9.3.4 Plot the Highest Number of Observations

W. Avy is thrilled. He's ready to make a decision about the surf shop location. But before he takes his proposal to the board of directors, he could benefit from a visualization of your results. You want to make sure that all stakeholders have all the information they need about the station closest to your proposed surf shop location.

You tell W. Avy that you're plotting the results of the analysis so that he can share a visual presentation with the board if needed, and convince them to invest in your shop.

We need to create a plot that shows all of the temperatures in a given year for the station with the highest number of temperature observations.

Create a Query for the Temperature Observations

To create a query, first select the column we are interested in. We want to pull `Measurement.tobs` in order to get our total observations count. Add this to your code:

```
session.query(Measurement.tobs)
```

Now filter out all the stations except the most active station with `filter(Measurement.station == 'USC00519281')`. Your code should look like this:

```
results = session.query(Measurement.tobs).\
filter(Measurement.station == 'USC00519281')
```

We need to apply another filter to consider only the most recent year. For this we can reuse some of the code we have written previously. Then we'll add the `.all()` function to save our results as a list. Here's what your query should look like:

```
results = session.query(Measurement.tobs).\
filter(Measurement.station == 'USC00519281').\
filter(Measurement.date >= prev_year).all()
```

To run this code, you will need to add a print statement around it.

```
print(results)
```

Your query results should look like the following.

Not too easy to read, right? Let's fix that, as investors will need to read this data.

To make the results easier to read, understand, and use, we'll put them in a DataFrame.

Convert the Temperature Observation Results to a DataFrame

REWIND

When creating a DataFrame, our first parameter is our list, and the second parameter is the column(s) we want to put our data in. In this case, we want to put our temperature observations result list into a DataFrame.

To convert the results to a DataFrame, add the following to your code:

```
df = pd.DataFrame(results, columns=['tobs'])
```

Add a `print(df)` statement after the last line and run the code. Below is what your data should now look like. Feel free to remove the index column.

Awesome! Now we'll use this data to create a plot.

Plot the Temperature Observations

We'll be creating a histogram from the temperature observations. This will allow us to quickly count how many temperature observations we have.

IMPORTANT

> A **histogram** is a graph made up of a range of data that is separated into different bins.

> When creating a histogram, you'll need to figure how many bins you need. It's recommended that you stay within a range of 5 to 20 bins. You may need to play around with the data a bit to find a good fit somewhere between 5 and 20. A "good fit" is one that represents the data well and highlights areas where there is a lot of data and areas where there is not a lot of data. It's all about finding the right balance.

We're going to divide our temperature observations into 12 different bins. This is intended to provide enough detail, but not too much. Note that we don't need to specify the ranges in which the data will be separated; we just need to specify the number of bins.

To create the histogram, we need to use the `plot()` function and the `hist()` function and add the number of bins as a parameter. Add the following to your code:

```
df.plot.hist(bins=12)
```

Using `plt.tight_layout()`, we can compress the x-axis labels so that they fit into the box holding our plot.

```
plt.tight_layout()
```

For this particular graph, using this function won't change much, but it can be a lifesaver in situations where the x-axis doesn't fit into the box. It's a cosmetic change, but it makes a big difference when presenting professional work.

When you run the code, your plot should look like the following. Notice how the 12 "bins" are visualized in this plot, just like you specified with your code `df.plot.hist(bins=12)`. "Bin" refers to each rectangular column in the plot, as shown below.

Looking at this plot, we can infer that a vast majority of the observations were over 67 degrees. If you count up the bins to the right of 67 degrees, you will get about 325 days where it was over 67 degrees when the temperature was observed.

ADD/COMMIT/PUSH

The weather station analysis is complete! Now is a good time to add, commit, and push your updates to GitHub. Remember the steps:

1. `git add < FILENAME>`
2. `git commit -m "< ADD COMMIT MESSAGE HERE>"`
3. `git push origin main`

Now test your skills in the following Skill Drill.

SKILL DRILL

Adjust the number of bins in the plot to 5, and then adjust the number to 20. Take note of any differences in the plot caused by changing the number of bins.

Our work with the precipitation and station analysis is complete, so let's share our findings!