Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project

Project Goals and Achievements

Goals: We originally planned to work with Weatherstack, Calendarific, and Spotify API, with plans to gather daily average temperature, a list of holidays, and how often certain song genres were played per day. We intended to look into listening trends and see if they had any relation to the weather and/or holidays.

What was achieved: While we still used Weatherstack and Calendarific, we switched to using OpenAQ and looking into air quality (in particular, ozone levels) in Ann Arbor from roughly the past three months. We gathered the daily average temperature and UV level (Weatherstack), a list of all the holidays celebrated in Ann Arbor, and the daily average ozone levels. We were able to create graphs from our data and observe certain trends between the variables.

Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project

Problems

We had difficulties figuring out how to not only store 25 table entries at a time, but how to add on a new 25 entries in the table during continued runs of the code. We were able to figure out how to do this with a combination of ChatGPT and searching how to use datetime objects/timedelta in Python, which allowed us to automatically store 25 new entries from within our specified time period each time we ran the code.

During the project, a database table error occurred where the error no such table: holidays appeared because the holidays table was missing or not initialized properly. This issue was resolved by adding an initialize_database function, which ensures that both tables (holidays and holiday_types) are created before inserting data into the database.

Another issue encountered was related to the database file path, where the database file could not be opened due to incorrect or relative paths. To resolve this, Python's os.path.abspath() was used to specify the absolute path to the database file, ensuring the script could locate and access the correct database file regardless of the working directory.

Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project

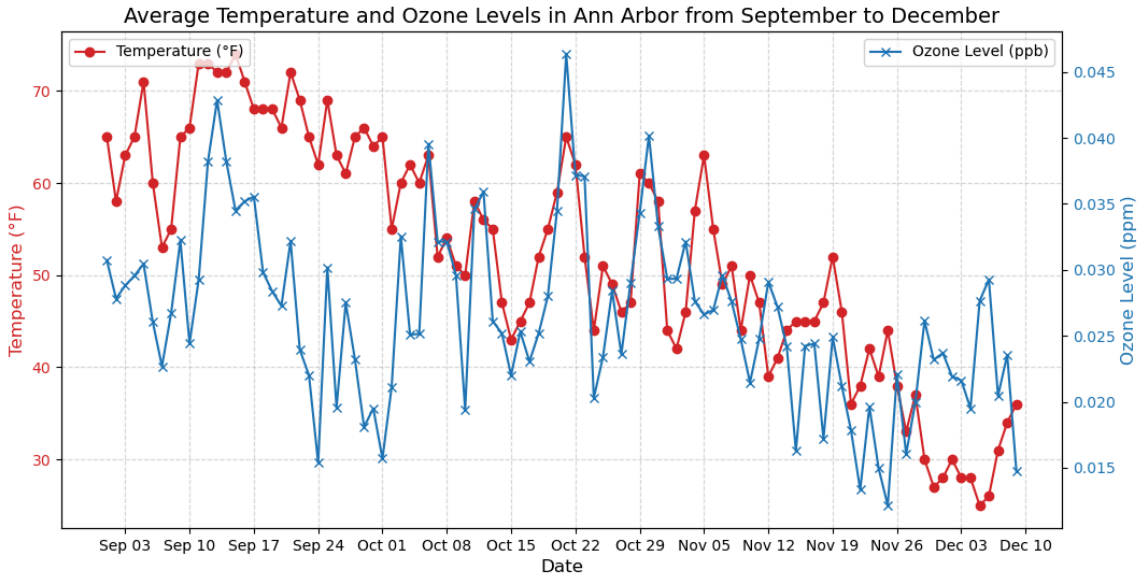
Calculations

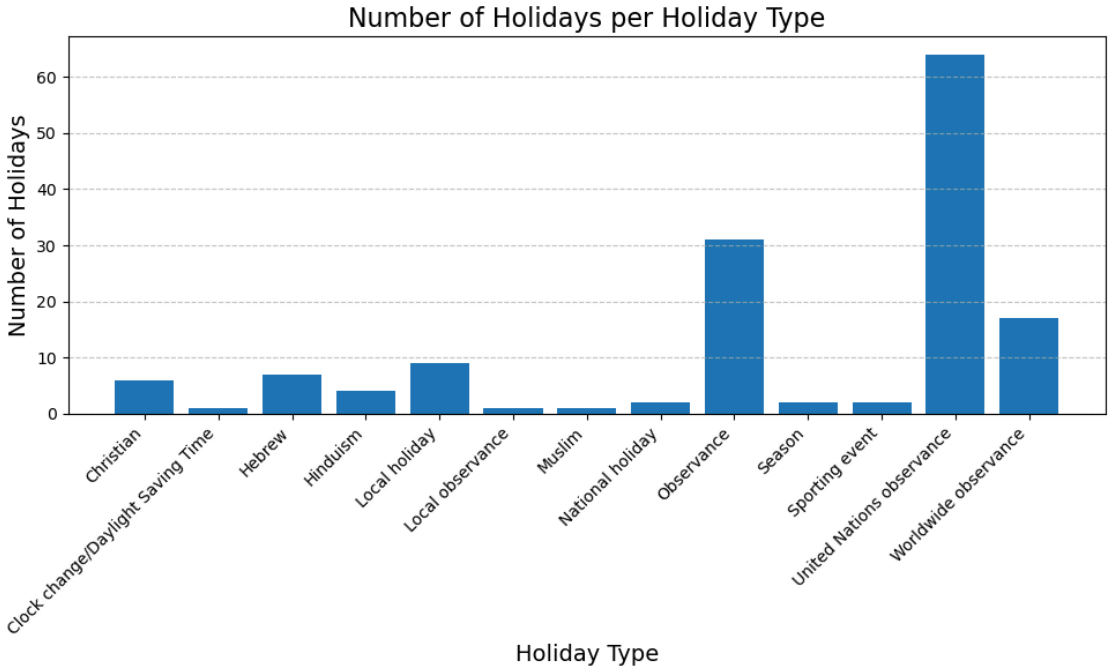
weekly_averages.csv > data

```
week,avg_temp,avg_ozone,holiday_count
2024-09-01,62.14,0.028,9
2024-09-08,68.0,0.0331,10
2024-09-15,69.57,0.0318,11
2024-09-22,64.86,0.0231,13
2024-09-29,61.71,0.0225,13
2024-10-06,54.86,0.0319,15
2024-10-13,49.14,0.025,11
2024-10-20,54.57,0.0325,6
2024-10-27,51.14,0.0313,12
2024-11-03,52.14,0.0279,8
2024-11-10,44.43,0.0239,8
2024-11-17,43.71,0.0198,9
2024-11-24,35.43,0.0192,9
2024-12-01,28.0,0.0234,12
```

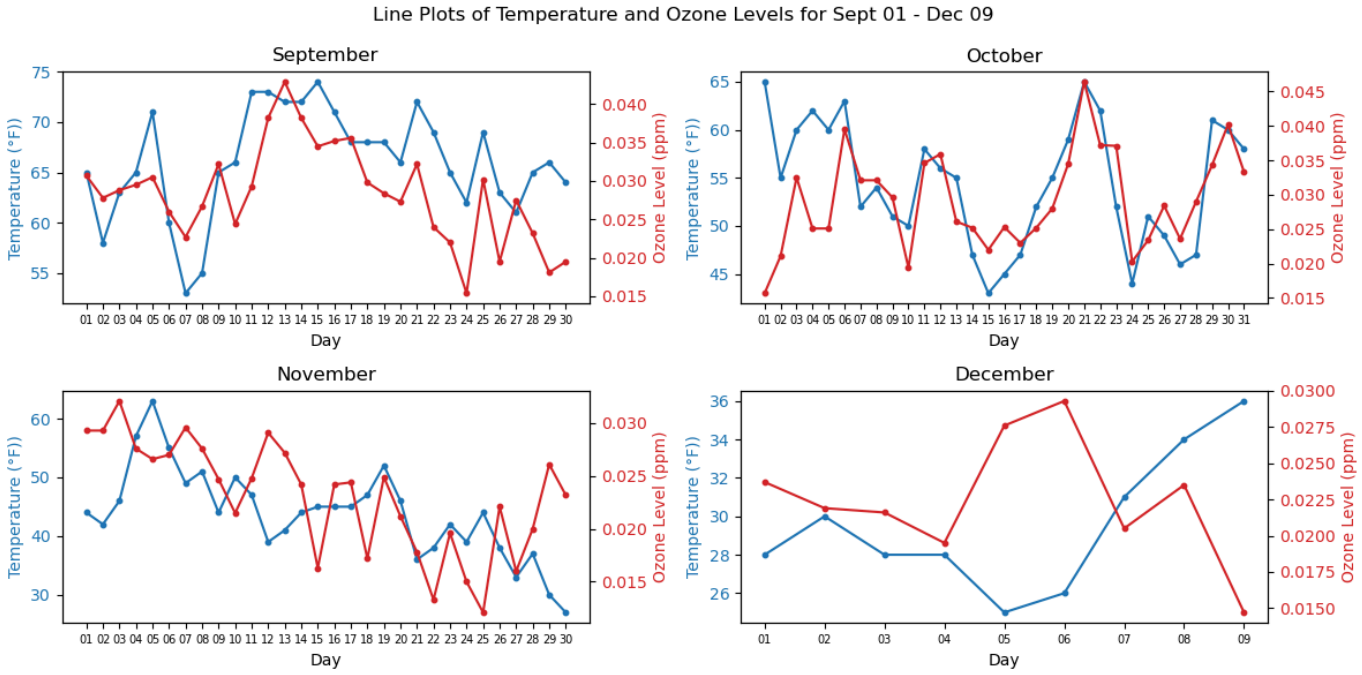
Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project

