
Creating Date-Partitioned Tables in BigQuery

Overview

This project explores **BigQuery**, Google's fully managed, serverless data warehouse, focusing on **date-partitioned tables** to optimize query performance and reduce costs. The lab uses an **ecommerce dataset** from Google Analytics to demonstrate how partitioning improves efficiency when querying large datasets.

Key Objectives

- Query partitioned tables.
- Create custom partitioned tables.
- Implement **auto-expiring partitions** for cost and compliance management.

Setup & Environment

Tools & Services Used

- **BigQuery** (Google Cloud's analytics database)
- **Public Datasets** (NOAA weather data, Google Analytics ecommerce data)
- **SQL** (Standard SQL dialect)

Dataset Sources

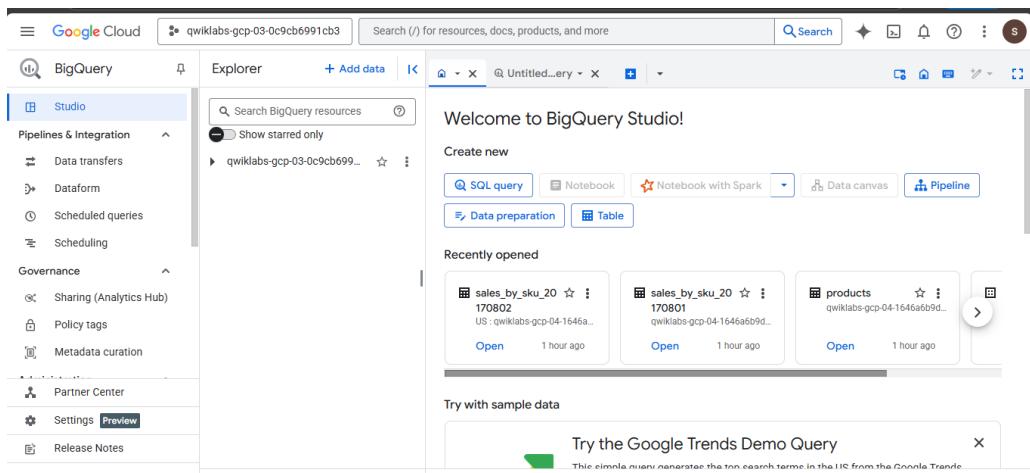
1. **Google Analytics Ecommerce Dataset** (data-to-insights.ecommerce.all_sessions_raw)
2. **NOAA GSOD Weather Dataset** (bigquery-public-data.noaa_gsod)

Open the 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 the release notes.

2. Click **Done**.



Task 1. Create a new dataset

1. First, you will create a dataset to store your tables.
2. In the **Explorer pane**, near your project id, click on **View actions** then click **Create dataset**.
3. Set **Dataset ID** to **ecommerce**.
Leave the other options at their default values (Data Location, Default table Expiration).
4. Click **Create dataset**.

Create dataset

Project ID *
qwiklabs-gcp-03-0c9cb6991cb3 [Change](#)

Dataset ID * Letters, numbers, and underscores allowed

Location type [?](#)

Region
Specify a region to colocate your datasets with other Google Cloud services.

Multi-region
Allow BigQuery to select a region within a group to achieve higher quota limits.

Some locations have been restricted due to a policy set by your organization. [Learn more about restricting locations.](#)

Multi-region *

External Dataset
The selected region supports the following external dataset types: Cloud Spanner

Link to an external dataset [?](#)

[Create dataset](#) Cancel

Welcome to BigQuery Studio!

Create new

SQL query Notebook Notebook with Spark Data canvas Pipeline

Data preparation Table

Recently opened

- ecommerce US : qwiklabs-gcp-03-0c9cb... Open Just now
- sales_by_sku_20 170802 US : qwiklabs-gcp-04-1646a... Open 1 hour ago
- sales_by_sku_20 170801 qwiklabs-gcp-04-1646a6b9d... Open 1 hour ago

Try with sample data

Try the Google Trends Demo Query

"commerce" created. Go to dataset X

generates the top search terms in the US from the Google Trends

Refresh ^

Task 2. Create 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 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 web page analytics for a sample of visitors in 2017

```
#standardSQL SELECT DISTINCT fullVisitorId, date, city, pageTitle FROM data-to-insights.ecommerce.all_sessions_raw WHERE date = '20170708' LIMIT 5
```

