Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 100%. We keep your highest score.

Next item →

Activity Overview

As data calculations become more complicated, there are many components to keep track of, such as range, cost, time elements, products, and more. Some people use sticky notes for this, while others use checklists. In the data profession, a temporary table is just like a sticky note.

You learned about temporary tables in SQL in earlier lessons, so take a moment to review. **Temporary tables**, or **temp tables**, store subsets of data from standard data tables for a certain period of time. Temp tables allow you to run calculations in temporary data tables without needing to make modifications to the primary tables in your database. Because they are temporary, they are automatically deleted at the end of your SQL session.

By the end of this activity, you will have gained more experience creating temp tables and using them to run queries.

Scenario

Review the following scenario. Then complete the step-by-step instructions.

A bikeshare company has reached a milestone, and their marketing team wants to write a blog post announcing the popularity of their most-used bike. They want to include the name of the station where that bike can most likely be found, so they ask you to determine which bike is used most often.

Step-By-Step Instructions

Follow the instructions to complete each step of the activity. Then answer the questions at the end of the activity before going to the next course item.

Step 1: Import your data

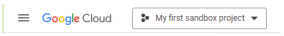
The first step is to import your data. For this activity, you will use a BigQuery dataset on bikesharing in Austin, Texas, which contains details about each public bike ride's duration, starting station, and ending station.

To load your data, follow these steps:

1. Log into BigQuery and open your Console.

- **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.

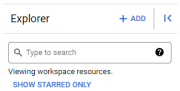
2. Select the project dropdown list to open the Select a project dialog box.



3. In the **Select a project** dialog box, select the **NEW PROJECT** button.

4. 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.

5. Now, access the Editor interface. You will use the Explorer menu to search for datasets.



Step 2: Pin the dataset

Note: Before you can load the `austin_bikeshare` dataset, you need to have the bigquery-public-data pinned in the Explorer menu of your SQL Workspace. Follow these steps to pin the dataset:

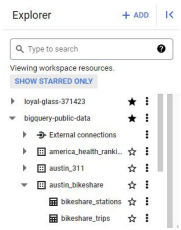
1. Enter **public** in the Explorer menu search box and press the **Enter** key.

2. Select **SEARCH ALL PROJECTS**.

3. Scroll to find **bigquery-public-data** and select the star to pin it.

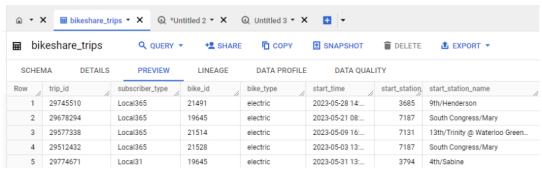
After you've highlighted the star and pinned the bigquery-public-data to the Explorer menu, you'll need to add the Austin bike dataset. Follow these steps:

1. Select the arrow to the left of bigquery-public-data to expand its contents and scroll to find **austin_bikeshare**.



2. Expand **austin_bikeshare** and select **bikeshare_trips**.

3. Select the **Preview** tab in the viewer on the right to examine the dataset.



Note: Many of the public databases on BigQuery are living records and, as such, are periodically updated with new data. Throughout this course, if your results differ from those you encounter in videos or screenshots, there's a good chance it is due to a data refresh.

Step 3: Create a temporary table

Now search the data to give the bikeshare company the information it needs for its blog post. Make sure it includes the name of the station where the bike is used most often.

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 was used for the most minutes. Follow these steps:

1. Return to your Editor tab or select **Compose new query** to open a query window.

2. Use `WITH` at the start of your query to set up a temp table and name your table **longest_used_bike** on a new indented line. Note: Always be sure to use the proper snake case (underscores between each word).

3. Add a space.

4. Enter `AS` and an open parenthesis [`(`]. Press **Enter** (Windows) or **Return** (Mac) to create a new indented line.

5. Enter `SELECT`, then press **Enter/Return**, then **Tab** to create a new indented line.

6. Enter `bike_id` followed by a comma and press **Enter/Return** to create a new line.

7. Enter `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/Return**.

8. Press **Backspace** to align the cursor with `SELECT`, then enter `FROM`. Press **Enter/Return**, then **Tab** to create a new indented line.

9. Enter `bigquery-public-data.austin_bikeshare.bikeshare_trips` to specify the dataset you'll be using.

10. Press **Enter/Return**, then **Backspace** to align the cursor with `SELECT`. Currently, your query should be something like this:

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

11. Continue writing your query by entering `GROUP BY`, then press **Enter/Return**, followed by **Tab** to create a new indented line.

12. Enter `bike_id` to group the data by the bike_id column and press **Enter/Return**.

13. Press **Backspace** to align the cursor with `SELECT` and enter `ORDER BY`. Then press **Enter/Return** followed by **Tab** to create a new indented line.