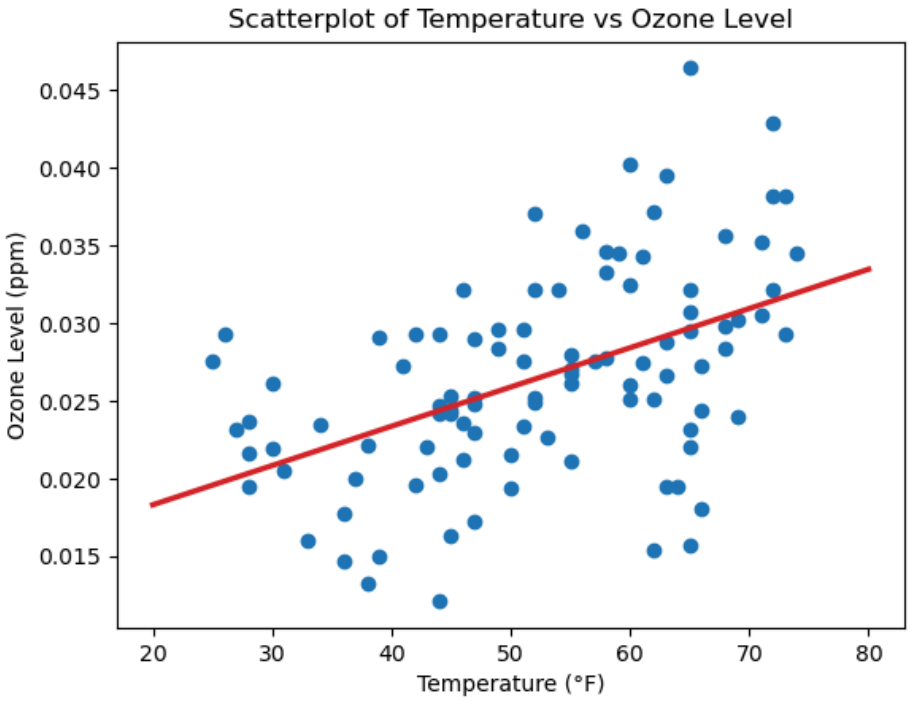
Visualizations



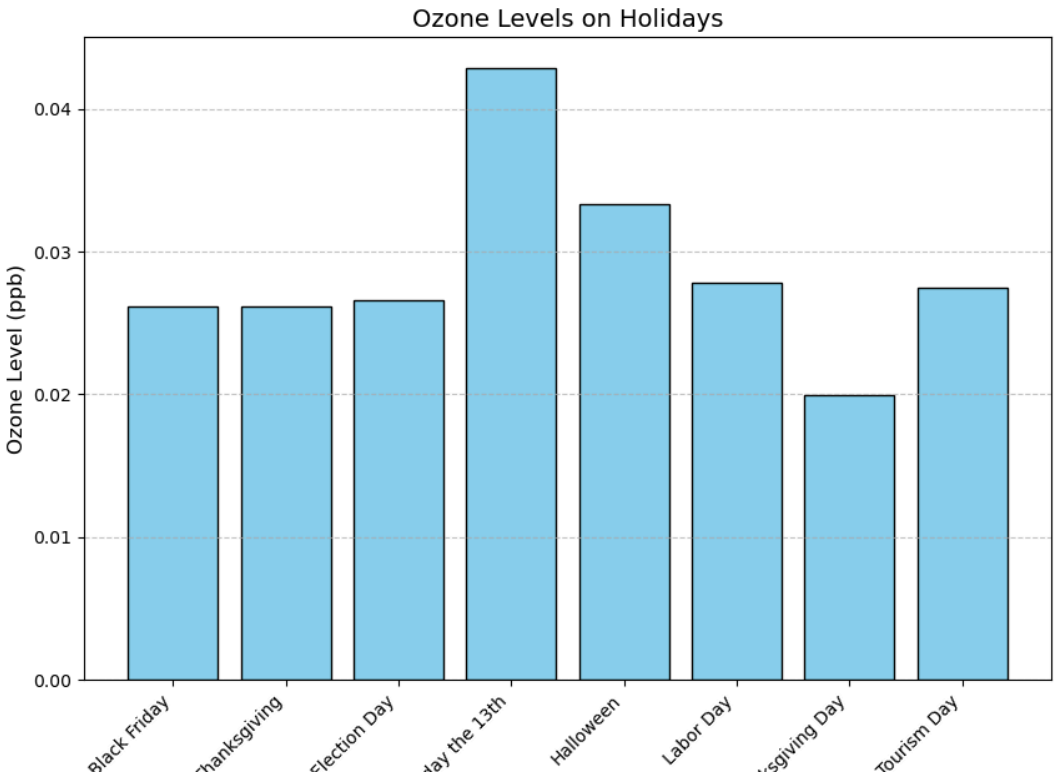


Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project





Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project



Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project

Instructions

If “database.db” does not exist, run main.py with an input of “1” six times in order for all the necessary data to be inserted into the tables. Then, input “2” to see the visualizations and “3” to create and complete the csv file with calculations (if not already created).

Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project

Function Documentation

main.py

- create_db()
    - Requires: Nothing
    - Returns: None
    - Function: Sets up tables within the database (i.e. creates tables and inserts data)
- visualizations()
    - Requires: Nothing
    - Returns: None
    - Function: calls main() functions from other Python files containing code to run visualizations. Includes temperature and ozone vs date line plot (across the months), holiday distribution bar graph, temperature and ozone vs date line plot (per month), temperature vs ozone scatterplot, and ozone levels on certain holidays bar graph.
- calculation()
    - Requires: Nothing
    - Returns: None
    - Function: calls main() function from calculations.py, which contains code to create and input data into a csv file.
- main()
    - Requires: Nothing
    - Returns: None
    - Function: Requests user input to either 1. set up the database/tables, 2. run the visualizations, or 3. run the calculations.

temp_and_ozone.py

- create_table()
    - Requires: Nothing
    - Returns: None
    - Function: Creates temp_and_airquality table in the database if it does not exist; creates date (as an id referencing a different table for the string value), average temperature (in °F), UV index, and average ozone level (in ppm) columns and ensures that entries are unique. Also creates dates table in database if it does not exist, which stores unique ids linked to string dates.
- insert_data(date_id, avg_temp, uv_index, avg_ozone)
    - Requires: date (str), average (int), parameter (str), units (str)
    - Returns: Nothing
    - Function: Takes in the date id, average temperature, UV ndex, and average ozone level and inserts it as an entry in the emp_and_airquality table.

- get_city_id()
    - Requires: Nothing
    - Returns: id (int)
    - Function: Makes an API request to retrieve the city id for Ann Arbor, Michigan.
- fetch_data(location_id)
    - Requires: location_id (int)