The screenshot shows the BigQuery Query Editor interface. At the top, there's a toolbar with icons for file operations like Open, Save, Download, and Share. Below the toolbar, the title bar says "Untitled query". The main area contains the following SQL code:

```

1 #standardSQL
2 SELECT DISTINCT
3   fullVisitorId,
4   date,
5   city,
6   pageTitle
7 FROM `data-to-insights.ecommerce.all_sessions_raw`
8 WHERE date = '20170708'
9 LIMIT 5

```

A green checkmark icon indicates "Query completed". Below the code, the "Query results" section displays the following data:

Row	fullVisitorId	date	city	pageTitle
1	3909426869160428588	20170708	San Jose	Nest-USA
2	595645219980861214	20170708	San Francisco	Nest-USA
3	1615330153985120765	20170708	not available in demo dataset	Nest-USA
4	5571504570820200013	20170708	not available in demo dataset	Nest-USA

At the bottom of the results table, it says "Results per page: 50 1 – 5 of 5".

2. Click Run.

Query web page analytics for a sample of visitors in 2018

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

1. Click **+ SQL query** to clear the **Query Editor**, then add this new query. Note the WHERE date parameter is changed to 20180708:

```
#standardSQL SELECT DISTINCT fullVisitorId, date, city, pageTitle FROM data-to-insights.ecommerce.all_sessions_raw WHERE date = '20180708' LIMIT 5
```

```

1 #standardSQL
2 SELECT DISTINCT
3   fullVisitorId,
4   date,
5   city,
6   pageTitle
7 FROM `data-to-insights.ecommerce.all_sessions_raw`
8 WHERE date = '20180708'
9 LIMIT 5

```

Query completed

Query results

Job information **Results** Chart JSON Execution details Execution graph

There is no data to display.

Results per page: 50 ▾ 1 – 0 of 0 |< < > >|

Click **Run**

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, Now set up a date-partitioned table. This allows you to completely ignore scanning records in certain partitions if they are irrelevant to our query.

Create a new partitioned table based on date

- 1. Click + SQL query , add the below query, then click 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
```

The screenshot shows the BigQuery web interface. At the top, there are four tabs labeled 'Untitled...ery' with the last one being active. Below the tabs is a toolbar with 'Run', 'Save', 'Download', and 'Share' buttons. The main area contains a code editor with the following SQL query:

```
1 #standardSQL
2 CREATE OR REPLACE TABLE ecommerce.partition_by_day
3 PARTITION BY date_formatted
4 OPTIONS(
5   description="a table partitioned by date"
6 ) AS
7
8 SELECT DISTINCT
9   PARSE_DATE("%Y%m%d", date) AS date_formatted,
10  fullvisitorId
11 FROM `data-to-insights.ecommerce.all sessions raw`
```

A green checkmark indicates 'Query completed'. Below the code editor is a 'Query results' section with tabs for 'Job information', 'Results' (which is selected), 'Execution details', and 'Execution graph'. A message states: 'This statement created a new table named partition_by_day.' with a 'Go to table' button. At the bottom left is a 'Job history' link, and at the bottom right is a 'Refresh' button.

2. Click on the **ecommerce** dataset, then select the new **partition_by_day** table:

The screenshot shows the BigQuery Studio interface. On the left is a sidebar with 'BigQuery Studio' and sections for 'Lines & Integration', 'Sharing', and 'Metadata curation'. The main area shows the 'Explorer' tab with a tree view of datasets and tables. Under the 'ecommerce' dataset, the 'partition_by_day' table is selected. The right panel shows the table's schema with two fields: 'date_formatted' (DATE type) and 'fullvisitorId' (STRING type). There are tabs for 'Schema', 'Details', 'Preview', 'Table Explorer', 'Preview' (which is selected), 'Insights', 'Lineage', and 'Data Pri'.

3. Click on the **Details** tab.

Confirm that you see:

- Partitioned by: Day
- Partitioning on: date_formatted

The screenshot shows the BigQuery UI with the 'partition_by_day' table selected. The left sidebar shows a tree view of datasets and tables. The main area displays the table's details, including its partitioning information (Partitioned, Partitioned by DAY, Partitioned on field date_formatted), storage info (478,323 rows, 0 partitions), and a preview section.

Task 3. Review results from queries on a partitioned table

1. 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'
```

The screenshot shows the execution of the provided SQL query. The query completed successfully, returning 888 rows. The results are displayed in a table format with columns 'Row', 'date_formatted', and 'fullvisitorid'. The total bytes processed for this query were 1.00 GB.

