Loading and Importing

First thing you need to do is load up your packages and then bring in the data.

This dataset contains daily values for Amazon's stock. This includes opening, closing, high price, low price, and also the amount of stock traded.

```
[169] import pandas as pd
      # also import matplotlib.pyplot and numpy with the proper aliases
      import numpy as np
      import matplotlib.pyplot as plt
```

```
[170] # Bring in your data. You just need to run this cell.
      price = pd.read_csv("https://docs.google.com/spreadsheets/d/1lCkFZhz-NGTuE1ZilzJA_ZYBZsbCSDpS2MllyGZWjX4/gviz/tq?tqx=out:csv")
```

Exploring the whole dataset

Now make some code cells to explore the whole dataset. I want you to do the following:

- Get the number of rows and columns
- Get the datatypes of each column
- Look at the first five rows
- Look at the last five rows
- Look at summary statistics

```
[171] # Obtaining the number of rows and columns
      price.shape
```

```
(5852, 7)
```

```
[172] # Obtaining the datatypes of every column
      price.dtypes
```

```
[172]
```

	0
Date	object
Open	float64
High	float64
Low	float64
Close	float64
Adj Close	float64
Volume	int64

dtype: object

```
# First 5 rows
price.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1997-05-15	2.437500	2.500000	1.927083	1.958333	1.958333	72156000
1	1997-05-16	1.968750	1.979167	1.708333	1.729167	1.729167	14700000
2	1997-05-19	1.760417	1.770833	1.625000	1.708333	1.708333	6106800
3	1997-05-20	1.729167	1.750000	1.635417	1.635417	1.635417	5467200
4	1997-05-21	1.635417	1.645833	1.375000	1.427083	1.427083	18853200

Next steps: Generate code with price | View recommended plots | New interactive sheet

```
[174] # Last 5 rows
      price.tail()
```

	Date	Open	High	Low	Close	Adj Close	Volume
5847	2020-08-10	3170.310059	3172.510010	3101.520020	3148.159912	3148.159912	3167300
5848	2020-08-11	3113.199951	3159.219971	3073.000000	3080.669922	3080.669922	3718100
5849	2020-08-12	3108.000000	3174.389893	3101.419922	3162.239990	3162.239990	3527200
5850	2020-08-13	3182.989990	3217.520020	3155.000000	3161.020020	3161.020020	3149000
5851	2020-08-14	3178.179932	3178.239990	3120.000000	3148.020020	3148.020020	2749200

```
# Summary
price.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	5852.000000	5852.000000	5852.000000	5852.000000	5852.000000	5.852000e+03
mean	377.469557	381.737827	372.815253	377.500736	377.500736	7.512673e+06
std	596.291033	602.615299	589.226694	596.287464	596.287464	7.278144e+06
min	1.406250	1.447917	1.312500	1.395833	1.395833	4.872000e+05
25%	37.995000	38.590000	37.247501	37.950001	37.950001	3.681975e+06
50%	83.699997	85.029999	82.055000	83.602501	83.602501	5.652850e+06
75%	360.852501	365.794991	357.877510	362.217506	362.217506	8.517450e+06
max	3251.060059	3344.290039	3165.429932	3225.000000	3225.000000	1.043292e+08

Questions

Write down Pandas instruction to answer these queries.

- How many rows are in this dataset?
- Do any datatypes need to be converted?
- What was the mean and all time high opening stock price?

Q1: [2 points] How many rows are in this dataset?

```
[176] #do not change the function name or number of parameters
      ## Q1 Your function starts here
      def price_number_of_row():
        return 0 ##replace this line with the return of the correct statement
      ## Q1 Your function ends here - Any code outside of these start/end markers won't be graded
```

As an example, I will provide the answer to this question:

```
[177] def price_number_of_row_key():
        return price.shape[0];
```

```
print(price_number_of_row_key())
```

```
5852
```

Question 2: [2 points] Convert the datatype of 'Date' to an appropriate type

```
[179] ## Q2 Your function starts here
      def convert_date_type():
        price['Date'] = pd.to_datetime(price['Date']) #replace this line with the correct state. You don't need to return
      ## Q2 Your function ends here - Any code outside of these start/end markers won't be graded
```

