9.3.2

Determine the Most Active Stations

Determining how active the stations are will tell us how much data has been collected from each station. In this case, active essentially means the number of recordings for each station. This will help us figure out how reliable our data is, which, in turn, will boost W. Avy's confidence in his investment.

Now that we've found the total number of stations, we need to run a query to determine the most active stations. This query is a bit more complicated, but with your solid understanding of queries, you'll be able to master it!

Begin with the function we use to start every query in SQLAlchemy:

```
session.query()
```

Next, we need to add a few parameters to our query. We'll list the stations and the counts, like this:

```
session.query(Measurement.station, func.count(Measurement.station))
```

Now that we have our core query figured out, let's add a few filters to narrow down the data to show only what we need.

We want to group the data by the station name, and then order by the count for each station in descending order. We're going to add `group_by()` first.

REWIND

You have previously used the `groupby()` function. The `group_by()` function for SQLite follows the same idea and groups data similarly.

Here's what your code should look like:

```
session.query(Measurement.station, func.count(Measurement.station)).\
    group_by(Measurement.station)
```

Now let's add the `order_by` function. This function will order our results in the order that we specify, in this case, descending order. Our query results will be returned as a list.

After the code above, add `order_by(func.count(Measurement.station).desc())`, as shown below.

```
session.query(Measurement.station, func.count(Measurement.station)).\
group_by(Measurement.station).order_by(func.count(Measurement.station).de
```

Now we need to add the `.all()` function here as well. This will return all of the results of our query. This is what your query should look like:

```
session.query(Measurement.station, func.count(Measurement.station)).\
group_by(Measurement.station).order_by(func.count(Measurement.station).de
```

Good work! Run this query. Your results should look something similar to this:

```
[('USC00519281', 2772),
 ('USC00519397', 2724),
 ('USC00513117', 2709),
 ('USC00519523', 2669),
 ('USC00516128', 2612),
 ('USC00514830', 2202),
 ('USC00511918', 1979),
 ('USC00517948', 1372),
 ('USC00518838', 511)]
```

In the left column is the station ID, and on the right are the counts for each station. The counts indicate which stations are most active. We can also see which stations are the least active.

Great work! This was a complicated SQLAlchemy query, so you should feel accomplished.