SI 206
Fiery Latinas

Marcela Passos and Carolina Janicke

Our Goal

We want to examine how the response time for fires between the the cities of Los Angeles and the New York differ during similar times in the day. Additionally we also want to determine if certain neighborhoods in NYC have more fires on average. Our database contains 4 tables two of them have fire ID, response time from the fire department, and time of day that the fires were reported as columns.

What We Achieved

- We were able to use two API's to gather our data into tables with 100 entries
- Created 4 tables- fire ID, response time from the fire department, and time of day that the fires were reported as columns.
- We converted time variables to only have 4 digits (hr and minutes) to be more easily read and understood
- Using the fire_data.db we were able to calculate the average number of fires in different NYC neighborhoods, and the average response time for fires during two hour periods during the day
- We used MatPlotLib to create a bar graph and line graphs to visualize and compare the data

Problems

- Finding two API's with data that was relevant to one another
- It was hard for us to ensure that each table in our database had exactly 100 data points
- We struggles with successfully parsing through the NYC_data.json because of the way it was formatted after getting data from the NYC API. Once we figured out that it was a list of lists of dictionaries than we were able to successfully parse through the Json file
- Figuring out how to calculate the average fire response for non-overlapping 2 hour periods was also a challenge
- Fixing our function calculations to ensure the averages were mathematically correct

Calculations

LA

The average response time of the firefighters to get on scene for every 2 hour period

NYC

The average number of fires per neighborhood, The average response time of the firefighters to arrive scene for every 2 hour period

Created a neighborhood table and calculated fires per neighborhood

Both

Comparing the average response time of firefighters between LA and NYC

How To Run Our Code (LA)

Set Up Database: Connect to the "fire_data.db" database and create the first table.
Insert Data: Load data from "LA_data.json" and insert it into the database table.
Calculate Averages: Calculate the average response time per period.
Close Database Connection: Close the database connection.
Handle Errors: Catch and print any errors that occur during execution.

```
def main():
    try:
        cur, conn = set_up_database("fire_data.db")
        create_first_table(cur, conn)
        with open("LA_data.json", "r") as json_file:
            LA_data = json.load(json_file)

        insert_data_to_fires_table(cur, conn, )
        calculate_avg_response_time_per_period(cur, conn)

        conn.close()
    except Exception as e:
        print("An error has occurred:", e)

if __name__ == "__main__":
    main()
```

How To Run Our Code (NYC)

Fetch Fire Data: Retrieve fire department data for "Structural Fires".
Set Up Database: Connect to the "fire_data.db" database and create tables for fire and neighborhood data.
Insert Data: Load data from "NYC_data.json" and insert it into the database tables.
Calculate Averages: Calculate the average number of fires per neighborhood and the average response time per period.
Close Database Connection: Close the database connection.
Handle Errors: Catch and print any errors that occur during execution

```
def main():
    get_fire_data("Structural Fires")

    try:
        #set up the database
        cur, conn = set_up_database("fire_data.db")
        create_first_nyc_table(cur, conn)
        create_neighborhood_table(cur, conn)
        create_fire_neighborhood_relationship_table(cur, conn)
        with open("NYC_data.json", "r") as json_file:
            NYC_data = json.load(json_file)

        insert_data_to_fires_table(cur, conn, NYC_data)
        insert_data_to_neighborhood_table(cur, conn, NYC_data)

        calculate_avg_fires_per_neighborhood(cur, conn)
        calculate_avg_response_time_per_period(cur, conn)

        conn.close()  # Close the database connection after insertion
    except Exception as e:
        print("An error has occured:", e)


if __name__ == "__main__":
```

Visuals


Figure 1
x=BROOKLYN y=5.39
Average Number of Fires per Neighborhood
Number of Fires
0
2
4
6
8
10
BRONX
BROOKLYN
MANHATTAN
QUEENS
RICHMOND / STATEN ISLAND
Neighborhood

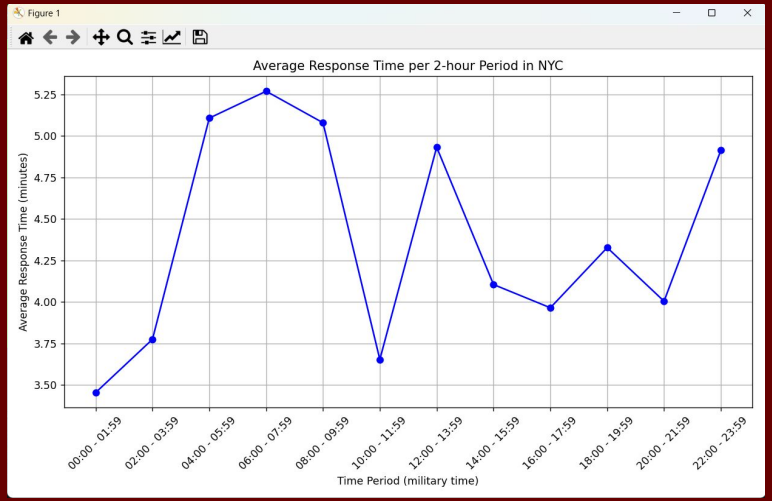


Figure 1
Average Response Time per 2-hour Period in NYC
Average Response Time (minutes)
3.50
3.75
4.00
4.25
4.50
4.75
5.00
5.25
00:00 - 01:59
02:00 - 03:59
04:00 - 05:59
06:00 - 07:59
08:00 - 09:59
10:00 - 11:59
12:00 - 13:59
14:00 - 15:59
16:00 - 17:59
18:00 - 19:59
20:00 - 21:59
22:00 - 23:59
Time Period (military time)

Comparison

Breakdown Of Our Functions (LA)

1. get_fire_dep_data(classification_group)
- Fetches fire department data from LA city government's dataset using an api.
- uses the classification_group argument to only get data that a recorded “On Scene Time”
- Appends received data to "LA_data.json" if available.
- Handles HTTP and JSON decoding errors.

