Hands-on Activity 8.1: Aggregating Data with Pandas

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
# Im following the Modules and yet I receive future warnings so I added this...
```

8.1.1 Intended Learning Outcomes

After this activity, the student should be able to:

- Demonstrate querying and merging of dataframes
- Perform advanced calculations on dataframes
- Aggregate dataframes with pandas and numpy
- Work with time series data

8.1.2 Resources

- Computing Environment using Python 3.x
- Attached Datasets (under Instructional Materials)

8.1.3 Procedures

The procedures can be found in the canvas module. Check the following under topics:

- 8.1 Weather Data Collection
- 8.2 Querying and Merging
- 8.3 Dataframe Operations
- 8.4 Aggregations
- 8.5 Time Series

8.1.4 Data Analysis

INSIGHTS AND COMMENTS IN MODULE 1:

This Module is the same as the first activity of the last moddule, when we are collecting data form an API (Appicatin Programming Interface). This API or the 'National Oceanic and Atmospheric Administration' NCDC API is the one that being used to this modue.

First I've learned to understand and read Documentation of a certain API or the items provided to give you knowledge about the software you want to use

- if i don't know or understand what to do, I refer first to the documentation before going to Internet or an AI.
- think of it as a manual even though some documentation has some overwhelming infoormation to throw at you.

INSIGHTS AND COMMENTS IN MODULE 2

Querying in Pandas/Dataframes are easy and easy to remember because of it's similarity in SQL.

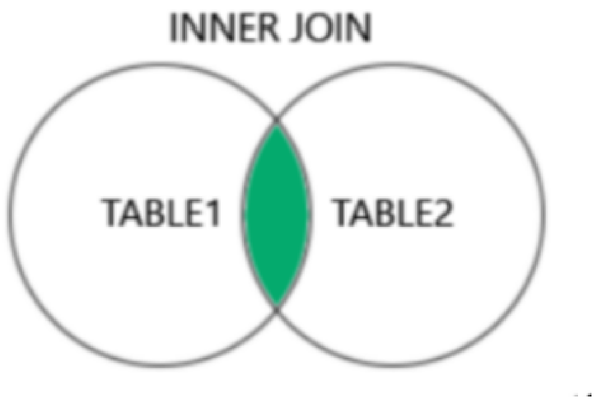
I've Learned that there are two ways to et the query you like/desire:

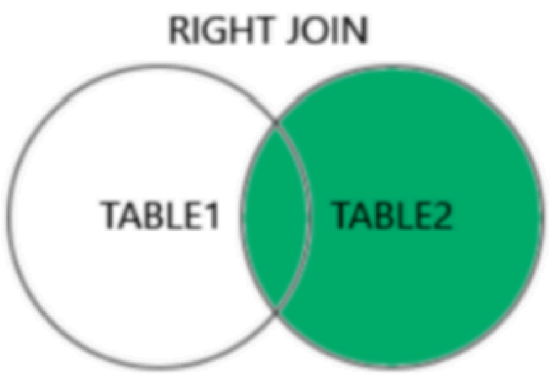
1. The Easy DataFrame.query():
    - The query accepts a SQL script as a Parameter.
    - `snow_data = weather.query('datatype == "SNOW" and value > 0')`
2. the bitwise Opt Cracker, The Booolean Mask :
    - uses bitwise to get what he/she wanted/desired in the query

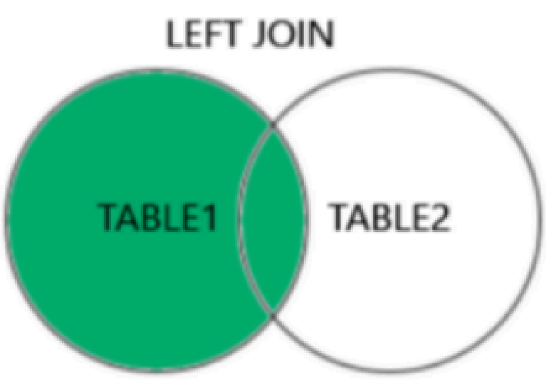
```
weather[(weather.datatype == 'SNOW') & (weather.value > 0)].equals(snow_data)
```

Merging folllows the JOIN in SQL

- using DataFrame.merge() we can perform Joins such as left, right, inner joins and many more. (Inner Join as the default)

- 



- 

INSIGHTS AND COMMENTS FOR MODULE 3

This module consists of the basic Operations you can do to your dataframes. The most memorable is these:

pct_change()

- lets you change the current number of a column to its percentage equivalent.

pd.cut(), pd.qcut()

- let youu create seperate bins in order for you to identify the rages/roupings of your data

.apply()

- lets you apply Functions in certain dataframe or specific coluumn for an easy configurations

Window calculations

- rolinng calculations can be done using this <-

INSIGHTS AND COMMENTS FOR MODULE 4

this Aggeration of dataframe is simply formating the data for future use. can be done uing the agg() function.

but this module alo includes:

group_by()

- aggreating the groups of a dataframe not the entire dataframe

pivot_tables

- Provides the simmplest form of aggregation

crosstabs

- in which you get the frequency of a specific column

INSIGHTS AND COMMENTS IN MODDULE 5

- basically this module talks about time. Yo can calulate time wiht TimeDelta() andd you can select data from simply inputting the dates/range of dates of the desired data.
- You can also resample your data by the DateTime Index, Period Index, and timedeltaIndex

8.1.5 Supplementary Activity

Using the CSV files provided and what we have learned so far in this module complete the following exercises

```
import pandas as pd
import numpy as np

earthquake_df = pd.DataFrame(pd.read_csv("/content/earthquakes.csv")) # reading and making a dtafr
faang_df = pd.DataFrame(pd.read_csv("/content/faang.csv", parse_dates=['date'], index_col = ['date
```

```
earthquake_df.head(10) # earthquake dataframe
```

	mag	magType	time	place	tsunami	parsed_place
0	1.35	ml	1539475168010	9km NE of Aguanga, CA	0	California
1	1.29	ml	1539475129610	9km NE of Aguanga, CA	0	California
2	3.42	ml	1539475062610	8km NE of Aguanga, CA	0	California
3	0.44	ml	1539474978070	9km NE of Aguanga, CA	0	California
4	2.16	md	1539474716050	10km NW of Avenal, CA	0	California
5	2.61	md	1539473686440	55km ESE of Punta Cana, Dominican Republic	0	Dominican Republic
6	1.70	ml	1539473176017	105km W of Talkeetna, Alaska	0	Alaska
7	1.13	md	1539473060280	10km NW of Parkfield, CA	0	California
8	0.92	md	1539473042310	6km NW of The Geysers, CA	0	California

Next steps: View recommended plots

```
faang_df.head(10) # faang dataframe
```

	ticker	open	high	low	close	volume
date						
2018-01-02	FB	177.68	181.58	177.5500	181.42	18151903
2018-01-03	FB	181.88	184.78	181.3300	184.67	16886563
2018-01-04	FB	184.90	186.21	184.0996	184.33	13880896
2018-01-05	FB	185.59	186.90	184.9300	186.85	13574535
2018-01-08	FB	187.20	188.90	186.3300	188.28	17994726
2018-01-09	FB	188.70	188.80	187.1000	187.87	12393057
2018-01-10	FB	186.94	187.89	185.6300	187.84	10529894
2018-01-11	FB	188.40	188.40	187.3800	187.77	9588587
2018-01-12	FB	178.06	181.48	177.4000	179.37	77551299
2018-01-16	FB	181.50	181.75	178.0400	178.39	36183842

Next steps: View recommended plots

1. With the earthquakes.csv file, select all the earthquakes in Japan with a magType of mb and a magnitude of 4.9 or greater.

```
earthquake_df.query('magType == "mb" and mag > 4.9 and parsed_place == "Japan"') # Simple method
```

	mag	magType	time	place	tsunami	parsed_place
2576	5.4	mb	1538697528010	37km E of Tomakomai, Japan	0	Japan

```
earthquake_df[
    (earthquake_df['magType'] == 'mb') & (earthquake_df['mag'] > 4.9) & (earthquake_df['parsed_pla
    ] # Boolean Mask method
```

	mag	magType	time	place	tsunami	parsed_place
2576	5.4	mb	1538697528010	37km E of Tomakomai, Japan	0	Japan

2. Create bins for each full number of magnitude (for example, the first bin is 0-1, the second is 1-2, and so on) with a magType of ml and count how many are in each bin.

