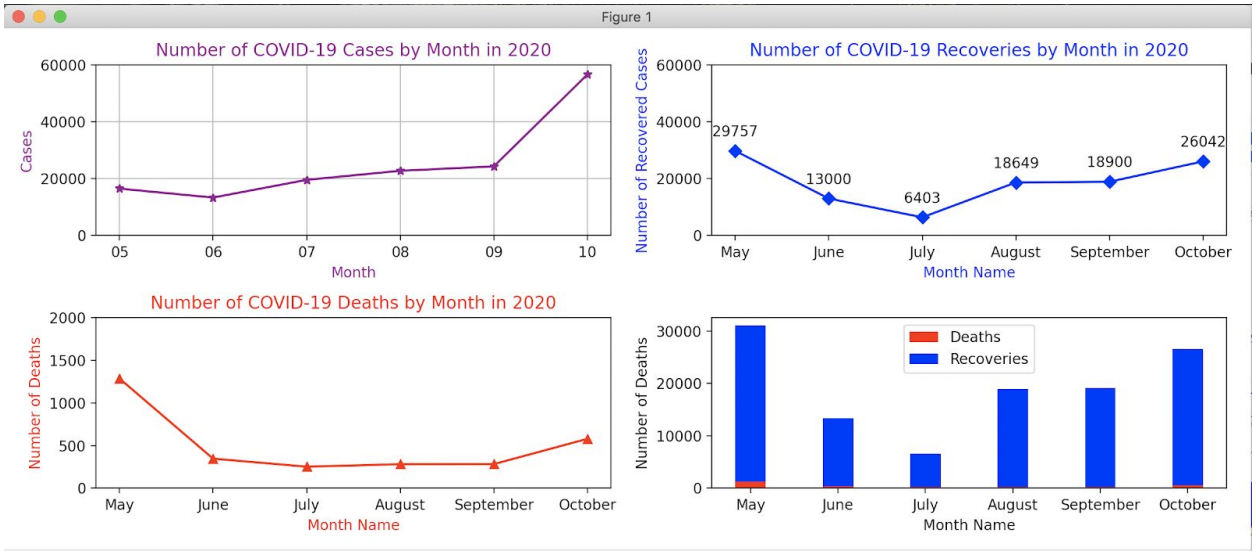
Github Repo: https://github.com/chalkerm/FinalProject

1. The goals for your project (10 points)
    a. Our original goals were to use Twitter, Instagram, and Youtube APIs to see if there was a correlation between likes and follower count for a company, and to compare the same company's interaction across the three platforms to see which one yielded the most engagement with followers. However, we realized that we were constrained by the APIs, and could not gather the data we had originally expected.
    b. Our new goal was to track COVID cases, deaths, and recoveries through its peak months (May-October), to see the upward and downward trends overtime in Michigan.
2. The goals that were achieved (10 points)
    a. Gathered data from three separate APIs related to COVID to store information on the number of cases, deaths, and recoveries in a database, covid_tracking.db.
    b. We conducted calculations to find total cases, deaths, and recoveries for each month.
    c. We created visualizations to depict total cases, deaths, and recoveries by month from May to October.
3. The problems that you faced (10 points)
    a. Our APIs did not have the functionality that we originally thought, so we had to change our topic.
    b. We originally thought that we could add data to the tables in increments of 25, not by running the function and adding 25 at a time, so we had to figure out how to add 25 unique items each time we ran the function, and have it pick up where it previously left off.
    c. When writing our calculations file, it kept writing one line on top of another. We then looked for alternative ways to write to text files.
    d. The data for the recovered.py was more nested than I realized at first, so initially I was collecting data on the US as a whole, not just the state of Michigan. The numbers looked high, so I checked my data and I was not pulling from the correct data.
    e. As we are all in different locations, we had to do this entire project virtually. As a result, we had a lot of merge errors on GitHub.
4. Your file that contains the calculations from the data in the database (10 points)

calculations_outfile

```
Total COVID Cases for May 2020: 16485
Total COVID Cases for June 2020: 13339
Total COVID Cases for July 2020: 19558
Total COVID Cases for August 2020: 22745
Total COVID Cases for September 2020: 24294
Total COVID Cases for October 2020: 56568
=======================================
Total Deaths from COVID for October 2020: 578
Total Deaths from COVID for September 2020: 282
Total Deaths from COVID for August 2020: 281
Total Deaths from COVID for July 2020: 252
Total Deaths from COVID for June 2020: 346
Total Deaths from COVID for May 2020: 1284
=======================================
Total Recoveries from COVID for May 2020: 29757
Total Recoveries from COVID for June 2020: 13000
Total Recoveries from COVID for July 2020: 6403
Total Recoveries from COVID for August 2020: 18649
Total Recoveries from COVID for September 2020: 18900
Total Recoveries from COVID for October 2020: 26042
```

