Hands-On Intro to SQL

ITP Camp 2019
Alice Liang

What is SQL?

- Structured query language
- Many dialects of SQL, today we’ll be using BigQuery
    - You might have heard of PostgreSQL, MySQL, MS SQL, Oracle, etc.
- Why not Excel?
    - Excel can only process ~1M rows and ~16K columns
    - SQL is the industry standard in analytics positions
    - SQL can more effectively & efficiently handle larger volumes of data for data cleaning and processing
    - SQL is easier to document and pass onto your teammates, assuming they know SQL
- Why not Python/etc?
    - SQL can be much faster!
    - Database storage
    - Easy to transfer outputs between languages

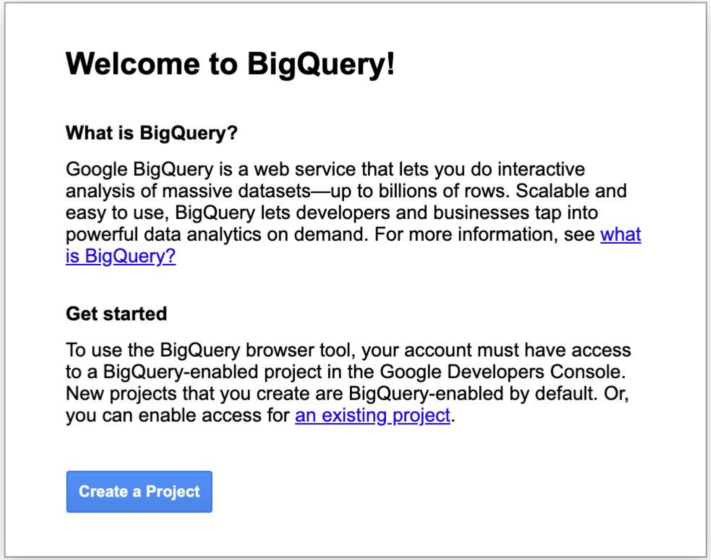
Overview

- Set Up (~15 mins)
- Basic Querying (~45 minutes)
- Exercises (~15 minutes)
- Advanced Querying (~15 minutes if time allows)

Set Up

Hang in there

Go to https://console.cloud.google.com/bigquery
Accept Terms of Service
Create a Project: Each Google account can make 12 projects



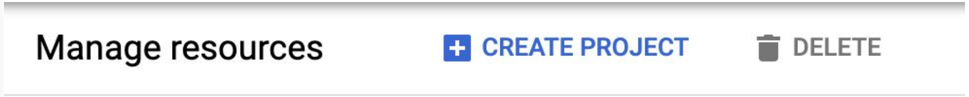
Welcome to BigQuery!

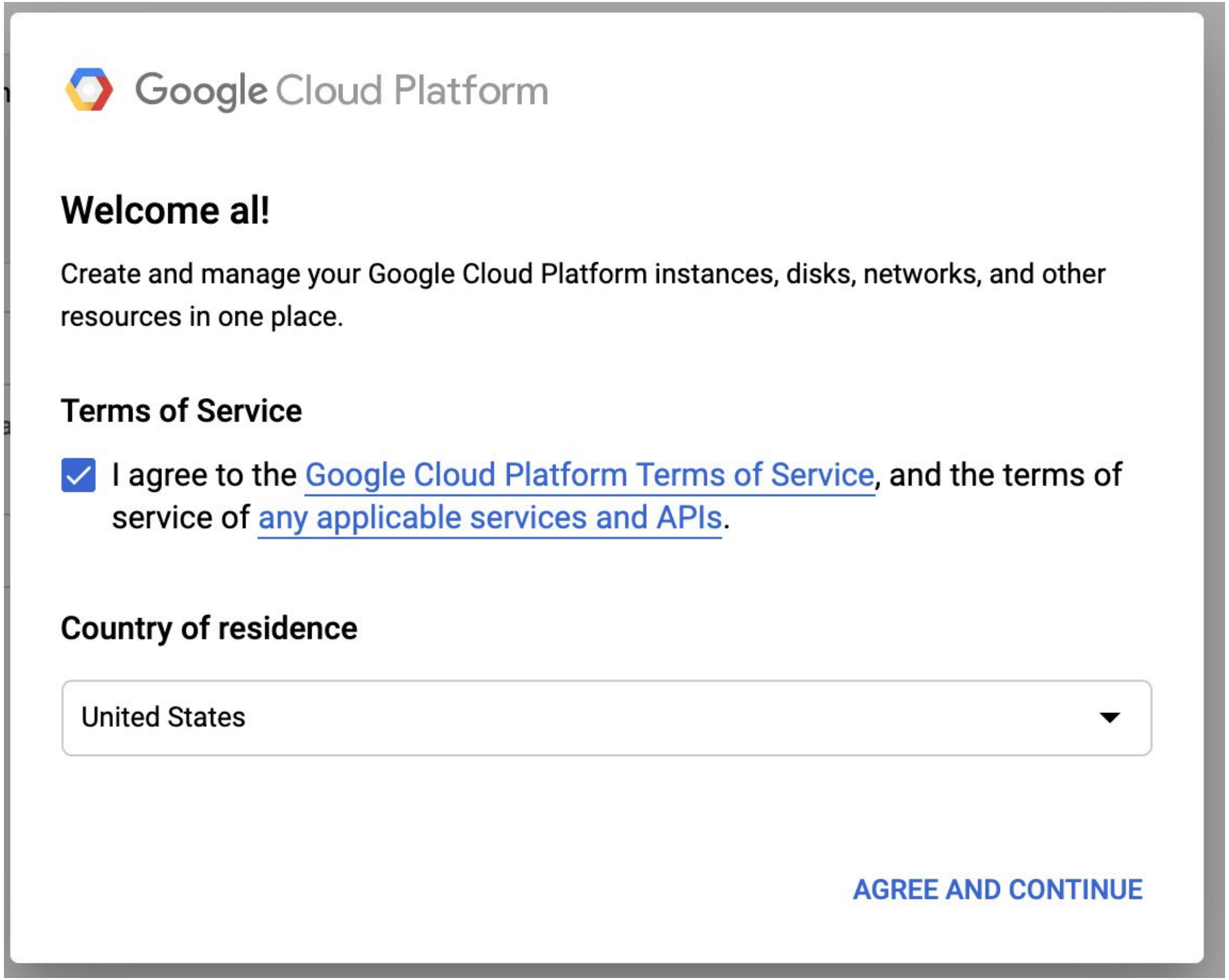
What is BigQuery?

