1. 1 point

Activity overview

In previous lessons, you learned about temporary tables. In this activity, you will practice making one and using it to run a query.
By the time you complete this activity, you will be able to use temporary tables to work with data without changing the original data. This will help you complete more complicated analytical tasks in your career as a data analyst.

What are temp tables?

As data calculations become more complicated, there are many components to keep track of. This is similar to keeping track of tasks in daily life. Some people use sticky notes while others use checklists. In data science, a temporary table is just like a sticky note.
Temporary tables, or temp tables, store subsets of data from standard data tables for a certain period of time. When you end your SQL database session, they are automatically deleted. Temp tables allow you to run calculations in temporary data tables without needing to make modifications to the primary tables in your database.
Now, you will practice creating a temp table.

Importing data

To begin, import your data. You will use a dataset on bikesharing in Austin, Texas. Specifically, you will work with a table that gives details about each public bike ride's duration, starting station, and ending station.
To load your data, follow these steps:

1. Log in to and open the BigQuery Console. If you have a free trial version of BigQuery, you can use that instead. On the BigQuery page, click the Go to BigQuery button.

- Note: BigQuery frequently updates its user interface. The latest changes may not be reflected in the screenshots presented in this activity, but the principles remain the same. Adapting to changes in software updates is an essential skill for data analysts, and it's helpful for you to practice troubleshooting. You can also reach out to your community of learners on the discussion forum for help.

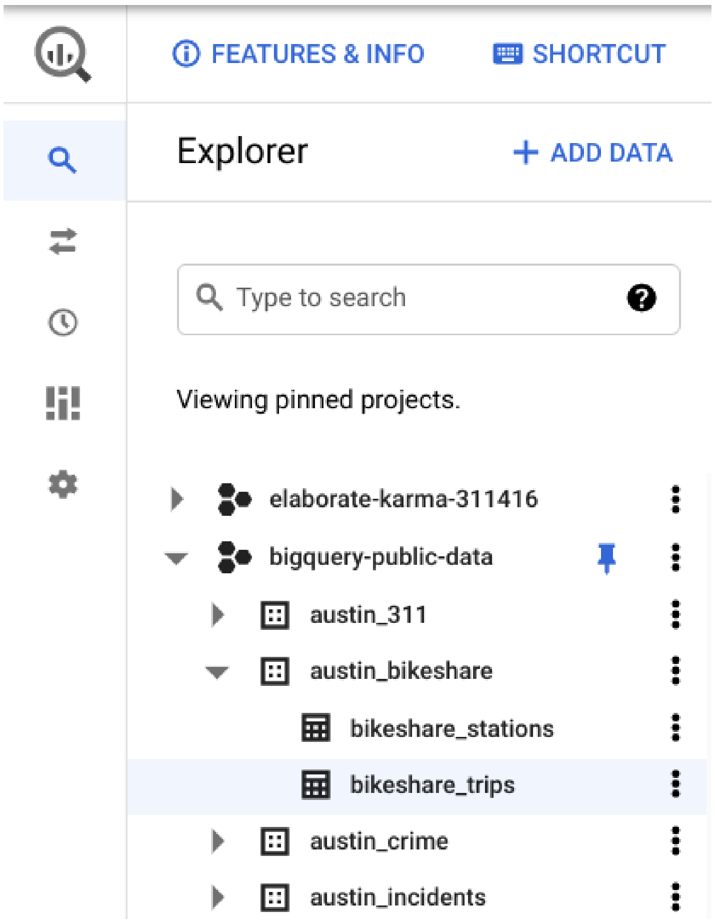
2. If you have never created a BigQuery project before, click CREATE PROJECT on the right side of the screen. If you have created a project before, you can use an existing one or create a new one by clicking the project dropdown in the blue header bar and selecting NEW PROJECT.
3. Name your project something that will help you identify it later. You can give it a unique project ID or use an auto-generated one. Don't worry about selecting an organization if you don't know what to put.
4. Now, you'll see the Editor interface. In the middle of the screen is a window where you can type code, and to the left is the Explorer menu where you can search for datasets.

- In order to find the austin_bikeshare dataset, you will want to make sure you have the bigquery-public-data pinned in your explorer menu. Follow these steps to pin the dataset:

1. Navigate to the Explorer menu in BigQuery.
2. Type the word public in the search box and enter.
3 Click "Broaden search to all projects"
4. Find the bigquery-public-data and pin it.

To examine the dataset, follow these steps:

1. Make sure that bigquery-public-data is pinned to the Explorer menu of your SQL Workspace.
2. Open the bigquery-public-data dropdown in the Explorer menu and scroll until you find austin_bikeshare.
3. Open the dropdown and click bikeshare_trips to open and examine the dataset.



4. Click on the Preview tab in the viewer on the right, then examine the dataset.

You'll need to use the WITH clause to create a temporary table so you can find at which station the longest (duration) bike ride started.

Create a temporary table

Consider the following scenario: A bikeshare company has reached a recent milestone, and their marketing team wants to write a blog post that "congratulates" their most-used bike on being so popular. They want to include the name of the station that the bike is most likely to be found.

