https://github.com/sunm22/final-project-SI206

SI 206 Final Report
Team Members: Mingxuan Sun and Tiara Amadia

GOALS

Original:
Our original goal was to visualize the relationship between the amount of incoming flights from DTW and the amount of positive COVID cases and/or deaths during March 2019 and 2020. We first planned to collect data from 2 APIs to do so, the Covid Tracking Project and the Skyscanner API. We would then pull Michigan state data from March 2019 and March 2020, store it into a database with the number of weeks (or time) as the shared key, and then make visualizations on each API as well as a combination of both.

This project was supposed to look at the relationships between covid cases and flight data. We would calculate the percentage change of COVID cases from our COVID database, and calculate the percentage change in flight numbers from our flight database. Then we would make 1 visualization about the starter data, 1 visualization about the percentage change for each API and then 1 combined visualization regarding percentage change in data of both databases.

Achieved:
Our project took quite a turn because of the difficulty in sourcing APIs compatible with the COVID Tracking Project. Instead, we achieved scraping the data from 1 API and 1 website. We stored all data into a database consisting of at least 1 main table per API/website, including 1 table with shared keys and 2 tables for calculations. The team managed to do at least 4 visualizations in total: 2 COVID visualizations, 1 population visualization and 1 visualization combining both aspects of the database. The COVID database also managed to have 2 tables that share a key whilst the population database has 1 table including the different population numbers for a different year as well as an extra column that tracks the date of the population. One table from each API/website has 100 entries. Both team members also managed to conduct calculations for each of their databases, deriving the percentage changes in COVID cases and population numbers from each API/website.

MAIN PROBLEMS ENCOUNTERED

1. Searching for APIs
   We started off our project by scraping data from the Covid Tracking Project API. Unfortunately, since it is a fairly new project it only started recording legitimate data from June 2020 onwards. The flight scanner API that we were going to use was no use because it only recorded real time flight data and not historic data. Thus, we decided to combine COVID data with government records. This was fairly incompatible with alot of the APIs that we were considering, since government APIs were only up to date until 2019 or at most 2020. In the end, we resolved this situation by scrapping the 2nd API totally and ended up just scraping data off of a Wikipedia website.

2. Overcoming duplicate data problems in each API/website's tables
    a. Tiara's Solution
    Tiara had to overcome the duplicate data problem because the project divided the data per state and unfortunately there are only 50 states in the US. Thus, she had to complete 100+ entries by combining population numbers from different years into a single column. One way she overcame this is to label the states according to their year dates. Instead of having duplicate string data where each state was replicated twice, the row labels were all unique with state name and year (i.e. California2020, California2021).
    b. Minnie's Solution
    Minnie also had 50 states worth of data for 2 years, totalling 100 rows of data. She overcame the duplicate data issue by creating a foreign key labeled state_id for duplicate state names and one for dates to remove all duplicate strings. These keys corresponded to two tables: one is States, which has all 50 state abbreviations in 50 rows, and one is Dates, which had the 2020 and 2021 dates stored in 2 rows. Thus, any duplicate strings were replaced with foreign key ids instead.
3. Limiting amount of data storage to 25 items at a time
    a. Tiara's solution:
    Tiara first scraped the data 50 items at a time by scraping the data off of the Wikipedia article through BeautifulSoup and putting it in a dictionary where the state is the key and the population number is the value. She ended up limiting the amount of data storage to 25 items at a time by splitting the given dictionaries she gained for each year into 2 so there is a total of 4 dictionaries to be scraped into the data each time it is run.
    b. Minnie's solution:
    Created a function to identify the value of the largest integer primary key in the table. Called that function in main, and wrote a set of if statements that would pull 25 lines of the appropriate data depending on how many rows were counted in the current table.
4. Joining each other's code into 1 database and code file
    Both team members had trouble combining all of our code into 1 file and database. One of the hardest challenges was combining the data we put into the same database and we overcame that by inserting the data into the same name of database (finalProject.db).
5. Technical Difficulties
    Other difficulties that we both encountered were technical difficulties that were easy to overcome by debugging the code. One example is the fact that scraping the website through BeautifulSoup seemed difficult until we realized there was a mistake in the writing of the variable name itself (ie. row.cell instead of row_cell as declared).