Google BigQuery is a web service that lets you do interactive analysis of massive datasets—up to billions of rows. Scalable and easy to use, BigQuery lets developers and businesses tap into powerful data analytics on demand. For more information, see what is BigQuery?

Get started

To use the BigQuery browser tool, your account must have access to a BigQuery-enabled project in the Google Developers Console. New projects that you create are BigQuery-enabled by default. Or, you can enable access for an existing project.





Welcome al!

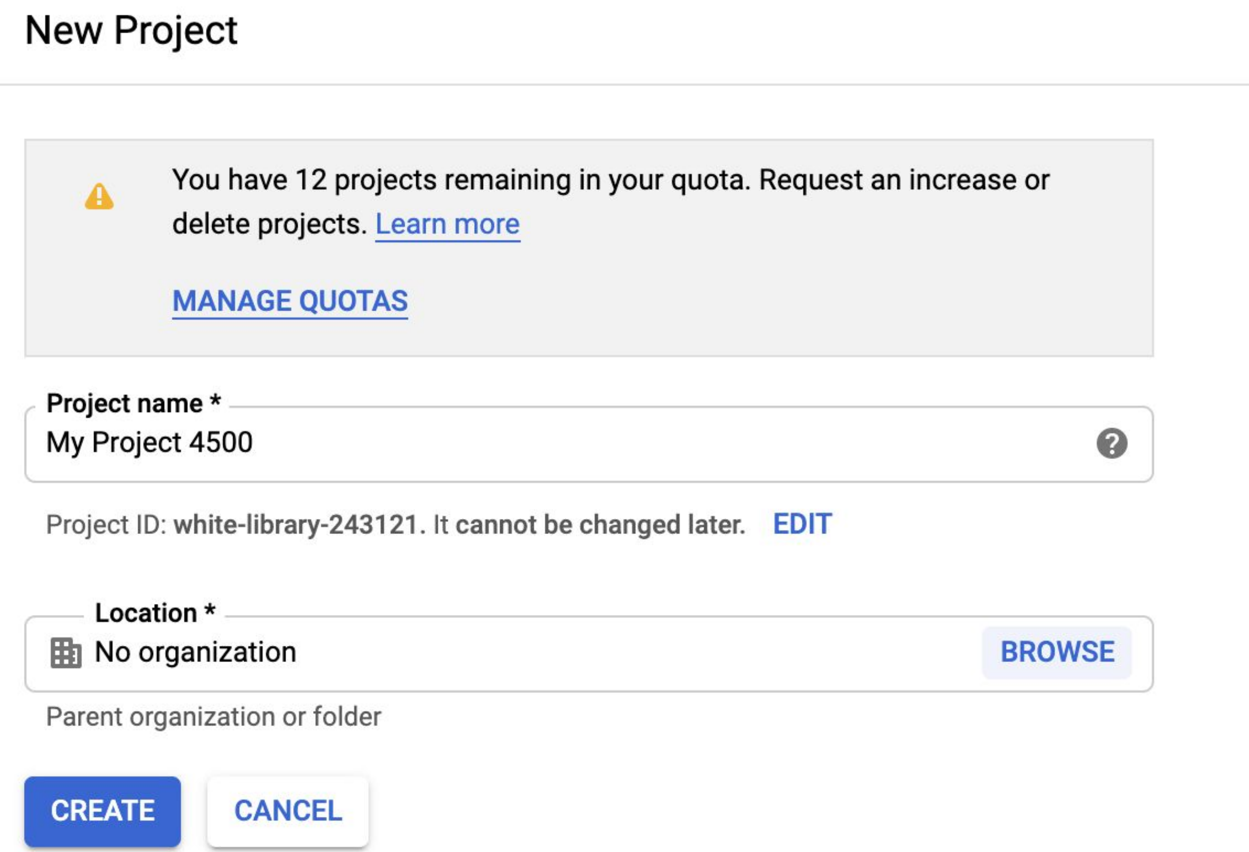
Create and manage your Google Cloud Platform instances, disks, networks, and other resources in one place.

Terms of Service

☑ I agree to the Google Cloud Platform Terms of Service, and the terms of service of any applicable services and APIs.

Country of residence

United States ▼



New Project

⚠ You have 12 projects remaining in your quota. Request an increase or delete projects. Learn more