    - Returns: None
    - Function: Fetches 25 new entries in chronological order between the start and end dates specified in the file (inclusive). For each entry, it calls process_data(data) to reformat the entry and insert it into the table.
- process_data(data):
    - Requires: data (dict)
    - Returns: None
    - Function: Extracts values for date, average temperature, UV ndex, and average ozone level from the two API responses. Inserts date information into the dates table if necessary, and finds the date id. Then, calls insert_data(date_id, avg_temp, uv_index, avg_ozone) to insert the entry into the table.
- main(db)
    - Requires: db (str)
    - Returns: None
    - Function: Sets up database connection and cursor (both global). Calls create_table() to create the air_quality table in the database, calls get_city_id() to get the id number for Ann Arbor, Michigan, and then calls fetch_data(id) to begin storing entries into the table.

holiday.py

- initialize_database(db_name)
    - Requires: db_name(str) - Path to the SQLite database file
    - Returns: None
    - Function: Creates two tables in the database if they do not already exists:
      - holiday_types: stores unique holiday types
      - holidays: stores details(name, type, country, state, and date id) and references holiday_type using the key

- fetch_holidays(api_key, country, state, year, month)
    - Requires: api_key(str): API key for the Calendarific API
    - country(str): country code (“US”)
    - state(str): state code (“us-mi”)
    - year(int): year for which holidays are fetched
    - month(int): month for which holidays are fetched
    - Returns: A list of holiday dictionaries from the API response

  - Function: Makes an API request to fetch holiday data for a specific country, state, year, and month - handles any requests errors and returns the holidays list

- store_holidays_in_db(holidays, db_name, state)
  - Requires: holidays (list): list of holiday dictionaries fetched from the API
  - db_name(str): path to the SQLite database file
  - State (str): state code where the holidays apply("us-mi")
  - Returns: An integer representing the number of rows successfully inserted into the database
  - Function: Inserts holiday types into the holiday_types table (ensures no duplicates using INSERT OR IGNORE).
  - Fetches the corresponding holiday_type_id for each holiday type.
  - Inserts holiday details (name, type, country, state, and date) into the holidays table.
  - Limits the total inserted rows to 25 to adhere to the defined HOLIDAY_LIMIT.

