Name	Last update
.. (/wustl-bootcamp/WASHSTL201809DATA3/tree/master/10-Advanced-Data-Storage-and-Retrieval/Homework/Instructions/..)	
📁 Images (/wustl-bootcamp/WASHSTL201809DATA3/tre...	a week ago
📁 Resources (/wustl-bootcamp/WASHSTL201809DATA3/...	a week ago
📄 README.md (/wustl-bootcamp/WASHSTL201809DATA...	a week ago
📄 climate_starter.ipynb (/wustl-bootcamp/WASHSTL2018...	a week ago

📄 **README.md** (/wustl-bootcamp/WASHSTL201809DATA3/blob/master/10-Advanced-Data-Storage-and-Retrieval/Homework/Instructions/README.md)

Surfs Up!

(/wustl-bootcamp/WASHSTL201809DATA3/raw/master/10-Advanced-Data-Storage-and-Retrieval/Homework/Instructions/Images/surfs-up.jpeg)

Congratulations! You've decided to treat yourself to a long holiday vacation in Honolulu, Hawaii! To help with your trip planning, you need to do some climate analysis on the area. The following outlines what you need to do.

Step 1 - Climate Analysis and Exploration

To begin, use Python and SQLAlchemy to do basic climate analysis and data exploration of your climate database. All of the following analysis should be completed using SQLAlchemy ORM queries, Pandas, and Matplotlib.

- Use the provided starter notebook (/wustl-bootcamp/WASHSTL201809DATA3/blob/master/10-Advanced-Data-Storage-and-Retrieval/Homework/Instructions/climate_starter.ipynb) and hawaii.sqlite (/wustl-bootcamp/WASHSTL201809DATA3/blob/master/10-Advanced-Data-Storage-and-Retrieval/Homework/Instructions/Resources/hawaii.sqlite) files to complete your climate analysis and data exploration.

- Choose a start date and end date for your trip. Make sure that your vacation range is approximately 3-15 days total.

- Use SQLAlchemy `create_engine` to connect to your sqlite database.

- Use SQLAlchemy `automap_base()` to reflect your tables into classes and save a reference to those classes called `Station` and `Measurement`.

Precipitation Analysis

- Design a query to retrieve the last 12 months of precipitation data.

- Select only the `date` and `prcp` values.

- Load the query results into a Pandas DataFrame and set the index to the date column.

- Sort the DataFrame values by `date`.

- Plot the results using the DataFrame `plot` method.
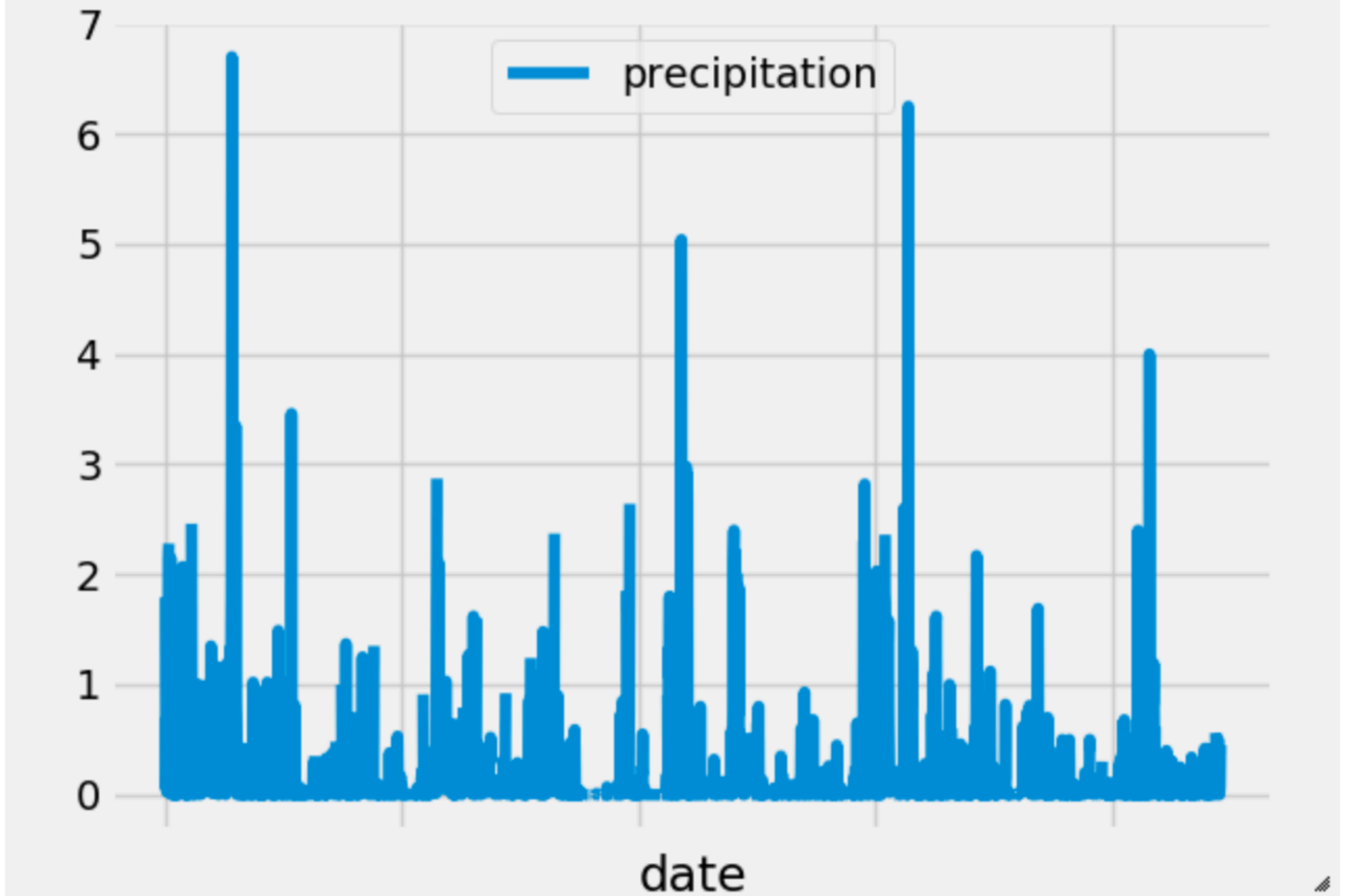
(/wustl-bootcamp/WASHSTL201809DATA3/raw/master/10-Advanced-Data-Storage-and-Retrieval/Homework/Instructions/Images/precipitation.png)