They task you with figuring out the station from which the bike begins a trip most frequently.

In order to do this, you will need to create a temp table to find the ID number of the bike that has taken the longest total trips (in minutes). You will take a sum of the minutes of each trip for each bike, then sort by descending order to find the bike that has spent the most minutes being used.

To do that, complete the following steps:

1. Return to your Editor tab or click Compose new query.

2. Begin your query with WITH to set up a temp table. Then on a new indented line, create the name of your temp table. Make sure your table name is in the proper snake case (with underscores between each word). Name it longest_used_bike. Then add a space.

3. Type AS and an open parenthesis (, then press Enter (Windows) or Return (Mac) to create a new indented line.

4. Type SELECT, then press Enter or Return and Tab to create a new indented line.

5. Type bikeid and a comma. Then press Enter or Return to create a new line and type SUM(duration_minutes) AS trip_duration. This creates a column in the temp table that contains the sum of the total minutes a bike has been used. Press Enter or Return again, then press Backspace to align the cursor with SELECT.

6. Type FROM, then press Enter or Return and Tab to create a new indented line.

7. Specify the dataset you'll be using. To do this, type bigquery-public-data.austin_bikeshare.bikeshare_trips. Press Enter or Return, then press Backspace to align the cursor with SELECT.

Your text should appear like this:

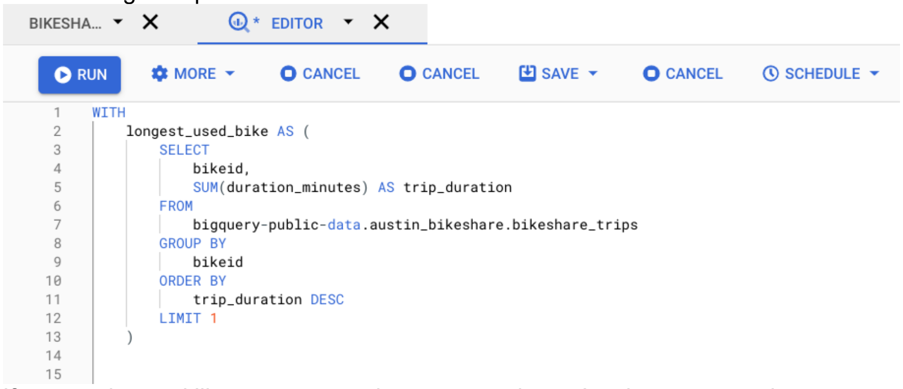
```sql
WITH longest_used_bike AS(
    SELECT
        bikeid,
        SUM(duration_minutes) AS trip_duration
    FROM
        bigquery-public-data.austin_bikeshare.bikeshare_trips

```

8. Type GROUP BY, then press Enter or Return and Tab to create a new indented line.
9. Type bikeid to group the data by the column bikeid. Press Enter or Return again, then press Backspace to align the cursor with SELECT.
10. Type ORDER BY, then press Enter or Return and Tab to create a new indented line.
11. Type trip_duration DESC to sort the data in descending order by the column trip_duration. Press Enter or Return again, then press Backspace to align the cursor with SELECT.
12. Type LIMIT 1.

13. Make sure there is a closed parenthesis) on the next line. If it is not there, add it.

This sets up your temporary table. This section identifies the specific bike (bikeid) with the longest trip duration.



```sql
WITH
    longest_used_bike AS (
        SELECT
            bikeid,
            SUM(duration_minutes) AS trip_duration
        FROM
            bigquery-public-data.austin_bikeshare.bikeshare_trips
        GROUP BY
            bikeid
        ORDER BY
            trip_duration DESC
        LIMIT 1
    )


```

If you run it now, it'll return an error because you haven't written any queries yet. Now, it's time to write a query that identifies the start station that this bike came out of.
Write your query

Now that you have found the ID of the bike that has been used the longest, you will write a query to find the station from which this bike leaves most frequently. To do this, you will join your temp table (containing just the bike's ID) with the original table and return the station ID with the highest number of trips started.
To find this station ID, follow these steps:

1. On a new line, type two # signs to begin a comment.

2. Describe the purpose of your query. This will help you remember the purpose of your query as you're writing it. It can also help you share your work with others. In this case, type find station at which the longest-used bike leaves most often or something similar. Then press Enter or Return to make a new line.
3. Begin the query with SELECT, then press Enter or Return and Tab to create a new indented line.
4. Type trips.start_station_id, and a comma. This line contains the start_station_id column from the trips table, which you will define with an alias later in this query. Then press Enter or Return to create a new line.
5. Type COUNT(*) AS trip_ct. This line will help you count how many times the bike has left each station. Press Enter or Return again, then press Backspace to align the cursor with SELECT.
6. Type FROM, then press Enter or Return and Tab to create a new indented line.