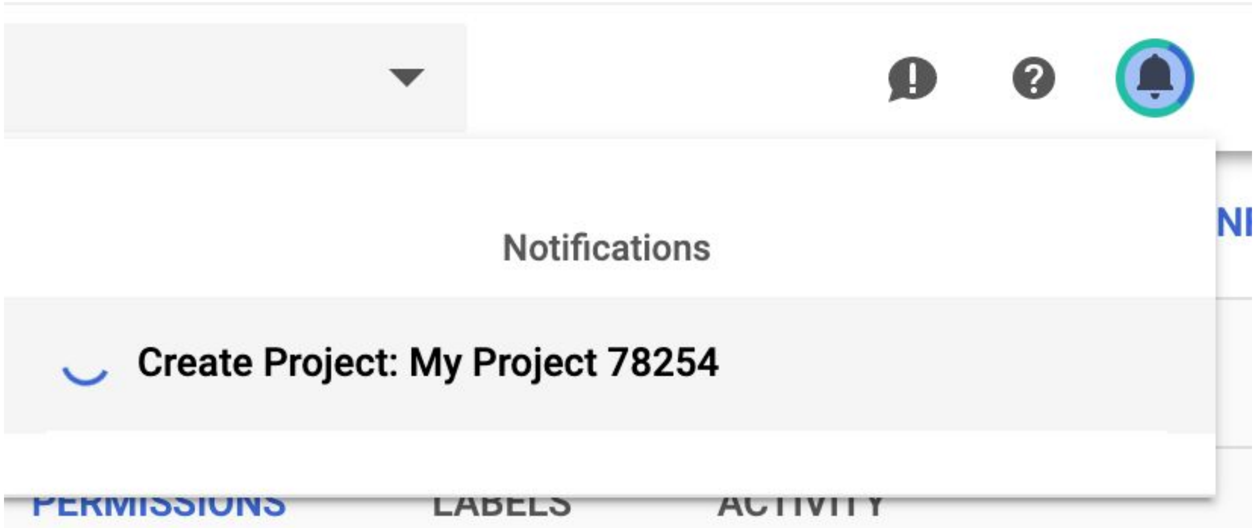
MANAGE QUOTAS

Project name *
My Project 4500

Project ID: white-library-243121. It cannot be changed later. EDIT

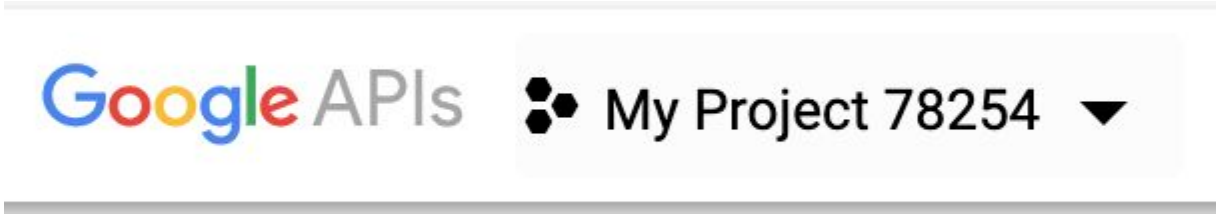
Location *
No organization

Parent organization or folder



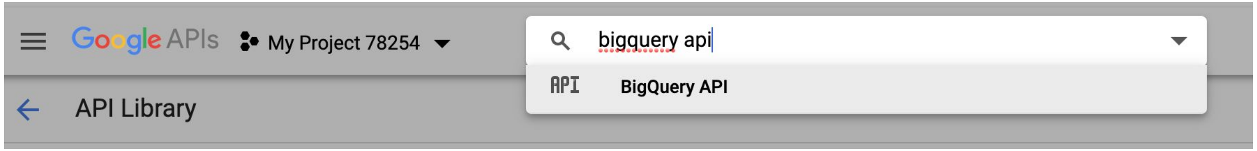
Notifications

Create Project: My Project 78254

