Group Members: Amelia Nam, Clare McNamara, & Allison Lee

Our github URL https://github.com/amelianam/si206_final_project.git

Goals for our project:

- Find the average temperature for each season
- Find the total number of covid cases for each season
- See if there is a correlation between temperature and rates of covid
- Visualize differences in average number of veterinary cases and average temperature based on the seasons
- Check if there is a correlation between daily temperature differences and the number of veterinary cases reported to the FDA on adverse animal drug cases

Goals that were achieved:

- We were able to find the total number of Covid-19 cases per season
- Found the average temperature each season
- We looked at and found that there wasn't a correlation between the temperature and number of COVID cases in the year 2020. The cases were the highest in the fall, lowest in the spring, and second highest in the winter, but the temperature was lowest in the winter and highest in the summer.
- There are many confounding variables that could have affected these results. Some that we considered were:
    - The dates we pulled from were in 2020, so it makes sense that there were the least amount of cases in the spring because covid was new.
    - We were shut down for a large part of summer so the cases were not super high
    - In the fall, people tried to return to their daily lives: school, jobs, etc. Because of this, more people interacting and the virus could spread easier
- We recognize that temperature is not the only factor involved, and our data reflects that there are other factors that affect the case numbers other than temperature alone.
- We found that there is no clear pattern between the daily temperature differences and the veterinary cases reported on that day
- We found that the number of veterinary cases reported is the greatest in the summer and the least in the winter. Some implications for this might be:
    - Heat stress/increased activity outside interacting with the drugs/animals health in general
    - Improper storage of pet drugs can mess up drug effects in hotter weathers

Problems that we faced:

- We had trouble using the strava API because we had to get authorization to access anyone's data. After this, we looked into a variety of other APIs, such as fitbit, Underarmour, Garmin, and multiple biking sites. Ultimately, we found that people's fitness data oftentimes required authorization and that we would not be able to access as much data as we needed to complete this project. As a result, instead of looking at exercise by state and weather, we decided to look at covid cases for one state (can use

our code to look at other states, but we chose Michigan) and weather- specifically weather.
- We had trouble with the COVID database because we originally thought that we could pull data from all of 2021 and up to the current date in 2022. We found that once we reached March of 2021, there was no longer data available. We decided to use the 15th day of the months that were available to us and organize data by season instead of by year.
- We had to rethink our visualization because originally, we planned to create a bar graph. Unfortunately, the bar graph could not show all three of our variables (season, temperature, and covid cases)
- We also had to change our graph from daily temperature in Michigan to daily temperature in Detroit because our API did not provide data that was broad to the whole state of Michigan.

File that contains the calculations from the data in the database:

covid_by_day.py
- Calculates average temperature (at midnight and for the whole day) per season
- Calculates the total number of covid cases in a given season

Vetinary.py
- Calculates absolute value of difference of daily average temperature and daily midnight temperature to get magnitude of daily temperature difference
- Calculates the average daily temperature for each season months (April, July, October, December)

Instructions for running our code:

COVID CODE
- Define a list of dates. Ours was stored in a variable called our_dates that included even dates from 2-30 (28 for february) for every month
- Call main using a name for the database in the parameter of database_name, a state code in the parameter state spot (two letter string, both lowercase letters), and a list of dates (in the format YYYY-MM-DD) - for us this was main('Covid_Temp_Animals.db', 'mi', our_dates)
- Each time you run the code, 25 lines of data will be added to the database named Covid_Temp_Animals.db into the table titled "Covid"
- Run this code as many times as necessary to get all the data from your dates list into the database (for us, this was 6 times because we had 134 dates in our list)

WEATHER CODE
- Call main using the name for a database (string format ending in .db) a city name in place of the region parameter (First letter capitalized, string format) and a list of dates (in the format YYYY-MM-DD). For us this was main('Covid_Temp_Animals.db', 'Detroit', our_dates)
- Same as above, each time you run the code, 25 data points will be added to 'Covid_Temp_Animals.db. This time, the data will be added to the database titled

"Temperature"