5. The visualization that you created (i.e. screenshot or image file) (10 points)



6. Instructions for running your code (10 points)

casesByDate.py: to enter COVID Cases data into covid_tracking.db you run casesByDate.py. Everytime you run it, it will enter 25 items. At the end of the program, it will tell you how many items are in the Cases table. It will tell you to run the program again if all the data has not been entered yet. Continue to run casesByDate.py to enter all the data in the database. Once all the possible data has been entered in the Cases table, the program will tell you that all data has been inputted.

recovered.py: Enter data into the database by running recovered.py. It first creates the covid_tracking.db, the database where we will store the data. It then creates a month table, which has the month in number in string form (i.e. January is the first month, so it is '01') and the name of the month in another column. It also creates the Recovered table, if it does not already exist. Then it asks for user input to see if the user would like to enter 25 rows of data into the

recovered table. If the user enters no, the program will exit. If the user enters yes, it will add 25 pieces of data and ask if the user would like to enter 25 more. It continues to do this until there is no more data to input, in which case it will print “Data input complete” then end the program.
Death.py: In order to ender COVID death data in the database (covid_tracking.db), run the program death.py. Everytime death.py is run, it will enter 25 distinct data values into the database. The program will continue to ask you, “Would you like to add 25 rows? Please enter 'yes' or 'no'." After all the data has been entered, that program will prompt you that “data input is complete”. The data being inputted into the table is the date, month, number of deaths, number of deaths probable, and number of deaths confirmed.
calculations.py: Running calculations.py will conduct calculations from our database, and write the calculations to an outfile, calculations_outfile. If that file does not already exist, it will create it. It then graphs three plots (cases by month, deaths by month, and recovered data by month) as well as a stacked bar graph of deaths and recoveries from COVID by month. It saves our graphs to a PNG, COVID_graphs.png.

7. Documentation for each function that you wrote. This includes the input and output for each function (20 points)

```
'''casesByDate.py function descriptions'''


def get_data():
    ''' this funciton tries to request the data from the API if there is an error it prints "error when requesting data from API"
     and creates an empty dict. if the request is successful, the function loops through the data and adds only the data from
    the months May through October to a list named MayThruOct and returns the list.'''

def clean_data():
    '''this function calls get_data() and loops through the data and creates a list of tuples and returns it.
     Each tuple includes (date, total cases, new cases)
     the date should be a string in the form of 2020-MM-DD
    total cases should become an int and new cases should become a real'''

def setUpDatabase(db_name):
    '''creates or finds a database with the passed in database name and returns cur,conn'''

def month_table(cur,conn):
    '''this function takes in cur,conn and loops through data returned from clean_data() and creates a list called month_list of all the unique month numbers
     creates a new table named months with two columns:
    (1) month_id (integer primary key)
    (2) month_num (text)
    the function doesn't return anything. '''


def add_data_to_table(cur,conn):
    ''' the input is in cur,conn. the function adds data (25 items at a time) returned from the clean_data() function to a new table named Cases with five columns:
    (1) id (integer primary key)
    (2) month_id (text) HINT: Found from the months2 table
    (3) date (text in form of 2020-DD-MM)
    (4) new_cases (integer)
    (5) total_cases (integer)
    the function also calls month_table() only when Cases is empty.
    the function doesn't return anything'''
```

Deaths.py function descriptions:

```
def setUpDatabase(db_name):
    ''' This function sets up a database. It will return a cursor and connector. '''
def create_month_table(cur, conn):
   ''' This function takes in cursor and connector (cur and conn) and creates a table within the
    database that has the name of the month. This function will be used to retrive the name of the month and
    insert it into the death table to allow for easier calculations. '''
def create_death_table(cur, conn):
''' This function takes in cur and conn as parameters and creates the table for COVID
    death data (total deaths and new deaths confirmed), located in the database. '''
def get_data():
    ''' This function creates the URL for the API request. It requires information on
    the state- michigan (in the US) for which data will be gathered. It returns the data.'''
def add_to_death_table(cur,conn,lst):
    ''' This function creates a list of strings for months May - October.
    This function loops adds the necessary data to the Death table. For each date in the time period,
    it adds a key, date (2020MONTHDAY), individual months (May- October), total deaths in michigan to date, and new confirmed deaths in michigan to date.
    This function adds 25 unique pieces of data into the table at a time. '''
def keep_running(cur, conn, lst):
    ''' This function asks the user if they would like to add the next 25 rows
    to the table. If the user inputs no, it ends the program. If the user inputs
    yes, it adds the next 25 unique rows of data, and asks if it should input 25 more. When
    there is no more data to input, the program prints "Data input complete" and exits the
    program. '''
def main():
    ''' This function sets up the database and tables, stores information in dictionary,
    loops through dictionary and adds appropriate data to death table, 25 unique items at a time. '''
```