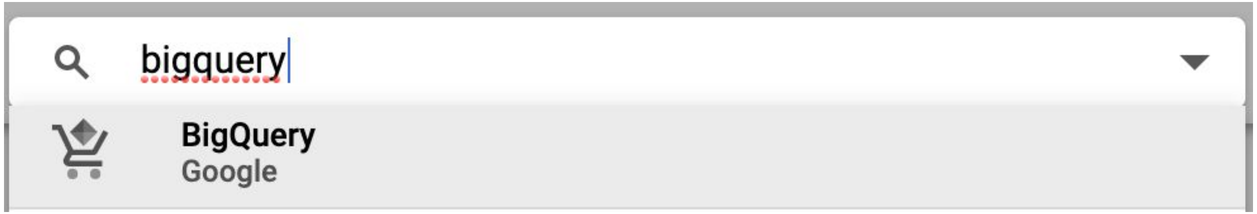


My Project 78254

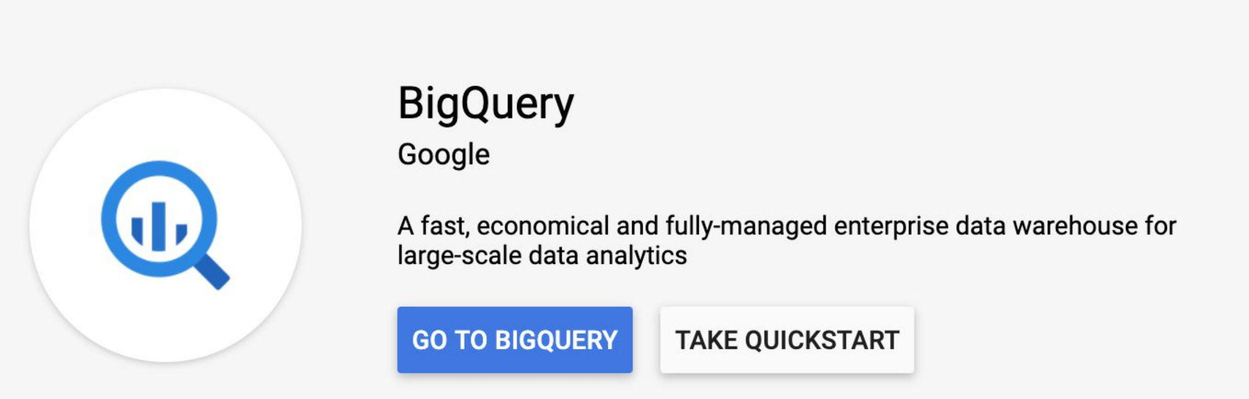
Enable the BigQuery API



bigquery api



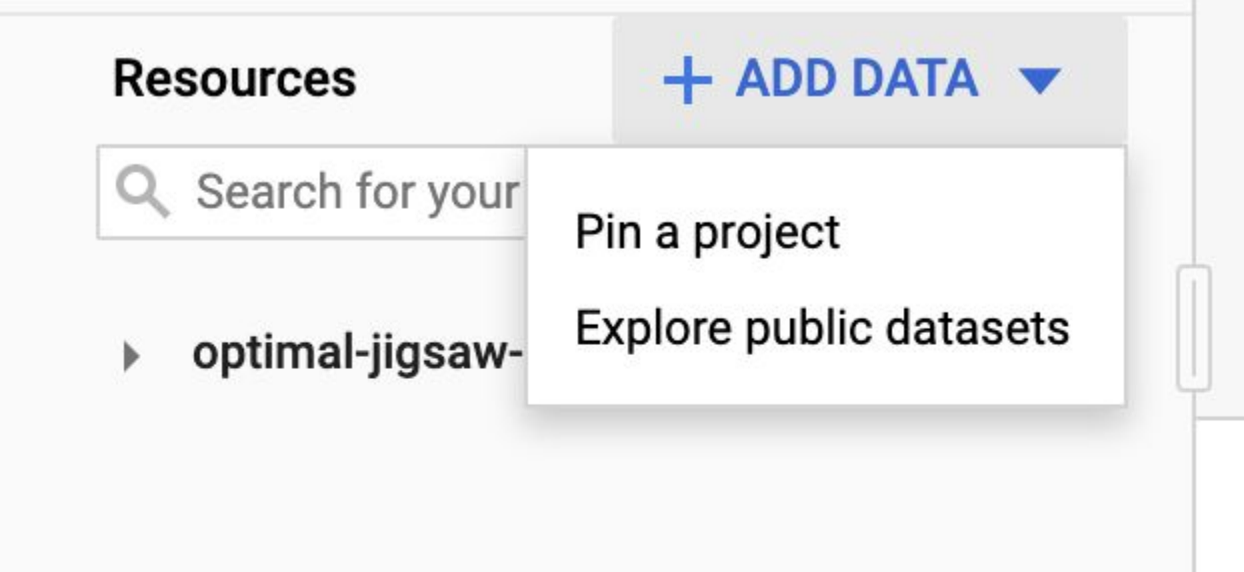
bigquery

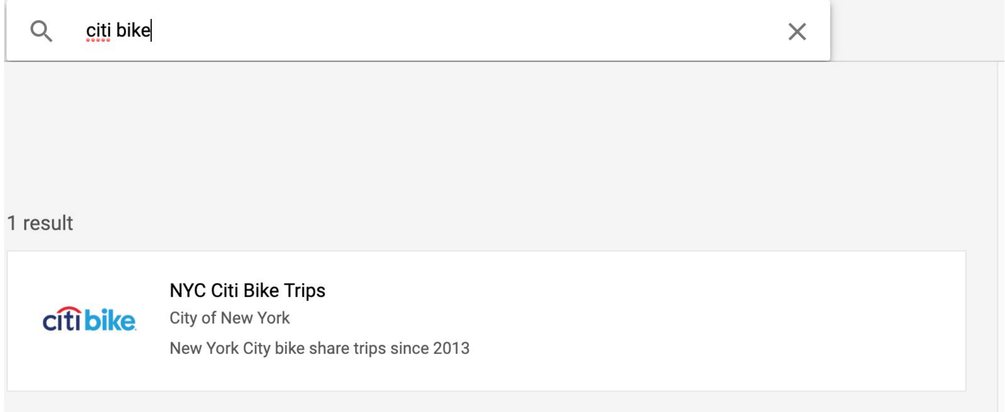


BigQuery

Google

A fast, economical and fully-managed enterprise data warehouse for large-scale data analytics

