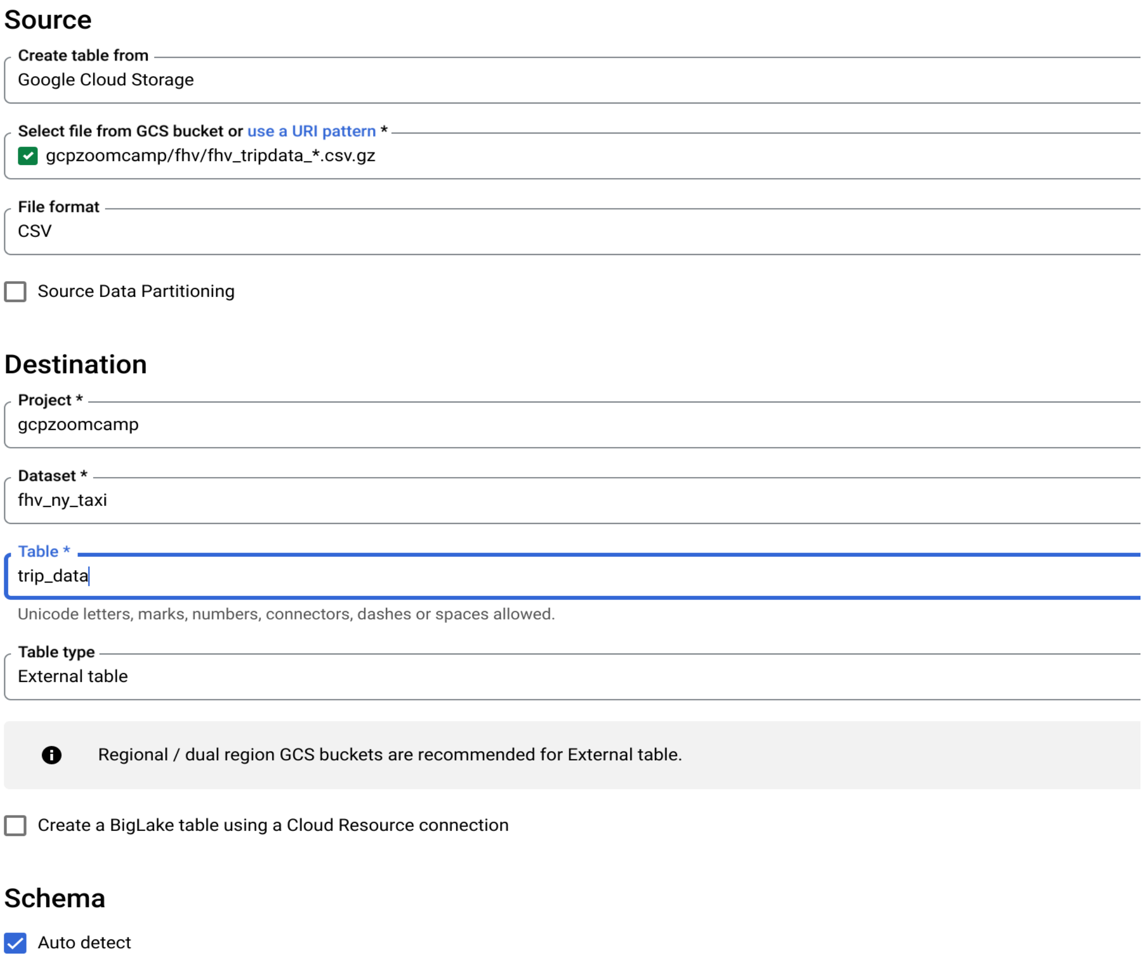


Source

Create table from: Google Cloud Storage

Select file from GCS bucket or use a URI pattern *: ✓ gcpzoomcamp/fhv/fhv_tripdata_*.csv.gz

File format: CSV

□ Source Data Partitioning

Destination

Project *: gcpzoomcamp

Dataset *: fhv_ny_taxi

Table *: trip_data

Unicode letters, marks, numbers, connectors, dashes or spaces allowed.