14. Enter `trip_duration DESC` to sort the data in descending order by the column trip_duration.

15. Press **Enter/Return**, followed by **Backspace** to align the cursor with `SELECT`.

16. Enter `LIMIT 1` and **Enter/Return**.

17. Press **Backspace**, then add a closed parenthesis [)] if there isn't one already on the next line. Now your text should be as follows:

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

This is how you set up your temporary table to identify the specific bike (bike_id) with the longest trip duration. However, if you run it now, it will return an error because you haven't written any queries for the temp table yet.

Step 4: Write your query

Now that you have written the portion of the query to createa temp table that will show the bike ID of the bike that was used most often, you need to write a query to find the station. To do this, join your temp table with the original table and return the station ID with the highest number of trips started.

But first, create a comment describing the purpose of the query. This will help you remember it as you're writing it, and it enables others to understand your work if you share it.

On a new line, enter two hashtags (`##`) to begin the comment and enter something like, `find station where the longest-used bike leaves most often.` Press **Enter/Return** to make a new line and follow these steps to write your query:

1. Enter `SELECT`, then **Enter/Return** followed by **Tab** to create a new indented line.

2. Enter `trips.start_station_id` followed by a comma. Then press **Enter/Return** to create a new line.

3. Enter `COUNT(*) AS trip_ct`. This will count the number of times the bike has left each station.

4. Press **Enter/Return**, followed by **Backspace** to align the cursor with `SELECT`.

5. Enter `FROM`, then press **Enter/Return** followed by **Tab** to create a new indented line.

6. Enter `longest_used_bike AS longest` to rename your temp table with an alias.

7. Press **Enter/Return** followed by **Backspace** to align the cursor with `SELECT`. At this point, your query should be this:

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


## find the station where the longest-used bike leaves most often
SELECT
  trips.start_station_id,
  COUNT(*) AS trip_ct,
FROM
  longest_used_bike AS longest
```

Next, write an `INNER JOIN` to pick out the station ID that corresponds to the bike you identified in the temporary table. Follow these steps:

1. Enter `INNER JOIN`. Press **Enter/Return**, then **Tab** to create a new indented line.

2. Enter `bigquery-public-data.austin_bikeshare.bikeshare_trips AS trips`.

3. Press **Enter/Return** followed by **Backspace** to align the cursor with `SELECT`.

4. Enter `ON longest.bike_id = trips.bike_id`. This specifies that the `JOIN` is on the bike_id column in the temp table you created and in the original dataset. Press **Enter/Return** to create a new line.

5. Enter `GROUP BY`, then press **Enter/Return** followed by **Tab** to create a new indented line.

6. Enter `trips.start_station_id` to group by the start_station_id column in the original dataset.

7. Press **Enter/Return** followed by **Backspace** to align the cursor with `SELECT`.

8. Enter `ORDER BY`, then press **Enter/Return** followed by **Tab** to create a new indented line.

9. Enter `trip_ct DESC` to sort by the trip_ct column in descending order.

10. Press **Enter/Return** followed by **Backspace** to align the cursor with **SELECT**.

11. Enter `LIMIT 1`. And with that, you've completed the query! The full query that creates a temporary table and then queries it should be this:

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


## find the station where the longest-used bike leaves most often
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.bike_id = trips.bike_id
GROUP BY
```

```
  trips.start_station_id
ORDER BY
  trip_ct DESC
  LIMIT 1
```

Now, select **Run**. The dataset is substantial, so the query might take a few seconds before showing you the count.

Your query should return 3798 in the start_station_id column and 177 in the trip_ct column. If your query returned a different result, review your steps, comparing them to the reading and checking for typos. Note, however, that this public dataset is updated from time to time and the results may change!

Step 5: Review other types of temp tables

There are even more 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.

Different clauses have their own strengths, so understanding how they work is helpful for using them effectively. The specific clause you use will depend on your preferences and one the project's demands.

1. **Reflection** — 1 / 1 point

How would you change the query to find the *least frequent* starting station of the bike ridden for the most time? Choose the correct code revision.

- (●)
```
-- In the second occurrence of ORDER BY:
ORDER BY
    trip_ct ASC
```

- ( )
```
-- In the first occurrence of ORDER BY:
ORDER BY
    trip_duration ASC
```

- ( )
```
ORDER BY
    trip_duration ASC
...
ORDER BY
    trip_ct ASC
```

- ( )
```
SELECT
    bike_id,
    MIN(duration_minutes) AS trip_duration
```

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: — 1 / 1 point

- 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?

The JOIN statement was necessary to link data from two or more separate tables based on a common key, such as an ID. This allowed us to combine related records into a single, cohesive dataset so that meaningful calculations and analysis could be performed on all the relevant information at once. Executing a query in a temporary table is beneficial because it acts as a safe, isolated sandbox for your calculations. This prevents you from accidentally modifying or corrupting the essential, permanent data stored in the primary tables, ensuring the integrity and stability of the main database.

In this activity, you used a temporary table to write a query. A good response would include how temporary tables are extremely helpful for complex calculations and queries.