Partitioned Tables in Google BigQuery

## Overview

[BigQuery](http://bigquery.cloud.google.com/) is Google's fully managed, NoOps, low cost analytics database. With BigQuery you can query terabytes and terabytes of data without having any infrastructure to manage or needing a database administrator. BigQuery uses SQL and can take advantage of the pay-as-you-go model. BigQuery allows you to focus on analyzing data to find meaningful insights.

The dataset you'll use is an [ecommerce dataset](https://www.en.advertisercommunity.com/t5/Articles/Introducing-the-Google-Analytics-Sample-Dataset-for-BigQuery/ba-p/1676331) that has millions of Google Analytics records for the [Google Merchandise Store](https://shop.googlemerchandisestore.com/) loaded into BigQuery. You have a copy of that dataset for this lab and will explore the available fields and row for insights.

In this lab you will query partitioned datasets and create your own dataset partitions to improve query performance and reduce cost.

### **Open BigQuery Console**

1. In the Google Cloud Console, select **Navigation menu** > **BigQuery**.

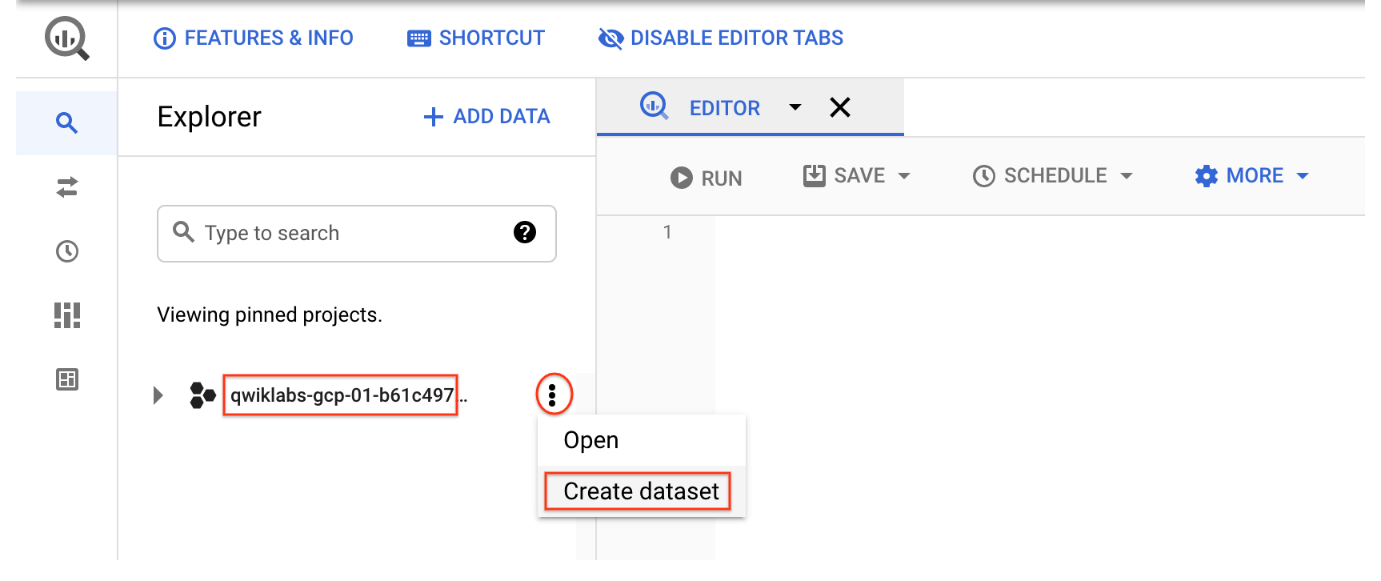
The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and lists UI updates.

1. Click **Done**.

## Create a new dataset

First, you will create a dataset to store your tables.

Create a new dataset within your project by clicking on the **View actions** icon next to your project ID in the Explorer section, and then selecting **CREATE DATASET**.



Set the Dataset ID to ecommerce. Leave the other options at their default values (Data Location, Default table Expiration). Click **CREATE DATASET**.