CALCULATIONS FILE

*** note: the calculations file is not included in the .zip folder. It will be created once you run the code according to the instructions below. Here are screenshots of what those files should look like. While we could've combined all our data into one csv or txt file, we created two files with different extensions to practice writing to files in different formats, and didn't see any restrictions in the rubric against this so hope it's acceptable.

covid_calculations.csv

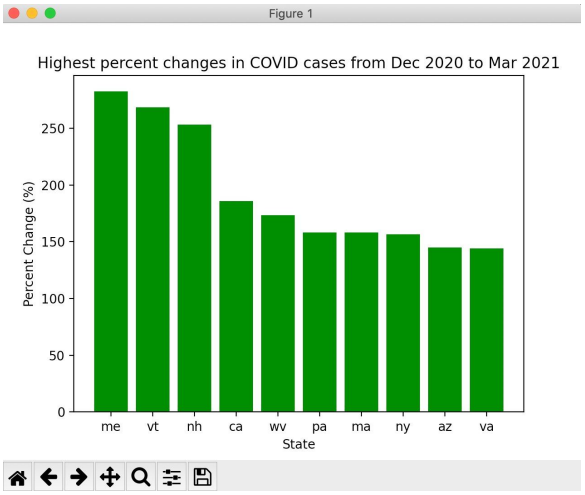
State	Percent Change COVID Cases
al	97.63503361012258
ak	78.6957341207514
az	145.13746555575
ar	103.89180774469742
ca	185.78398924574086
co	83.97960473642073
ct	140.26980143826734
de	143.11146575681698
fl	92.47804740349427
ga	114.83548661328071
hi	57.11704806744772
id	70.04365867568684
il	62.1900910338555
in	93.76141567428341
ia	38.83322353218812
ks	87.91268117322765
ky	124.22530136268344
la	82.46193320434088
me	282.3814295257181
md	92.56668406791458
ma	158.00309766366354
mi	65.9039127279348
mn	52.03002060115664
ms	92.72007823276839
mo	58.7899871486103
mt	59.66141919151966
ne	58.11131791880505
nv	91.25443122162889
nh	253.12413856473398
nj	118.24322328188408
nm	88.0143634516541
ny	156.56322251302907
nc	137.39462975816218
nd	23.71803561525664
oh	127.50219138651408
ok	115.05549372875747
or	108.24196948204319
pa	158.3872637141145
ri	122.39664283493939
sc	140.21753033182281
sd	40.385851295234325
tn	105.74297843687037
tx	107.16083307889453
ut	89.11187795132582
vt	268.45360824742266
va	143.97762254074973
wa	94.59148056525129
wv	173.35204227948708
wi	49.305056861167486
wy	61.99970418577133

