Jason Kemp, Christopher Sayah
The SQL Injectors

Final Project Report

GitHub Link: https://github.com/sayahchris/206_final_project

1) **The goals for your project (10 points)**
    a) This project was oriented toward cross-analyzing the relationship between temperature and total points scored in football and baseball games. We set out to find if there is a correlation between these variables in order to determine if the outdoor temperature has an impact on the overall point count within sports games.
    b) Specifically, we were conducting this analysis in regard to football and baseball games played by the University of Michigan's team over the past decade.

2) **The goals that were achieved (10 points)**
    a) Our end goal throughout this project was partially achieved. Upon analysis of the final data, we were able to notice a significant correlation between temperature and total points scored within a football game. Typically, as the temperature increased, the points scored also increased. As the temperature decreased, the contrary was true.
    b) Although we were able to achieve one of our goals, we were unable to conduct the same analysis on baseball games due to the lack of functionality associated with the APIs we were attempting to pull data from.

3) **The problems that you faced (10 points)**
    a) One of the main problems that we faced was an elongated request timeframe while attempting to execute our code. It often took upwards of 15 minutes to collect the data in whole, which made for challenges while testing the code throughout the process. We were able to solve this problem by directly inserting the data into the code for testing purposes.
    b) Another problem that we faced was the fact that we were unable to find a functional—for our purposes—API associated with baseball statistics. We attempted to switch from a baseball API to a cryptocurrency API, however we were also unsuccessful in the same manner. Due to this dilemma, we simply focused on the football and weather API.

4) **Your file that contains the calculations from the data in the database (10 points)**
    a) See "data_output.txt" file. This file should appear on the desktop when running our code.

5) **The visualization that you created (i.e. screenshot or image file) (10 points)**
    a) See "visualizations.pdf" file. When running the code, four of the visualizations will appear as images. The other two visualizations will appear in your browser.

6) Instructions for running your code (10 points)

a) There are two ways to run the code. Due to the lengthy time frame (approximately 15 minutes) required in order to directly request the data and run the code, we have inputted the data on line 85 (data = ([‘2009-09-05’…) to make the process faster. This specific action was done simply to save time, however the code can be executed another way.

   i) In order to utilize the faster process, simply comment lines 11 (def get_michigan_date…) through 84 (data = data_storing…) and uncomment line 85 ([‘2009-09-05’…).

b) If direct requests are preferred, ensure that line 85 (data = ([‘2009-09-05’…) is commented and that lines 11 (def get_michigan_date…) through 84 (data = data_storing…) are uncommented.

   i) Press the play button to run the code and the database will begin to fill with the first 25 requests.
   ii) Upon completion of the first cycle, the database will be created and filed with the information provided by the initial 25 requests.
   iii) Run the code three more times in order to fully fill up the database with the subsequent information.
   iv) After executing the code, a text file will appear on the desktop depicting the calculations made.
   v) Additionally, the visualizations will pop up one by one. Four visualizations will show in a preview screen, and the following two will populate in a browser window.

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

Function Name	Input	Output
get_michigan_date	- Takes year and week from a football season - Uses information to pull data about the University of Michigan’s football game history - Utilizes the “start_date”	- After inputting the week and year that you want to search for, we are able to access a json response and see information - Specifically, this function returns the date that the game in question was played on
get_michigan_score	- Similarly to the function before, the week and year of a University of Michigan’s football game is the input	- This function calculates the total points score in a game by adding the home score and the away score

		- The total score is the output for this function
get_michigan_temp	- In order to maintain consistency, this function intakes the week and year of a football game - Utilizes get_michigan_date function, and searches for the weather in Ann Arbor on that given date	- The output of the function is the average temperature in Ann Arbor on the given week and year that was imputed
data_storing	- There is no input for this function - This function is used to gather all of the data from the above three functions for the years between 2009 and 2019 - Because of bye weeks in football, we had to utilize the try/except functions to ensure that we did not get errors once a year for the bye weeks - In addition, we stored all of the information into a tuple that way we could easily access the data when putting it into the database	- The output of this function is a tuple that has 100 dates, 100 temperatures, and 100 scores - The tuple is returned as final_tuple
setUpDatabase	- Establishing database titled “final_project_1.db”	- “final_project_1.db” database established
create_date_table	- Creating “Dates” table (columns: id, date) - Ensuring only 25 requests are conducted per run (count and if/else statement) - Adding information to columns in “Dates” table (id and date values)	- “Dates” table created within database that displays two columns (id and date) - Information is filtered into each column upon execution of code (limited to 25 per run)
create_weather_table	- Creating “Temperature” table (columns: id, temp) - Ensuring only 25 requests are conducted per run (count and if/else statement) - Adding information to	- “Temperature” table created within database that displays two columns (id and temp) - Information is filtered into each column upon execution of code (limited to 25 per run)

