

Lillian Joelson and Karen Lin  
SI 201 - Data-Oriented Programming  
Barbara Jane Ericson  
15 Dec 2025  
GitHub Link: [https://github.com/lkarenin/SI\\_201-Final-Project](https://github.com/lkarenin/SI_201-Final-Project)

## Final Project Report

### 1. Project Goals

Planned APIs

- i. OpenWeather: A weather API that allows us to access the weather conditions of any location we desired; we specifically accessed reports for Detroit, Michigan
  1. **Historical** weather data
    - a. Airport location (city)
    - b. Date/time
    - c. Temperature
    - d. Humidity
    - e. Wind speed
    - f. Weather description (sunny, rainy, cloudy, snow, etc.)
- ii. AviationStack: A flight API that would give us the schedule of flights and their actual departure times in Detroit
  1. **Historical** flight data
    - a. Flight number
    - b. Airline
    - c. Departure airport (IATA code)
    - d. Arrival airport (IATA code)
    - e. Scheduled departure & arrival time
    - f. Actual departure & arrival time
    - g. Delay (minutes)

Our goal: Discover the average amount of delay time for flights departing from Detroit, Michigan during weather with precipitation

### 2. Achieved Goals

Actual APIs - we used the same ones as planned however the data changed slightly

- i. AviationStack: A flight API that would give us the schedule of flights and their actual departure times in Detroit
  1. **Real-time** flight data - we couldn't access historical flight data for free, so we changed the data to collect real time departure flights from the airport instead

- a. Flight number
  - b. Airline
  - c. Departure airport (IATA code)
  - d. Arrival airport (IATA code)
  - e. Scheduled departure & arrival time
  - f. Actual departure & arrival time
  - g. Delay (minutes)
- ii. OpenWeather: A weather API that allows us to access the weather conditions of any location we desired; we specifically accessed reports for Detroit, Michigan
  - 1. **Real-time** weather data - because of the constraint with the AviationStack API, we also transitioned our weather data to be collected in real-time so that it would be compatible with our flight data
    - a. Airport location (city)
    - b. Date/time
    - c. Temperature
    - d. Humidity
    - e. Wind speed
    - f. Weather description (sunny, rainy, cloudy, snow, etc.)

### 3. Problems

#### API Constraints

- i. Because we could only gather real time information instead of historical information, we couldn't organize our information by month in the way that we planned. We changed our plan from gathering monthly data to gathering data by the hour over the course of one day. This way, we could still calculate the average while allotting for the time constraints.