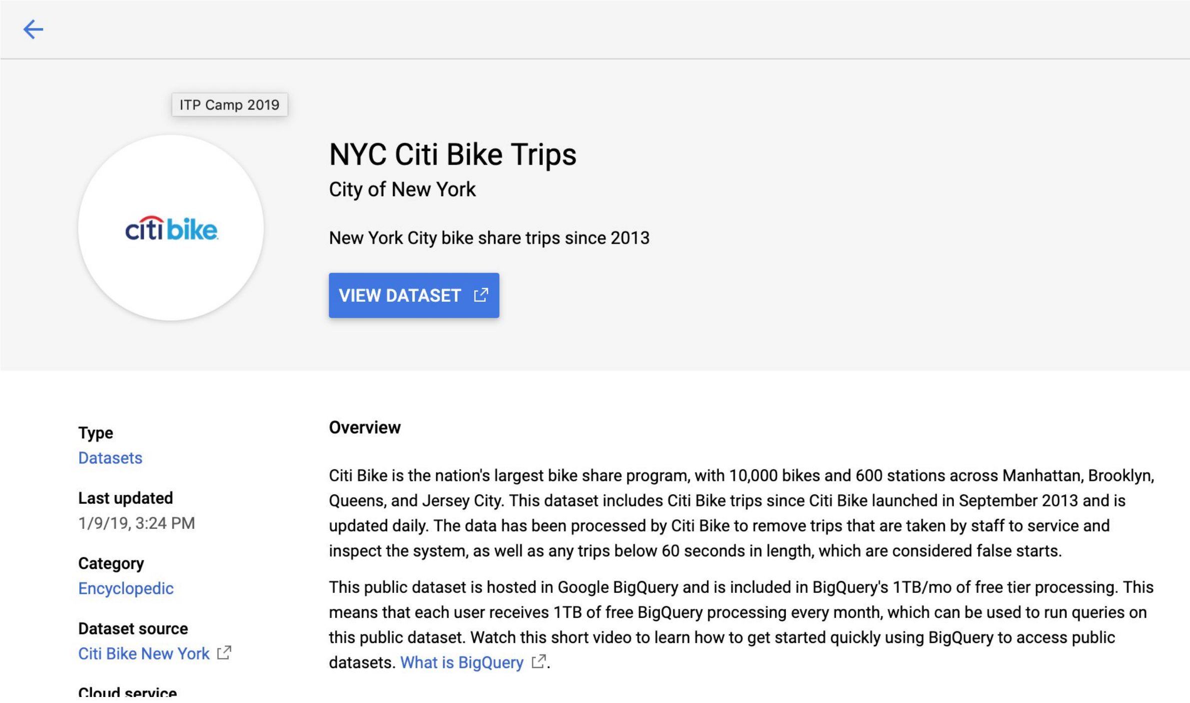




citi bike

1 result

NYC Citi Bike Trips
City of New York
New York City bike share trips since 2013



ITP Camp 2019

NYC Citi Bike Trips

City of New York

New York City bike share trips since 2013

Type
Datasets

Last updated
1/9/19, 3:24 PM

Category
Encyclopedic

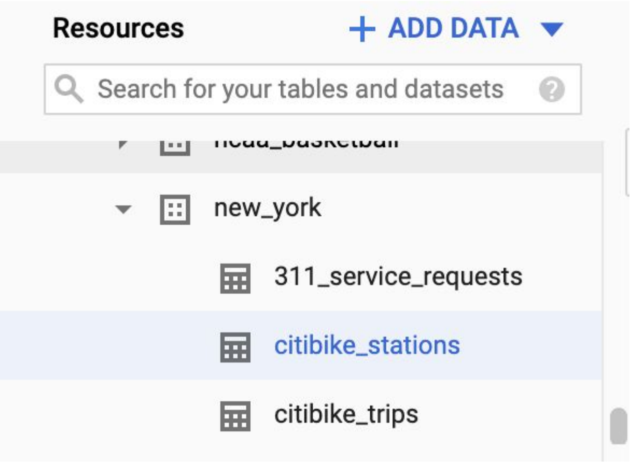
Dataset source
Citi Bike New York

Cloud service

Overview

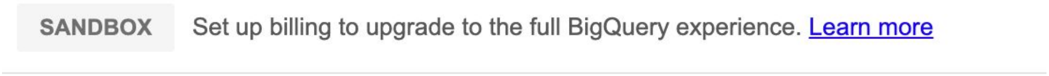
Citi Bike is the nation's largest bike share program, with 10,000 bikes and 600 stations across Manhattan, Brooklyn, Queens, and Jersey City. This dataset includes Citi Bike trips since Citi Bike launched in September 2013 and is updated daily. The data has been processed by Citi Bike to remove trips that are taken by staff to service and inspect the system, as well as any trips below 60 seconds in length, which are considered false starts.

This public dataset is hosted in Google BigQuery and is included in BigQuery's 1TB/mo of free tier processing. This means that each user receives 1TB of free BigQuery processing every month, which can be used to run queries on this public dataset. Watch this short video to learn how to get started quickly using BigQuery to access public datasets. What is BigQuery.



Working in the BQ Sandbox

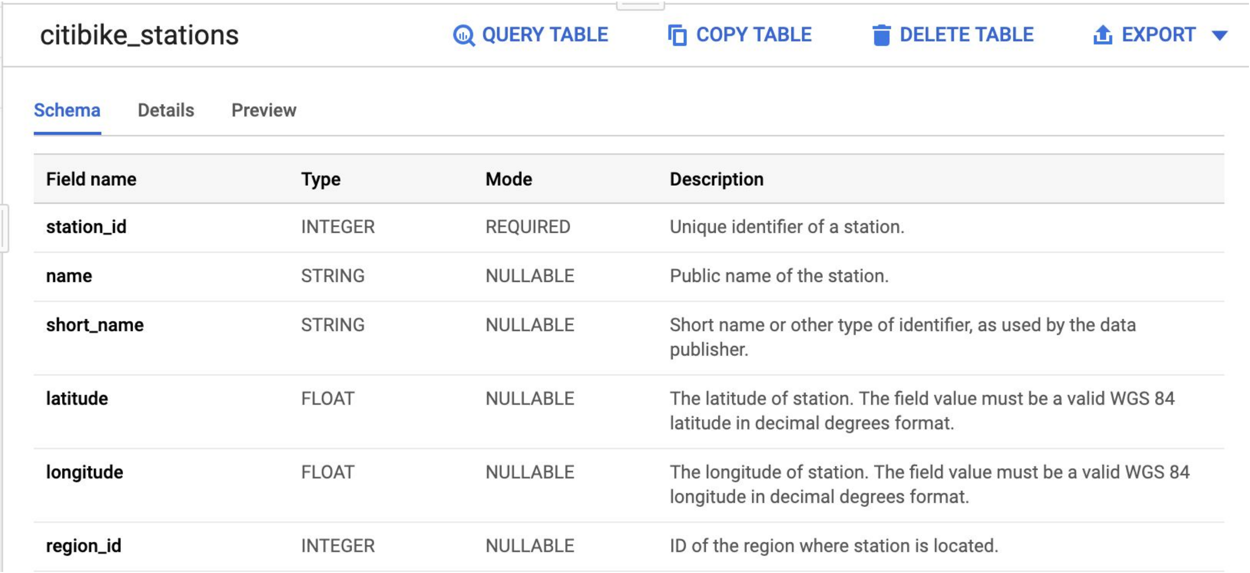
10 GB of active storage and 1 TB of processed query data per month in the sandbox for free! You can also upgrade to a free tier or a paid tier.