I give you the grader statement for this one too:

```
[180] convert_date_type();
      print(price.Date.dtype);
```

```
datetime64[ns]
```

Question 3: [2 points] What is the mean of all opening stock price?

```
[181] ## Q3 Your function starts here
      def mean_of_opening():
        return price.describe().loc['mean']['Open']; #replace this with the correct statement
      ## Q3 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
print(mean_of_opening())
```

```
377.4695570299043
```

Question 4: [4 points] What is the daily min volume for trades after 2010 (including 2010-1-1)

```
[183] ## Q4 Your function starts here
      def min_daily_volume_after_2010():
        price['Date'] = pd.to_datetime(price['Date'])
        i = price[price['Date'] >= '2010-01-01']
        min_volume = i['Volume'].min()

        return min_volume
      ## Q4 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
[184] print(min_daily_volume_after_2010())
```

```
881300
```

Question 5: [4 points] Make a new column called up_binom. The value of up_binom is 1 if the closing price of the stock is higher (>=) than opening price, and 0 otherwise. Then find the number days the stock closed higher than openning.

```
[185] ## Q5 Your function starts here
      def number_of_green_days():
        price['up_binom'] = np.where(price['Close'] >= price['Open'], 1, 0)
        ct = 0
        for i in price['up_binom']:
          if i == 1:
            ct += 1
        return(ct)
      ## Q5 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
[186] print(number_of_green_days());
```

```
2964
```

Question 6 [4 points] Write a function that plots the Closing price of the stock over the entire timeline of the dataset. The x-axis is day and the y-axis is the Closing price. Look at the examples here for more help https://matplotlib.org/stable/gallery/index.html)

```
#plot
## Q6 Your function starts here
def plot_closing_price_after_2020():
  #Add your code for your plot
  plt.plot(price['Date'], price['Close'], color = "purple")
  plt.title("Closing Price of Amazon")
  plt.xlabel("Date")
  plt.ylabel("Closing Price")
  plt.show()

# function call to
plot_closing_price_after_2020()
## Q6 Your function ends here - Any code outside of these start/end markers won't be graded
```

Question 7: [4 points] Word problem time!

When I was 18 I had saved up $2000 washing dishes at a local restaurant to buy a new gaming rig (a cutting edge Pentium 5 :D). This was actually right around the start of this dataset. Let's say instead of buying that computer, I had put all that money into Amazon stock, what would my total profit have been if I'd sold all of my shares on the last day of the dataset?

Let's say that I bought the shares at opening on Aug 1, 1997 and sold them at close on the last day of the dataset (don't assume that the dataset is sorted by date).

Keep in mind that these are real world data and, as such, have constraints that you need to consider: e.g., you can only buy full shares and not fractions of shares. Also keep in mind that the question is about total **profit**, which should account for the money I spent on the shares. To keep things simple, your answer can (and should for ease of grading) include fractions of cents even though that wouldn't really be possible.

Write a function ***market_value(buy_date, sell_date, starting_cash)*** that calculates the total profit from a number of shares bought at open of one date and sold at close of another date and use it to solve the word problem

```
[188] # I'm going to make your life easier and set the date column you created as the index.
      # This will make searching and extracting the values much easier
      price = price.set_index('Date')
```

```
[189] ## Q7 Your function starts here

      def market_value(buy_date, sell_date, starting_cash):
          global price
          price = price.reset_index()
          price['Date'] = pd.to_datetime(price['Date'])
          price = price.sort_values('Date')
          price = price.set_index('Date')

          buy = price.loc[buy_date]
          sell = price.loc[sell_date]

          shares_bought = starting_cash // buy['Open']

          total_profit = (shares_bought * sell['Close']) - (shares_bought * buy['Open'])

          return total_profit, shares_bought, buy['Open'], sell['Close']

      ## Q7 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
      last_date_in_dataset = price.index.max().strftime('%Y-%m-%d') # replace with code that finds the last date in the dataset
      print(market_value('1997-08-01', last_date_in_dataset, 2000))
```

```
(2683261.85831, 853.0, 2.34375, 3148.02002)
```