```
earthquake_magBin = pd.cut(earthquake_df.query('magType == "ml"').mag,
                           bins=6,
                           labels=['0-1', '1-2', '2-3', '3-4', '4-5', '5-6'])
earthquake_magBin.value_counts() # Creating bins
```

```
2-3    3436
1-2    1889
3-4    1027
0-1     288
4-5     160
5-6       3
Name: mag, dtype: int64
```

3. Using the faang.csv file, group by the ticker and resample to monthly frequency. Make the following aggregations

- Mean of the opening price
- Maximum of the high price
- Minimum of the low price
- Mean of the closing price
- Sum of the volume traded

```
aggFang = faang_df.groupby("ticker").agg( # agg() does the job for aggregation
    {
        'open': np.mean,  # Self explainatory things is descriptive statictics
        'high': np.max,
        'low': np.min,
        'close': np.mean,
        'volume': np.sum
    }
)
aggFang
```

	open	high	low	close	volume
ticker					
AAPL	187.038674	231.6645	145.9639	186.986218	8539383858
AMZN	1644.072669	2050.5000	1170.5100	1641.726175	1418040266
FB	171.454424	218.6200	123.0200	171.510936	6949682394
GOOG	1113.554104	1273.8900	970.1100	1113.225139	437403914
NFLX	319.620533	423.2056	195.4200	319.290299	2879045091

Next steps: View recommended plots

4. Build a crosstab with the earthquake data between the tsunami column and the magType column. Rather than showing the frequency count, show the maximum magnitude that was observed for each combination. Put the magType along the columns.

```
pd.crosstab(
    index=earthquake_df.tsunami,
    columns=earthquake_df.magType,
    values=earthquake_df.mag, # As said, instead of the frequency, we get the Magnitude of the com
    aggfunc=lambda x:np.max(x)
).fillna(0) # since there are Empty shells, it is better to fill it up
```

magType	mb	mb_lg	md	mh	ml	ms_20	mw	mwb	mwr	mww
tsunami										
0	5.6	3.5	4.11	1.1	4.2	0.0	3.83	5.8	4.8	6.0
1	6.1	0.0	0.00	0.0	5.1	5.7	4.41	0.0	0.0	7.5

4. Calculate the rolling 60-day aggregations of OHLC data by ticker for the FAANG data. Use the same aggregations as exercise no. 3.

```
faang_df.groupby(['ticker']).rolling('60D').agg( # Grouuping them by the Ticker and setting Rollin
    {
        'open': np.mean,
        'high': np.max,
        'low': np.min,
        'close': np.mean,
        'volume': np.sum
    }

).join(aggFang[['open', # Join the past aggregation for a comparison of the data
        'high',
        'low',
        'close',
        'volume']], lsuffix="_rolling").sort_index(axis=1) # adding suffixes to prevent errors
```

ticker	date	close	close_rolling	high	high_rolling	low	low_rollin
AAPL	2018-01-02	186.986218	168.987200	231.6645	169.0264	145.9639	166.044
	2018-01-03	186.986218	168.972500	231.6645	171.2337	145.9639	166.044
	2018-01-04	186.986218	169.229200	231.6645	171.2337	145.9639	166.044
	2018-01-05	186.986218	169.840675	231.6645	172.0381	145.9639	166.044
	2018-01-08	186.986218	170.080040	231.6645	172.2736	145.9639	166.044
...	...	...	...	...	...	...	
NFLX	2018-12-24	319.290299	281.931750	423.2056	332.0499	195.4200	233.680
	2018-12-26	319.290299	280.777750	423.2056	332.0499	195.4200	231.230
	2018-12-27	319.290299	280.162805	423.2056	332.0499	195.4200	231.230
	2018-12-28	319.290299	279.461341	423.2056	332.0499	195.4200	231.230
	2018-12-31	319.290299	277.451410	423.2056	332.0499	195.4200	231.230

1255 rows × 10 columns

6. Create a pivot table of the FAANG data that compares the stocks. Put the ticker in the rows and show the averages of the OHLC and volume traded data.

```
faang_df.pivot_table(
  index = 'ticker',
  values = ['open', 'high', 'low','close', 'volume'],
  aggfunc = 'mean' # getting the mean == average
).reset_index()
```

	ticker	close	high	low	open	volume
0	AAPL	186.986218	188.906858	185.135729	187.038674	3.402145e+07
1	AMZN	1641.726175	1662.839801	1619.840398	1644.072669	5.649563e+06
2	FB	171.510936	173.615298	169.303110	171.454424	2.768798e+07
3	GOOG	1113.225139	1125.777649	1101.001594	1113.554104	1.742645e+06
4	NFLX	319.290299	325.224583	313.187273	319.620533	1.147030e+07

7. Calculate the Z-scores for each numeric column of Netflix's data (ticker is NFLX) using apply().