SANDBOX Set up billing to upgrade to the full BigQuery experience. Learn more

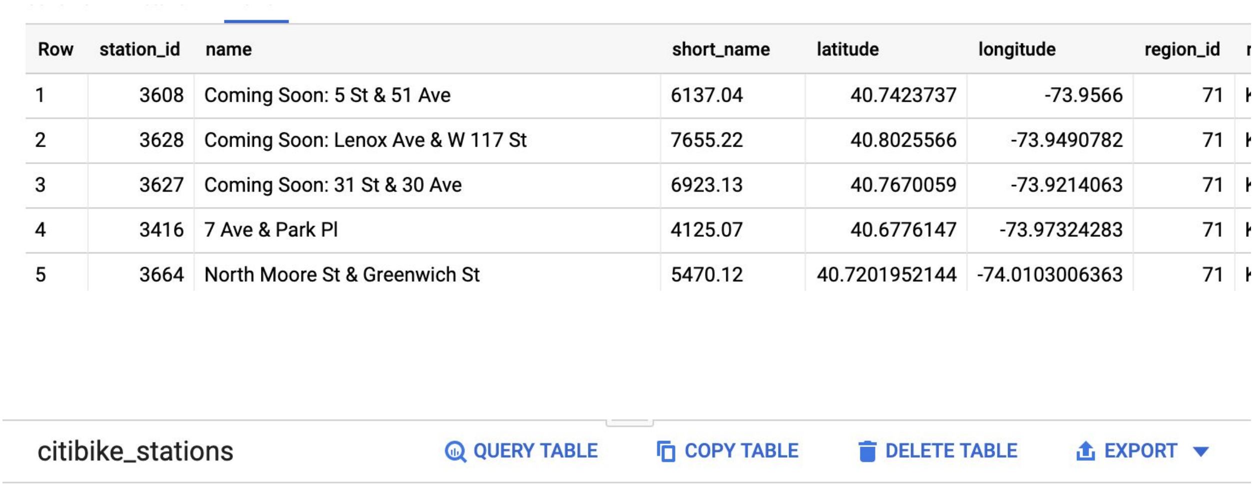
The Data

Our Schema



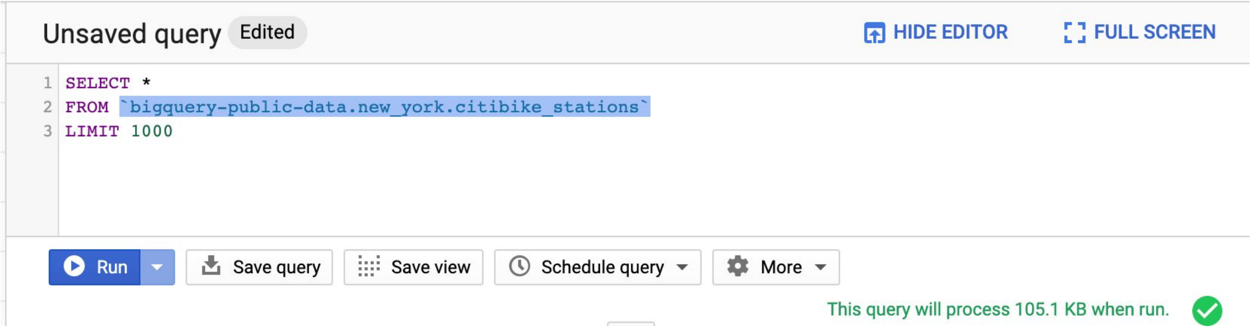
Field name	Type	Mode	Description
station_id	INTEGER	REQUIRED	Unique identifier of a station.
name	STRING	NULLABLE	Public name of the station.
short_name	STRING	NULLABLE	Short name or other type of identifier, as used by the data publisher.
latitude	FLOAT	NULLABLE	The latitude of station. The field value must be a valid WGS 84 latitude in decimal degrees format.
longitude	FLOAT	NULLABLE	The longitude of station. The field value must be a valid WGS 84 longitude in decimal degrees format.
region_id	INTEGER	NULLABLE	ID of the region where station is located.

A Preview



Row	station_id	name	short_name	latitude	longitude	region_id
1	3608	Coming Soon: 5 St & 51 Ave	6137.04	40.7423737	-73.9566	71
2	3628	Coming Soon: Lenox Ave & W 117 St	7655.22	40.8025566	-73.9490782	71
3	3627	Coming Soon: 31 St & 30 Ave	6923.13	40.7670059	-73.9214063	71
4	3416	7 Ave & Park Pl	4125.07	40.6776147	-73.97324283	71
5	3664	North Moore St & Greenwich St	5470.12	40.7201952144	-74.0103006363	71

Query Set Up



```
SELECT *
FROM `bigquery-public-data.new_york.citibike_stations`
LIMIT 1000
```

This query will process 105.1 KB when run.

- SELECT : What columns do you want to include in your output table?
- FROM : Where are we getting the data from?
- LIMIT : # of rows to include, we don’t really need this
- * : All columns
- COUNT() : Count operator

Let’s Query!

How many trips have been taken?

```
SELECT COUNT(1) AS total_rides
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
```

How many trips have been taken?

```
SELECT COUNT(1) AS total_rides
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
```

58,937,715 trips — that’s a lot of trips!!

How many bikes have been used?

```
SELECT COUNT(DISTINCT bikeid) as number_of_bikes
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
```

How many trips are people taking by different generations?

```
SELECT
  CASE WHEN birth_year >= 1997 THEN 'Gen Z'
       WHEN birth_year >= 1980 AND birth_year < 1997 THEN 'Millennial'
       WHEN birth_year >= 1965 AND birth_year < 1980 THEN 'Gen X'
       WHEN birth_year >= 1944 AND birth_year < 1965 THEN 'Baby Boomer'
       WHEN birth_year IS NULL then 'Unknown'
       ELSE 'Other'
       END as generation,
  COUNT(1) as trips
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
GROUP BY generation
ORDER BY generation
```

Have male or female riders taken more trips? *

```
SELECT
  COUNT(CASE WHEN gender = 'female' THEN 1 END) AS rides_from_female_riders,
  _____________________________________________ AS ________________
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
```

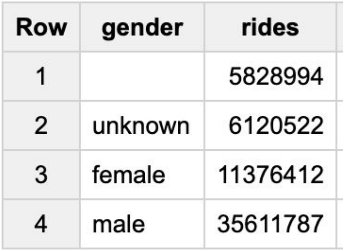
Have male or female riders taken more trips? *

```
SELECT
  COUNT(CASE WHEN gender = 'female' THEN 1 END) AS rides_from_female_riders,
  COUNT(CASE WHEN gender = 'male' THEN 1 END) AS rides_from_male_riders
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
```

Another way of answering

```
SELECT gender, COUNT(1) as rides
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
GROUP BY gender
ORDER BY rides
```

What’s with the blank field?