JSON

The last part of the assignment will have you working with some basic JSON data. The URL links to a JSON file with stats on every episode of the TV show Silicon Valley

```
[191] # First just run this to import the data
      import requests
      url = 'http://api.tvmaze.com/singlesearch/shows?q=Silicon Valley&embed=episodes'
      sv_json_obj = requests.get(url)
      sv_json = sv_json_obj.json()
```

Viewing your JSON

Now just to look at what's in the JSON a bit

- Make a code cell that just calls the JSON we named above.
- Also run the .keys() function on the object.

```
      # Calling JSON
      sv_json
```


```
{'id': 143,
 'url': 'https://www.tvmaze.com/shows/143/silicon-valley',
 'name': 'Silicon Valley',
 'type': 'Scripted',
 'language': 'English',
 'genres': ['Comedy'],
 'status': 'Ended',
 'runtime': 30,
 'averageRuntime': 30,
 'premiered': '2014-04-06',
 'ended': '2019-12-08',
 'officialSite': 'http://www.hbo.com/silicon-valley/',
 'schedule': {'time': '22:00', 'days': ['Sunday']},
 'rating': {'average': 8.4},
 'weight': 93,
 'network': {'id': 8,
  'name': 'HBO',
  'country': {'name': 'United States',
   'code': 'US',
   'timezone': 'America/New_York'},
  'officialSite': 'https://www.hbo.com/'},
 'webChannel': None,
 'dvdCountry': None,
 'externals': {'tvrage': 33759, 'thetvdb': 277165, 'imdb': 'tt2575988'},
 'image': {'medium': 'https://static.tvmaze.com/uploads/images/medium_portrait/215/538434.jpg',
  'original': 'https://static.tvmaze.com/uploads/images/original_untouched/215/538434.jpg'},
 'summary': '<p>In the high-tech gold rush of modern Silicon Valley, the people most qualified to succeed are the least capable of handling success. From Mike Judge comes this satire about a
programmer whose game-changing algorithm becomes the subject of a valley-wide bidding war.</p>',
 'updated': 1704794037,
 '_links': {'self': {'href': 'https://api.tvmaze.com/shows/143'},
  'previousepisode': {'href': 'https://api.tvmaze.com/episodes/1757884',
   'name': 'Exit Event'}},
 '_embedded': {'episodes': [{'id': 10897,
    'url': 'https://www.tvmaze.com/episodes/10897/silicon-valley-1x01-minimum-viable-product',
    'name': 'Minimum Viable Product',
    'season': 1,
```

```
[193] # Running the .keys() function
      sv_json.keys()
```

```
dict_keys(['id', 'url', 'name', 'type', 'language', 'genres', 'status', 'runtime', 'averageRuntime', 'premiered', 'ended', 'officialSite', 'schedule', 'rating', 'weight', 'network',
 'webChannel', 'dvdCountry', 'externals', 'image', 'summary', 'updated', '_links', '_embedded'])
```

Questions

Based on these responses, what keys are present in the JSON. More importantly, are there any keys that don't get returned by .keys()?

Keys that are present:

- ['id', 'url', 'name', 'type', 'language', 'genres', 'status', 'runtime', 'averageRuntime', 'premiered', 'ended', 'officialSite', 'schedule', 'rating', 'weight', 'network', 'webChannel', 'dvdCountry', 'externals', 'image', 'summary', 'updated', '_links', '_embedded']

Keys that don't get returned:

