9.2.1 Retrieve the Precipitation Data

Your dependencies are added, you've connected the SQLite database, and your code is appropriately documented. It's time to start the analysis.

Now, let's think about precipitation. W. Avy is concerned about the amount of precipitation on Oahu. There needs to be enough rain to keep everything green, but not so much that you lose out on that ideal surfing and ice cream weather.

You know that you can set W. Avy's mind at ease by analyzing precipitation levels and showing him the cold, hard, data that backs up Oahu as the perfect place to surf. You have the last 12 months of precipitation data already loaded into your SQLite database, so you are ready to go.

W. Avy supplied you with the data he wants us to use and has asked you to look at a full year of data. When deciding how to parse the data, you remember that his favorite day is August 23, 2017 because it's the anniversary of the first time he ever went surfing and had ice cream on the same day. So, you decide to start the analysis there.

In the weather database, let's calculate the date one year from August 23,

2017. We'll be creating a variable called `prev_year` and using the datetime dependency that we imported previously.

The datetime dependency has a function called `dt.date()`, which specifies the date in the following format: year, month, day.

Find the Date One Year Ago

Add the most recent date, August 23, 2017, with the following code:

```
prev_year = dt.date(2017, 8, 23)
```

This code specifies the most recent date, but we want to calculate the date one year back. To do this, add the `dt.timedelta()` function to the previous line of code. This function allows us to trace back a certain number of days. In this case, we want to go back 365 days. Go ahead and add the `dt.timedelta()` function to your code.

```
prev_year = dt.date(2017, 8, 23) - dt.timedelta(days=365)
```

Now that we've got our date from the previous year (August 23, 2016), let's retrieve the amount of precipitation that was recorded, or the **precipitation score**.

Retrieve the Precipitation Scores

We'll begin by creating a variable to store the results of the query we'll write. This variable will be called `results`:

```
results = []
```

This code defines our new variable. Next, let's add our session that we created earlier so that we can query our database. For this we'll use the `session.query()` function, which is how we'll begin all of our queries in SQLAlchemy. From a bird's-eye view, this is how we query a SQLite database using Python.

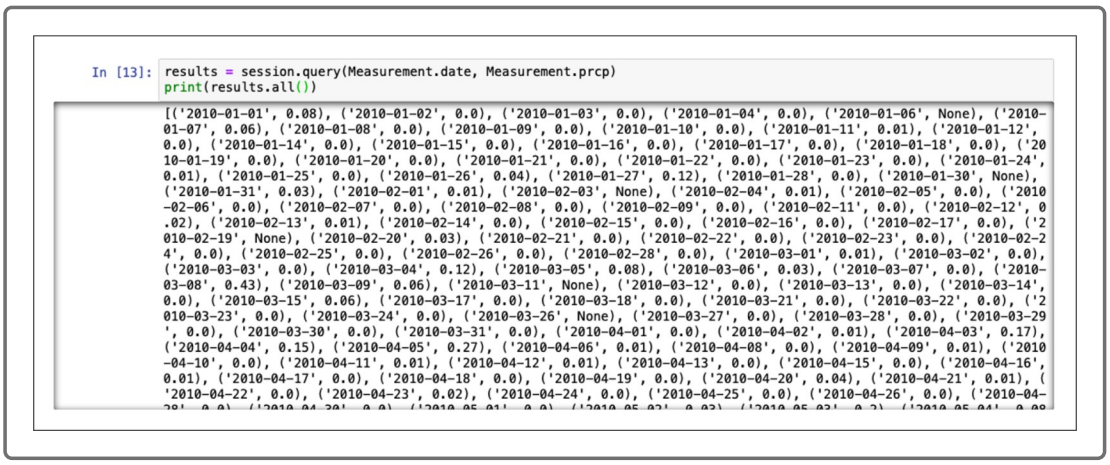
The `session.query()` function for this query will take two parameters. We will reference the Measurement table using `Measurement.date` and `Measurement.prcp`. Add the following to your code:

```
results = session.query(Measurement.date, Measurement.prcp)
```

Let's give this a shot and run the code. You might notice that there isn't anything returned. Let's go ahead and add a new line here. This will print everything that is returned in the query.

```
print(results.all())
```

Here's what you can expect to see.



We still have a few aspects to add to our query, but we'll get to that shortly.

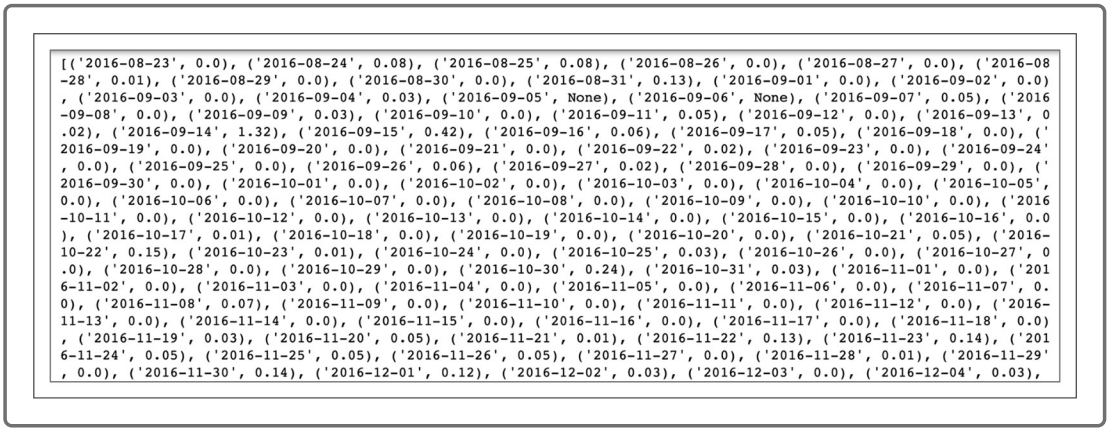
Since we only want to see the most recent data, we need to filter out all of the data that is older than a year from the last record date. We'll use the `filter()` function to filter out the data we don't need. Add the `filter()` function to the existing query.

```
results = session.query(Measurement.date, Measurement.prcp).filter(Measureme
```