## Creating tables with date partitions

A partitioned table is a table that is divided into segments, called partitions, that make it easier to manage and query your data. By dividing a large table into smaller partitions, you can improve query performance, and control costs by reducing the number of bytes read by a query.

Now you will create a new table and bind a date or timestamp column as a partition. Before we do that, let's explore the data in the non-partitioned table first.

### **Query webpage analytics for a sample of visitors in 2017**

In the **Query Editor**, add the below query. Before running, note the total amount of data it will process as indicated next to the query validator icon: "This query will process 1.74 GB when run".

#standardSQL

SELECT DISTINCT

fullVisitorId,

date,

city,

pageTitle

FROM `data-to-insights.ecommerce.all\_sessions\_raw`

WHERE date = '20170708'

LIMIT 5

Copied!

content\_copy

Click **Run**.

The query returns 5 results.

### **Query webpage analytics for a sample of visitors in 2018**

Let's modify the query to look at visitors for 2018 now.

In the **Query Editor**, add the below query:

#standardSQL

SELECT DISTINCT

fullVisitorId,

date,

city,

pageTitle

FROM `data-to-insights.ecommerce.all\_sessions\_raw`

WHERE date = '20180708'

LIMIT 5

Copied!

content\_copy

The **Query results** will tell you how much data this query will process.

Click **RUN**.

Notice that the query still processes 1.74 GB even though it returns 0 results. Why? The query engine needs to scan all records in the dataset to see if they satisfy the date matching condition in the WHERE clause. It must look at each record to compare the date against the condition of ‘20180708'.

Additionally, the LIMIT 5 does not reduce the total amount of data processed, which is a common misconception.

Why did the previous query return 0 records but still scan through 1.74GB of data?



The query engine has the metadata for each partition stored but still needs to scan all records even if the table is partitioned.



The query was written incorrectly



Before the query runs, the query engine does not know whether 2018 data exists to satisfy the WHERE clause condition and it needs to scan through all records in a non-partitioned table.

Submit

#### Common use-cases for date-partitioned tables

Scanning through the entire dataset everytime to compare rows against a WHERE condition is wasteful. This is especially true if you only really care about records for a specific period of time like:

* All transactions for the last year
* All visitor interactions within the last 7 days
* All products sold in the last month

Instead of scanning the entire dataset and filtering on a date field like we did in the earlier queries, we will now setup a date-partitioned table. This will allow us to completely ignore scanning records in certain partitions if they are irrelevant to our query.

#### Create a new partitioned table based on date

Click **COMPOSE NEW QUERY** and add the below query, then **RUN**:

#standardSQL

CREATE OR REPLACE TABLE ecommerce.partition\_by\_day

PARTITION BY date\_formatted

OPTIONS(

description="a table partitioned by date"

) AS

SELECT DISTINCT

PARSE\_DATE("%Y%m%d", date) AS date\_formatted,

fullvisitorId

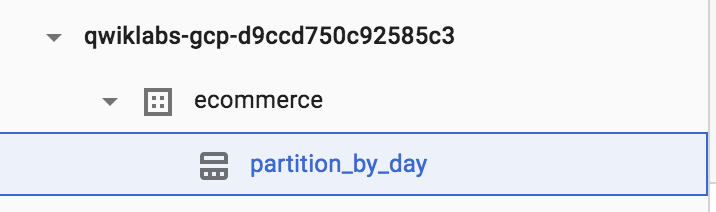
FROM `data-to-insights.ecommerce.all\_sessions\_raw`

Copied!

content\_copy

In this query, note the new option - PARTITION BY a field. The two options available to partition are DATE and TIMESTAMP. The PARSE\_DATE function is used on the date field (stored as a string) to get it into the proper DATE type for partitioning.