	columns in “Temperature” table (id and temp values)	
create_game_table	- Creating “Scores” table (columns: id, scores) - Ensuring only 25 requests are conducted per run (count and if/else statement) - Adding information to columns in “Scores” table (id and score values)	- “Scores” table created within database that displays two columns (id and score) - Information is filtered into each column upon execution of code (limited to 25 per run)
join_tables	- Joining scores and temperature tables by utilizing “ID” category as key - Developing list based upon joined data	- Associated temperatures and scores in accordance with matching “ID” keys - Developed and returned list based upon joined data
writing_file	- Creating and saving a text file named “data_output.txt” in “write” format to desktop - Calculating mean of scores from indexed data - Calculating mean of temperatures from indexed data - Writing calculations to text file with both outputs	- Created and saving a text file named “data_output.txt” in “write” format to desktop - Places into text file “Mean of Temperatures” and resulting calculation - Places into text file “Mean of Scores” and resulting calculation
plotly_Scores_Vs_temp	- Creating graph with scores on x-axis and temperature on y-axis - Adding and comparing collected data regarding scores and temperatures - Adding a line of best fit to regression in accordance with data	- Graph created with scores on x-axis and temperature on y-axis - Data filtered into graph to show collected scores and temperature - Data shows negative linear relationship between scores and temperatures
make_hist_score	- Creating histogram with “Scores” on x-axis and “Number of Times” on y-axis - Adding collected data regarding scores	- Histogram created with “Scores” on x-axis and “Number of Times” on y-axis - Data filtered into histogram to show collected scores - Data shows standardized residual regression

make_mean_hist_score	- Creating histogram with "Mean of Scores" on x-axis and "Number of Times" on y-axis - Adding mean data regarding scores	- Histogram created with "Mean of Scores" on x-axis and "Number of Times" on y-axis - Data filtered into histogram to show mean of scores - Mean of scores is slightly above 50
make_mean_hist_temp	- Creating histogram with "Mean of Temperatures" on x-axis and "Number of Times" on y-axis - Adding mean data regarding temperatures	- Histogram created with "Mean of Temperatures" on x-axis and "Number of Times" on y-axis - Data filtered into histogram to show mean of temperatures - Mean of temperatures is slightly below 60
make_hist_temp	- Creating histogram with "Temperatures" on x-axis and "Number of Times" on y-axis - Adding collected data regarding temperatures	- Histogram created with "Temperatures" on x-axis and "Number of Times" on y-axis - Data filtered into histogram to show collected temperatures - Data shows standardized residual regression
make_plotly	- Creating graph with years on x-axis and temperature/points on y-axis - Adding and comparing collected data regarding scores (blue) and temperatures (red)	- Graph created with years on x-axis and temperatures/points on y-axis - Data filtered into graph to show collected temperatures and scores - Data shows typically standardized residual regression on an annual basis
main	- Does not have any inputs	- Stores all functions and is called to run all functions - Does not have an output

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

Date	**Issue Description**	**Location of Resource**	**Result (Did it solve the issue?)**

November 20, 2022	- API that allows us to access data about the weather	https://weatherstack.com/login?u=https%3A%2F%2Fweatherstack.com%2Fdashboard	- N/A
November 20, 2022	- API that allows us to access data about University of Michigan's football games	https://api.collegefootballdata.com/api/docs/?url=/api-docs.json#/	- N/A
November 28, 2022	- Taught us how to effectively join tables within our database	https://www.w3schools.com/python/python_mysql_join.asp	- Issue was resolved
November 28, 2022	- Taught us how to effectively join tables within our database	https://medium.com/analytics-vidhya/introduction-to-sql-using-python-using-join-statements-to-merge-multiple-tables-3b3513fe2c98	- Issue was resolved
November 28, 2022	- Helped us insert data into specific tables within database	https://www.w3schools.com/sql/sql_insert.asp	- Issue was resolved
December 6, 2022	- Taught us how to utilize the "mean" function in our code	https://www.geeksforgeeks.org/python-statistics-mean-function/	- Issue was resolved
December 8, 2022	- Taught us how to utilize "matplotlib" within our code	https://matplotlib.org/	- Issue was resolved
December 8, 2022	- Taught us how to utilize "matplotlib" within our code	https://www.w3schools.com/python/matplotlib_pyplot.asp	- Issue was resolved
December 8, 2022	-- Taught us how to utilize "plotly" within our code	https://plotly.com/python/line-and-scatter/	- Issue was resolved
December 8, 2022	- Taught us how to utilize "plotly" within our code	https://plotly.com/python/getting-started/	- Issue was resolved