**Methodology**

This methodology describes the structure of the Hawaii Climate Analysis API, which allows users to interact with a SQLite database containing historical climate data. By utilizing Flask for the web interface and SQLAlchemy for database interaction, this API serves data on precipitation, station, and temperature.

**1. Database Setup**

* **SQLite Database**: The data for the project resides in a SQLite database file (hawaii.sqlite). This database contains two primary tables:
  + Measurement: Stores precipitation and temperature data.
  + Station: Contains information on various weather stations.
* **SQLAlchemy**: SQLAlchemy's automap\_base feature was used to automatically reflect the structure of the existing database.
* **Session**: A session was created using SQLAlchemy's Session class, which allows for interaction with the database through Python code.

**2. Flask API Setup**

* **Flask Framework**: The API is built using Flask.
* **API Routes**:
  + **Welcome Route (/)**: Provides a basic introduction to the API and lists available routes.
  + **Precipitation Route (/api/v1.0/precipitation)**: Returns precipitation data for the last year.
  + **Stations Route (/api/v1.0/stations)**: Returns a list of weather stations.
  + **Temperature Observations Route (/api/v1.0/tobs)**: Returns temperature observations for the last year.
  + **Temperature Statistics Route (/api/v1.0/temp/start)**: Returns the minimum, average, and maximum temperatures for a given start date or a date range.

**3. Database Queries**

* **Date Handling**:
  + The date format used is MMDDYY (month, day, year). The Python ‘datetime’ function is used to convert the string dates into datetime objects for comparison.
  + For queries requiring a range of dates (start and end), the datetime.strptime function is used to convert string parameters into datetime objects.

**4. Response Formatting**

* **JSON Format**: All data returned by the API is formatted in JSON. The Flask jsonify function is used to return the data in a valid JSON format.
* **Data Structures**:
  + For routes like /api/v1.0/precipitation, the results are returned as a dictionary with the date as the key and precipitation value as the value.
  + For temperature statistics (/api/v1.0/temp/start), the response includes a list of aggregate temperature values like minimum, average, and maximum.