values* for which one has to specify a column to sort on. The default is an ascending sort. Note that unless you tell Pandas to do so, the sort does not reshuffle the underlying data. We will first read in a remote dataset.

```
url='http://bit.ly/imdbratings (http://bit.ly/imdbratings)'
movies=pd.read_csv(url)
```

Q10. Generate a new dataframe (movies) by reading in the imdb ratings data as shown above. Browse the first few rows to get a sense of the data.

Q11. How many movies are there in the dataframe?

Q12. Sort the dataframe on star_rating and display the first 10 and then the last 10.

Q13. Sort the dataframe on star_rating and then on duration, and display the first 10 and then the last 10.

Q14. Which is the longest movie in the data?

Q15. Generate a new column in the dataframe that has the length of the title (number of characters in the title including any spaces). Use the **apply** method and the **len** function.

Q16. Generate a new column that has the starting character of the movie title. Display a frequency count of the starting character. Hint: Use the **str[]** function to get the first character of the title string.

Q17. List the movie titles starting with 'Z.'

Visualization

Python offers a wide range of graphing capabilities. A commonly used graphing library is *matplotlib*. Another is *seaborn*. Import matplotlib.pyplot as shown below. The next line allows you to generate plots within Notebook.

```
import matplotlib.pyplot as plt
%matplotlib notebook
```

As for most of Python modules, there are numerous excellent free online resources that can serve as quick introduction to Matplotlib. The official documentation with illustrative examples is at:
https://matplotlib.org/users/pyplot_tutorial.html (https://matplotlib.org/users/pyplot_tutorial.html)

One of the many nice introductory videos: https://www.youtube.com/watch?v=a9UrKTVEeZA (https://www.youtube.com/watch?v=a9UrKTVEeZA)

Q18. Define two series X and Y, with X ranging from 0 to 9, and Y = 3X+4.

Q19. Graph Y against X in a line plot. Add axes labels ('X' and 'Y'). Add a graph title "My first plot in Python!'.

Q20. Graph Y against X once again, but use 'o' as a market symbol to get a scatter plot instead of a line plot. Add axes labels ('X' and 'Y'). Add a graph title "My first plot in Python!'.

Q21. Graph the exponential function $\exp(x)$ for 20 equally spaced values of x ranging from -1 to +1. Hint: You may find the **numpy linspace** and **numpy exp** functions useful.

In [ ]: