Chapter 4, Task 2

Exercise 01

With the earthquakes.csv file, select all the earthquakes in Japan with a magnitude of 4.9 or greater using the mb magnitude type.

```
filtered_df = df.query("parsed_place == 'Japan'and mag >= 4.9and magType == 'mb'")
# query by location and mag and magtype
print(filtered_df)
```

Exercise 02

Create bins for each full number of earthquake magnitude (for instance, the first bin is (0, 1], the second is (1, 2], and so on) with the ml magnitude type and count how many are in each bin.

```
count_by_bin = (df.query("magType == 'ml'")
                .assign(mag_bin=pd.cut(df['mag'], bins=range(0, 11), right=True))
                .groupby('mag_bin')
                .size())
    # query by the magtype ml
    # create bins using pd.cut () from 0 to 10 (maximum magnitude)
    # assign those to new column
    # groupby() and size() to count # in each bin

print(count_by_bin)
```

Exercise 03

Using the faang.csv file, group by the ticker and resample to monthly frequency.
Make the following aggregations:
a) Mean of the opening price
b) Maximum of the high price
c) Minimum of the low price
d) Mean of the closing price
e) Sum of the volume traded

```
faang = pd.read_csv('./exercises/faang.csv')
faang['date'] = pd.to_datetime(faang['date'])
faang.set_index('date', inplace=True)       # I had to convert the column to dt for it to w
monthly_data = (faang.groupby('ticker').resample('M').agg({
                    'open': np.mean,
                    'high': np.max,
                    'low': np.min,
                    'close': np.mean,
                    'volume': np.sum
                }))
```

```
    # group by ticker
    # resample to monthly freq

print(monthly_data)
```

Exercise 04

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 magnitude type along the columns.

```
earthquakes = pd.read_csv('./exercises/earthquakes.csv')

crosstab_result = pd.crosstab(
    index=earthquakes['tsunami'],       # uniques from 'tsunami' column to be rows
    columns=earthquakes['magType'],     # uniques from 'magType' column to be columns
    values=earthquakes['mag'],          # values to aggregate
    aggfunc=np.max                      # aggregate based on max
)

print(crosstab_result)
```

Exercise 05

Calculate the rolling 60-day aggregations of the OHLC data by ticker for the FAANG data. Use the same aggregations as exercise 3

```
rolling60 = faang.groupby('ticker').apply(      # group by ticker
    lambda x: x.rolling(window='60D').agg({     # use 60 day rolling window
        'open': np.mean,                        # use same agg from exercise 3
        'high': np.max,
        'low': np.min,
        'close': np.mean
    })
)

print(rolling60)
```

Exercise 06

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

```
pivot_table = faang.pivot_table(index='ticker')

print(pivot_table)
```

Exercise 07

Calculate the Z-scores for each numeric column of Amazon's data (ticker is AMZN) in Q4 2018 using apply()

```
# look at AMZN ticker i nrange of 2018 Q4
amzn = faang.loc['2018-Q4'].query("ticker == 'AMZN'")

z_amzn = amzn.drop(columns='ticker').apply(
    # z score = (x - mean) / stdev
    lambda x: (x - x.mean()) / x.std()
)

print(z_amzn)
```

Exercise 08

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

```
faang = pd.read_csv('./exercises/faang.csv')
faang['date'] = pd.to_datetime(faang['date'])

event_data = 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'
    ]
})
# create the dataframe of facebook ticker, the specific dates, and the respective events

merged_data = pd.merge(
    faang.set_index(['date', 'ticker']),
    event_data.set_index(['date', 'ticker']),
    left_index=True,
    right_index=True,
    how='outer',
    sort=False
```

```
).reset_index()

# set index of both dfs to date and ticker and merge them

# print(merged_data)
print(event_data)
```

```
  ticker        date                                              event
0     FB  2018-07-25  Disappointing user growth announced after close.
1     FB  2018-03-19                          Cambridge Analytica story
2     FB  2018-03-20                                FTC investigation
```

Exercise 09

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:Statistical_concept_-_Index_and_base_year). When data is in this format, we can easily see growth over time. Hint: transform() can take a function name.

```
faang = pd.read_csv('./exercises/faang.csv')
faang['date'] = pd.to_datetime(faang['date'])
faang.sort_values(['ticker', 'date'], inplace=True)      # set index
transform_faang = faang.groupby('ticker').transform(     # group by ticker
    lambda x: x / x.iloc[0]                              # get first value of each group
    )

print(transform_faang)
```

Exercise 10

The European Centre for Disease Prevention and Control (ECDC) provides an open dataset on COVID-19 cases called daily number of new reported cases of COVID-19 by country worldwide. This dataset is updated daily, but we will use a snapshot that contains data through September 18, 2020. Complete the following tasks to practice the skills you've learned up to this point in the book:
a) Prepare the data:
i) Read in the data in the covid19_cases.csv file.
ii) Create a date column by parsing the dateRep column into a datetime.
iii) Set the date column as the index.
iv) Use the replace() method to update all occurrences of United_ States_of_America and United_Kingdom to USA and UK, respectively.
v) Sort the index.
b) For the five countries with the most cases (cumulative), find the day with

the largest number of cases.
c) Find the 7-day average change in COVID-19 cases for the last week in the data for the five countries with the most cases.
d) Find the first date that each country other than China had cases.
e) Rank the countries by cumulative cases using percentiles.

```
# Part A
# read data
covid = pd.read_csv('./exercises/covid19_cases.csv')
# create date column
covid['date'] = pd.to_datetime(covid['dateRep'])
# make date column the index
covid.set_index('date', inplace=True)
# replace countries
covid.replace({'countriesAndTerritories': {'United_States_of_America': 'USA', 'United_Kingdo
# sort the index
covid.sort_index(inplace=True)
```

```
# Part B
# Sum the cases by date and country
grouped = covid.groupby('countriesAndTerritories').cases.sum().nlargest(5).index

covid[covid.countriesAndTerritories.isin(grouped)].groupby('countriesAndTerritories').cases
```

```
# Part C
# Calculate 7-day rolling mean of the daily case difference
seven_day_avg = covid[covid['countriesAndTerritories'].isin(grouped)] \
    .groupby(['countriesAndTerritories', 'date']).cases.sum() \
    .groupby('countriesAndTerritories').apply(lambda x: x.diff().rolling(7).mean())

# Get the last week's data
last_week_seven_day_avg = seven_day_avg.groupby('countriesAndTerritories').last('1W')

print(last_week_seven_day_avg)
```

```
# Part D
# make new df without china
remove_china = covid[covid['countriesAndTerritories'] != 'China']
# find date of first case
remove_china.groupby('countriesAndTerritories')['cases'].idxmax()
```

```
# Part E
# find total cases
cumulative_cases = covid.groupby('countriesAndTerritories')['cases'].sum()
# sort them and convert to percentile
cumulative_cases.rank(pct=True).sort_values(ascending=False)
```