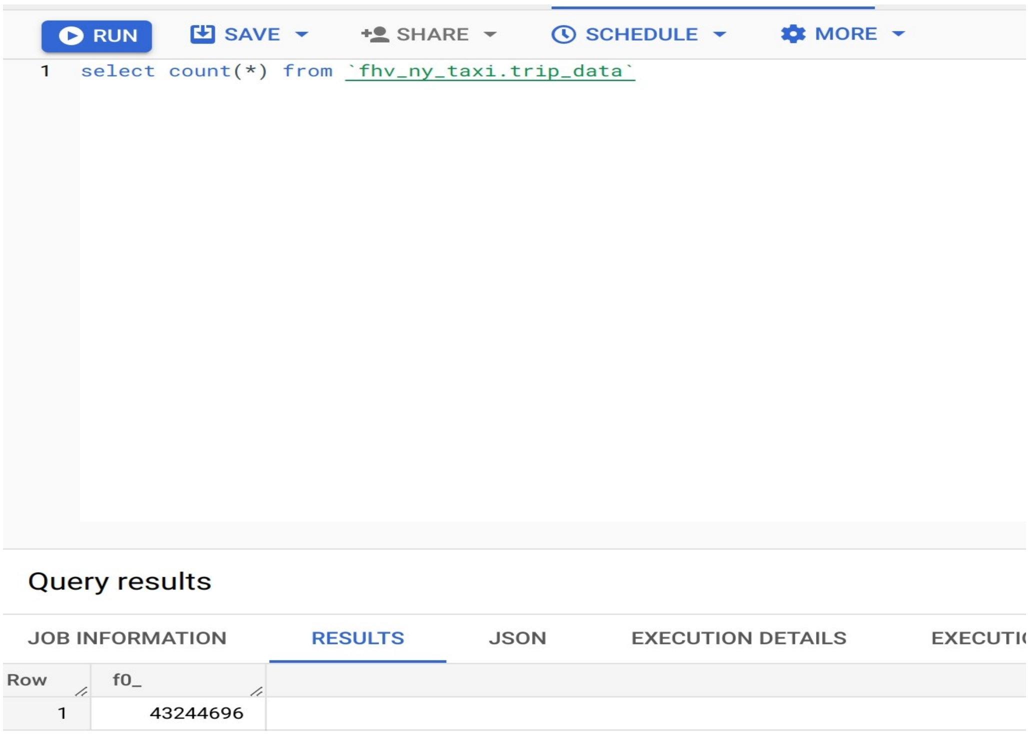
Table type: External table

Regional / dual region GCS buckets are recommended for External table.

□ Create a BigLake table using a Cloud Resource connection

Schema

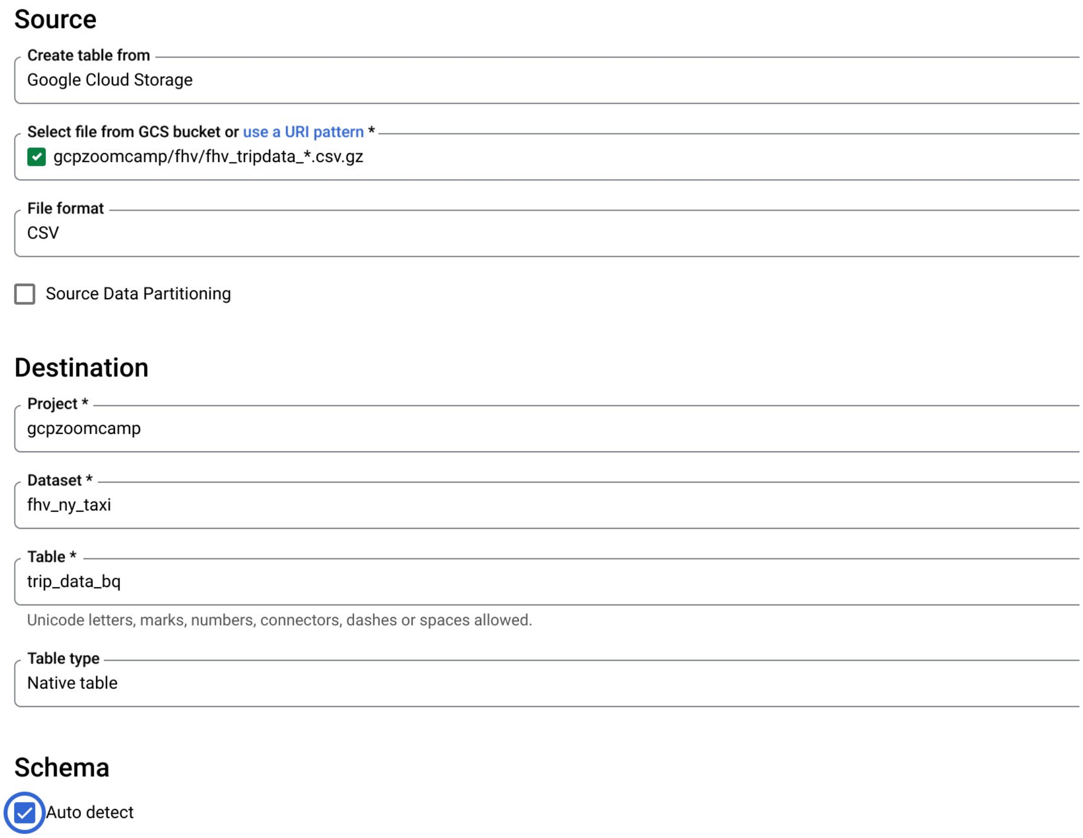
☑ Auto detect



```
select count(*) from `fhv_ny_taxi.trip_data`
```