VETINARY CODE
***Actions italicized**

- *In the main function, located at the end of the file, comment out from* `data1 = get_graph_data(cur, conn)` *till the end of the file*
- *Press run, and the main function will run automatically*
    - Main function automatically creates databaseDict which is an empty dictionary
    - Main function automatically creates list of dates we will be pulling from API
    - Main function automatically will loop through the list of dates and pulls each data from API, filter data so that only the right parts of JSON is retrieved, and finally making dictionary that will be inserted into database
    - Then, functions are automatically called to connect to the database, create the Vetinary table, and 25 or less data points are added each time you run the code (run until all data points are added to to the database)
- *Then, uncomment what was commented earlier and run file again to get both graph figures*
    - Automatically gets graph data for the first visualization
    - Automatically gets first visualization
    - Automatically gets graph data for the second visualization
    - Automatically gets second visualization

COVID AND WEATHER COMBINATION CODE
This code is contained in the covid_by_day.py file
connect_temp_and_covid_by_date(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Initializes three dictionaries, all of them have the same four keys 'fall', 'winter', 'spring', and 'summer'. Their values are as follows
        - seasons_cases_dictionaries
            - integer, starts at 0
        - seasons_avg_temp_and_days
            - A list with two values, both initialized to 0, the first item is the total temperature, the second is the total number of days, these values will later be used to calculate the average temperature for each season
        - return_dict
            - A list with two values, both initialized to 0. The first item will be the total number of cases for that season, and the second will be the average temperature, rounded to two decimal places.
    - Defines each season by the two number string of the months
    - Loops through each tuple in the list returned by the join statement

        - For each tuple, the date is at index 0, the average temperature is at index one, and the number of cases is at index 2
- Returns
    - return_dict
        - A dictionary with seasons as the keys and a list as the value
        - The list has the total number of cases during that season as the first value (index 0) and the average temperature in Temperature, rounded to two decimal places as the second value (index 1).

Documentation for each function that we wrote (input/output/what it does):

Vetinary.py

(also contains code for running the code that combines and visualizes this data with weather data)

getData(search, dateStart, dateEnd, limitnum)

- Takes in
    - search - The variable of interest which in this case is the “original_receive_date” of a veterinary case
    - dateStart - the start date you want data from
    - dateEnt - the end date you want data from
    - limitnum - the number of data you want to retrieve, 1000 max
- Function
    - Queries data from API using API URL and requesting
    - Loads the json file as text and returns
- Returns
    - The API retrieved data json file loaded as text

filterData(jsonData)

- Takes in
    - jsonData - json text queried from API
- Function
    - Removing unnecessary data from the json file and only the needed “original_receive_date” dates
- Returns
    - A list of all dates from the jsonData input

makeDict(dateList, modifyDict)

- Takes in
    - List of dates dateList that is created in filterData
    - modifyDict which is a dictionary compiling the number of cases each day
- Function
    - Creating/modifying a dictionary (modifyDict) that goes through the list of dates and counts the number of times each date occurs (basically finding number of cases each day)
- Returns

    - modifyDict which includes the number of cases each day that is continually modified everytime makeDict is called

setupdb(dbname)

- Takes in
    - dbname, a database name in string ending in .db
- Function
    - Creates a database with a cur and conn, a cursor and a connection to the database
- Returns
    - Cur and conn, cursor and connection for SQL

makedb(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Create a table if it does not exist in the connected database
- Returns
    - Nothing

rowPosition(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Select all rows currently in Vetinary table and computes the length of the rows
- Returns
    - Length of rows in Vetinary table as an int

storedb(cur, conn, dataDict)

- Takes in
    - Cursor as cur and connection as conn
    - dataDict, a dictionary of keys being the dates and values being the number of cases in the corresponding date
- Function
    - Calls rowPosition to check the current number of rows in database
    - Creates a list of dataDict dictionary items
    - Calls for loop to that list, initially changes the date format from 2020401 to 2020-04-01
    - Based on the returned row number, adds a maximum of 25 rows into the database that is not a duplicate data
- Returns
    - Nothing

get_graph_data(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Joins table Vetinary and Temperature on the dates to get a list of tuples in format (date, midnight temperature, average temperature, cases)

        - Calculates absolute value of average temperature and midnight temperature to find daily temperature difference
    - Returns
        - Nested list with index 0 a list of calculated temperature difference every day in April, July, October, and December, and at index 1 a list of cases every day in April, July, October, and December

visualization1(data_list)