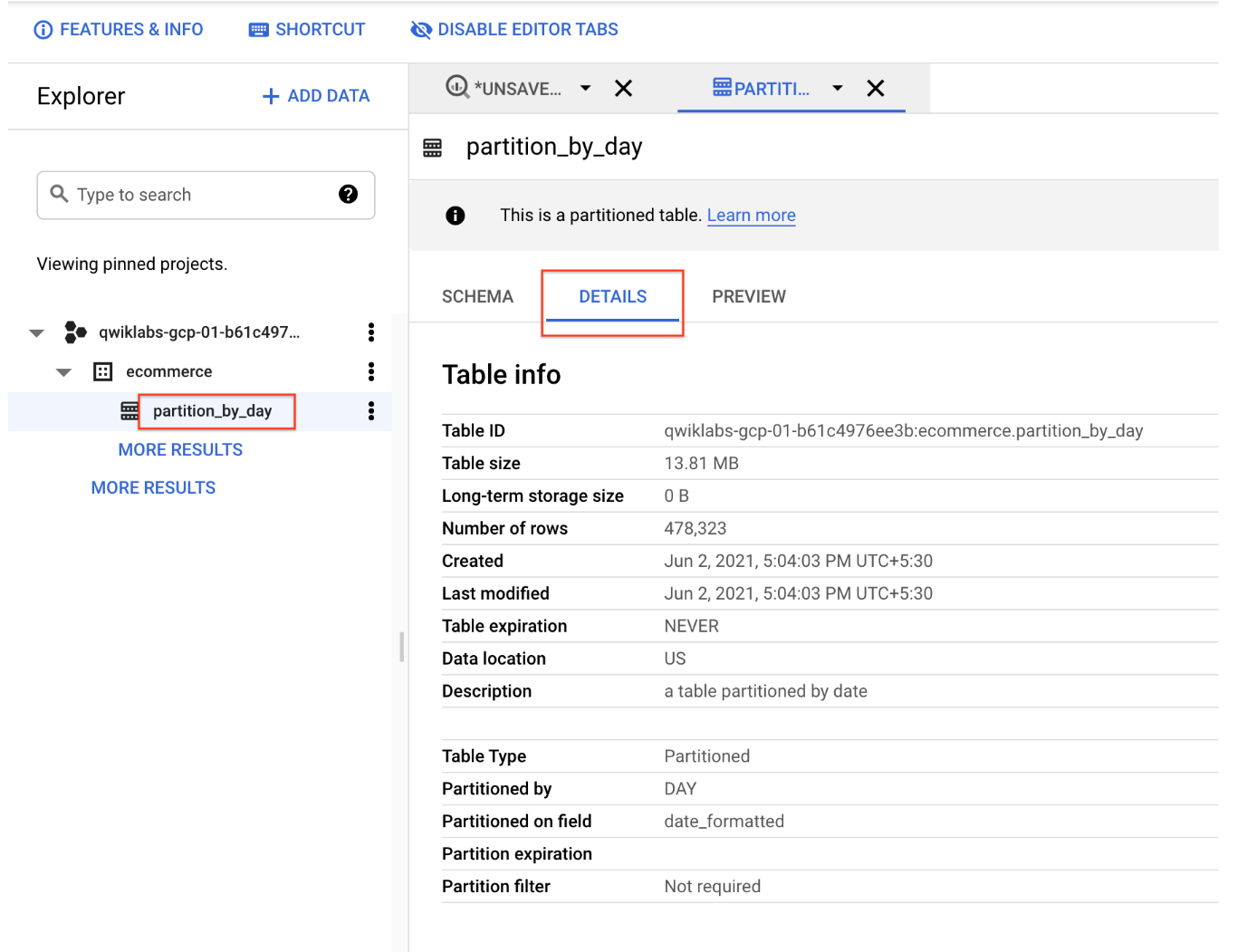
Click on the **ecommerce** dataset, then select the new **partiton\_by\_day** table:



Click on the **Details** tab.

Confirm that you see:

* Partitioned by: Day
* Partitioning on: date\_formatted



Note: Partitions within partitioned tables on your Qwiklabs account will auto-expire after 60 days from the value in your date column. Your personal GCP account with billing-enabled will let you have partitioned tables that don't expire. For the purposes of this lab, the remaining queries will be ran against partitioned tables that have already been created.

