9.5.1

Set Up the Database and Flask

With your Flask application set up, you know there is only one more step separating you and long days filled with surfing and ice cream on Oahu: creating the appropriate routes so that W. Avy's board of directors will be able to easily access the analysis. You know you'll want to put together a route for each segment of your analysis: Precipitation, Stations, Monthly Temperature, and Statistics, as well as a welcome route that will orient W. Avy and his associates to the webpage.

You know this task might be tough, but motivated by the fact that this is the final step, you refill your coffee and get back to making your dreams come true.

We've learned how to set up and create a Flask application. Now it's time to create our routes so that W. Avy's board of directors can easily access our analysis. We're so close, so stay focused because this will be good stuff! Let's begin by creating a new Python file and importing dependencies our app requires.

Set Up the Flask Weather App

We need to create a new Python file and import our dependencies to our code environment. Begin by creating a new Python file named `app.py`. This will be the file we use to create our Flask application.

Once the Python file is created, let's get our dependencies imported. The first thing we'll need to import is datetime, NumPy, and Pandas. We assign each of these an alias so we can easily reference them later. Add these dependencies to the top of your `app.py` file.

```
import datetime as dt
import numpy as np
import pandas as pd
```

Now let's get the dependencies we need for SQLAlchemy, which will help us access our data in the SQLite database. Add the SQLAlchemy dependencies after the other dependencies you already imported in `app.py`.

```
import sqlalchemy
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine, func
```

Finally, add the code to import the dependencies that we need for Flask. You'll import these right after your SQLAlchemy dependencies.

```
from flask import Flask, jsonify
```

Good work! Now that we've created the Python file and imported dependencies, we're ready to set up our database engine.

Set Up the Database

We'll set up our database engine for the Flask application in much the same way we did for `climate_analysis.ipynb`, so most of this setup process will be familiar. Add the following code to your file:

```
engine = create_engine("sqlite:///hawaii.sqlite")
```

What does the following line of code allow you to do?

```
engine = create_engine("sqlite:///hawaii.sqlite")
```

- [✓ Access the SQLite database.] ✓
- Change data in the SQLite database.
- Create a new Flask application.

Feedback
Correct. Nice work! With this code, you will be able to access the SQLite database.

Retake

The `create_engine()` function allows us to access and query our SQLite database file. Now let's reflect the database into our classes.

```
Base = automap_base()
```

Just as we did previously, we're going to reflect our tables.

What Python Flask function will you use to reflect the tables?

[✓ `Base.prepare()`] ✓

`Base.reflect()`

Feedback
Correct. Nice work! You will use this function to reflect the tables into SQLAlchemy.

Finish ▶

Add the following code to reflect the database:

```
Base.prepare(engine, reflect=True)
```

With the database reflected, we can save our references to each table. Again, they'll be the same references as the ones we wrote earlier in this module. We'll create a variable for each of the classes so that we can reference them later, as shown below.

```
Measurement = Base.classes.measurement
Station = Base.classes.station
```

Finally, create a session link from Python to our database with the following code:

```
session = Session(engine)
```

Next, we need to define our app for our Flask application.

Set Up Flask

To define our Flask app, add the following line of code. This will create a Flask application called "app."

```
app = Flask(__name__)
```

Notice the `__name__` variable in this code. This is a special type of variable in Python. Its value depends on where and how the code is run. For example, if we wanted to import our `app.py` file into another Python file named `example.py`, the variable `__name__` would be set to `example`. Here's an example of what that might look like:

```
import app

print("example __name__ = %s", __name__)

if __name__ == "__main__":
    print("example is being run directly.")
else:
    print("example is being imported")
```

However, when we run the script with `python app.py`, the `__name__` variable will be set to `__main__`. This indicates that we are not using any other file to run this code.

Now we're ready to build our Flask routes!