- Takes in
    - Data_list that was returned by get_graph_data that is a nested list with index 0 being the difference in daily temperatures for every day in April, July, October, and December and index 1 being the number of cases everyday in April, July, October, and December
- Function
    - Plots the data in data_list with index 0 being on x axis and index 1 being in y axis
- Returns
    - A visualization graph of scatterplot showing the number of Veterinary cases in April, July, October, and December of 2020 based on the daily temperature differences (Average Temperature - Midnight Temperature).

get_graph2_data(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Joins table Vetinary and Temperature on the dates to get a list of tuples in format (date, average temperature, cases)
    - Adds up the temperature and average it for April, July, October, and December
    - Adds up the number of cases and average it for April, July, October, and December
- Returns
    - A nested list with index 0 the average temperatures of April, July, October, and December and index 1 the average number of cases in April, July, October, and December

visualization2(list_data)

- Takes in
    - list_data that was returned by get_graph2_data that is a nested list with index 0 being a list of average temperatures each month and index 1 being a list of average cases each month
- Function
    - Plots the data in list_data, index 0 list being on the x axis and index 1 list being on the y axis
- Returns
    - A visualization graph of scatterplot showing the average number of Veterinary Cases in each season based on the average temperature of each season

create_file(file_name, data_dict)

- Takes in

    - file_name - what you want to name the text file to be named. This should be a string that ends in .txt
- Function
    - Opens the file in write mode
    - Uses json.dumps() to add the data in the dictionary to a text file
- Returns
    - Nothing, but it will create a new text file

covid_by_day.py

get_all_data(state_code, date_iso)

- Takes in
    - state_code- two letter, lowercase state code
    - date_iso- date in the format YYYY-MM-DD
- Function
    - Inserts date and state into the API link
    - Makes a call to the API
    - Turns the returned data into a JSON formatted string
- Returns
    - Data- formatted in JSON

get_important_data (data)

- Takes in
    - Data, formatted in json format (from get_data)
- Returns a list
    - Index 0 = state
    - Index 1 = date
    - Index 2 = number of cases (already integer)
    - OR - if date not in database it will return an empty list

state_data(state, date_list)

- Takes in
    - State code, list of dates
- Function
    - Calls get_data then calls get_important_data
    - Loops through all the dates in date_list
    - For each date:
        - Creates a dictionary with date as the key and a smaller dictionary as the value
            - Within the smaller dictionary, the key is the state, and the value is the number of cases in that state on that date
    - It checks to make sure that the date is not already in the dictionary, and if it is not then it adds it to the larger dictionary
- Returns single_date_data_dict - a dictionary of all the dates in date list as keys for a state that gives cases of covid per day

setUpDatabase (db_name)

- Takes in

    - Db_name = a string that ends in .dp - name of the database that you want the table data will be saved to
- Function
    - Creates a database with a cur and conn, a cursor and a connection to the database
- Returns
    - curr and conn

create_table(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Creates a table named Covid if it does not already exist
    - Adds the following into the table:
        - Date as the primary key
        - State as text
        - Cases as an integer
    - Commits these into the table
- Returns
    - Nothing

check_rows(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Selects all of the rows from the table Covid
    - Use fetchall with cur to return the data we selected
- Returns
    - The length of the fetchall() statement
        - This is the number of rows currently in the table

add_data_to_table(big_data_dict, cur, conn)

- Takes in
    - big_data_dict
        - A large dictionary that was created by state_data() that consists of dates as keys and a small dictionary in the format {state: cases} as the value
    - Cursor as cur and connection as conn
- Function
    - Calls check_rows to check how many rows are currently in the table, this is saved as current_rows
    - Creates a variable called target_rows that is 25 more than the value of current_rows
    - Checks to make sure that target rows is not larger than the amount of data we have available

