Final Project Report
Hannah Nadel and Nadia Reddin
SI 206
April 15th, 2025

Github link: https://github.com/hnadel123/SI-206-Final-Project.git

**Corrections:**

We added a legend to our Matplotlib graphs using the plt.legend() function to clearly label different data series, improving the readability of the visualizations. To keep our CSV file organized, we ensured that each time the code runs, the new data entry is assigned a sequential number indicating the run order, making it easy to track the most recent data. Additionally, when working with the Disneyland Parks API, we removed the "country", "continent", and "timezone" keys from the data to prevent repeating IDs.

## Goals and Challenges

The primary goal of our project, Weathering the Wait Time, was to investigate how weather conditions affect ride wait times at Disneyland. We aimed to identify patterns between environmental factors and the length of time guests spend waiting for rides, ultimately helping park visitors plan their trips more efficiently. We intended to use two APIs to accomplish this goal. The first one was a Queue-Times API, which gave Disneyland access to real-time ride wait times and park information. We extracted data about each Disneyland park, including ride titles, wait times, park location, and whether it was open or closed. We also used the Open-Meteo API, which provided us with up-to-date weather information for the geographic coordinates of Disneyland parks, including temperature, precipitation, rain, and cloud cover. We collected and stored time-stamped information on local weather conditions and ride wait times in a SQLite database. Next, we calculated the average wait times under various weather conditions and used scatter plots to illustrate the relationship between temperature and wait times. We wanted to know if the park's queues are shorter or longer because of conditions like heat or rain.

Our primary project objective, which was to examine how weather affects wait times at Disneyland rides, was accomplished. We used the two APIs that were initially planned. We gathered real-time wait times, park associations, ride names, and ride status from Disneyland using the Queue-Times API. This information was saved in a local SQLite database and timestamped. Current weather data for the same time and place as the ride data was provided by the Open-Meteo API. We collected data on temperature, precipitation, rain levels, and cloud cover for every Disneyland park. We merged these two datasets using location coordinates. We then analyzed and visualized trends such as how ride wait times varied with temperature. For example, we made four scatter plot graphs showing the weather versus various Disney park ride wait times.

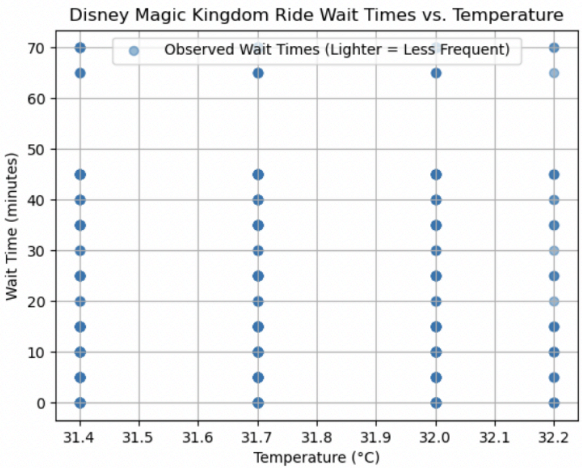
However, we encountered some complex challenges. Depending on when we executed the code, the values could vary significantly because both weather and ride wait times were gathered in real-time. We had to run the software numerous times over a few days to collect enough data for analysis. Furthermore, we were only able to retrieve data for 25 trips at a time using the Queue-Times API. Therefore, to create a more comprehensive dataset, we had to run our script multiple times. We divided the project between the two of us because it was challenging to work on the code simultaneously. One person worked on the weather API, while the other concentrated on the ride data API. This allowed us to work together without overlapping our work.