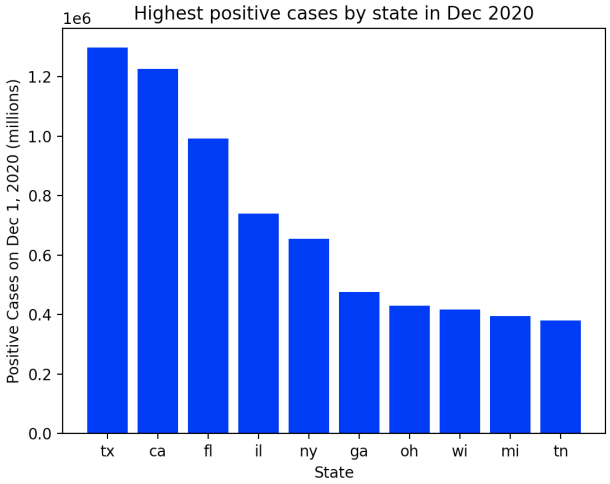
pop_calculations.txt

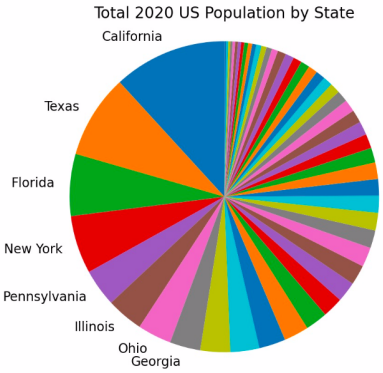
(50 rows of text total, couldn't fit in one screenshot)

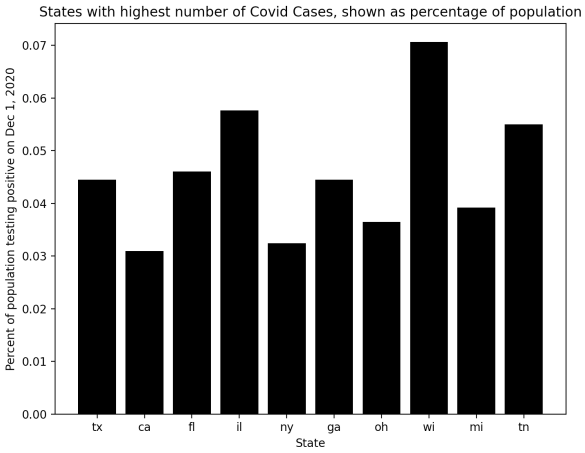
```
California has had a 1.0623504521237959 change in population
Texas has had a 1.1605742262023901 change in population
Florida has had a 1.1472885134067785 change in population
New York has had a 1.0432265760599257 change in population
Pennsylvania has had a 1.024362759133545 change in population
Illinois has had a 0.9993848315499969 change in population
Ohio has had a 1.023607151698643 change in population
Georgia has had a 1.1071075729074937 change in population
North Carolina has had a 1.0963207631957395 change in population
Michigan has had a 1.0203166040041929 change in population
New Jersey has had a 1.0571661805749706 change in population
Virginia has had a 1.0816792950502336 change in population
Washington has had a 1.1474310510458707 change in population
Arizona has had a 1.1199787172030362 change in population
Massachusetts has had a 1.074200905396442 change in population
Tennessee has had a 1.089943674111916 change in population
Indiana has had a 1.047268254027498 change in population
Maryland has had a 1.0713124260420621 change in population
Missouri has had a 1.0286118030825888 change in population
Wisconsin has had a 1.037012048209719 6 change in population
Colorado has had a 1.1497207505931366 change in population
Minnesota has had a 1.0765144680590317 change in population
South Carolina has had a 1.1079586384985052 change in population
Alabama has had a 1.0523704656491488 change in population
Louisiana has had a 1.0282562295792184 change in population
Kentucky has had a 1.0391704596545994 change in population
Oregon has had a 1.1071307941324025 change in population
Oklahoma has had a 1.0565569577466891 change in population
Connecticut has had a 1.0095691303285836 change in population
Utah has had a 1.185017466356234 change in population
Puerto Rico has had a 0.8819270227057947 change in population
Iowa has had a 1.047942869429203 change in population
Nevada has had a 1.1510473233055032 change in population
Arkansas has had a 1.033553069736529 change in population
Mississippi has had a 0.9988599051594768 change in population
Kansas has had a 1.0307547742504866 change in population
New Mexico has had a 1.0296433675751355 change in population
Nebraska has had a 1.075008993391705 change in population
Idaho has had a 1.1746607195030307 change in population
West Virginia has had a 0.9687268280415371 change in population
```