#### Collaboration cross VS Code and GitHub

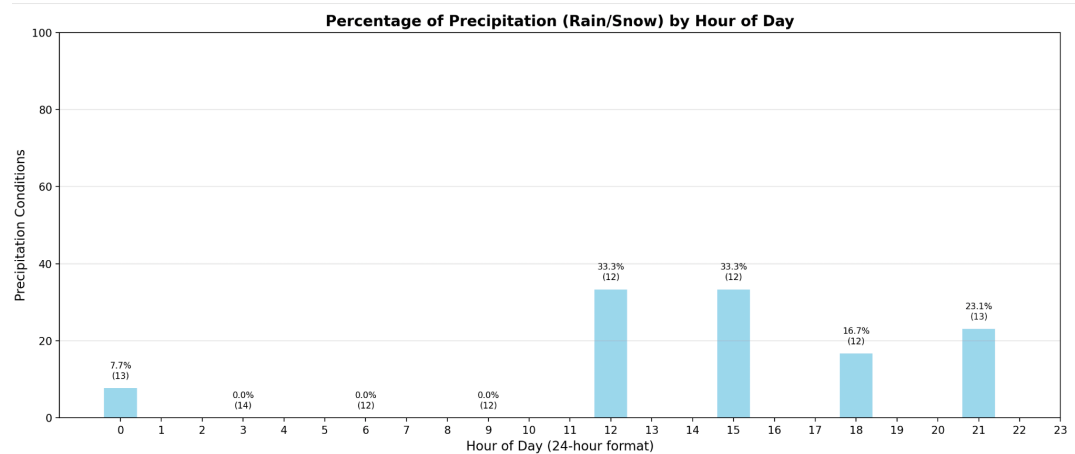
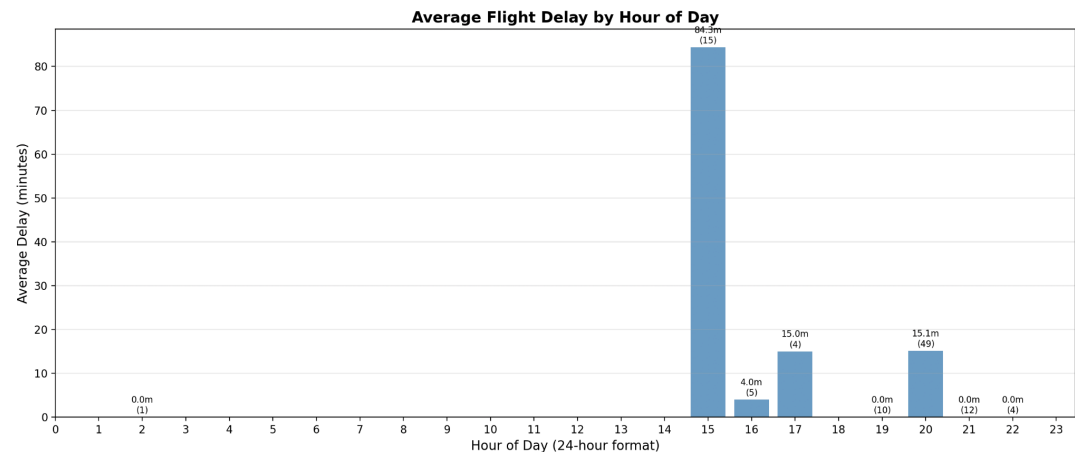
- ii. On project 1 we had tried to work on one VS Code file simultaneously, but because Karen was the owner, Lilly couldn't access the file unless Karen had it open on her computer, making asynchronous collaboration difficult. For this project we took a different approach. Karen was responsible for working with the weather API and Lilly was responsible for working with the flight API. We each created our own GitHub repositories and our own VS Code files that we could work with our individual portion of the data on. We then shared the GitHub repositories with each other, and as we make updates and commits to our own repositories, we would text the other person about our updates. Then that person could

create a file for the code the other team member wrote and copy + paste their work into their own VS Code file. This way we could work asynchronously while still having access to the other person's code.

#### 4. The calculations from the data in the database

```
delay_calculations.txt
1  FLIGHT DELAY DURING PRECIPITATION
2  Total flights: 100
3  Flights with delay data: 100
4  Total weather records: 100
5  Flight dates range: 2025-12-13T17:55:00+00:00 to 2025-12-15T02:00:00+00:00
6  Weather fetch dates range: 2025-12-13T15:21:56.894660 to 2025-12-14T20:15:20.591934
7  Weather records with precipitation: 14
8
9  Total flight-weather matches within 3 hours: 266
10 RESULTS
11 Flights during precipitation: 114
12 Total flight-weather matches: 266
13 Average departure delay during precipitation: 13.00 minutes
14
```