Query results

Row	f0_
1	43244696



Source

Create table from
Google Cloud Storage

Select file from GCS bucket or use a URI pattern *
☑ gcpzoomcamp/fhv/fhv_tripdata_*.csv.gz

File format
CSV

□ Source Data Partitioning

Destination

Project *
gcpzoomcamp

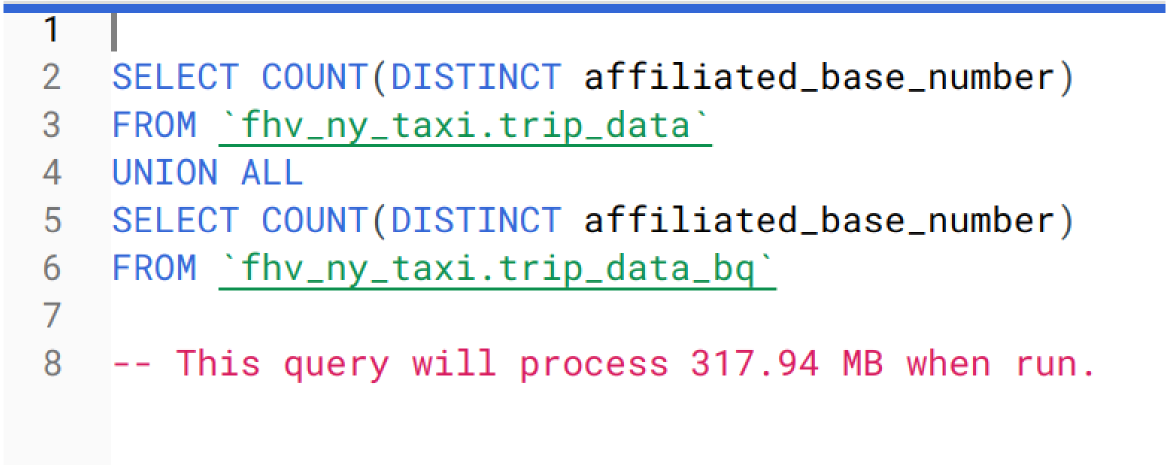
Dataset *
fhv_ny_taxi

Table *
trip_data_bq
Unicode letters, marks, numbers, connectors, dashes or spaces allowed.