        - If it is, it is set to the length of our data so it will finish uploading all of the data we have, but nothing more
    - Turns the data dictionary into a list of the dictionary items
    - Loops through the range of numbers from current_rows to target_rows
        - Uses indexing in this loop to access each dictionary
        - The key at each index is the date
        - Pulls the smaller dictionary (values) and sets the key to the state and the value to the number of covid cases
        - Sets variables equal to the states (string in 2 letter, lowercase format), number of covid cases (integer), and date (string in YYYY-MM-DD format)
    - Inserts data into the table we've created
- Returns
    - Nothing- adds information into the table in the database

connect_avg_temp_and_covid_by_date(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Joins the Temperature and Covid tables where the dates are the same
    - Selects the average temperature, date, and number of covid cases
    - Creates an empty dictionary (seasons_cases_dict) to keep track of the number of cases for each season
    - Creates a return dictionary (return_dict) with the seasons as keys and the value a list of two numbers, the first being cases and the second being average temperature for that season
    - Uses a dictionary called seasons_avg_temp_and_days, with keys as the seasons and the values as two numbers used to calculate the average temperature for the return dictionary. One keeps track of the total temperature, the other keeps track of the days (total temp / days = avg temp)
    - Lists for each season that contains the months in a list
    - Loops through the selected data in the table
        - Sets date, avg temp, and cases based on their position in the tuple
        - Uses date to find month
    - Checks to see what season the month is in then adds to the above dictionaries accordingly
- Returns
    - Return_dictionary keys = seasons, value = list of two numbers(first is number of covid cases, second is the average temperature)

connect_normal_temp_and_covid_by_date(cur, conn)

- Same as the above function, but instead of using the average temperature throughout the day, it uses the temperature taken at midnight.

visualization

- Takes in season_dict
    - A dictionary returned by either connect_normal_temp_and_covid_by_date or from connect_avg_temp_and_covid_by_date
- Function
    - Creates lists from the dictionary for number of covid cases, average temp, and season
    - Uses indexing to loop through this data and add it onto the table
    - Labels axes and points for clearer understanding of the data
- Returns
    - Nothing, but creates a scatter plot with number of covid cases on the Y axis, temperature on the X axis, and different points on the graph for each season (each season has a different color and those colors are in the key)

visualization2

- Does the same as visualization, but instead takes in the averages from temperature at midnight, and labels the title accordingly

main(database_name, state, date_list)

- Takes in
    - database_name - what you want to name the database that will be created (for our project, we used 'Covid_Temp_Animals.db') - should be a string, and should end with db
    - state - a string for any state in the US, should be the two letter code for the state (both letters lowercase)
- Function
    - Gets data on the state and for the dates you inputted by calling full_data_in_list()
    - Creates cur and conn by calling setUpDatabase and inputting the database name you provide to create a file in db browser
    - Creates a table using cur and conn to be within that database
    - Adds the data for the state and dates provided into the table in the database (uses add_data_to_table to do this)
    - Creates two visualizations by calling connect_avg_temp_and_covid_by_date and connect_normal_temp_and_covid_by_date then inputting those into the functions visualization and visualization2
- Returns
    - Nothing, executes the above code

weather.py

get_data(region, date_iso)

- Takes in
    - region as a major US city (for our data, we pulled from Detroit) the format of this is a string with the first letter capitalized
    - date_iso as a date in a string format of YYYY-MM-DD

- Function
    - Inserts the region and date into the API link in order to make the request
    - Pulls from the API, turns it into text, in json format
- Returns
    - data- a json string of all the data pulled for that city and date

get_specific_data(region, dates_list)

- Takes in
    - region, same as above (a major US city with the first letter capitalized, ours was Detroit)
    - dates_list = a list of dates in the format YYYY-MM-DD
- Function
    - Creates a dictionary called big_dict
    - Loops through dates_list
    - Creates a list called info_list (will start empty at the beginning of each loop)
        - Calls the API for each date and gathers data values for the date, state, and temperature at the zero hour (which is at midnight), and average temperature in that area on that day.
        - Adds that data to info_list
        - Adds a key, value pair to the dictionary with the date as the key and the list of date, state, temp at the zero hour (midnight), and average temp as the value.
- Returns
    - big_dict- a dictionary with all of the dates as keys and info lists as values.

setUpDatabase(database_name)

- Takes in
    - database_name = a string that ends in .db that the table data will be saved to
- Function
    - Creates a database with a cur and conn, a cursor and a connection to the database
- Returns
    - curr and conn

create_table(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Creates a table named Temperature if it does not already exist
    - Adds the following into the table:
        - Date as the primary key
        - State as text (full name of the state, first letter capitalized)

