Partitioned tables in BigQuery

Introduction to partitioned tables

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

You can partition BigQuery tables by:

- **Time-unit column**: Tables are partitioned based on a TIMESTAMP, DATE, or DATETIME column in the table. BigQuery automatically puts the data into the correct partition, based on the values in the column.

  For TIMESTAMP and DATETIME columns, the partitions can have either hourly, daily, monthly, or yearly granularity. For DATE columns, the partitions can have daily, monthly, or yearly granularity. Partitions boundaries are based on UTC time

- **Ingestion time**: Tables are partitioned based on the timestamp when BigQuery ingests the data.
- **Integer range**: Tables are partitioned based on an integer column.

Creating partitioned tables

Create an empty partitioned table

The steps to create a partitioned table in BigQuery are similar to creating a standard table, except that you specify the partitioning options, along with any other table options.

1.Create a time-unit column-partitioned table

To create an empty time-unit column-partitioned table with a schema definition:

1. Open the BigQuery page in the Cloud Console.
   Go to the BigQuery page
2. In the **Explorer** panel, expand your project and select a dataset.
3. Expand the more_vert **Actions** option and click **Open**.
4. In the details panel, click **Create table** add_box.
5. On the **Create table** page, in the **Source** section, select **Empty table.**
6. In the **Destination** section:
   - For **Dataset name**, choose the appropriate dataset.
   - In the **Table name** field, enter the name of the table.
   - Verify that **Table type** is set to **Native table**.
7. In the **Schema** section, enter the schema definition. Make sure the schema includes a DATE, TIMESTAMP, or DATETIME column for the partitioning column. For more information, see Specifying a schema.
8. In the **Partition and cluster settings** section, in the **Partitioning** drop-down list, select **Partition by field** and choose the partitioning column. This option is only available if the schema contains a DATE, TIMESTAMP, or DATETIME column.
9. Select the **Partitioning type** to choose daily, hourly, monthly, or yearly partitioning.
10. (Optional) To require a partition filter on all queries for this table, select the **Require partition filter** checkbox. Requiring a partition filter can reduce cost and improve performance. For more information, see Set partition filter requirements.
11. Click **Create table**.

OR

Create table using SQL

```
CREATE TABLE
  mydataset.newtable (transaction_id INT64, transaction_date DATE)
PARTITION BY
  transaction_date
OPTIONS(
  partition_expiration_days=3,
  require_partition_filter=true
)
```

2. Create an ingestion-time partitioned table

To create an empty ingestion-time partitioned table with a schema definition:

1. Open the BigQuery page in the Cloud Console.
   Go to the BigQuery page
2. In the **Explorer** panel, expand your project and select a dataset.
3. Expand the more_vert **Actions** option and click **Open**.
4. In the details panel, click **Create table** add_box.
5. On the **Create table** page, in the **Source** section, select **Empty table.**
6. In the **Destination** section:
   - For **Dataset name**, choose the appropriate dataset.
   - In the **Table name** field, enter the name of the table.
   - Verify that **Table type** is set to **Native table**.
7. In the **Schema** section, enter the schema definition.
8. In the **Partition and cluster settings** section, for **Partitioning**, click **Partition by ingestion time**.
9. (Optional) To require a partition filter on all queries for this table, select the **Require partition filter** checkbox. Requiring a partition filter can reduce cost and improve performance. For more information, see Set partition filter requirements.
10. Click **Create table**.

3. Create an integer-range partitioned table

To create an empty integer-range partitioned table with a schema definition:

1. Open the BigQuery page in the Cloud Console.
   Go to the BigQuery page
2. In the **Explorer** panel, expand your project and select a dataset.
3. Expand the more_vert **Actions** option and click **Open**.
4. In the details panel, click **Create table** add_box.
5. On the **Create table** page, in the **Source** section, select **Empty table.**
6. In the **Destination** section:
   - For **Dataset name**, choose the appropriate dataset.
   - In the **Table name** field, enter the name of the table.
   - Verify that **Table type** is set to **Native table**.
7. In the **Schema** section, enter the schema definition. Make sure the schema includes an INTEGER column for the partitioning column. For more information, see Specifying a schema.
8. In the **Partition and cluster settings** section, in the **Partitioning** drop-down list, select **Partition by field** and choose the partitioning column. This option is only available if the schema contains an INTEGER column.
9. Provide values for **Start**, **End**, and **Interval**:
   - **Start** is the start of first partition range (inclusive).
   - **End** is the end of last partition range (exclusive).
   - **Interval** is the width of each partition range.
10. Values outside of these ranges go into a special __UNPARTITIONED__ partition.
11. (Optional) To require a partition filter on all queries for this table, select the **Require partition filter** checkbox. Requiring a partition filter can reduce cost and improve performance. For more information, see Set partition filter requirements.
12. Click **Create table**.

Managing partitioned tables

Getting partition metadata using INFORMATION_SCHEMA views

When you query the INFORMATION_SCHEMA.PARTITIONS view, the query results contain one row for each partition. For example, the following query lists all of the table partitions in the dataset named mydataset:

```
SELECT table_name, partition_id, total_rows
FROM `mydataset.INFORMATION_SCHEMA.PARTITIONS`
WHERE partition_id IS NOT NULL
```