One last thing: we'll add a function that extracts all of the results from our query and put them in a list. To do this, add `.all()` to the end of our existing query. All said and done, your query should look something like this:

```
results = session.query(Measurement.date, Measurement.prcp).filter(Measureme
```

Let's run this code. We'll print the results in order to ensure that we're getting output. Add `print(results)` after the last line of code. Here's what your results should look like:



```
[('2016-08-23', 0.0), ('2016-08-24', 0.08), ('2016-08-25', 0.08), ('2016-08-26', 0.0), ('2016-08-27', 0.0), ('2016-08
-28', 0.01), ('2016-08-29', 0.0), ('2016-08-30', 0.0), ('2016-08-31', 0.13), ('2016-09-01', 0.0), ('2016-09-02', 0.0)
, ('2016-09-03', 0.0), ('2016-09-04', 0.03), ('2016-09-05', None), ('2016-09-06', None), ('2016-09-07', 0.05), ('2016
-09-08', 0.0), ('2016-09-09', 0.03), ('2016-09-10', 0.0), ('2016-09-11', 0.05), ('2016-09-12', 0.0), ('2016-09-13', 0
.02), ('2016-09-14', 1.32), ('2016-09-15', 0.42), ('2016-09-16', 0.06), ('2016-09-17', 0.05), ('2016-09-18', 0.0), ('
2016-09-19', 0.0), ('2016-09-20', 0.0), ('2016-09-21', 0.0), ('2016-09-22', 0.02), ('2016-09-23', 0.0), ('2016-09-24'
, 0.0), ('2016-09-25', 0.0), ('2016-09-26', 0.06), ('2016-09-27', 0.02), ('2016-09-28', 0.0), ('2016-09-29', 0.0), ('
2016-09-30', 0.0), ('2016-10-01', 0.0), ('2016-10-02', 0.0), ('2016-10-03', 0.0), ('2016-10-04', 0.0), ('2016-10-05',
0.0), ('2016-10-06', 0.0), ('2016-10-07', 0.0), ('2016-10-08', 0.0), ('2016-10-09', 0.0), ('2016-10-10', 0.0), ('2016
-10-11', 0.0), ('2016-10-12', 0.0), ('2016-10-13', 0.0), ('2016-10-14', 0.0), ('2016-10-15', 0.0), ('2016-10-16', 0.0
), ('2016-10-17', 0.01), ('2016-10-18', 0.0), ('2016-10-19', 0.0), ('2016-10-20', 0.0), ('2016-10-21', 0.05), ('2016-
10-22', 0.15), ('2016-10-23', 0.01), ('2016-10-24', 0.0), ('2016-10-25', 0.03), ('2016-10-26', 0.0), ('2016-10-27', 0
.0), ('2016-10-28', 0.0), ('2016-10-29', 0.0), ('2016-10-30', 0.24), ('2016-10-31', 0.03), ('2016-11-01', 0.0), ('201
6-11-02', 0.0), ('2016-11-03', 0.0), ('2016-11-04', 0.0), ('2016-11-05', 0.0), ('2016-11-06', 0.0), ('2016-11-07', 0.
0), ('2016-11-08', 0.07), ('2016-11-09', 0.0), ('2016-11-10', 0.0), ('2016-11-11', 0.0), ('2016-11-12', 0.0), ('2016-
11-13', 0.0), ('2016-11-14', 0.0), ('2016-11-15', 0.0), ('2016-11-16', 0.0), ('2016-11-17', 0.0), ('2016-11-18', 0.0)
, ('2016-11-19', 0.03), ('2016-11-20', 0.05), ('2016-11-21', 0.01), ('2016-11-22', 0.13), ('2016-11-23', 0.14), ('201
6-11-24', 0.05), ('2016-11-25', 0.05), ('2016-11-26', 0.05), ('2016-11-27', 0.0), ('2016-11-28', 0.01), ('2016-11-29'
, 0.0), ('2016-11-30', 0.14), ('2016-12-01', 0.12), ('2016-12-02', 0.03), ('2016-12-03', 0.0), ('2016-12-04', 0.03),
```

IMPORTANT

> When you're handling a data analysis problem, printing your results is one of the most important tasks you can do. This can help you debug your code and ensure that you're getting all the data that you are expecting.
>
> You should print your results frequently so that you can make sure you're getting the data that you expect. Otherwise, you might spend hours working on code only to discover you're way off track.

Good work! Now that we've created the query, let's save it so that we can easily access it later, when we dive into Flask. Let's walk through how to do that now.