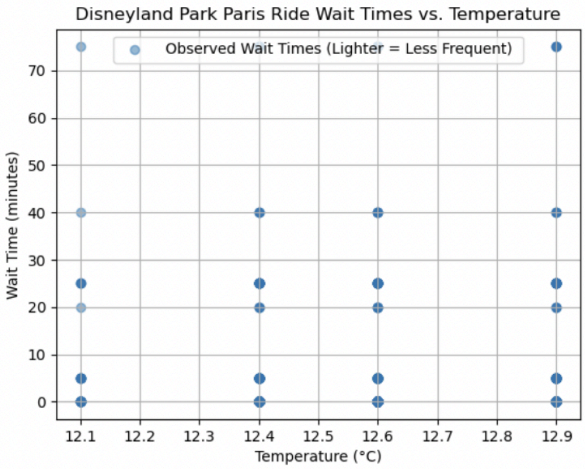
## Calculations from Data Table

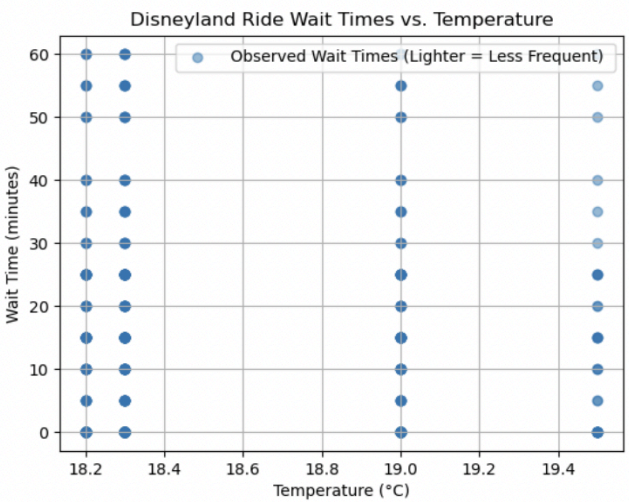
park_wait_summary.csv

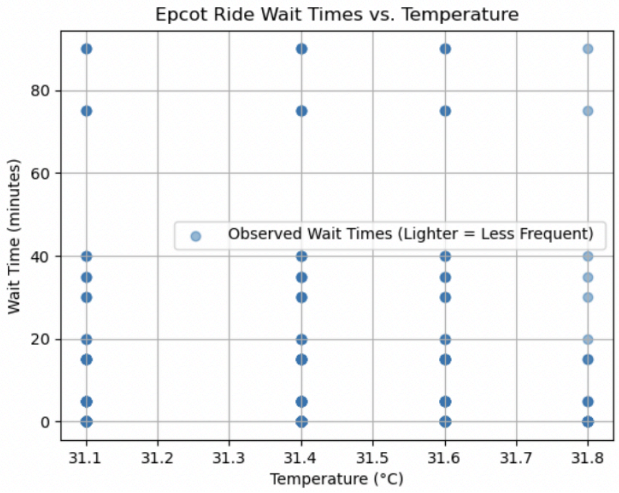
```
Park Name,Average Wait Time (min),Temperature (°C),LatestUpdate
Disney California Adventure,10.48,18.2,0
Disney California Adventure,10.48,18.3,1
Disney California Adventure,10.48,19.0,2
Disneyland,15.00,18.2,3
Disneyland,15.00,18.3,4
Disneyland,15.00,19.0,5
Disneyland Park Paris,6.83,12.4,6
Disneyland Park Paris,6.83,12.6,7
Disneyland Park Paris,6.83,12.9,8
Shanghai Disney Resort,0.16,19.2,9
Shanghai Disney Resort,0.16,19.3,10
Walt Disney Studios Paris,10.00,12.5,11
Walt Disney Studios Paris,10.00,12.8,12
Walt Disney Studios Paris,10.00,13.0,13

```

## Visualizations






## Instructions:

There are three Python files for this project:

- **main.py** — The file you run to launch the complete code
- **Nadia.py** — Collects Disneyland ride and park data using the Queue-Times API.
- **hannah.py** — Collects weather data using the Open-Meteo API.

## Step-by-Step Instructions:

1. Open the file titled main.py and run the code.
2. When the code runs, it automatically performs the following actions:
    - Creates a new SQLite database file called weathniering_the_wait_time.db
    - Sets up 3 tables inside the database: DisneyParks, DisneyRides, and Weather
    - Gets the Disneyland park and ride data using the Queue-Times API

- Gets the current weather data (temperature, precipitation, cloud cover, etc.) for each park using the Open-Meteo API
- Matches each ride's wait time with weather conditions at that exact time
- Filters data where the temperature is under 30°C, and calculates average wait times for each park
- Prints the results to the VS Code terminal
- Saves the results to a file called park_wait_summary.csv
- Creates and displays a graph showing the relationship between temperature and ride wait times

## What You'll See:

- A CSV file in your folder with the park name, average wait time, and temperature
- The VS Code terminal will print the average wait time per park when the temperature is below 30°C

## Documentation

a. **create_tables()**
   i. **Purpose**: Creates three database tables: DisneyParks, DisneyRides, and Weather.
   ii. **Inputs:** None
   iii. **Outputs:** Creates tables in weathering_the_wait_time.db

b. **plot_wait_times_vs_temperature()**
   i. **Purpose:** Makes a scatterplot of how the wait times are related to temperature
   ii. **Inputs:** None
   iii. **Outputs:** A Matplotlib plot with temperature vs wait time

c. **test()**
   i. **Purpose:** Calculates each park's average ride wait times when the temperature is below 30°C.
   ii. **Inputs:** None
   iii. **Outputs:** Displays the results in the VS Code terminal and saves them to a CSV file using the write_results_to_csv() function.

d. **write_results_to_csv(results)**
   i. **Purpose:** Sends ride wait time and temperature to a CSV file.
   ii. **Inputs:** A list of tuples: Park Name, Avg Wait Time, Temperature
   iii. **Outputs:** Creates park_wait_summary.csv with that data

e. **main()**
   i. **Purpose:** Runs the code by setting up the database, collecting data, and analyzing the results.
   ii. **Inputs:** None
   iii. **Outputs:** Shows results in the VS Code terminal, stores data in a database, creates a graph, and exports a CSV file.

f. **fetch_disneyland_data(timestamp)** ***(from Nadia.py)***
   i. **Purpose:** Takes the ride information and park details from the Queue-Times API for Disneyland parks.
   ii. **Inputs:** Timestamp: Shows the exact time something was saved in the database.
   iii. **Outputs:** Inserts ride and park information into DisneyParks and DisneyRides tables

g. **fetch_weather_data(timestamp)** ***(from hannah.py)***
   i. **Purpose:** Gets the current weather information for park locations using the Open-Meteo weather API.
   ii. **Inputs:** timestamp: Shows the current time something was saved to the database.
   iii. **Outputs:** Inserts weather info into the Weather table

## Documentation

| Date | Issue Description | Location of Resource | Result<br>(Did it solve the issue?) |
|---|---|---|---|
| 4/1/2025 | Fetching Disneyland ride wait times using API | Nadia.py – Disneyland API function | Yes, ride data was successfully retrieved and stored in the database |
| 4/1/2025 | Fetching weather data based on Disney Park locations | hannah.py – Open-Meteo API function | Yes, weather data was successfully pulled using latitude/longitude |
| 4/13/2025 | Plotting only 25 rides per park in the visualizations | | Used a loop counter to restrict the number of rides plotted |
| 4/14/2025 | Visualizing data with Python | Matplotlib | Successfully created data visualizations. |

| | | | |
|---|---|---|---|
| 4/16/2025 | Overlapping graphs when plotting multiple charts | plt.figure() and plt.close() in plotting functions | Yes, each graph is now separate without overlap |