        - avg_temp as an integer
        - temp as an integer (this is the temperature at the zero hour (midnight))
    - Commits these into the table
- Returns
    - Nothing

check_rows(cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Selects all of the rows from the table Temperature
    - Use fetchall with cur to return the data we selected
- Returns
    - The length of the fetchall() statement
        - This is the number of rows currently in the table

addition_data(data_dict, cur, conn)

- Takes in
    - Cursor as cur and connection as conn
- Function
    - Calls check_rows to check how many rows are currently in the table, this is saved as curr_rows
    - Creates a variable called target_rows that is 25 more than the value of curr_rows
    - Checks to make sure that target rows is not larger than the amount of data we have available
        - If it is, it is set to the length of our data so it will finish uploading all of the data we have, but nothing more
    - Turns the data dictionary into a list of the dictionary items
    - Loops through the range of numbers from current_rows to target_rows
        - Uses indexing in this loop to access each dictionary
        - The key at each index is the date
        - Pulls the list from inside the dictionary (the value) and sets the following:
            - Index 0 is the state
            - Index 1 is the average temp
            - Index 2 is the temp at midnight (the zero hour)
        - Sets variables equal to the states (full name of state as a string, first letter capitalized), avg_temp (integer in degrees fahrenheit), temp at midnight (the zero hour) (integer in degrees fahrenheit) and date (string in YYYY-MM-DD format)
    - Inserts data into the table we've created
- Returns
    - Nothing- adds information into the table in the database
    -

create_file(filename, data_dict)

- Takes in
    - filename - what you want to name the text file to be named. This should be a string that ends in .txt
- Function
    - Opens the file in write mode
    - Uses json.dumps() to add the data in the dictionary to a text file
- Returns
    - Nothing, but it will create a new text file

main(database_name, region, date_list)

- Takes in
    - database_name - what you want to name the database that will be created (for our project, we used 'Covid_Temp_Animals.db') - should be a string, and should end with db
    - region - a string of a major city in the US, the first letter should be capitalized
    - date_list - a list of dates that you want to gather data from
- Function
    - Gets data on the city and for the dates you inputted by calling get_specific_data()
    - Creates cur and conn by calling setUpDatabase and inputting the database name you provide to create a file in db browser
    - Creates a table using cur and conn to be within that database
    - Adds the data for the city and dates provided into the table in the database
    - Creates a text file with the data calculations by calling create_textfile
- Returns
    - Nothing, executes the above code

Resources we used:

Date	Issue Description	Location of Resource	Result (did it solve the issue)
11-30-2022	Figuring out how to add more than one key, value pair into a dictionary (wanted to use append)	https://www.w3schools.com/python/ref_dictionary_update.asp	Yes, using the update feature, I was able to add multiple dictionaries as values to create a nested dictionary.
12-02-2022	Trying to figure out how to loop through the dictionary to access all of the items in the same way that we can	https://realpython.com/iterate-through-dictionary-python/#iterating-through-items	Yes, using this website I was able to figure out how to loop through a dictionary and realized I needed to use .items(),

	loop through lists and use indexing		save that to a list and then index through that.
12-08-2022	Adding labels to the points on the scatter plot and creating a legend	https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.annotate.html	Yes, we were able to use .annotate() to add labels to our points to make the scatter plot more readable
12-08-2022	Adding a side comment on the scatter plot and working with the sizing/positioning of the visualizations	https://matplotlib.org/stable/tutorials/text/text_props.html	Yes, we were able to find out how to add in a side comment on the scatter plot and we were also able to learn how to position the certain elements onto the scatter plot