Table type
Native table

Schema

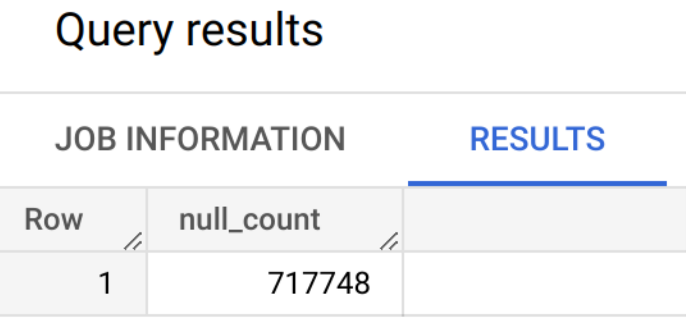
☑ Auto detect



```

SELECT COUNT(DISTINCT affiliated_base_number)
FROM `fhv_ny_taxi.trip_data`
UNION ALL
SELECT COUNT(DISTINCT affiliated_base_number)
FROM `fhv_ny_taxi.trip_data_bq`

-- This query will process 317.94 MB when run.
```

```
SELECT COUNT(DISTINCT affiliated_base_number)
FROM `fhv_ny_taxi.trip_data`
UNION ALL
SELECT COUNT(DISTINCT affiliated_base_number)
FROM `fhv_ny_taxi.trip_data_bq`

-- This query will process 317.94 MB when run.
```

```
# counting null rows in external table
SELECT COUNT(1) AS null_count
FROM `fhv_ny_taxi.trip_data`
WHERE PUlocationID IS NULL AND DOlocationID IS NULL
```



Query results

Row	null_count
1	717748

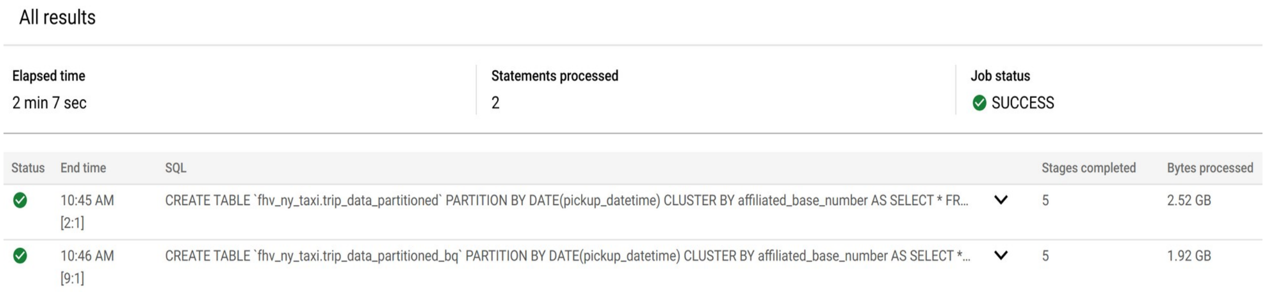
What is the best strategy to optimize the table if query always filter by pickup_datetime and order by affiliated_base_number?

Partitioning allows BigQuery to prune unneeded data, which can greatly improve query performance. Partitioning by pickup_datetime means that BigQuery will only scan data for the relevant date range, rather than having to scan the entire table.

Clustering the table on affiliated_base_number reorders the data within each partition based on the values in a set of columns, which can further optimize queries that filter and order by those columns. In this case, clustering on affiliated_base_number would allow BigQuery to efficiently retrieve the data for each affiliation_base_number value, as the data for each value would be physically stored together.

So, partition by pickup_datetime, cluster on affiliated_base_number".

```
-- Partition external table – create new
CREATE TABLE `fhv_ny_taxi.trip_data_partitioned`
PARTITION BY DATE(pickup_datetime)
CLUSTER BY affiliated_base_number
AS
SELECT *
FROM `fhv_ny_taxi.trip_data`;

-- Partition internal table – create new
CREATE TABLE `fhv_ny_taxi.trip_data_partitioned_bq`
PARTITION BY DATE(pickup_datetime)
CLUSTER BY affiliated_base_number
AS
SELECT *
FROM `fhv_ny_taxi.trip_data_bq`
```



All results

Elapsed time	Statements processed	Job status
2 min 7 sec	2	✓ SUCCESS

Status	End time	SQL		Stages completed	Bytes processed
✓	10:45 AM [2:1]	CREATE TABLE `fhv_ny_taxi.trip_data_partitioned` PARTITION BY DATE(pickup_datetime) CLUSTER BY affiliated_base_number AS SELECT * FR...	⌄	5	2.52 GB
✓	10:46 AM [9:1]	CREATE TABLE `fhv_ny_taxi.trip_data_partitioned_bq` PARTITION BY DATE(pickup_datetime) CLUSTER BY affiliated_base_number AS SELECT *...	⌄	5	1.92 GB