Row	date_formatted	fullvisitorid
1	2016-08-01	8346614539128137085
2	2016-08-01	1856237131266550302
3	2016-08-01	8422029627538180622
4	2016-08-01	7298538238612725446
5	2016-08-01	8271170844108113200
6	2016-08-01	6030957980134486247
7	2016-08-01	7604200805116024111

2. 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'
```

The screenshot shows a BigQuery query editor interface. At the top, there are tabs for 'Query' and 'Partitioned by day'. Below the tabs, the title bar says 'Untitled query'. There are buttons for 'Run', 'Save', 'Download', and 'Share'. The main area contains the following SQL code:

```
1 #standardSQL  
2 SELECT *  
3 FROM `data-to-insights.ecommerce.partition_by_day`  
4 WHERE date_formatted = '2018-07-08'
```

Below the code, a message says 'Query completed'. Underneath that, a section titled 'Query results' shows a message: 'There is no data to display.'

Task 4. Create 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

1. In the left menu, in Explorer, click on **+ Add data** and select **Public datasets**.
2. Search for **GSOD NOAA** then select the dataset.
3. Click on **View Dataset**.
4. **Scroll through** the tables in the **noaa_gsod** dataset (which are manually sharded and not partitioned):

The screenshot shows the Marketplace search results for "GSOD NOAA". The search bar at the top contains the query. Below it, the results are displayed under the heading "Marketplace > GSOD NOAA > Data". A filter bar on the left allows filtering by Category (Science & research, Sustainability, Financial services), Type (Data), and Price (Free). The result section shows one item: "GSOD NOAA" from NOAA, which is described as a public dataset created by the National Oceanic and Atmospheric Administration from the USAF Climatology Center, covering GSOD data between 1929 and present from over 9000 stations. The item has a small NOAA logo icon.

The screenshot shows the BigQuery Explorer interface. On the left is the sidebar with various datasets listed. In the center, the details for the "noaa_gsod" dataset are shown. The "noaa_gsod" dataset is selected in the main pane. The "Details" tab is active, displaying information such as Dataset ID (bigquery-public-data.noaa_gsod), Created (Mar 14, 2016, 8:24:51 AM UTC+5:30), Default table expiration (Never), Last modified (Sep 20, 2022, 1:27:49 PM UTC+5:30), Data location (US), and Description (Overview: This public dataset was created by the National Oceanic and Atmospheric Administration (NOAA) and includes global data obtained from the USAF Climatology Center. This dataset covers GSOD data between 1929 and present, collected from over 9000 stations.). There are tabs for "Insights" and "Preview" as well.

Your goal is to create a table that:

- Queries on weather data from 2018 onward
- Filters to only include days that have had some precipitation (rain, snow, etc.)
- Only stores each partition of data for 90 days from that partition's date (rolling window)

1. First, **copy and paste** this below query:

```
#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 >= '2018' ORDER BY date DESC -- Where has it rained/snowed recently LIMIT 10

```

1 #standardSQL
2 SELECT
3   DATE(CAST(year AS INT64)), CAST(mo AS INT64), CAST(da AS INT64)) AS date,
4   (SELECT ANY_VALUE(name) FROM `bigquery-public-data.noaa_gsod.stations` AS stations
5    WHERE stations.usaf = strn) AS station_name, -- Stations may have multiple names
6   prcp
7   FROM `bigquery-public-data.noaa_gsod.gsod*` AS weather
8   WHERE prcp < 99.9 -- Filter unknown values
9   AND prcp > 0 -- Filter stations/days with no precipitation
10  AND _TABLE_SUFFIX >= '2018'
11  ORDER BY date DESC -- Where has it rained/snowed recently