## View data processed with a partitioned table

Run the below query, and note the total bytes to be processed:

#standardSQL

SELECT \*

FROM `data-to-insights.ecommerce.partition\_by\_day`

WHERE date\_formatted = '2016-08-01'

Copied!

content\_copy

This time ~25 KB or 0.025MB is processed, which is a fraction of what you queried.

Now run the below query, and note the total bytes to be processed:

#standardSQL

SELECT \*

FROM `data-to-insights.ecommerce.partition\_by\_day`

WHERE date\_formatted = '2018-07-08'

Copied!

content\_copy

You should see This query will process 0 B when run.

Why is there 0 bytes processed?

Why is there 0 bytes processed?



The query engine knows which partitions already exist and knows that no partition exists for 2018-07-08 (the ecommerce dataset ranges from 2016-08-01 to 2017-08-01).



The query engine has cached the results from a query we ran earlier and will return the same 10 records



The query engine processes many fewer rows of data when you use partitions and caches each row for all users so 0 bytes are processed

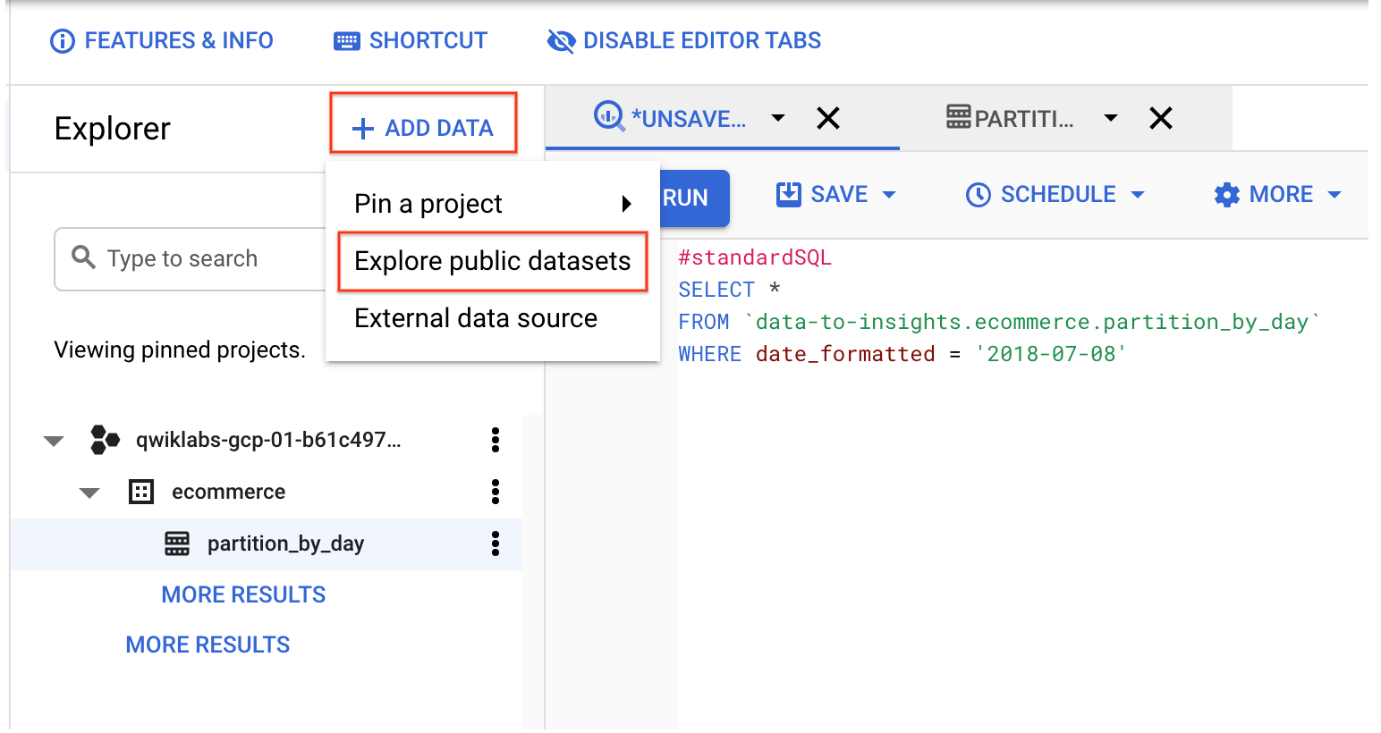
Submit

## Creating an auto-expiring partitioned table

Auto-expiring partitioned tables are used to comply with data privacy statutes, and can be used to avoid unnecessary storage (which you'll be charged for in a production environment). If you want to create a rolling window of data, add an expiration date so the partition disappears after you're finished using it.

### **Explore the available NOAA weather data tables**

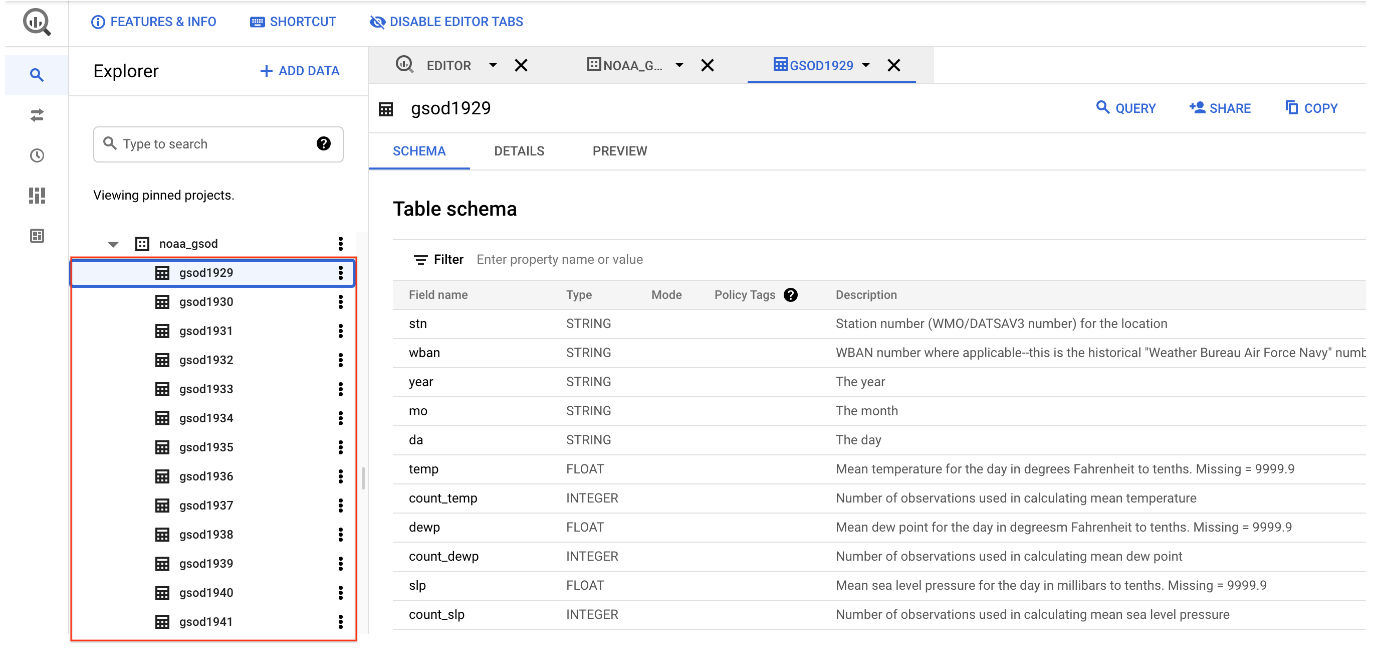
In the left panel, click on **+ ADD DATA** and select **Explore public datasets**.



Search for **GSOD NOAA**, and then select the dataset.

Click on **View dataset**.

Scroll through the tables in the **noaa\_gsod** dataset (which are manually sharded and not partitioned).