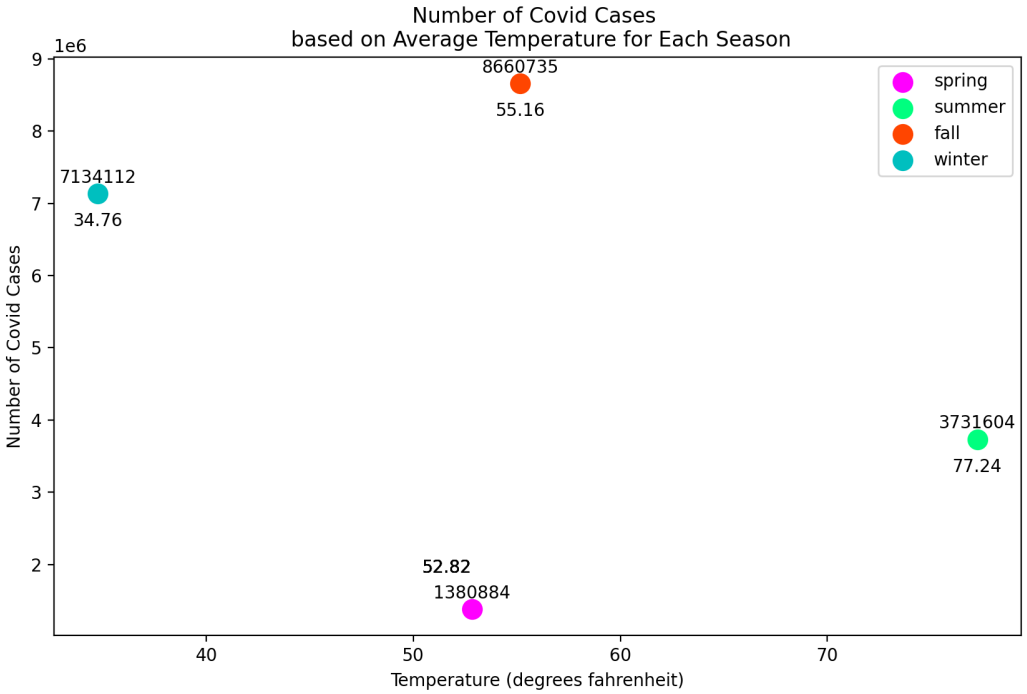
Visualizations & Output that we created:

Output 1:

Avg_Temp_vs_Covid.txt

```
1   {"spring": [1380884, 52.82], "summer": [3731604, 77.24], "fall": [8660735, 55.16], "winter": [7134112, 34.76]}
```

Visualization 1:



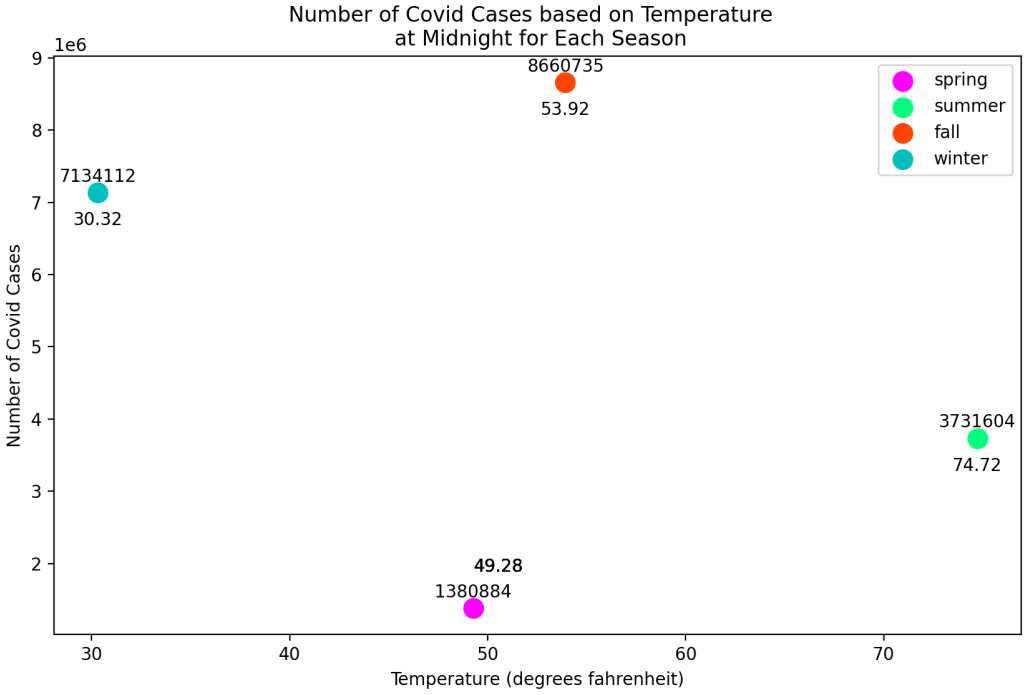
Scatterplot showing the number of Covid-19 cases based on the average temperature (calculated throughout the whole day) for each season.