```

Query completed

Row	date	station_name	prcp
1	2025-06-16	BLUE RIDGE	0.15
2	2025-06-16	CHARLOTTESVILLE ALB	0.13
3	2025-06-16	CRAVEN COUNTY REG AIRPORT	0.01
4	2025-06-16	CHIANGMAI INTERNATIONAL AIRPORT	0.01

Results per page: 50 ▾ 1 – 10 of 10 |< < > >|

2. Click **Run**.

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

Task 5. 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 >= '2018'
```

The screenshot shows the BigQuery web interface. In the top navigation bar, there are tabs for 'noaa_gsod' and '*Untitled...ery'. Below the tabs, the title 'Untitled query' is displayed along with a 'Run' button, a 'Save' dropdown, a 'Download' button, a 'Share' button, a 'Schedule' button, an 'Open in' dropdown, and a three-dot menu. The main area contains the SQL code from the previous block. A green checkmark icon and the text 'Query completed' are visible at the bottom of the code area. Below this, a section titled 'Query results' is shown, with tabs for 'Job information', 'Results' (which is selected), 'Execution details', and 'Execution graph'. A message in the results section states, 'This statement created a new table named days_with_rain.' At the bottom right of the results section is a 'Go to table' button.

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](#) which has significant precipitation.

- Add this query and run it:

The screenshot shows a BigQuery interface with the following details:

- Query:**

```

1 #standardSQL
2 # avg monthly precipitation
3 SELECT
4   AVG(prcp) AS average,
5   station_name,
6   date,
7   CURRENT_DATE() AS today,
8   DATE_DIFF(CURRENT_DATE(), date, DAY) AS partition_age,
9   EXTRACT(MONTH FROM date) AS month
10  FROM ecommerce.days_with_rain
11 WHERE station name = 'WAKAYAMA' #Japan

```
- Status:** Query completed.
- Results:**

Row	average	station_name	date	today	partition_age	month
1	1.06	WAKAYAMA	2025-06-15	2025-06-17	2	6
2	1.69	WAKAYAMA	2025-06-14	2025-06-17	3	6
3	2.05	WAKAYAMA	2025-06-10	2025-06-17	7	6
4	1.06	WAKAYAMA	2025-06-09	2025-06-17	0	6
- Page Controls:** Results per page: 50, 1 – 20 of 20, navigation arrows.

Task 6. Confirm the oldest partition_age is at or below 60 days

Update the ORDER BY clause to show the oldest partitions first.

- Add this query and run it:

The screenshot shows a BigQuery interface with the following details:

- Query:**

```

1 #standardSQL
2 # avg monthly precipitation
3
4 SELECT
5   AVG(prcp) AS average,
6   station_name,
7   date,
8   CURRENT_DATE() AS today,
9   DATE_DIFF(CURRENT_DATE(), date, DAY) AS partition_age,
10  EXTRACT(MONTH FROM date) AS month
11  FROM ecommerce.days with rain

```
- Status:** Query completed.
- Results:**

Row	average	station_name	date	today	partition_age	month
1	0.02	WAKAYAMA	2025-04-22	2025-06-17	56	4
2	0.43	WAKAYAMA	2025-04-23	2025-06-17	55	4
3	0.67	WAKAYAMA	2025-04-28	2025-06-17	50	4
4	1.06	WAKAYAMA	2025-05-09	2025-06-17	46	5
- Page Controls:** Results per page: 50, 1 – 20 of 20, navigation arrows.

Key Learnings

1. Partitioning Improves Efficiency

- a. Reduces **query costs** by scanning only relevant partitions.
- b. Avoids full table scans (e.g., 1.74 GB → 25 KB).

2. Auto-Expiring Partitions for Cost Control

- a. Ensures **old data is automatically purged**.
- b. Useful for **rolling time windows** (e.g., last 60 days).

3. BigQuery Optimization Best Practices

- a. Use PARTITION BY for large tables.
- b. Apply partition_expiration_days for transient data.

Conclusion

This project successfully demonstrated:

- **Creating and querying partitioned tables** in BigQuery.
- **Reducing query costs** by avoiding full scans.
- **Auto-expiring partitions** for efficient data lifecycle management.