# Module 10 Challenge

Start Assignment

**Due** Feb 5 by 10:59am **Points** 100 **Submitting** a text entry box or a website url

## Before You Begin

1. Create a new repository for this project called `sqlalchemy-challenge`. **Do not add this assignment to an existing repository**.
2. Clone the new repository to your computer.
3. Inside your local Git repository, create a directory for this Challenge. Use a folder name that corresponds to the Challenge, such as `SurfsUp`.
4. Add your Jupyter notebook and `app.py` to this folder. They'll contain the main scripts to run for analysis.
5. Push the changes to GitHub or GitLab.

## Files

Download the following files to help you get started:

**Module 10 Challenge files (https://static.bc-edx.com/data/dla-1-2/m10/lms/starter/Starter_Code.zip)**

## Instructions

Congratulations! You've decided to treat yourself to a long holiday vacation in Honolulu, Hawaii. To help with your trip planning, you decide to do a climate analysis about the area. The following sections outline the steps that you need to take to accomplish this task.

## Part 1: Analyse and Explore the Climate Data

In this section, you'll use Python and SQLAlchemy to do a basic climate analysis and data exploration of your climate database. Specifically, you'll use SQLAlchemy ORM queries, Pandas, and Matplotlib. To do so, complete the following steps:

1. Note that you'll use the provided files (`climate_starter.ipynb` and `hawaii.sqlite`) to complete your climate analysis and data exploration.
2. Use the SQLAlchemy `create_engine()` function to connect to your SQLite database.
3. Use the SQLAlchemy `automap_base()` function to reflect your tables into classes, and then save references to the classes named `station` and `measurement`.
4. Link Python to the database by creating a SQLAlchemy session.

**IMPORTANT**

> Remember to close your session at the end of your notebook.

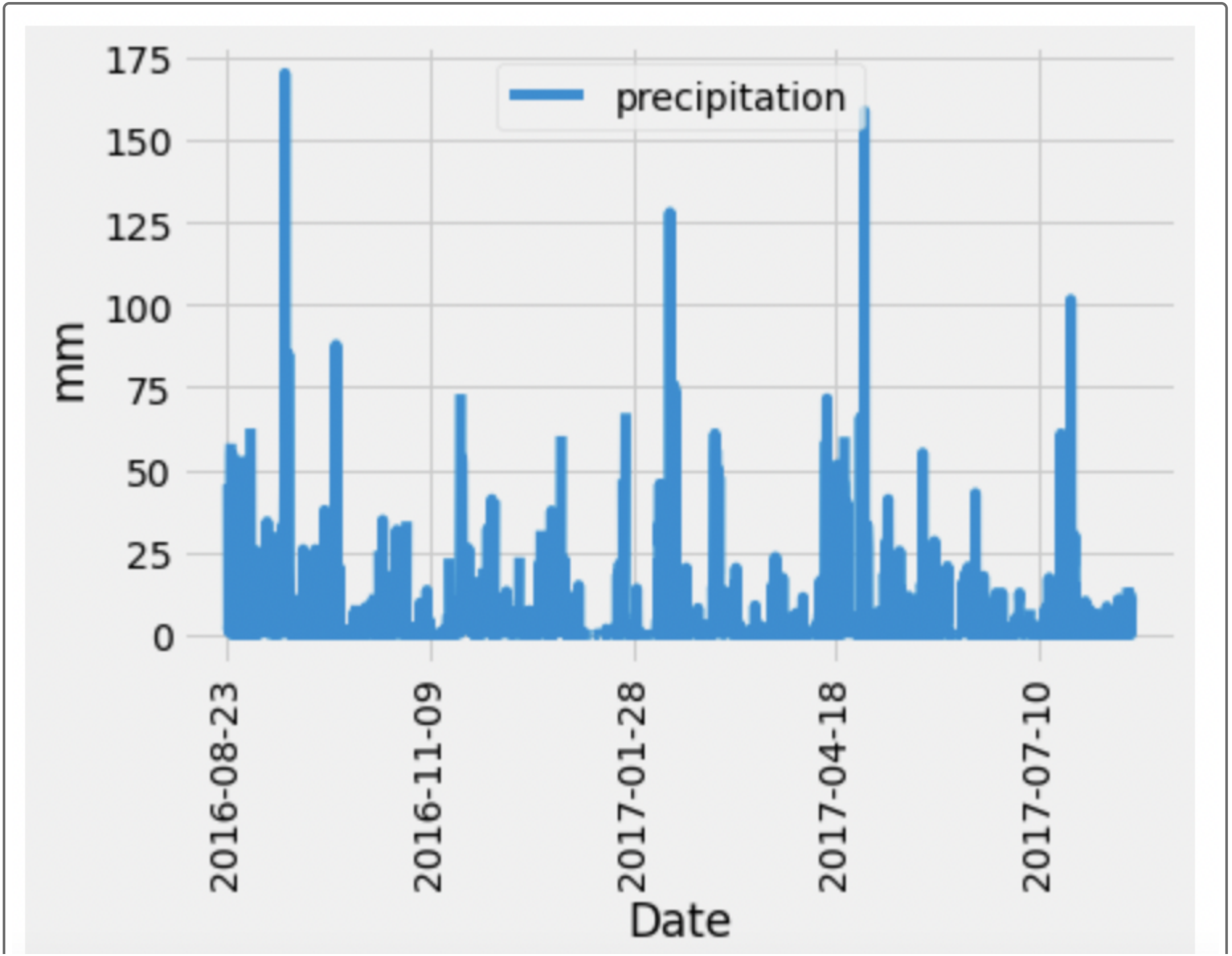
5. Perform a precipitation analysis and then a station analysis by completing the steps in the following two subsections.