VISUALIZATIONS CREATED









INSTRUCTIONS FOR RUNNING THE CODE

1. Install Anaconda and SQLite Database Browser if not already installed. Download all files from the .zip folder and make sure there are only three .py files and a README.
2. Open the files “population_data.py”, “covid_data.py”, and viz.py”.
3. Run the “population_data.py” file. You should see a new file created in the folder called “finalProject.db”. Open the .db file in DB Browser for SQLite. At this point, there should be one table in the database titled Population with 25 rows.
4. Run the “population_data.py” file three more times. At this point, you should have a database with one table with 100 rows. You should also see a new file called “pop_calculations.txt”, which contains the calculations from scraping the population website.
5. Run the “covid_data.py” file. You should see three new tables in the database titled States, Dates, and CovidData. The CovidData table should have 25 rows.
6. Run the “covid_data.py” file three more times. The CovidData table should now have 100 rows. You should also see a new table called PercentChange, which is a table used later in the visualizations. You should also see a new file called “covid_calculations.csv”, which contains the calculations from the Covid API.
7. Run the “viz.py” file once. Four visualizations total will be created. There is a chance that only one will show up at a time. If that happens, close one visualization in order to see the next one.

CODE DOCUMENTATION

covid_data.py

```
def setUpDatabase(db_name):
    '''This function takes in the name of the database, makes a
connection to server
    using name given, and returns cur and conn as the cursor and
connection variable
    to allow database access.'''
def state_table(cur, conn):
    '''Takes in the cur and conn variables. Creates a table called
States that has
    lowercase abbreviations for all 50 states and a state_id primary
key for each.'''
def date_table(cur, conn):
    '''Takes in the cur and conn variables. Creates a table called
Dates that holds the
    two date values and a date_id primary key for each.'''
def covid_table(cur, conn, state, date, positive):
    '''This function takes in cursor and connection variables to
database, state,
    date, and number of positive COVID cases for that state. It
creates a table in the
    database if it doesn't exist and inserts the state, date, and
number of positive
    cases. Returns nothing.'''
def percent_change_table(cur, conn, state_id, percent):
    '''This function takes in cursor and connection variables to
database, state,
    and percent change calculated from percent_change. It creates a
table in the
    database if it doesn't exist and inserts the state and percent
change. Returns
    nothing.'''
```

```
def covid_table_length(cur, conn):
    '''This function calculates the number of rows in the CovidData
table to help with
    extracting 25 lines at a time. Returns the number of rows in the
table as an
    int.'''
def get_mar_data(cur, conn, state, state_id, date_id):
    '''This function takes in the cursor and connection variables, and
the lowercase state abbreviation. It sends requests to COVID Tracking
Project API for the latest data for the given state (mostly March 7th
2021, as that's when the project ended). Uses json to extract date
and number of positive cases. Calls covid_table to create and add to
table. Returns nothing.'''
def get_dec_data(cur, conn, state, state_id, date_id):
    '''This function takes in the cursor and connection variables, and
the lowercase state abbreviation. It sends requests to COVID Tracking
Project API for Dec 1st, 2020 data for the given state. Uses json to
extract date and number of positive cases. Calls covid_table to
create and add to table. Returns nothing.'''
def percent_change(cur, conn, states_list):
    '''This function takes in cursor and connection variables, and the
lowercase state abbreviation. It calculates the percent change from
Dec 2020 to Mar 2021 in number of COVID cases for the given state.
Returns a list with all the percent changes.'''
def write_to_file(filename, cur, conn, states_list):
    '''Takes in a filename, cur and conn variables, and a list of
state abbreviations.
Writes to a csv file the column headers and values of state
abbreviations and percent change in COVID cases as calculated in
percent_change().'''
def main():
    '''Main includes two state abbreviation lists with 25 states in
each. It calls covid_table_length()and uses the results to determine
what set of data to collect, storing 25 rows each time until
CovidData is populated with 100 rows. Then calculates and populates
```

```
PercentChange, and writes calculations to csv file. Returns
nothing.'''
```

