Lab 1: Basic Data Analysis ¶

The purpose of this lab is to get you started with running Jupyter notebooks, and getting you familiar with loading and analyzing data in a notebook. You could use the notebooks from the lectures as reference points, as well.

Learning Objectives

The purpose of this lab is to give you a gentle introduction to Python notebooks, as well as an introduction to loading a dataset into a notebook and performing some basic queries with the notebook.

After completing this lab, you should be able to:

1. Start a Juypter notebook and write/execute basic Python code in the notebook.
2. Understand basic Python operations, including how to write a function, and how to load data.
3. Understand the purpose of Pandas and plotting libraries (e.g., matplotlib) and gain some initial, basic experience with them.

1. Python and Notebooks: Hello World

If you've gotten this far, hopefully you have the notebook running, but you should have started this from the command-line (using the instructions on the course page), or via Anaconda.

Elements in a notebook are divided into cells, which might be markdown (such as this cell), which contains text, or code. The cell below contains code. You can execute a cell by clicking on the cell and typing "Shift + Enter". Try your first "Hello World" Python program below.

```
In [79]: "Hello world"
Out[79]: 'Hello world'
```

You can also define and call functions in cells. In the cell below, write a simple function that takes a string as an argument and prints that string as an output.

```
In [2]: print("Mayarak Quintero")
        Mayarak Quintero
```

2. Basic Operations in Python: Loading Data

You should famliarize yourself with loading data into a Python notebook in various formats. Some of the notebooks we have provided have some examples for loading files.

Much of the data we will use is in formats such as comma-separated value (CSV), but you may also find the need to load data that is in other formats (e.g., JSON, SQL databases). The Python Pandas library provides easy ways to load these types of files into Pandas "data frames".

2.1 Loading a CSV File into Pandas

Load the Divvy Trip data (https://data.cityofchicago.org/Transportation/Divvy-Trips/fg6s-gzvg) from the City of Chicago data portal into a Pandas data frame.

Note: The file is large (5 GB), and so this will possibly take a fair bit of time to download/load. Maybe 5-10 minutes. Be patient! If you have problems, the course staff has a smaller, truncated version of the file to work with; let us know.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

In [2]:
```
df = pd.read_csv('/Users/mayar/Downloads/Divvy_Trips.csv')
```

In [3]:
```
df.drop(columns=['FROM STATION ID', 'FROM STATION NAME', 'TO STATION ID', 'TO
 STATION NAME', 'FROM LATITUDE','FROM LONGITUDE', 'FROM LOCATION','TO LATITUD
E', 'TO LONGITUDE', 'TO LOCATION'], inplace = True)
```

In [4]:
```
df[['DATE','HOUR']] = df['START TIME'].str.split(" ",n=1,expand=True)
```

In [8]:
```
df.head(5)
```

Out[8]:

	TRIP ID	START TIME	STOP TIME	BIKE ID	TRIP DURATION	USER TYPE	GENDER	BIRTH YEAR	DATE	
0	8546790	12/31/2015 05:35:00 PM	12/31/2015 05:44:00 PM	979	521	Subscriber	Female	1991.0	12/31/2015	(
1	8546793	12/31/2015 05:37:00 PM	12/31/2015 05:41:00 PM	1932	256	Subscriber	Male	1992.0	12/31/2015	(
2	8546795	12/31/2015 05:37:00 PM	12/31/2015 05:40:00 PM	1693	134	Subscriber	Female	1987.0	12/31/2015	(
3	8546797	12/31/2015 05:38:00 PM	12/31/2015 05:55:00 PM	3370	995	Subscriber	Male	1975.0	12/31/2015	(
4	8546798	12/31/2015 05:38:00 PM	12/31/2015 05:41:00 PM	2563	177	Subscriber	Male	1990.0	12/31/2015	(

In [5]:
```
def lookup(s):
    dates = {date:pd.to_datetime(date) for date in s.unique()}
    return s.map(dates)
```

In [6]:
```
df['DATE'] = lookup(df['DATE'])
```

In [7]:
```
df['WEEK DAY'] = [d.day_name() for d in df['DATE']]
```

2.2 Basic Data Analysis in Pandas

Now that you have the data loaded into a basic dataframe, you can ask some questions about the data, using Pandas. Each of these questions is intended to give you practice manipulating and analyzing a data frame.