### Precipitation Analysis

1. Find the most recent date in the dataset.
2. Using that date, get the previous 12 months of precipitation data by querying the previous 12 months of data.

**SHOW HINT**

3. Select only the "date" and "prcp" values.
4. Load the query results into a Pandas DataFrame. Explicitly set the column names.
5. Sort the DataFrame values by "date".
6. Plot the results by using the DataFrame `plot` method, as the following image shows:


precipitation
175
150
125
100
75
50
25
0
mm
2016-08-23
2016-11-09
2017-01-28
2017-04-18
2017-07-10
Date

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

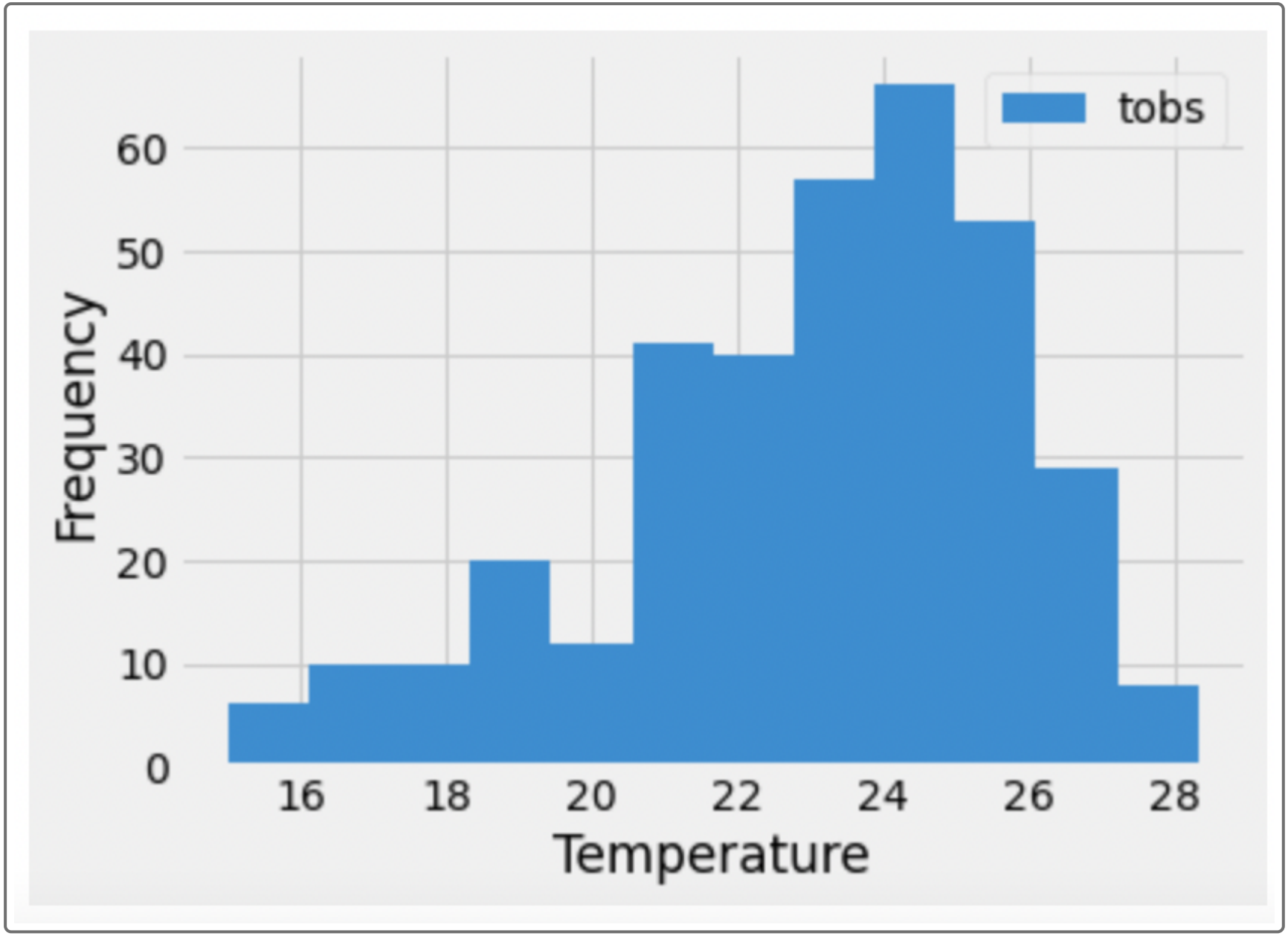
### Station Analysis

1. Design a query to calculate the total number of stations in the dataset.
2. Design a query to find the most-active stations (that is, the stations that have the most rows). To do so, complete the following steps:

**SHOW HINT**

   - List the stations and observation counts in descending order.
   - Answer the following question: which station id has the greatest number of observations?
   - Using the most-active station id, calculate the lowest, highest, and average temperatures.

3. Design a query to get the previous 12 months of temperature observation (TOBS) data. To do so, complete the following steps:
   - Filter by the station that has the greatest number of observations.
   - Query the previous 12 months of TOBS data for that station.
   - Plot the results as a histogram with `bins=12`, as the following image shows:




4. Close your session.

## Part 2: Design Your Climate App

Now that you've completed your initial analysis, you'll design a Flask API based on the queries that you just developed. To do so, use Flask to create your routes as follows:

1. `/`

- Start at the homepage.
- List all the available routes.

2. `/api/v1.0/precipitation`

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

3. `/api/v1.0/stations`

- Return a JSON list of stations from the dataset.

4. `/api/v1.0/tobs`

- Query the dates and temperature observations of the most-active station for the previous year of data.
- Return a JSON list of temperature observations for the previous year.

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

- Return 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, calculate `TMIN`, `TAVG`, and `TMAX` for all the dates greater than or equal to the start date.