#### 5. Visualizations



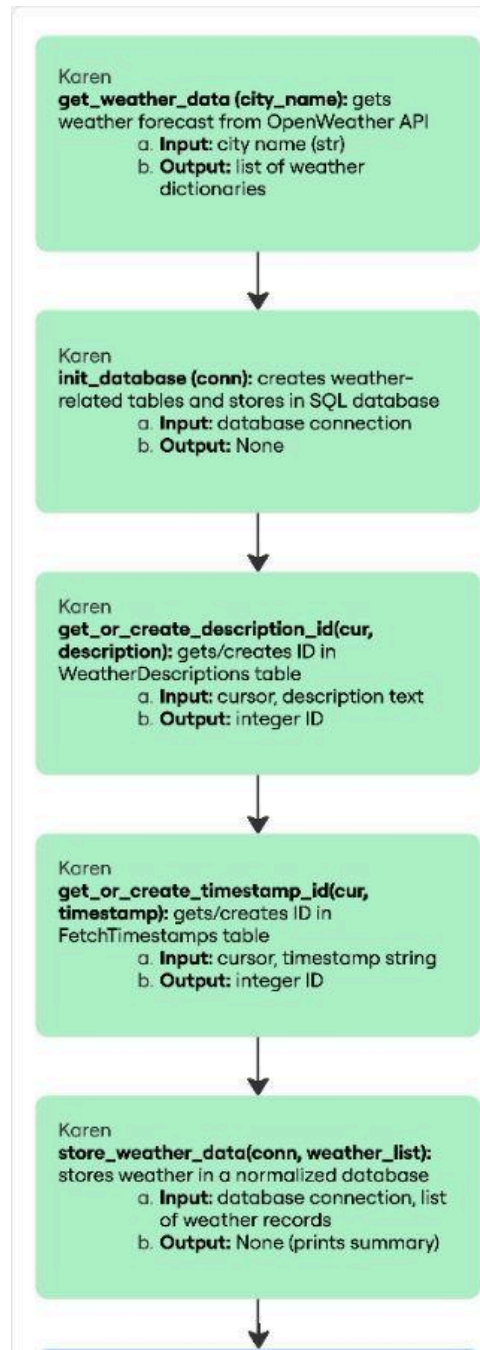
## 6. Instructions

AviationStack API key: 52866fb0c723c8b58cd3b8f8eaa6f169

OpenWeather API key: 2aee95c86e2cf484f13b865933689b2a

Run **flights\_api.py** to access flight information in Detroit via the AviationStack API and then **weather\_api.py** to access weather information in Detroit via the OpenWeatherMap API. Since ea  
age amount of time in delays for flights departing in Detroit during weather  
with precipitatory conditions in **delay\_calculations.txt**.

## 7. Function diagram



Lilly  
**get\_flight\_data** (airport, month=None):  
fetches flight data from AviationStack API

- **Input:** Airport code (str), optional month (str)
- **Output:** list of flight dictionaries

Lilly  
**get\_or\_create\_id** (cursor, table, column, value): get existing IDs or create new ones in lookup table

- **Input:** Cursor, table name, column name, value
- **Output:** integer ID

Lilly  
**store\_flight\_data** (db\_conn, flights\_list):  
store flights in database

- **Input:** database connection, list of flights
- **Output:** count of inserted records (int), creates table categories: Airlines, FLightStatuses, Timestamps, Flights, FlightDelays

Karen  
**calculate\_avg\_delay\_precipitation**(db\_conn)

- **Input:** SQLite connection
- **Output:** Average departure delay (float) during rainy/snowy weather, saved to .txt file.

Lilly  
**plot\_avg\_delay\_by\_hour**(db\_name="project\_data.db"): plot average flight delays by hour of day

- **Input:** database name (str)
- **Output:** matplotlib chart and console output

Lilly  
**plot\_avg\_precipitation\_by\_hour**(db\_name="project\_data.db"): plot percentage of precipitation by hour of day

- **Input:** database name (str)
- **Output:** matplotlib chart and console output

## 8. Resource documentation

Date	Issue Description	Location of Resource	Result Did it solve the issue?
12/10/2025	Formatting and debugging flights_api	Claude AI	Lilly Yes, I was having some issues that I couldn't work out with the formatting of the function and Claude helped me debug and gave advice on how to restructure it
12/10/2025	Assistance with SQLite tables	Claude AI	Karen Yes, I requested guidance on structuring my SQLite database tables to properly store flight and weather data from the OpenWeather API
12/11/2025	Got advice on how to change formatting to adjust for data only being collected in real time instead of historically	Claude AI	Lilly and Karen Yes, it advised us to calculate by hour instead of by month and helped adjust our code to accommodate this
12/12/2025	Debugging SQLite errors	Claude AI	Karen Yes, I ran into a couple SQLite-related errors throughout my project so I had Claude help with debugging based on outputted error messages
12/13/2025	Making integer keys for repeating strings in flights_api	Claude AI	Lilly Yes, Claude helped guide me through how to create integer keys for flights_api because it had a lot of repeating strings
12/13/2025	Making integer keys to prevent duplicate string data in the database	Claude AI	Karen Yes, I asked Claude for help in creating integer keys to prevent duplicate string data after we received feedback from our grading session
12/15/2025	Adjusting visualizations for	Claude AI	Lilly Yes, Karen had to change some

	changes made in the weather_api file		formatting in her weather_api file, so the visualizations file needed to be adjusted for that, Claude gave me advice on how to do that because I wasn't familiar with the changes Karen made in her code
--	--------------------------------------	--	--