- Season
- Number
- Airdate
- Airtime
- Airstamp
- Show

```
[194] # First, you can see the structure after moving down a level into '_embedded'
      sv_json['_embedded']
```


```
    'number': 8,
    'type': 'regular',
    'airdate': '2018-05-13',
    'airtime': '22:15',
    'airstamp': '2018-05-14T02:15:00+00:00',
    'runtime': 30,
    'rating': {'average': 6.9},
    'image': {'medium': 'https://static.tvmaze.com/uploads/images/medium_landscape/155/389554.jpg',
     'original': 'https://static.tvmaze.com/uploads/images/original_untouched/155/389554.jpg'},
    'summary': "<p>The launch of PiperNet finds Monica suspicious of an early success, and the team must race against the clock as their future is threatened. Realizing he's made more enemies
then friends, Richard makes a surprising move.</p>",
    '_links': {'self': {'href': 'https://api.tvmaze.com/episodes/1445227'},
     'show': {'href': 'https://api.tvmaze.com/shows/143',
      'name': 'Silicon Valley'}}},
  {'id': 1698000,
   'url': 'https://www.tvmaze.com/episodes/1698000/silicon-valley-6x01-artificial-lack-of-intelligence',
   'name': 'Artificial Lack of Intelligence',
   'season': 6,
   'number': 1,
   'type': 'regular',
   'airdate': '2019-10-27',
   'airtime': '22:00',
   'airstamp': '2019-10-28T02:00:00+00:00',
   'runtime': 30,
   'rating': {'average': 6.5},
   'image': {'medium': 'https://static.tvmaze.com/uploads/images/medium_landscape/217/543706.jpg',
    'original': 'https://static.tvmaze.com/uploads/images/original_untouched/217/543706.jpg'},
   'summary': "<p>Richard discovers his promise to keep Pied Piper free from collecting user data is under threat. Jared finds himself missing his role as Richard's go-to guy and revisits the
hacker hostel. Gilfoyle devises a creative way to deal with Dinesh's complaining.</p>",
   '_links': {'self': {'href': 'https://api.tvmaze.com/episodes/1698000'},
    'show': {'href': 'https://api.tvmaze.com/shows/143',
     'name': 'Silicon Valley'}}},
  {'id': 1730750,
   'url': 'https://www.tvmaze.com/episodes/1730750/silicon-valley-6x02-blood-money',
   'name': 'Blood Money',
   'season': 6,
   'number': 2,
   'type': 'regular',
   'airdate': '2019-11-03',
   'airtime': '22:00',
   'airstamp': '2019-11-04T03:00:00+00:00',
   'runtime': 30,
   'rating': {'average': 6.6},
   'image': {'medium': 'https://static.tvmaze.com/uploads/images/medium_landscape/218/545800.jpg',
    'original': 'https://static.tvmaze.com/uploads/images/original_untouched/218/545800.jpg'},
   'summary': '<p>When Richard meets an interested investor with a sketchy past, he must weigh his integrity against a tempting financial offer. Jared seeks a new future, putting him at odds
with Richard. A defiant Gilfoyle butts heads with HR over his lack of direct reports. Hoover and Denpok worry when Gavin explores alternative solutions to keep a leaner Hooli afloat.</p>',
   '_links': {'self': {'href': 'https://api.tvmaze.com/episodes/1730750'},
    'show': {'href': 'https://api.tvmaze.com/shows/143',
     'name': 'Silicon Valley'}}},
  {'id': 1730751
```


Question 8: [4 points] Find the day that show premiered (ie the date that the very first episode first aired).

```
[195] ## Q8 Your function starts here
      def get_show_premiered():
        return sv_json['_embedded']['episodes'][0]['airdate']
      ## Q8 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
[196] print(get_show_premiered());
```

```
2014-04-06
```

Question 9: [4 points] Get the summary of a specific episode.

```
[197] ## Q9 Your function starts here
      def get_summary(season, episode):
        for i in range(len(sv_json['_embedded']['episodes'])):
          if sv_json['_embedded']['episodes'][i]['season'] == season and sv_json['_embedded']['episodes'][i]['number'] == episode:
            return sv_json['_embedded']['episodes'][i]['summary']
      ## Q9 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
[198] print(get_summary(2,5))
```

```
<p>Gavin creates interference that hinders Pied Piper's expansion. Meanwhile, the guys could be threatened by a nosy neighbor; Guilfoyle sets out to build servers; Richard's reluctant to let
```