- For a specified start date and end date, calculate `TMIN`, `TAVG`, and `TMAX` for the dates from the start date to the end date, inclusive.

# Hints

- Join the station and measurement tables for some of the queries.
- Use the Flask `jsonify` function to convert your API data to a valid JSON response object.

# Requirements

## Jupyter Notebook Database Connection (10 points)

### To receive all points, you must:

- Use the SQLAlchemy `create_engine()` function to connect to your SQLite database (1 point)
- Use the SQLAlchemy `automap_base()` function to reflect your tables into classes (3 points)
- Save references to the classes named `station` and `measurement` (4 points)
- Link Python to the database by creating a SQLAlchemy session (1 point)
- Close your session at the end of your notebook (1 point)

## Precipitation Analysis (16 points)

### To receive all points, you must:

- Create a query that finds the most recent date in the dataset (8/23/2017) (2 points)
- Create a query that collects only the `date` and `precipitation` for the last year of data without passing the date as a variable (4 points)
- Save the query results to a Pandas DataFrame to create `date` and `precipitation` columns (2 points)
- Sort the DataFrame by `date` (2 points)
- Plot the results by using the DataFrame `plot` method with `date` as the x and `precipitation` as the y variables (4 points)

- Use Pandas to print the summary statistics for the precipitation data (2 points)

## Station Analysis (16 points)

### To receive all points, you must:

- Design a query that correctly finds the number of stations in the dataset (9) (2 points)
- Design a query that correctly lists the stations and observation counts in descending order and finds the most active station (USC00519281) (2 points)
- Design a query that correctly finds the min, max, and average temperatures for the most active station (USC00519281) (3 points)
- Design a query to get the previous 12 months of temperature observation (TOBS) data that filters by the station that has the greatest number of observations (3 points)
- Save the query results to a Pandas DataFrame (2 points)
- Correctly plot a histogram with `bins=12` for the last year of data using `tobs` as the column to count. (4 points)