Next, **copy and paste** this below query to **Query editor**:

#standardSQL

SELECT

DATE(CAST(year AS INT64), CAST(mo AS INT64), CAST(da AS INT64)) AS date,

(SELECT ANY\_VALUE(name) FROM `bigquery-public-data.noaa\_gsod.stations` AS stations

WHERE stations.usaf = stn) AS station\_name, -- Stations may have multiple names

prcp

FROM `bigquery-public-data.noaa\_gsod.gsod\*` AS weather

WHERE prcp < 99.9 -- Filter unknown values

AND prcp > 0 -- Filter stations/days with no precipitation

AND \_TABLE\_SUFFIX >= '2021'

ORDER BY date DESC -- Where has it rained/snowed recently

LIMIT 10

Copied!

content\_copy

Note that the table wildcard \* used in the FROM clause to limit the amount of tables referred to in the TABLE\_SUFFIX filter.

Note that although a LIMIT 10 was added, this still does not reduce the total amount of data scanned (about 141.6 MB) since there are no partitions yet.

Click **Run**.

Confirm the date is properly formatted and the precipitation field is showing non-zero values.

## Your turn: Create a Partitioned Table

Modify the previous query to create a table with the below specifications:

* Table name: ecommerce.days\_with\_rain
* Use the date field as your PARTITION BY
* For OPTIONS, specify partition\_expiration\_days = 60
* Add the table description = "weather stations with precipitation, partitioned by day"

Your query should look like this:

#standardSQL

CREATE OR REPLACE TABLE ecommerce.days\_with\_rain

PARTITION BY date

OPTIONS (

partition\_expiration\_days=60,

description="weather stations with precipitation, partitioned by day"

) AS

SELECT

DATE(CAST(year AS INT64), CAST(mo AS INT64), CAST(da AS INT64)) AS date,

(SELECT ANY\_VALUE(name) FROM `bigquery-public-data.noaa\_gsod.stations` AS stations

WHERE stations.usaf = stn) AS station\_name, -- Stations may have multiple names

prcp

FROM `bigquery-public-data.noaa\_gsod.gsod\*` AS weather

WHERE prcp < 99.9 -- Filter unknown values

AND prcp > 0 -- Filter

AND \_TABLE\_SUFFIX >= '2021'

Copied!

content\_copy

#### Confirm data partition expiration is working

To confirm you are only storing data from 60 days in the past up until today, run the DATE\_DIFF query to get the age of your partitions, which are set to expire after 60 days.

Below is a query which tracks the average rainfall for the NOAA weather station in [Wakayama, Japan](https://en.wikipedia.org/wiki/Wakayama,_Wakayama#Climate) which has significant precipitation.

Add this query and run it:

#standardSQL

# avg monthly precipitation

SELECT

AVG(prcp) AS average,

station\_name,

date,

CURRENT\_DATE() AS today,

DATE\_DIFF(CURRENT\_DATE(), date, DAY) AS partition\_age,

EXTRACT(MONTH FROM date) AS month

FROM ecommerce.days\_with\_rain

WHERE station\_name = 'WAKAYAMA' #Japan

GROUP BY station\_name, date, today, month, partition\_age

ORDER BY date DESC; # most recent days first

Copied!

content\_copy

## Confirm the oldest partition\_age is at or below 60 days

Update the ORDER BY clause to show the oldest partitions first. The date you see there Add this query and run it:

#standardSQL

# avg monthly precipitation

SELECT

AVG(prcp) AS average,

station\_name,

date,

CURRENT\_DATE() AS today,

DATE\_DIFF(CURRENT\_DATE(), date, DAY) AS partition\_age,

EXTRACT(MONTH FROM date) AS month

FROM ecommerce.days\_with\_rain

WHERE station\_name = 'WAKAYAMA' #Japan

GROUP BY station\_name, date, today, month, partition\_age

ORDER BY partition\_age DESC

Copied!

content\_copy

Note: Your results will vary if you re-run the query in the future, as the weather data, and your partitions, are continuously updated.

## Congratulations!

You've successfully created and queried partitioned tables in BigQuery.

## End your lab