2.2.1 What is the number of rows in the data frame?

This question is intended to help you understand one of the most basic questions about your data: How many data points does it have?

Call a single Pandas dataframe function to figure out how large this dataset is. Your answer should produce a single integer.

In [12]:
```
num_rows = df.shape[0]
print(f"{num_rows:,d}")
```

```
20,538,686
```

2.2.2 What are the start and end dates of the rides in the data set?

It is typically important to understand basic information about the data, such as when it starts and ends.

This is also an example of *looking for outliers*. The Divvy program started in Chicago somewhat recently (find out when!) and so if the earliest ride in the dataset predates that, you know you the dataset has a problem! Performing these kinds of basic sanity checks on the data is critical and something you should always be doing.

In [14]:
```
min(df['DATE']) #On June 28, 2013, Divvy launched with 750 bikes at 75 station
s in Chicago
```

Out[14]:
```
Timestamp('2013-06-27 00:00:00')
```

In [15]:
```
max(df['DATE'])
```

Out[15]:
```
Timestamp('2019-09-30 00:00:00')
```

2.2.3 What is the mean duration of all trips?

This question is intended to give you exposure to (1) selecting a column from a Pandas dataframe; (2) applying an aggregate function (e.g., a mean) to a column of the dataframe. Of course, it is possible to apply other aggregate functions, and you should familiarize yourself with those.

In [16]:
```
min_secs = min(df['TRIP DURATION']) # The minimum duration of a trip is 60 sec
onds which can make sense if a user took a
    # bike and decided to return it right after.
print("Minimum number of trip seconds:", min_secs)
```

```
Minimum number of trip seconds: 60
```

In [17]:
```
max_hrs = max(df['TRIP DURATION'])/3600
my_string = '{:,.2f}'.format(max_hrs)
print("Maximum number of trip hours:", my_string) #The maximum duration of a t
rip is 13,453,200 seconds or 3,737 hours.
# This maximum trip duration number is an outlier in the data
```

```
Maximum number of trip hours: 3,737.00
```

In [88]:
```
df.dtypes
```

Out[88]:
```
TRIP ID                    int64
START TIME        datetime64[ns]
STOP TIME                 object
BIKE ID                    int64
TRIP DURATION              int64
USER TYPE                 object
GENDER                    object
BIRTH YEAR               float64
NEW DATE                  object
NEW TIME                  object
WEEK DAY                  object
dtype: object
```

In [8]:
```
df = df[df['TRIP DURATION'] <= 86400] ## I kept only the observations below th
is threshold because it is the equivalent
# bike for 24 hours.
```

In [19]:
```
mean = df['TRIP DURATION'].mean()
my_string_mean = '{:,.2f}'.format(mean)
print("The mean duration of trips is:", my_string_mean)
```

```
The mean duration of trips is: 1,052.13
```

2.2.4 Do men or women take longer trips on average?

The goal of this question is to give you experience with the groupby function in Pandas, as well as how to combine groupby with an aggregation operation. There are a couple of ways to answer this question, actually; you could also do it with a conditional select. Try it both ways!

Using Groupby

In [22]:
```
df.groupby(['GENDER']).mean().round(1)
```

Out[22]:

GENDER	TRIP ID	BIKE ID	TRIP DURATION	BIRTH YEAR
Female	14007588.5	2928.6	894.1	1982.9
Male	13708027.0	2964.1	736.5	1980.8

Using Conditional Selection

In [23]:
```
women = df[df['GENDER'] == 'Female']
men = df[df['GENDER'] == 'Male']
```

In [24]:
```
women_mean = women['TRIP DURATION'].mean()
print('The mean duration of women trips is:', women_mean)
```

```
The mean duration of women trips is: 894.075387865949
```

In [25]:
```
men_mean = men['TRIP DURATION'].mean()
print('The mean duration of men trips is:', men_mean)
```

```
The mean duration of men trips is: 736.5033995643389
```

2.2.5 Sanity Checks

We just performed the above operations without checking how many rides were taken by males and females. Quickly do that below. Also compute the sum of male and female rides.

In [26]:
```
df['GENDER'].value_counts()
```

Out[26]:
```
Male      11748123
Female     3960969
Name: GENDER, dtype: int64
```

In [27]:
```
total_gender = df['GENDER'].value_counts().sum()
print(f"{total_gender:,d}")
```

```
15,709,092
```

In [30]:
```
gender_null = df['GENDER'].isnull().sum()
print(f"{gender_null:,d}")
```

```
4,826,742
```

Compare the number you just computed to the total number of data points. Do they match? Why or why not?