```
columns = faang_df.columns[1:]

for i in columns:
  faang_df[f'{i}_zscore'] = faang_df[f'{i}'].apply(
      lambda x: (x - (faang_df.mean()[f'{i}'])) / (faang_df.std()[f'{i}'])
      ).abs()

faang_df.query('ticker == "NFLX"')
```

	ticker	open	high	low	close	volume	open_zscore	high_zscore
date								
2018-01-02	NFLX	196.10	201.6500	195.4200	201.070	10966889	0.819007	0.814892
2018-01-03	NFLX	202.05	206.2100	201.5000	205.050	8591369	0.809083	0.807364
2018-01-04	NFLX	206.20	207.0500	204.0006	205.630	6029616	0.802161	0.805978
2018-01-05	NFLX	207.25	210.0200	205.5900	209.990	7033240	0.800410	0.801075
2018-01-08	NFLX	210.02	212.5000	208.4400	212.050	5580178	0.795790	0.796981
...	...	...	...	...	...	...	...	...
2018-12-24	NFLX	242.00	250.6500	233.6800	233.880	9547616	0.742451	0.734001
2018-12-26	NFLX	233.92	254.5000	231.2300	253.670	14402735	0.755928	0.727645
2018-12-27	NFLX	250.11	255.5900	240.1000	255.565	12235217	0.728925	0.725846
2018-12-28	NFLX	257.94	261.9144	249.8000	256.080	10987286	0.715865	0.715405
2018-12-31	NFLX	260.16	270.1001	260.0000	267.660	13508920	0.712163	0.701892

251 rows × 11 columns

8. Add event descriptions:

- Create a dataframe with the following three columns: ticker, date, and event. The columns should have the following values:
  - ticker: 'FB'
  - date: ['2018-07-25', '2018-03-19', '2018-03-20']
  - event: ['Disappointing user growth announced after close.', 'Cambridge Analytica story', 'FTC investigation']
- Set the index to ['date', 'ticker']
- Merge this data with the FAANG data using an outer join

```
event_df = pd.DataFrame({
    'ticker':'FB',
    'date':["2018-07-25", "2018-03-19", "2018-03-20"],
    'event':['Disappointing user growth announced after close.',
             'Cambridge Analytica story', 'FTC investigation']
})
```

```
faang_df = pd.DataFrame(pd.read_csv("/content/faang.csv"))
faang_df.merge(event_df, how = 'outer')
```

	ticker	date	open	high	low	close	volume	event

9. Use the transform() method on the FAANG data to represent all the values in terms of the first date in the data. To do so, divide all the values for each ticker by the values for the first date in the data for that ticker. This is referred to as an index, and the data for the first date is the base (https://ec.europa.eu/eurostat/statistics-explained/ index.php/ Beginners:Statisticalconcept-Indexandbaseyear). When data is in this format, we can easily see growth over time. Hint: transform() can take a function name.

...	...	...	...	...	...	...	...	...

```
faang_df.groupby('ticker').head()
```

	ticker	date	open	high	low	close	volume
0	FB	2018-01-02	177.6800	181.5800	177.5500	181.4200	18151903
1	FB	2018-01-03	181.8800	184.7800	181.3300	184.6700	16886563
2	FB	2018-01-04	184.9000	186.2100	184.0996	184.3300	13880896
3	FB	2018-01-05	185.5900	186.9000	184.9300	186.8500	13574535
4	FB	2018-01-08	187.2000	188.9000	186.3300	188.2800	17994726
251	AAPL	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934
252	AAPL	2018-01-03	169.2521	171.2337	168.6929	168.9578	29517899
253	AAPL	2018-01-04	169.2619	170.1742	168.8106	169.7426	22434597
254	AAPL	2018-01-05	170.1448	172.0381	169.7622	171.6751	23660018
255	AAPL	2018-01-08	171.0375	172.2736	170.6255	171.0375	20567766
502	AMZN	2018-01-02	1172.0000	1190.0000	1170.5100	1189.0100	2694494
503	AMZN	2018-01-03	1188.3000	1205.4900	1188.3000	1204.2000	3108793