

Week 3 Homework: Data Warehouse & BigQuery

Georgios Grigoriou

georgiosvgrigoriou@gmail.com

Deadline: 13 February (Monday), 22:00 CET

Important Note:

You can load the data however you would like, but keep the files in .GZ Format. If you are using orchestration such as Airflow or Prefect do not load the data into Big Query using the orchestrator.

Stop with loading the files into a bucket.

NOTE: You can use the CSV option for the GZ files when creating an External Table

SETUP:

Create an external table using the fhv 2019 data.

Create a table in BQ using the fhv 2019 data (do not partition or cluster this table).

Data can be found here:

<https://github.com/DataTalksClub/nyc-tlc-data/releases/tag/fhv>

In this homework, I familiarised myself with BigQuery orchestration tool as part of the [Data Engineering Zoomcamp course](#). The code that was used can be found [here](#)

Preparation

Data were loaded in Google Cloud Storage

Google Cloud dtc-de Search (/) for resources, docs, products and more

Bucket details

prefect-de-zoomcampgg

Location: eu (multiple regions in European Union) Storage class: Standard Public access: Not public Protection: None

OBJECTS CONFIGURATION PERMISSION PROTECTION LIFECYCLE OBSERVABILITY NEW

Buckets > prefect-de-zoomcampgg > data > fhv

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER TRANSFER DATA MANAGE HOLDS DOWNLOAD DELETE

Filter by name prefix only Filter Filter objects and folders

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history
<input type="checkbox"/>	fhv_tripdata_2019-01.csv.gz	231 MB	application/x-gzip	13 Feb 2023, 13:40:37	Standard	13 Feb 2023, 13:40:37	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-02.csv.gz	14.3 MB	application/x-gzip	13 Feb 2023, 13:40:09	Standard	13 Feb 2023, 13:40:09	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-03.csv.gz	12.7 MB	application/x-gzip	13 Feb 2023, 13:40:08	Standard	13 Feb 2023, 13:40:08	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-04.csv.gz	16.4 MB	application/x-gzip	13 Feb 2023, 13:40:12	Standard	13 Feb 2023, 13:40:12	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-05.csv.gz	17.6 MB	application/x-gzip	13 Feb 2023, 13:40:13	Standard	13 Feb 2023, 13:40:13	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-06.csv.gz	17.2 MB	application/x-gzip	13 Feb 2023, 13:40:16	Standard	13 Feb 2023, 13:40:16	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-07.csv.gz	16.7 MB	application/x-gzip	13 Feb 2023, 13:40:17	Standard	13 Feb 2023, 13:40:17	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-08.csv.gz	16.7 MB	application/x-gzip	13 Feb 2023, 13:40:20	Standard	13 Feb 2023, 13:40:20	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-09.csv.gz	12 MB	application/x-gzip	13 Feb 2023, 13:40:20	Standard	13 Feb 2023, 13:40:20	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-10.csv.gz	18.5 MB	application/x-gzip	13 Feb 2023, 13:40:25	Standard	13 Feb 2023, 13:40:25	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-11.csv.gz	18.3 MB	application/x-gzip	13 Feb 2023, 13:40:25	Standard	13 Feb 2023, 13:40:25	Not public	—
<input type="checkbox"/>	fhv_tripdata_2019-12.csv.gz	19.9 MB	application/x-gzip	13 Feb 2023, 13:40:28	Standard	13 Feb 2023, 13:40:28	Not public	—

Create an external table using the fhv 2019 data.

-- Creating external table referring to gcs path

```
CREATE OR REPLACE EXTERNAL TABLE `<project id>.dezoomcamp.fhv_trip_data`
OPTIONS (
  format = 'CSV',
  uris = ['gs://prefect-de-zoomcampgg/data/fhv/fhv_tripdata_2019-*.csv.gz']
);
```

Create a table in BQ using the fhv 2019 data (do not partition or cluster this table).

```
CREATE OR REPLACE TABLE `<project id>.dezoomcamp.fhv_nonpartitioned_trip_data`
AS SELECT * FROM `<project id>.dezoomcamp.fhv_trip_data`;
```

The screenshot shows the Google Cloud BigQuery console. On the left, the Explorer pane shows a project named 'braided-case-375422' with a dataset 'dezoomcamp' containing tables 'fhv_nonpartitioned_trip_data', 'fhv_trip_data', and 'rides'. The main editor shows a SQL query: `SELECT count(1) FROM `braided-case-375422.dezoomcamp.fhv_nonpartitioned_trip_data` WHERE PUlocationID is NULL and DOlocationID is NULL;`. The 'Query results' pane at the bottom shows a single row with the count 717748.

Row	count(1)
1	717748

Question 1: What is the count for fhv vehicle records for year 2019?

- 65,623,481
- 43,244,696
- 22,978,333
- 13,942,414

Question 1: Solution

The correct answer is **43,244,696**.