Row	gender	rides
1		5828994
2	unknown	6120522
3	female	11376412
4	male	35611787

Let’s get rid of it

```
SELECT gender, COUNT(1) as rides
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
GROUP BY gender
HAVING gender <> ''
ORDER BY rides
```

Logical operators

<	less than	>	greater than
<=	less than or equal to	>=	greater than or equal to
LIKE		NOT LIKE	
=	equals	!=, <>	
IN ()		NOT IN ()	
IS NULL		IS NOT NULL	
IS TRUE		IS FALSE	
AND		OR	
%	wildcard		

So was our original answer wrong?

Let’s look at the weird fields…

```
SELECT *
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE gender = ''
```

So was our original answer wrong?

Let’s look at the weird fields…

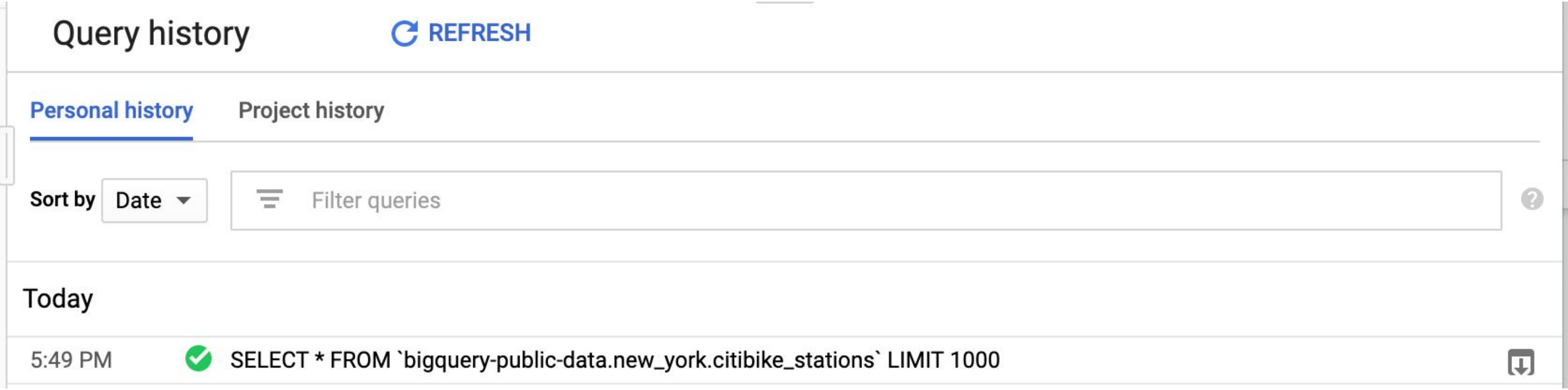
```
SELECT *
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE gender = ''
```

So it turns out we should fix this!

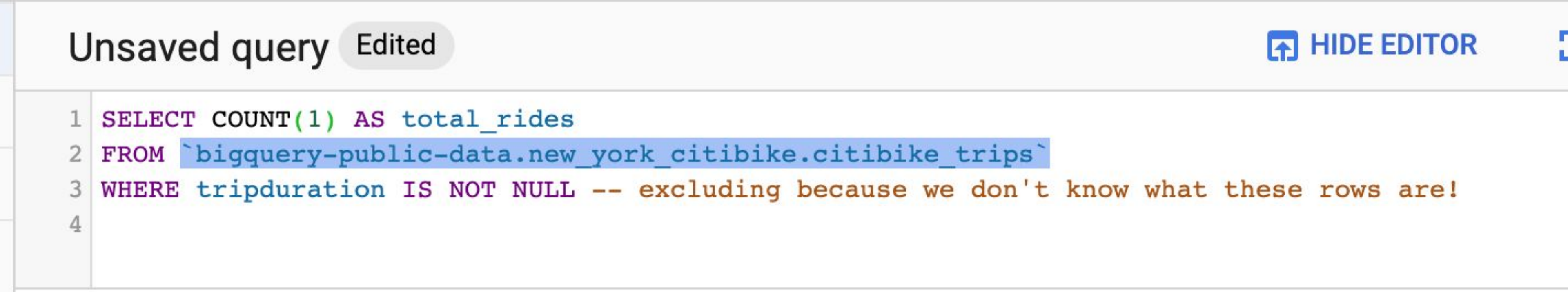
```
SELECT COUNT(1) AS total_rides
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE gender = ''
```

Ans: 53108721 — still a ton of rides!!

Commenting & query history



5:49 PM ✓ SELECT * FROM `bigquery-public-data.new_york.citibike_stations` LIMIT 1000



```sql
SELECT COUNT(1) AS total_rides
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE tripduration IS NOT NULL -- excluding because we don't know what these rows are!
```

Do Gen-Zers ride bikes?

```
SELECT COUNT(1)
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE birth_year >= 1997
```

Do Gen-Zers ride bikes?

Has this pattern changed over time?

```
SELECT EXTRACT(year from starttime) AS year, COUNT(1) AS trips
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE birth_year >= 1997
GROUP BY 1
ORDER BY 2
```

* This data only goes through May 31, 2018 for some reason so that's why 2018 is low

You can look at lots of timestamp functions here
https://cloud.google.com/bigquery/docs/reference/standard-sql/timestamp_functions

Quick exercise

How many trips were taken by people the same age as you, in each year?