2. set_up_database(db_name)
- Sets up an SQLite database
- Returns cursor (`cur`) and connection (`conn`) objects.

3. create_first_table(cur, conn)
- Creates the "LA_Fires" table in the database.
- Table has columns for fire ID, time, and response time.

4. insert_data_to_fires_table(cur, conn)
- Inserts data from the LA_fires.json file into the database.
- Limits entries to 100 rows and 8 unique fire data per hour.
- Calculates response time (on scene time - time of call) and adds it to the table.

5. calculate_avg_response_time_per_period(cur, conn)
- Calculates average response time per 2-hour period.
- Writes results to "calculations.txt".

Breakdown Of Our Functions (NYC)

1. get_fire_data(classification_group)
- Retrieves fire data from the NYC government API.
- uses the classification_group argument to filter data by the classification group of “Structural Fires”
- Appends the data to "NYC_data.json".
- Handles HTTP errors and JSON decoding errors.

2. set_up_database(db_name)
- Sets up an SQLite database connection using the provided name.
- Drops existing tables if they exist to ensure a clean start.

3. create_first_nyc_table(cur, conn)
- Creates the "NYC_Fires" table in the database.
- Table includes columns for fire ID, date, time, and response time.

4. create_neighborhood_table(cur, conn)
- Creates the "neighborhood_ID" table in the database.
- Table includes columns for neighborhood ID and neighborhood name.

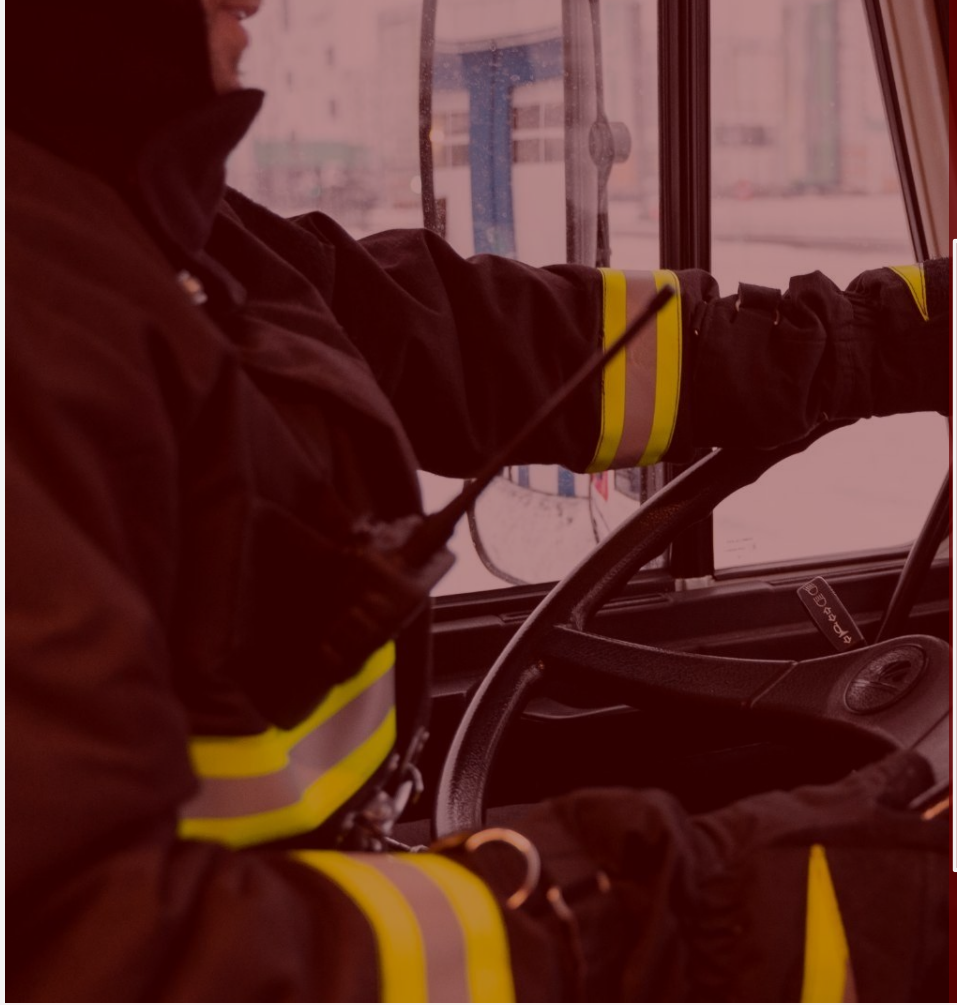
5. create_fire_neighborhood_relationship_table(cur, conn)
- Creates the "Fire_Neighborhood_Relationship" table in the database.
- Table establishes a relationship between fire_ID and neighborhood_ID.

6. insert_data_to_fires_table(cur, conn, json_data)
- Inserts NYC data from NYC_data.json into the "NYC_Fires" table.
- Processes the data in batches to handle large datasets.
- Stops at 100 entries
- converts fire_response_time from seconds to minutes

7. insert_data_to_neighborhood_table(cur, conn, json_data)
- Inserts neighborhood data into the "neighborhood_ID" table.
- Establishes relationships between fires and neighborhoods.

8. calculate_avg_fires_per_neighborhood(cur, conn)
- Calculates the average number of fires per neighborhood.
- Writes results to "calculations.txt".

9. calculate_avg_response_time_per_period(cur, conn)
- Calculates the average response time per 2-hour period in NYC.
- Writes results to "calculations.txt".



Resources

ChatGpt
Socrata
GeeksforGeeks

Thank You!