Day 8: *DataFrames in Pandas*

Dot enjoyed some great food, weather, and sports entertainment in Barcelona. They made their way to the airport and boarded a plane for their next destination, this time with no airport dramatics. They were headed over to one of Europe's romantic heartlands, the so-called City of Lights – Paris! They arrived at the busy Roissy airport and made their way into the city center or centre-ville. The cobblestoned streets were flecked with old and modern buildings, an eclectic mix that gave the city a timeless feeling.

Dot saw quaint little boats paddling down the Seine river and the skeletal Eiffel tower rising over the landscape. Dot was so excited to take in the beauty and artfulness of the French capital! They consulted a little booklet they brought with them, listing popular things to do as a tourist in Paris. The botanical gardens, Notre Dame cathedral, the Louvre...Dot wanted to visit all of it, but they simply didn't have the time. To optimize their trip, Dot decided to examine the queue times and entry prices for Parisian landmarks. Let's engage with this dataset alongside Dot!

Tutorial

Pandas is one of the most widely used Python libraries. Pandas can be used when working with large datasets, or when performing data cleaning, wrangling, and analysis.

In this week of challenges we will be using some basic Pandas functionalities to help us gain insights into the datasets we're working with.

But before we can use the Pandas library, we have to import it.

In [ ]:
```
#import the pandas plugin
import pandas as pd # pd is the alias we have given to pandas.
```

Because Pandas exists externally to basic Python, we need to import it. When importing, we give an alias to Pandas so it shortens our code when calling on functions from pandas. Instead of writing **pandas.name_of_function()** we'll be able to write **pd.name_of_function()**. The alias pd is standard within the Python community.

The first pandas function we'll use is called using **pd.read_csv()**. This function reads the path to a **.csv** file and stores it in a Pandas DataFrame. A DataFrame in Pandas is a representation of data in a table, similar to what we typically see when working within Excel.

```
df = pd.read_csv('path/to/csv/sample.csv')
```

The second function we'll use is **df.head()**, which by default displays the top 5 rows of a dataframe. However, we can change the number of rows it displays by inputting an integer value between the parentheses after .head with the number we would like to observe.

```
df.head(21)
```

The third function we'll use is ***df.tail()***, which by default displays the bottom five rows of a dataframe. However, we can change the number of rows it displays by inputting an integer value between the parentheses after *.tail* with the number we would like to observe.

```
df.tail(4)
```

In [ ]:
```
# To Read a Dataset
# paris_landmarks.csv is stored into a Pandas DataFrame variable called df.
df = pd.read_csv('paris_landmarks.csv')

#df.head() function displays the first 5 rows of the dataset
df.head()
```

Try out the following functions on our Pandas DataFrame *df*, and see what you can learn from the dataset.

DataFrame Functions

- `df.describe()` *provides descriptive statistics of all numerical columns*
- `df.unique()` *provides the number of unique items in a column*
- `df.shape()` *gets the number of rows and columns in the dataframe*
- `df.sort_values()` *sorts the dataframe by specific column*

DataFrame Column Functions

- `.info()` *provides an overview of all the columns, number of non-nulls, and data types in a DataFrame*
- `.max()` *gets the max value from a column*
- `.min()` *gets the min value from a column*
- `.mean()` *get the mean value from a column*
- `.idxmax()` *gets the integer index position of the max value from a column*
- `.idxmin()` *gets the integer index position of the min value from a column*
- `.loc()` *gets rows (or columns) with particular labels from the index*
- `.iloc()` *gets rows (or columns) with particular positions in the index (only takes integers)*

```
#example using info()
df.info()

#example calling the max number from a column
df['column_name'].max()
```

The `df.sort_values()` allows us to reorder our dataframe in an ascending or descending order given a column for pandas to work from. This is similar to the excel sort function.

```
# parameter ascending indicates the direction we order in
df.sort_values(by=['column_name'], ascending = False)
```

To learn more about the various pandas functions, check out the user guide in the pandas documentation.

In [ ]:
```
import pandas as pd
```

In [ ]:
```
df = pd.read_csv('paris_landmarks.csv')
# we sort our data by queue_time, starting from the longest
df.sort_values(by="queue_time",ascending=False).head()
```

In [ ]:
```
df.sort_values(by="queue_time",ascending=False)
```

Challenge

After playing around with the functions above, you can help Dot by answering following questions about Paris landmarks:

1. **What is the most expensive landmark in Paris?**
2. **What is the average wait time for all landmarks?**

In [ ]:
```
import pandas as pd
df = pd.read_csv('paris_landmarks.csv')
df.head()
```

In [ ]:
```
# SOLUTION

df.iloc[df['price'].idxmax()]['landmark']
```

In [ ]:
```
df['queue_time'].mean()
```