# Internal table

```
-- This query will process 647.87 MB when run
SELECT DISTINCT affiliated_base_number
FROM `fhv_ny_taxi.trip_data_bq`
WHERE pickup_datetime >= '2019-03-01'
AND pickup_datetime <= '2019-03-31 23:59:59'
ORDER BY affiliated_base_number;

-- This query will process 23.05 MB when run
SELECT DISTINCT affiliated_base_number
FROM `fhv_ny_taxi.trip_data_partitioned_bq`
WHERE pickup_datetime >= '2019-03-01'
AND pickup_datetime <= '2019-03-31 23:59:59'
ORDER BY affiliated_base_number;
```

# External table

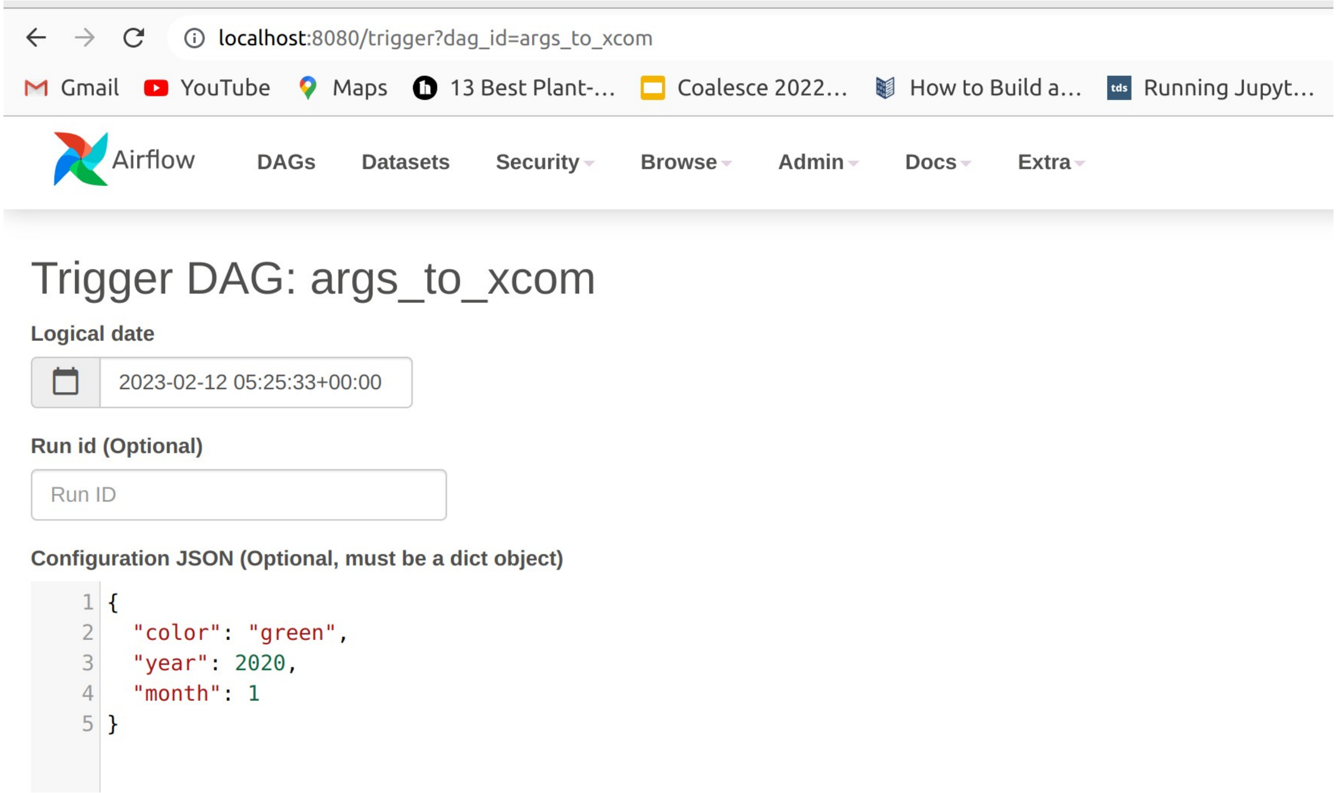
```
-- This query will process 0 MB when run (when executed shuffled 25.05 KB)
SELECT DISTINCT affiliated_base_number
FROM `fhv_ny_taxi.trip_data`
WHERE pickup_datetime >= '2019-03-01'
AND pickup_datetime <= '2019-03-31 23:59:59'
ORDER BY affiliated_base_number;

-- This query will process 23.05 MB when run
SELECT DISTINCT affiliated_base_number
FROM `fhv_ny_taxi.trip_data_partitioned`
WHERE pickup_datetime >= '2019-03-01'
AND pickup_datetime <= '2019-03-31 23:59:59'
ORDER BY affiliated_base_number
```