population_data.py

```
def setUpDatabase(db_name):
   '''This function takes in the name of the database, makes a
connection to server
   using name given, and returns cur and conn as the cursor and
connection variable
   to allow database access.'''
def pop_table(cur, conn, pop_dict, date, count):
   '''This function takes in the cursor and connection variables to
database, state, year and number of US Population for that state. It
creates a table in the database if it doesn't exist and inserts the
state, date and number of population. Returns nothing '''
def percent_changes(cur, conn):
   '''This function takes in cursor and connection variables. It
grabs the population and states from the database, seeds out the 2020
and 2010 population, append the state name into a labels list and its
population numbers to the population list. Calculate percentage
changes and write the calculations onto a txt file. Returns
nothing'''
def pop_table_length(cur, conn):
   '''This function calculates the number of rows in the CovidData
table to help with extracting 25 lines at a time. Returns the number
of rows in the table as an int.'''
def get_pop_2020(soup):
   '''This function takes in a soup request called in the main() and
finds the table retaining population numbers through class tag. Then
it finds the rows of the table and iterates through the rows to
scrape state names and population numbers. It appends the state and
population number into a dictionary where state is key and population
number is value. It removes District of Columbia to get 50 states.
Returns dictionary containing state name and 2020 population
numbers.'''
```

```
def get_pop_2010(soup):
    '''This function takes in a soup request called in the main() and
finds the table retaining population numbers through class tag. Then
it finds the rows of the table and iterates through the rows to
scrape state names and population numbers. It appends the state and
population number into a dictionary where state is key and population
number is value. It removes District of Columbia to get 50 states.
Returns dictionary containing state name and 2010 population
numbers.'''
```

viz.py

```
def setUpDatabase(db_name):
    '''This function takes in the name of the database, makes a
connection to server
    using name given, and returns cur and conn as the cursor and
connection variable
    to allow database access.'''
def cases_percent_change(cur, conn):
    '''This function takes in the cursor and connection variables. It
uses matplotlib to create a bar graph of the 10 states with highest %
increase in COVID cases from Dec 2020 to Mar 2021 by using the
PercentChange table. Output is the creation of the graph.'''
def highest_positives_viz(cur, conn):
    '''This function takes in the cursor and connection variables. It
uses matplotlib
to create a bar graph of the 10 states with highest # of COVID cases
in Mar 2021 by using the CovidData table. Output is the creation of
the graph.'''
def pop_chart(cur, conn):
    '''This function takes in the cursor and connection variables. It
uses matplotlib to create a pie chart of the 50 United States and
their 2020 population numbers to create the division of the 2020
total US Population per state population. Output is the creation of
the pie chart.'''
def comparison_chart(cur, conn):
```

```
    '''This function takes in the cursor and connection variables. It
uses matplotlib to create a bar graph of the 10 states with highest #
of COVID cases on Dec 1 2020, exhibited as a percentage of their
overall population. Output is the creation of the graph.'''
def main():
    '''Establishes connection to server and creates visualizations.''
```

DOCUMENTATION OF RESOURCES USED:

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
04/21/2021	Forgot how to parse through Beautiful Soup via class	https://www.crummy.com/software/BeautifulSoup/bs4/doc/	Yes
04/21/2021	Didn't know what the difference between shared key and foreign key was	https://dba.stackexchange.com/questions/234108/what-is-the-difference-between-a-foreign-key-and-a-primary-foreign-key#:~:text=Look%20up%20%22shared%20primary%20key%22.&text=In%20Shared%20Primary%20Key%2C%20one,key%20of%20some%20other%20table.	Yes
04/24/2021	Didn't know how to split a dictionary in half	https://stackoverflow.com/questions/12988351/split-a-dictionary-in-half	Yes
04/26/2021	Didn't know how to delete an element from a dictionary	https://www.programiz.com/python-programming/dictionary#:~:text=Removing%20elements%20from%20Dictionary,item%20pair%20from%20the%20dictionary.	Yes