Exploring more with stations

```
SELECT EXTRACT(year from starttime) as year,
       start_station_name,
       COUNT(1) as trips
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE start_station_name IN
  ('Washington Pl & Broadway','University Pl & E 8 St','LaGuardia Pl & W 3
St')
GROUP BY 1,2
ORDER BY 1,2
```

Exploring more with stations

EXERCISE: Look at the trips from stations on your street! e.g.:

```
WHERE start_station_name LIKE '%Washington Ave%'
```

We can also export this data to Google Sheets :)

Adding in Other Tables

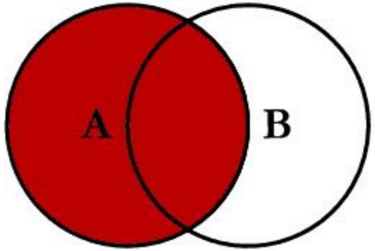
Let’s look at citibike_stattions

- In citibike_trips, there is a start_station_id and a stop_station_id
- These correspond to station_id in citibike_stations
- This makes station_id a FOREIGN KEY joining the two tables
- Technically our trips table should have a PRIMARY KEY uniquely identifying each row, but there isn’t one :(

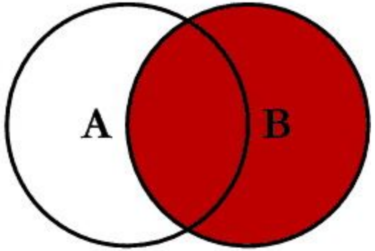
How many of the stations from the ones that were used on Citibike trips have a key dispenser?

```
SELECT stations.eightd_has_key_dispenser,
       COUNT(distinct trips.start_station_id) as stations
FROM `bigquery-public-data.new_york.citibike_trips` as trips
LEFT JOIN `bigquery-public-data.new_york.citibike_stations` as stations
  ON trips.start_station_id = stations.station_id
GROUP BY 1
```

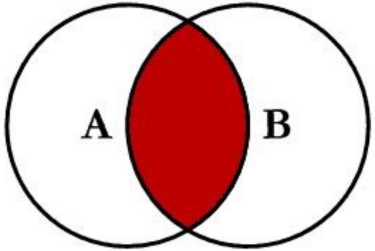
SQL JOINS



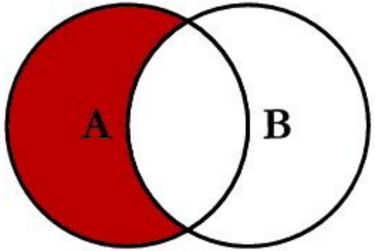
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



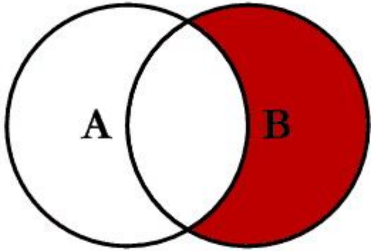
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



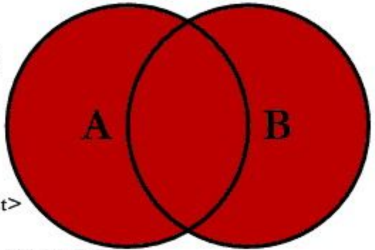
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



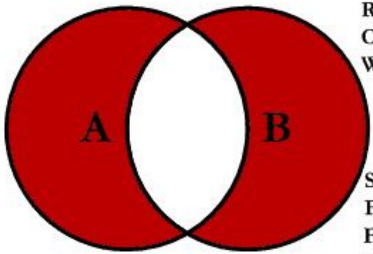
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



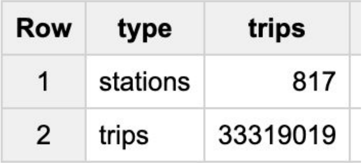
```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

You can also UNION but you probably won’t need to

```
SELECT 'trips' AS type, COUNT(1) AS trips
FROM `bigquery-public-data.new_york.citibike_trips`
UNION ALL
(SELECT 'stations' AS type, COUNT(1) AS stations
FROM `bigquery-public-data.new_york.citibike_stations`)
```



Row	type	trips
1	stations	817
2	trips	33319019

Exercises

1. On average, do Subscribers take longer trips or Customers? By how much?
2. What hours of the day are most popular for starting bike trips? What about stopping?
3. What month is the least popular for taking a bike trip?
4. What are the longest bike times? Where did people travel to and from in those long trips?
5. Explore your own questions! Shout out a complex one to explore together!

More advanced stuff (not for slides)

Which stations are in the top 10 ranked most popular start stations?

```
with starts as (
  SELECT start_station_id, start_station_name, COUNT(1) as start_trips
  FROM `bigquery-public-data.new_york_citibike.citibike_trips`
  WHERE tripduration IS NOT NULL
  GROUP BY 1,2
), ranked_starts as (
  SELECT *, ROW_NUMBER() OVER(ORDER BY start_trips DESC) AS start_station_rank
  FROM starts
)

select *
from ranked_starts
where start_station_rank <= 10
order by start_station_rank
```

What about the ranking for millennials vs. Gen Z vs. Gen X?

(there are many ways to do this but this is showing you certain methods)

```
with starts as (
```

Let’s add another struct for stops

```
WITH starts AS (
  SELECT start_station_id, start_station_name,
  CASE WHEN birth_year >= 1997 THEN 'Gen Z'
       WHEN birth_year >= 1980 AND birth_year < 1997 THEN 'Millennial'
       WHEN birth_year >= 1965 AND birth_year < 1980 THEN 'Gen X'
    END AS generation,
  COUNT(1) as start_trips
  FROM `bigquery-public-data.new_york_citibike.citibike_trips`
  WHERE tripduration IS NOT NULL
        AND birth_year >= 1965
  GROUP BY 1,2,3
), ranked_starts as (
  SELECT *, ROW_NUMBER() OVER(PARTITION BY generation ORDER BY start_trips DESC) AS start_station_rank
  FROM starts
), top10_starts AS (
  SELECT *
  FROM ranked_starts
  WHERE start_station_rank <= 10
  ORDER BY generation, start_station_rank
), structured_starts AS (
  SELECT start_station_rank, array_agg(struct(generation, start_station_id, start_station_name, start_trips) ORDER BY generation) as
starts
  FROM top10_starts
  GROUP BY 1
  ORDER BY start_station_rank
), ends as (
```

What we didn’t cover

- Uploading your own data to BigQuery
- The other public datasets! Explore & combine
    - E.g. When are Citibikes faster than taxis?
- DMLs - Inserting, updating, deleting tables
- Partitioning tables — a powerful storage solution in BigQuery
- Other SQL tutorials