External tables in BigQuery are stored outside of BigQuery in a storage location - Google Cloud Storage (GCS) in my case.
The data remains in the external storage and is not imported into BigQuery.

False.
Clustering in BigQuery can be used to improve query performance by rearranging the data based on one or more columns.
Clustering works by creating a sorted, immutable, table that is physically stored in the same order as the clustered columns.
When you run a query that filters or sorts by the clustered columns, BigQuery can access the data more efficiently, reducing query latency and cost.

However, not all use cases benefit from clustering, and there may be additional costs associated with creating and maintaining a clustered table.
It is therefore important to carefully evaluate your data and query patterns to determine whether clustering is appropriate for your needs.
In some cases, it may be more beneficial to use other performance optimization techniques such as partitioning, indexing, or materialized views.

Running dag with config to push parameters to xcom in airflow:



Trigger DAG: args_to_xcom

Logical date

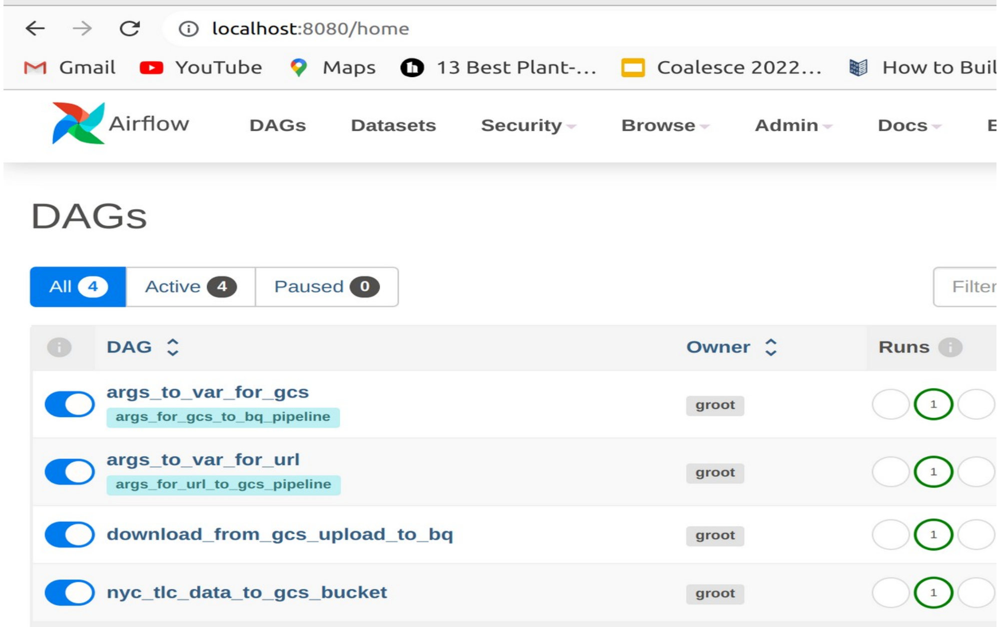
2023-02-12 05:25:33+00:00

Run id (Optional)

Run ID

Configuration JSON (Optional, must be a dict object)

```
{
  "color": "green",
  "year": 2020,
  "month": 1
}
```



DAGs

All 4 | Active 4 | Paused 0

DAG	Owner	Runs
args_to_var_for_gcs args_for_gcs_to_bq_pipeline	groot	1
args_to_var_for_url args_for_url_to_gcs_pipeline	groot	1
download_from_gcs_upload_to_bq	groot	1
nyc_tlc_data_to_gcs_bucket	groot	1