In [ ]:
```
# The sum of male and female rides (15,709,092) does not match with the total
 number of data points which is (20,535,834).
# That is because 'GENDER' has 4,826,742 null values.
```

2.2.6 Other checks

We know anecdotally that the birth year column (`BIRTH YEAR`) has several missing values. How many rows exactly are missing a birth year?

In [31]:
```
birth_null = df['BIRTH YEAR'].isnull().sum()
print(f"{birth_null:,d}")
```

```
4,803,517
```

What proportion of rows in the dataset have a missing birth year?

```
In [32]: (df['BIRTH YEAR'].isnull().sum() / df.shape[0]).round(4)
```

```
Out[32]: 0.2339
```

3. Basic Plotting and Visualization

The goal of this part of the assignment is to give you basic experience with plotting data from datasets. You can use the same types of operations as were demonstrated in lecture to perform this part of the lab.

3.1 Plotting Ride Volumes Over Time

Below will provide some experience with plotting rides over time.

3.1.1 Setting an index in the data frame

Recall the first steps include importing plotting libraries and setting one of the columns in the data frame to be the index.

```
In [16]: df['TRIPS DAY'] = 1
```

```
In [39]: df1 = df.set_index('DATE')
```

3.1.2 Plotting the total trip duration by day

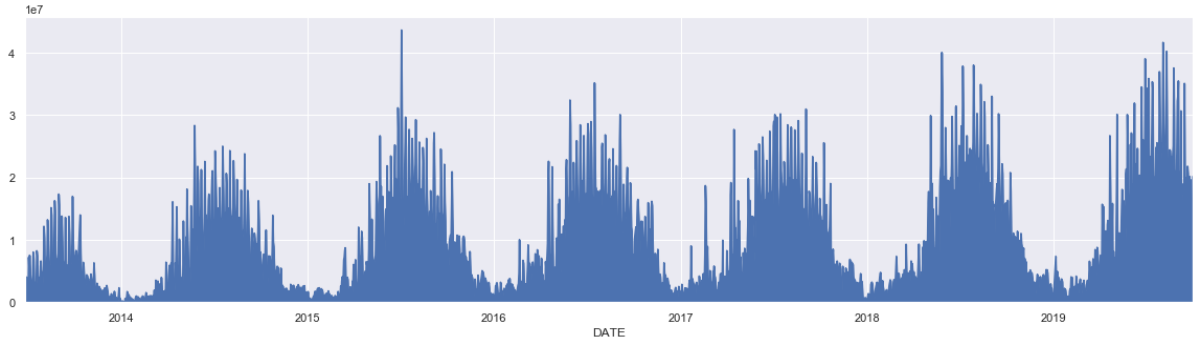
Plot the total trip duration by day. While there are a number of ways to perform this operation, you may find the `resample` function in Pandas useful.

```
In [40]: df1 = df1.resample('1D').sum()
```

In [45]:
```
%matplotlib inline


sns.set(rc={'figure.figsize':(20, 5)})
df1['TRIP DURATION'].plot.area()
```

Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26962771808>
```

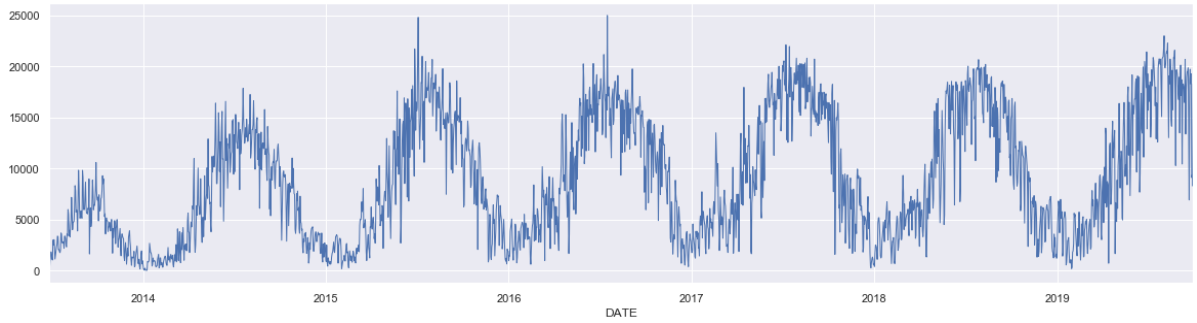


3.1.3 Plot the total number of trips by day

This calcuation is a little bit trickier, since there's no number in the data frame to plot: each row is what you're trying to count. You may find `resample` useful, but you may have to add some data to the data frame to make it work. group by and count may also work.

In [30]:
```
df1['TRIPS DAY'].plot(linewidth=1)
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26963f73a48>
```



3.2 Data Exploration on Your Own

Pick a question or hypothesis, justify **why** you picked that question (i.e., why it might be an interesting question to some audience, such as city officials), and present a simple analysis.

Optionally, you may pick a related dataset from the Chicago data portal and join it onto the Divvy trips data. As above, be sure to justify your choice and explain what analysis you can now perform (as well as the analysis itself).

Some example questions might include:

- Adjusting for seasons, is ridership increasing? (You could use conditional selection on dates or months.)
- Are rides getting longer? (on average? max?)
- Do ride characteristics differ by user type?
- Are certain trip routes (e.g. pairs of start and end stations) more popular than others? Does this change during peak and non-peak "rush" hours (defined loosely)?
- Which neighborhoods are seeing the most ridership? (More difficult! Requires spatial analysis!)

Exploration on my own

The question I want to answer is, adjusting for weekdays, "Does ridership increase during business days and hours?". This preliminary question could be interesting to A) people who are responsible for public policy related to urban development and less polluting transport. B) Divvy, if the company want to design new strategies to capture more or different clients. The assumption here is that if people are using bikes more during the week than during weekends, as well as more during business hours, that would mean that people are using Divvy bikes more as the transport to daily activities (e.g. commute to school) than for recreational purposes.

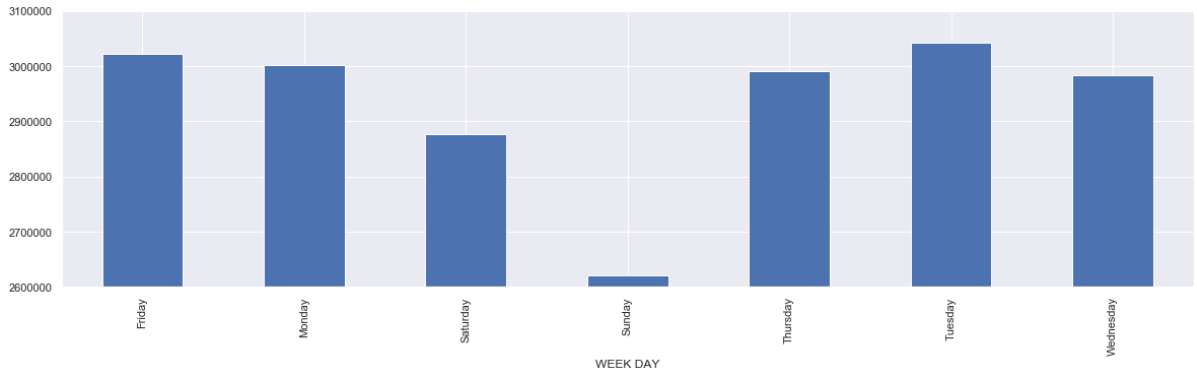
Trips per week day

In the next figure, we can observe that the day with less trips is Sunday, followed by Saturday. That means that people is using Divvy bikes more during the business days than during weekends.

In [48]:
```
df2 = df.set_index('WEEK DAY')
```

In [78]:
```
df2['TRIPS DAY'].groupby('WEEK DAY').count().plot.bar(ylim = (2600000,3100000
))
```

Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x26962498748>



Trips per hour

In the last figure, we can observe that the hours with more trips are between 04:00 - 08:00 pm. Interestingly, people don't use Divvy bikes during busy hours in the morning. That could indicate that the people are just using Divvy bikes for one part of the commute the do every day. Less use of bikes between 08:00 am - 12:00 pm could be explained by different factors such as faster transport options.

In [ ]:
```
df['HOUR'] = lookup(df['HOUR'])
```

In [17]:
```
df3 = df.set_index('HOUR')
```

In [18]:
```
df3 = df3.resample('4H').sum()
```

In [22]:
```
%matplotlib inline


sns.set(rc={'figure.figsize':(20, 5)})
df3['TRIPS DAY'].plot.bar()
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x28fe41e3fc8>