Output 2:

Normal_Temp_vs_Covid.txt

```
1   {"spring": [1380884, 49.28], "summer": [3731604, 74.72], "fall": [8660735, 53.92], "winter": [7134112, 30.32]}
```

Visualization 2:

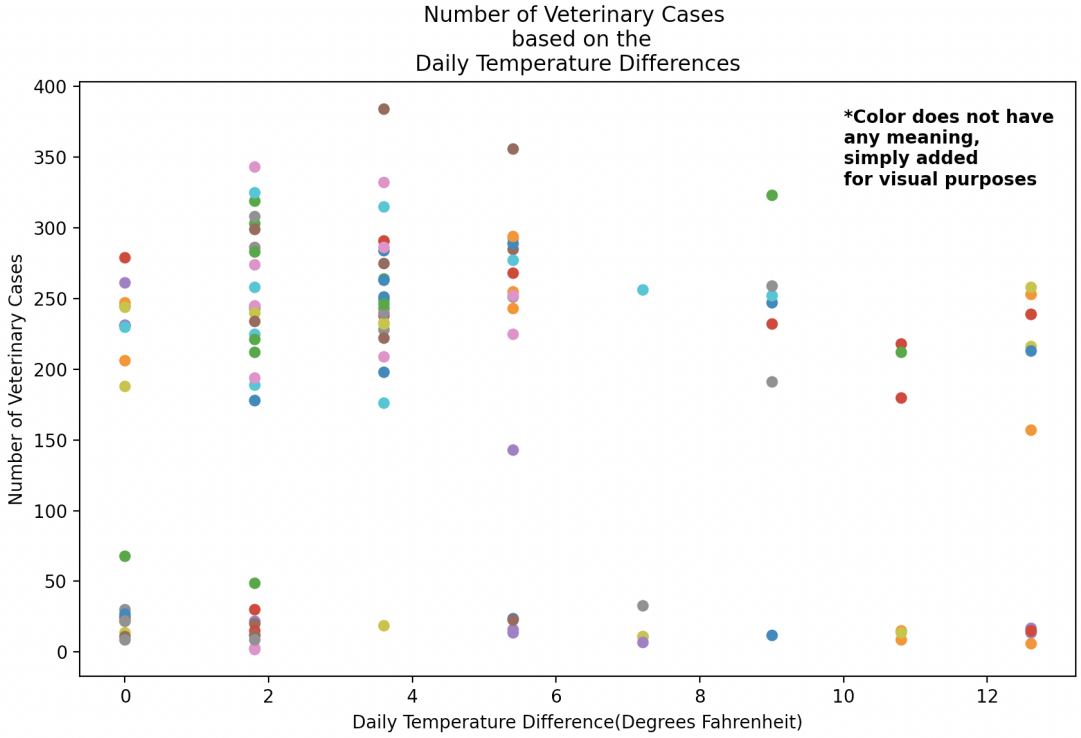


Scatterplot showing the number of Covid-19 cases based on the average temperature (taken at midnight) for each season.

Output 3:

```
[[0.0, 12.6, 1.8, 1.8, 12.6, 5.4, 5.4, 3.6, 12.6, 1.8, 5.4, 10.8, 1.8, 3.6, 0.0, 3.6, 3.6, 7.2, 0.0, 5.4, 9, 5.4, 1.8, 9.0, 1.8, 1.8, 5.4,
5.4, 3.6, 1.8, 5.4, 5.4, 1.8, 1.8, 5.4, 3.6, 3.6, 1.8, 3.6, 7.2, 0, 1.8, 1.8, 3.6, 0.0, 1.8, 5.4, 0.0, 3.6, 1.8, 3.6, 5.4, 3.6, 5.4, 5.4,
0.0, 1.8, 1.8, 3.6, 3.6, 3.6, 0.0, 3.6, 0.0, 7.2, 5.4, 3.6, 3.6, 0.0, 9.0, 0.0, 0, 9.0, 0.0, 1.8, 3.6, 1.8, 0, 7.2, 3.6, 3.6, 12.6, 3.6,
10.8, 12.6, 1.8, 1.8, 9.0, 1.8, 1.8, 3.6, 10.8, 1.8, 12.6, 0.0, 3.6, 1.8, 0.0, 12.6, 0.0, 12.6, 0, 1.8, 12.6, 7.2, 1.8, 1.8, 9.0, 0.0, 1.
8, 9.0, 12.6, 10.8, 10.8, 5.4, 5.4, 1.8, 1.8, 10.8, 3.6, 1.8, 12.6, 0], [231, 253, 212, 30, 14, 285, 225, 228, 216, 189, 24, 15, 303, 238,
245, 249, 209, 33, 14, 277, 247, 243, 243, 232, 22, 20, 292, 251, 232, 258, 289, 255, 49, 15, 14, 384, 332, 286, 291, 256, 24, 12, 319,
291, 261, 299, 252, 30, 19, 325, 284, 294, 243, 268, 16, 11, 343, 308, 233, 250, 251, 247, 264, 25, 11, 356, 286, 240, 244, 252, 27, 22,
323, 279, 245, 275, 245, 22, 11, 315, 263, 239, 246, 218, 17, 12, 274, 259, 240, 225, 198, 9, 283, 239, 231, 222, 3, 9, 258, 230, 213,
206, 221, 15, 7, 234, 194, 191, 188, 178, 12, 6, 212, 180, 143, 23, 2, 9, 14, 176, 178, 157, 68]]
```