## API SQLite Connection & Landing Page (10 points)

### To receive all points, your Flask application must:

- Correctly generate the engine to the correct sqlite file (2 points)
- Use `automap_base()` and reflect the database schema (2 points)
- Correctly save references to the tables in the sqlite file (`measurement` and `station`) (2 points)
- Correctly create and binds the session between the python app and database (2 points)
- Display the available routes on the landing page (2 points)

## API Static Routes (15 points)

### To receive all points, your Flask application must include:

A **precipitation route** that:

- Returns json with the date as the key and the value as the precipitation (3 points)
- Only returns the jsonified precipitation data for the last year in the database (3 points)

A **stations route** that:

- Returns jsonified data of all of the stations in the database (3 points)

A **tobs route** that:

- Returns jsonified data for the most active station (USC00519281) (3 points)
- Only returns the jsonified data for the last year of data (3 points)

## API Dynamic Route (15 points)

### To receive all points, your Flask application must include:

A **start route** that:

- Accepts the start date as a parameter from the URL (2 points)
- Returns the min, max, and average temperatures calculated from the given start date to the end of the dataset (4 points)

A **start/end route** that:

- Accepts the start and end dates as parameters from the URL (3 points)
- Returns the min, max, and average temperatures calculated from the given start date to the given end date (6 points)

## Coding Conventions and Formatting (8 points)

### To receive all points, your code must:

- Place imports at the top of the file, just after any module comments and docstrings, and before module globals and constants. (2 points)

- Name functions and variables with lowercase characters, with words separated by underscores. (2 points)
- Follow DRY (Don't Repeat Yourself) principles, creating maintainable and reusable code. (2 points)
- Use concise logic and creative engineering where possible. (2 points)

## Deployment and Submission (6 points)

### To receive all points, you must:

- Submit a link to a GitHub repository that's cloned to your local machine and contains your files. (2 points)
- Use the command line to add your files to the repository. (2 points)
- Include appropriate commit messages in your files. (2 points)

## Comments (4 points)

### To receive all points, your code must:

- Be well commented with concise, relevant notes that other developers can understand. (4 points)

## Grading

This assignment will be evaluated against the requirements and assigned a grade according to the following table:

| Grade | Points |
| --- | --- |
| A (+/-) | 90+ |
| B (+/-) | 80–89 |
| C (+/-) | 70–79 |

| Grade | Points |
| --- | --- |
| D (+/-) | 60–69 |
| F (+/-) | < 60 |

## Submission

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.

**NOTE**

> You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next module.

Comments are disabled for graded submissions in Bootcamp Spot. If you have questions about your feedback, please notify your instructional staff or your Student Success Advisor. If you would like to resubmit your work for an additional review, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.

**IMPORTANT**

> **It is your responsibility to include a note in the README section of your repo specifying code source and its location within your repo**. This applies if you have worked with a peer on an assignment, used code in which you did not author or create sourced from a forum such as Stack Overflow, or you received code outside curriculum content from support staff such as an Instructor, TA, Tutor, or Learning Assistant. This will provide visibility to grading staff of your circumstance in order to avoid flagging your work as plagiarized.

If you are struggling with a challenge assignment or any aspect of the academic curriculum, please remember that there are student support services available for you:

1. Ask the class Slack channel/peer support.
2. AskBCS Learning Assistants exists in your class Slack application.
3. Office hours facilitated by your instructional staff before and after each class session.
4. **Tutoring Guidelines (https://docs.google.com/document/d/1hTldEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing)** - schedule a tutor session in the Tutor Sessions section of Bootcampspot - Canvas
5. If the above resources are not applicable and you have a need, please reach out to a member of your instructional team, your Student Success Advisor, or submit a support ticket in the Student Support section of your BCS application.

## References

Menne, M.J., I. Durre, R.S. Vose, B.E. Gleason, and T.G. Houston, 2012: An overview of the Global Historical Climatology Network-Daily Database. Journal of Atmospheric and Oceanic Technology, 29, 897-910, **https://doi.org/10.1175/JTECH-D-11-00103.1 (https://doi.org/10.1175/JTECH-D-11-**