**Background**

I have decided to treat my family to a long overdue holiday in Honolulu, Hawaii. To help plan our trip, I decided to do a climate analysis about the area. The following sections outline the steps that I needed to take to accomplish this task.



**Part 1: Analysed and Explored the Climate Data**

For this part of the challenge, I have used Python and SQLAlchemy to do a basic climate analysis and data exploration of your climate database.

Specifically, I used SQLAlchemy ORM queries, Pandas, and Matplotlib and completed the following steps:

1. Used the provided files (climate\_starter.ipynb and hawaii.sqlite) to complete the climate analysis and data exploration.
2. Used the SQLAlchemy create\_engine() function to connect to my SQLite database.
3. Used the SQLAlchemy automap\_base() function to reflect my tables into classes, and then saved references to the classes named station and measurement.
4. Linked Python to the database by creating an SQLAlchemy session.
5. Performed a precipitation analysis and then a station analysis by completing the steps in the following two subsections.

* **Precipitation Analysis**

1. Found the most recent date in the dataset; used this date and got the previous 12 months of precipitation data by querying the previous 12 months of data.
2. Ensured not to pass the date as a variable to my query.
3. Selected only the "date" and "prcp" values.
4. Loaded the query results into a Pandas DataFrame and explicitly set the column names.
5. Sorted the DataFrame values by "date".
6. Plotted the results by using the DataFrame plot method, as the following image shows:
7. Use Pandas to print the summary statistics for the precipitation data.

* **Station Analysis**

1. Designed a query to calculate the total number of stations in the dataset.
2. Designed a query to find the most-active stations (i.e. the stations that had the most rows). To achieve this, the following steps were completed:
   * Used functions such as func.min, func.max, func.avg, and func.count in my queries.
   * Listed the stations and observation counts in descending order.
   * Answered the following question: *which station id has the greatest number of observations?*
   * Used the most-active station id to calculate the lowest, highest, and average temperatures.
3. Designed a query to get the previous 12 months of temperature observation (TOBS) data. To do this, I completed the following steps:
   * Filtered by the station that had the greatest number of observations.
   * Queried the previous 12 months of TOBS data for that station.
   * Plotted the results as a histogram with bins=12, as the following image shows:
4. Closed my session.

**Part 2: Designed my Climate App**

Now that the initial analysis has been completed, I proceeded to design a Flask API based on the queries that I had just developed. To do so, I used Flask to create my routes as follows:

1. /

* Started at the homepage and listed all the available routes.

1. /api/v1.0/precipitation

* Converted the query results to a dictionary by using date as the key and prcp as the value.
* Returned the JSON representation of my dictionary.

1. /api/v1.0/stations

* Returned a JSON list of stations from the dataset.

1. /api/v1.0/tobs

* Queried the dates and temperature observations of the most-active station for the previous year of data.
* Returned a JSON list of temperature observations for the previous year.

1. /api/v1.0/<start> and /api/v1.0/<start>/<end>

* Returned a JSON list of the minimum temperature, the average temperature, and the maximum temperature for a specified start or start-end range.
* For a specified start, calculated TMIN, TAVG, and TMAX for all the dates greater than or equal to the start date.
* For a specified start date and end date, calculated TMIN, TAVG, and TMAX for the dates from the start date to the end date, inclusive.
* Joined the station and measurement tables for some of the queries.
* Used the Flask jsonify function to convert my API data to a valid JSON response object.