- Use Pandas to print the summary statistics for the precipitation data.

Station Analysis

- Design a query to calculate the total number of stations.

- Design a query to find the most active stations.

  - List the stations and observation counts in descending order.
  - Which station has the highest number of observations?
  - Hint: You may need to use functions such as `func.min`, `func.max`, `func.avg`, and `func.count` in your queries.

- Design a query to retrieve the last 12 months of temperature observation data (tobs).

  - Filter by the station with the highest number of observations.
  - Plot the results as a histogram with `bins=12`.

(/wustl-bootcamp/WASHSTL201809DATA3/raw/master/10-

Advanced-Data-Storage-and-Retrieval/Homework/Instructions/Images/station-histogram.png)

Temperature Analysis (Optional)

- The starter notebook contains a function called `calc_temps` that will accept a start date and end date in the format `%Y-%m-%d` and return the minimum, average, and maximum temperatures for that range of dates.

- Use the `calc_temps` function to calculate the min, avg, and max temperatures for your trip using the matching dates from the previous year (i.e., use "2017-01-01" if your trip start date was "2018-01-01").

- Plot the min, avg, and max temperature from your previous query as a bar chart.

    - Use the average temperature as the bar height.
    - Use the peak-to-peak (tmax-tmin) value as the y error bar (yerr).

(/wustl-bootcamp/WASHSTL201809DATA3/raw/master/10-

Advanced-Data-Storage-and-Retrieval/Homework/Instructions/Images/temperature.png)

Other Recommended Analysis (Optional)

- The following are optional challenge queries. These are highly recommended to attempt, but not required for the homework.

    - Calculate the rainfall per weather station using the previous year's matching dates.

- Calculate the daily normals. Normals are the averages for the min, avg, and max temperatures.

  - You are provided with a function called `daily_normals` that will calculate the daily normals for a specific date. This date string will be in the format `%m-%d`. Be sure to use all historic tobs that match that date string.
  - Create a list of dates for your trip in the format `%m-%d`. Use the `daily_normals` function to calculate the normals for each date string and append the results to a list.
  - Load the list of daily normals into a Pandas DataFrame and set the index equal to the date.
  - Use Pandas to plot an area plot (`stacked=False`) for the daily normals.

(/wustl-bootcamp/WASHSTL201809DATA3/raw/master/10-

Advanced-Data-Storage-and-Retrieval/Homework/Instructions/Images/daily-normals.png)

Step 2 - Climate App

Now that you have completed your initial analysis, design a Flask API based on the queries that you have just developed.

- Use FLASK to create your routes.

Routes

- `/api/v1.0/precipitation`

  - Convert the query results to a Dictionary using `date` as the key and `prcp` as the value.
  - Return the JSON representation of your dictionary.

- `/api/v1.0/stations`

  - Return a JSON list of stations from the dataset.

- `/api/v1.0/tobs`

  - query for the dates and temperature observations from a year from the last data point.

- Return a JSON list of Temperature Observations (tobs) for the previous year.

- `/api/v1.0/<start>` and `/api/v1.0/<start>/<end>`

  - Return a JSON list of the minimum temperature, the average temperature, and the max temperature for a given start or start-end range.
  - When given the start only, calculate `TMIN`, `TAVG`, and `TMAX` for all dates greater than and equal to the start date.
  - When given the start and the end date, calculate the `TMIN`, `TAVG`, and `TMAX` for dates between the start and end date inclusive.

Hints

- You will need to join the station and measurement tables for some of the analysis queries.

- Use Flask `jsonify` to convert your API data into a valid JSON response object.

Copyright