Visualization 3:

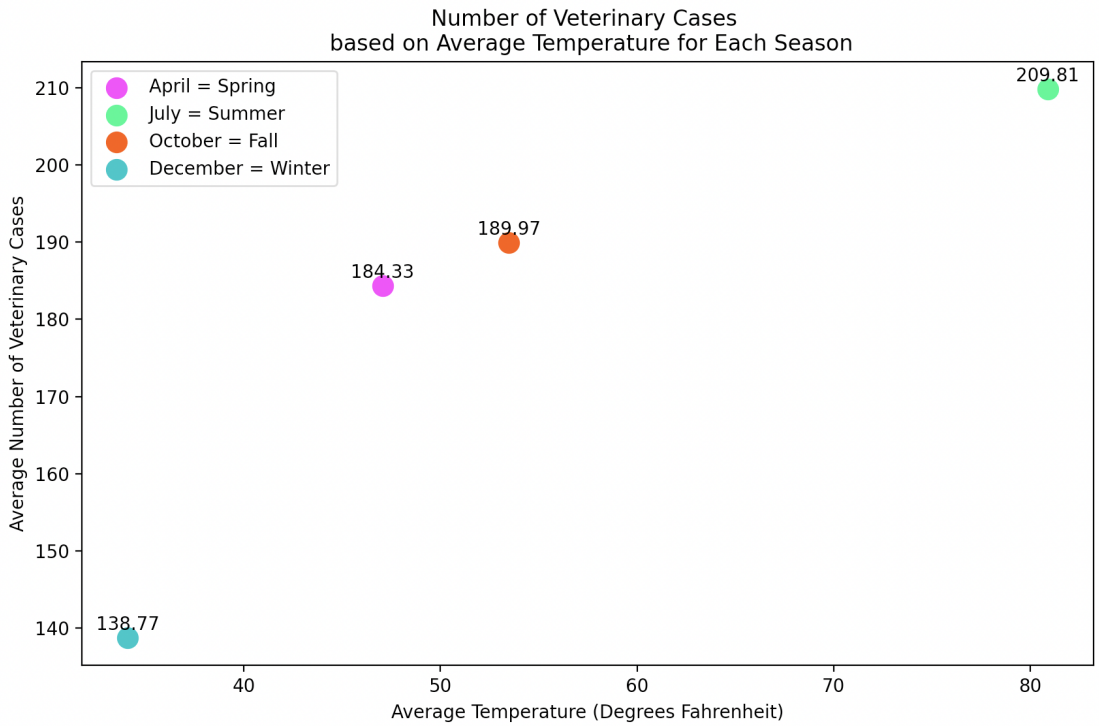


Scatterplot showing the number of Veterinary cases in April, July, October, and December of 2020 based on the daily temperature differences (Average Temperature - Midnight Temperature).

Output 4:

```
1    [[47.06, 80.89, 53.48, 34.09], [184.33, 209.81, 189.97, 138.77]]
```

Visualization 4:



Scatterplot showing the average number of Vetinary Cases in each season based on the average temperature of each season.