- main(db)
  - Requires: db
  - Returns: none
  - Function: Calls initialize_database() to set up the database structure.
  - Iterates over the months September to December.
  - Calls fetch_holidays() to retrieve holiday data for each month.Calls store_holidays_in_db() to insert the data into the database, limiting the total to 25 rows.
  - Displays progress and final count of holidays stored in the database.

holidayvisual.py

- load_holiday_data(db_path)
  - Requires: db_path (str): the path to the SQLite database file
  - Returns: type_name: the name of the holiday type ("national holiday")
  - Holiday_count: count holidays for each holiday type
  - Function: SQL query to join the holidays table with the holiday_types table, grouping the data by holiday type

- plot_holidays_per_type(df)
  - Requires: df - pandas dataframe containing columns type_name and holiday_count
  - Return: none
  - Function: X-Axis: holiday types ("national holiday" "local holiday"

  - Y-Axis : count of holidays for each type
  - The bars are colored blue and gridlines are included for clarity, the x-axis labels are rotated for readability

- main()
  - Requires ; SQLite database file (holidays.db) must exist at the specified database_path
  - Return (none)
  - Function: loads holiday data using load_holiday_data()
  - Bar graph using the Plot_holidays_per_type()

temperature_ozone_visualization.py

- main()
  - Required: Nothing
  - Returns: None
  - Function: Connects to SQLite database and uses SQL query to fetch weather data and air quality data
  - Loads data from query into pandas DataFrame
  - Converts date column to a pandas datetime for readability
  - Creates a dual-axis line plot to visualize:
    - Left Y-Axis showing average temperature
    - Right Y-Axis showing average ozone level

temp_ozone_vis.py

- plot_data(x, y1, y2, ax1, month)
  - Requires: x (x-axis values, list), y1 (first y-axis values, list), y2 (second y-axis values, list), ax1 (plot location, array), month (str)
  - Returns: None
  - Function: Given the input values, plots a line graph for temperature and ozone levels vs date.
- graph_setup()
  - Requires: Nothing
  - Returns: None
  - Function: Processes database data, separates data into months, and then calls plot_data(x, y1, y2, ax1, month) to plot each graph. Shows graph at the end.
- main()
  - Requires: Nothing
  - Returns: None

    - Function: Sets up database connection and cursor; calls graph_setup().

temp_ozone_scatter.py

- graph()
    - Requires: Nothing
    - Returns: None
    - Function: Processes database data and then plots a scatterplot of temperature vs ozone level. Also plots a regression line for the scatterplot.
- main()
    - Requires: Nothing
    - Returns: None
    - Function: Calls graph().

ozone_holiday_vis.py

- main()
    - Requires: Nothing
    - Returns: None.
    - Function:
        - Connects to the SQLite database and fetches holiday names and corresponding air quality from holidays and air_quality tables.
        - Filters data to include specific holidays, chosen at random.
        - Creates a bar chart to visualize average ozone levels on selected holidays.

Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project

Resources

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
12/05/24	Didn't understand when and how to convert string to datetime and back to string for API requests and for storage	Internet (docs.python.org)	Yes
12/05/24	Had trouble making sure no duplicate dates were entered in the table and the program would continue on the next date after 25 entries	ChatGPT	Yes
12/08/24	Couldn't figure out how to work requests (website only had bash commands)	Internet (OpenAQ Python tutorial, found via OpenAQ Slack)	Yes
12/10/24	Didn't know how to make the table for foreign key	ChatGPT	Yes
12/13/24	Didn't understand details of datetime objects/how to work timedelta	Internet (dataquest.io)	Yes
12/15/24	Unsure how to make line plot with two different scales	Internet (matplotlib documentation)	Yes
12/15/24	Didn't know how to add a regression line to a scatterplot	Internet (python-graph-gallery .com)	Yes

Bryan Holmes, Faye Guan, Jenny Shin
github link: https://github.com/fjguan/SI206-Final-Project

12/15/24	Didn't know how to separate days in x axis by week in a line plot	Internet (matplotlib documentation)	Yes
12/15/24	Forgot how to reference correct database path for visualizations	Internet (stackoverflow)	Yes
12/15/24	script inserts only 25 new rows each time it runs, you need to modify the logic to:	ChatGPT	Yes