Recovered.py function descriptions

```
def setUpDatabase(db_name):
    ''' This function sets up a database in the current location. It takes in the name of the database
    and will return a cursor and connector, cur and conn. '''
def create_month_table(cur, conn):
    ''' This function takes in parameters, cur and conn and creates a table within the
    database that has the number of the month (i.e. January is the first month, '01') and
    the name of the month. This function will be used to grab the name of the month and
    insert it into the recovered table to allow for easier calculations later. '''
def create_recovered_table(cur, conn):
    ''' This function takes in cur and conn as parameters and creates the table for COVID
    recovery data, located in the database. '''
def create_request_url(state, start_date, end_date):
    ''' This function creates the URL for the API request. It requires information on
    the state (in the US) for which data will be gathered, as well as a start date
    and end date for the time period it will be getting data for. All required parameters,
    state, start_date, and end_date should be strings. It returns the request URL as a string.
    Date should be in the format YYYY-MM-DD '''
def get_data():
    ''' This function executes the API request for each month in our designated time
    period. The month must be in the form of a string for the API request url.
    After each execution, it adds the requested data to a dictionary, and returns the
    dictionary with the data in it. '''
def add_recovered_data(cur, conn, d):
    ''' This function takes in cur, conn, and a dictionary (returned by get_data), and loops through
    the data from the dictionary returned in get_data and adds the necessary data to the Recovered table.
    For each date in the time period,
    it adds:
    1. a primary key (integer)
    2. date (string)
    3. month name (string selected from Month table)
    4. number of new people who recovered from COVID on that date (integer)
    5. total number of people recovered from COVID in Michigan to date (integer)
    This function adds 25 unique pieces of data into the table at a time. '''
def keep_running(cur, conn, d):
    ''' This function asks the user if they would like to add the next 25 rows
    to the table. If the user inputs no, it ends the program. If the user inputs
    yes, it adds the next 25 rows of data, and asks if it should input 25 more. If
    there is no more data to input, it prints "Data input complete" and exits the
    program. '''
```

```
'''calculations.py descriptions'''


def setUpDatabase(db_name):
    '''creates a new database with the passed in database name and returns cur,conn'''

def cases_calculations(cur,conn):
    '''takes in cur,conn as input. reates a list of the months from the month_num column in months2 table.
    returns a dictionary named newCases_dict where the key is the month_num from months2 table and the value is the total new cases from that month
    the total new cases are calculated by adding up the new_cases column from that specific month by joining Cases and months2 tables '''


def recovered_dictionary(cur, conn):
    ''' This function takes cur and conn as parameters. It fetches distinct months
    from the Recovered table and adds them to a list. It then adds up the total new
    recovered cases for each month, and adds the month to a dictionary as a key, and
    the total number of recoveries for that month as the value. It returns this dictionary. '''


def death_dictionary(cur, conn):
    ''' This function takes cur and conn as parameters. It fetches distinct months
    from the death table and adds them to a list. It then adds up the total new
    death confirmed cases for each month, and adds the month to a dictionary as a key, and
    the total number of confirmed deaths for that month as the value. It returns this dictionary. '''

def write_calculations(filename, d, data, dic):
    '''takes in a filename and 3 dictionaries where the key is the month and the value is the total. the function
    writes to the filename the total cases,deaths, and recoveries for each month. the function doesn't return anything.'''

def graphs(cases_data,recov_d, death_d):
    '''takes 3 dictionaries as input, each have the month as they key, and the value as the total for that month'''
    '''the function creates 4 subplots displayed in 2 columns, 2 rows'''
    '''the function doesn't return anything but it shows the graphs and saves them as a png to COVID_graphs.png'''
```

8. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

Date	Issue Description	Location of Resource	Result (Did it solve the issue?)
12/1/2020	Add 25 to database every time the program runs, not in increments of 25	Lecture Slides “ProjectExWith Vis-v5.pdf“	Yes, I was able to add 25 pieces of data at a time and created another function, keep_running to delegate how many times to run it (or stop when table is complete).
12/9/20	My calculations were writing on top of each other so I could only see the last one.	https://www.w3schools.com/python/python_file_write.asp	Yes
12/10/20	The subplots were overlapping each other whenever the graph was generated.	https://matplotlib.org/3.2.1/tutorials/intermediate/tight_layout_guide.html	Yes, the graphs can now be easily read
12/10	Didn’t know how to customize the	https://www.w3schools.com/python/matplot	Yes

	Matplotlib graph color and markers	lib_markers.asp	
12/10	Was having trouble making an accurate graph	https://matplotlib.org/tutorials/introductory/pyplot.html	Yes, it was a small build error
12/10	I wanted to add labels to my points, but didn't know how.	https://queirozf.com/entries/add-labels-and-text-to-matplotlib-plots-annotation-examples	Yes

APIs used:
Death API: https://api.covidtracking.com/v1/states/mi/daily.json
Recovered API: https://covid19tracking.narrativa.com/index_en.html
Cases API: https://data.cdc.gov/resource/9mfq-cb36.json?state=MI