7. Type longest_used_bike AS longest to rename your temp table with an alias. Then press Backspace to align the cursor with SELECT.
Your text should appear like this:

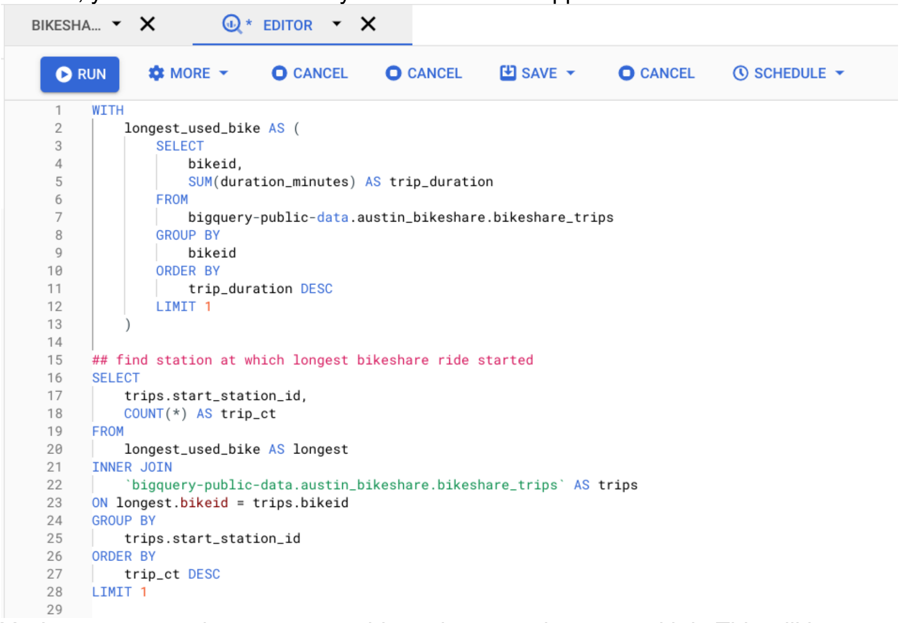
```sql
## find the station at which longest bikeshare ride started
SELECT
    trips.start_station_id,
    COUNT(*) AS trip_ct
FROM
    longest_used_bike AS longest

```

Now, it's time to write an INNER JOIN, which you will use to pick out the station ID that corresponds to the bike you identified in the temporary table.
8. Type INNER JOIN. Press Enter or Return and Tab to create a new indented line.

9. Type `bigquery-public-data.austin_bikeshare.bikeshare_trips` AS trips. Press Enter or Return again, then press Backspace to align the cursor with SELECT.
10. Type ON longest.bikeid = trips.bikeid. This specifies that the JOIN is on the bikeid column in the temp table you created and the original dataset. Then press Enter or Return to create a new line.
11. Type GROUP BY, then press Enter or Return and Tab to create a new indented line.
12. Type trips.start_station_id to group by the start_station_id column in the original dataset. Press Enter or Return again, then press Backspace to align the cursor with SELECT.
13. Type ORDER BY, then press Enter or Return and Tab to create a new indented line.
14, Type trip_ct DESC to sort by the trip_ct column in descending order. Press Enter or Return again, then press Backspace to align the cursor with SELECT.
15. Type LIMIT 1.

16. Finally, click Run. The query might take a few seconds before showing you the count. If your query returns 2575 in the start_station_id column and 90 in the trip_ct column, you've written it correctly. Your text should appear like this:



```
WITH
    longest_used_bike AS (
        SELECT
            bikeid,
            SUM(duration_minutes) AS trip_duration
        FROM
            bigquery-public-data.austin_bikeshare.bikeshare_trips
        GROUP BY
            bikeid
        ORDER BY
            trip_duration DESC
        LIMIT 1
    )

## find station at which longest bikeshare ride started
SELECT
    trips.start_station_id,
    COUNT(*) AS trip_ct
FROM
    longest_used_bike AS longest
INNER JOIN
    `bigquery-public-data.austin_bikeshare.bikeshare_trips` AS trips
ON longest.bikeid = trips.bikeid
GROUP BY
    trips.start_station_id
ORDER BY
    trip_ct DESC
LIMIT 1
```

You've now created a temporary table and executed a query with it. This will be helpful when you are performing several calculations at once.
Other types of temp tables

There are also other ways to create a temp table. Instead of using the WITH clause, you can use the SELECT INTO or the CREATE TABLE clauses.
The SELECT INTO clause copies data from one table into a new table, but doesn't add the new table to the database. It's useful if you want to make a copy of a table with a specific condition.
The CREATE TABLE clause is a good option when several people need to access the same temp table. This statement adds the table into the database.
Which clause you use depends on your preference and the project's demands. Different clauses have their own strengths, so understanding how each of them work is helpful for using them effectively.
Confirmation and reflection

In a past activity, you learned about the importance of using the right type of join. In this activity, you wrote a query with an INNER JOIN to join your temporary table with the original bikeshare_trips table. Which station ID would your query return if you used a FULL JOIN instead of an INNER JOIN?

- ○ 3798
- ○ 2758
- ○ 2575
- ○ 3575

2. In this activity, you created a temporary table to run calculations without needing to make modifications to the primary tables in your database. In the text box below, write 2-3 sentences (40-60 words) in response to each of the following questions:
- Why was the JOIN statement necessary to use in this activity?

- What is the benefit of executing a query in a temporary table rather than a primary table in a database?