By running the following SQL Command in BigQuery

```
-- Count of fhv trips
```

```
SELECT count(*)
```

```
FROM `<project id>.dezoomcamp.fhv_trip_data`
```

```
WHERE DATE(pickup_datetime) BETWEEN "2019-01-01" AND "2019-12-31";
```

Or

```
-- Count of fhv trips
```

```
SELECT count(*) FROM `<project id>.dezoomcamp.fhv_nonpartitioned_trip_data`;
```

We get the result of 43,244,696

The screenshot shows a BigQuery interface. At the top, there are buttons for RUN, SAVE, SHARE, SCHEDULE, and MORE. Below these is a text area containing a SQL query:

```
1 SELECT count(*)
2 FROM `braided-case-375422.dezoomcamp.fhv_trip_data`
3 WHERE DATE(pickup_datetime) BETWEEN "2019-01-01" AND "2019-12-31";
```

Below the query editor, there is a section for "Query results" with options to SAVE RESULTS, EXPLORE DATA, and a refresh icon. Below this is a tabbed interface with tabs for JOB INFORMATION, RESULTS (selected), JSON, EXECUTION DETAILS, and EXECUTION. The RESULTS tab shows a table with one row and one column:

Row	f0_
1	43244696

At the bottom right of the interface, there is a text prompt: "Press Alt+F1 for accessibility options".

Question 2: Write a query to count the distinct number of affiliated_base_number for the entire dataset on both the tables. What is the estimated amount of data that will be read when this query is executed on the External Table and the Table?

- 25.2 MB for the External Table and 100.87MB for the BQ Table
- 225.82 MB for the External Table and 47.60MB for the BQ Table
- 0 MB for the External Table and 0MB for the BQ Table
- **0 MB for the External Table and 317.94MB for the BQ Table**

Question 2. Solution

The correct answer is

- **0 MB for the External Table and 317.94MB for the BQ Table**

By running the following SQL Commands we get the above-mentioned answer

```
SELECT COUNT(DISTINCT(Affiliated_base_number)) FROM `<project  
id>.dezoomcamp.fhv_trip_data`;
```

The screenshot displays the Google Cloud BigQuery interface. At the top, a toolbar includes buttons for RUN, SAVE, SHARE, SCHEDULE, and MORE, along with a 'Query' status indicator. The SQL query is entered in the editor: `SELECT COUNT(DISTINCT(Affiliated_base_number)) FROM `braided-case-375422.dezoomcamp.fhv_trip_data`;`. Below the editor, the 'Query results' section is active, showing a table of job information. The table lists details such as Job ID, User, Location, Creation time, Start time, End time, Duration, Bytes processed, Bytes billed, Job priority, Use legacy SQL, and Destination table. The 'Destination table' is noted as a 'Temporary table'.

Query results	
JOB INFORMATION	RESULTS
Job ID	braided-case-375422:EU.bqjob_749e4422_1864c68c654
User	g.v.grigiousb@gmail.com
Location	EU
Creation time	13 Feb 2023, 20:13:00 UTC
Start time	13 Feb 2023, 20:13:00 UTC
End time	13 Feb 2023, 20:13:01 UTC
Duration	0 sec
Bytes processed	0 B (results cached)
Bytes billed	0 B
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table

```
1 SELECT COUNT(DISTINCT(Affiliated_base_number)) FROM `braided-case-375422.dezoomcamp.fhv_nonpartitioned_trip_data`;
```

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

[JOB INFORMATION](#) RESULTS JSON EXECUTION DETAILS EXECUTION

Job ID	braided-case-375422:EU.bquxjob_751189d2_1864c35ed9c
User	g.v.grigoriousb@gmail.com
Location	EU
Creation time	13 Feb 2023, 19:17:28 UTC
Start time	13 Feb 2023, 19:17:28 UTC
End time	13 Feb 2023, 19:17:29 UTC
Duration	0 sec
Bytes processed	317.94 MB
Bytes billed	318 MB
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table

Question 3: How many records have both a blank (null) PUlocationID and DOlocationID in the entire dataset?

- **717,748**
- 1,215,687
- 5
- 20,332

Question 3. Solution

The correct answer is **717,748**. By running the SQL command

```
SELECT count(1) FROM `<project id>.dezoomcamp.fhv_nonpartitioned_trip_data`  
WHERE PUlocationID is NULL and DOlocationID is NULL;
```

</

Partitioning my table by pickup_datetime will allow BigQuery to only scan the relevant partitions for my query, reducing the amount of data that needs to be read and improving query performance. This is particularly useful if data is time-sensitive and you run queries for specific time ranges.

Clustering on affiliated_base_number will physically store the data in a way that makes it more efficient to retrieve records that are sorted by affiliated_base_number. Since my queries are filtering by pickup_datetime and ordering by affiliated_base_number, this strategy will provide significant performance benefits by reducing the amount of data that needs to be read and returning the results in the order you need.

With this approach, I am optimising my table for the specific queries that I am running, making my queries faster and more efficient.

Question 5: Implement the optimized solution you chose for question 4. Write a query to retrieve the distinct affiliated_base_number between pickup_datetime 2019/03/01 and 2019/03/31 (inclusive). Use the BQ table you created earlier in your from clause and note the estimated bytes. Now change the table in the from clause to the partitioned table you created for question 4 and note the estimated bytes processed. What are these values? Choose the answer which most closely matches.

- 12.82 MB for non-partitioned table and 647.87 MB for the partitioned table
- **647.87 MB for non-partitioned table and 23.06 MB for the partitioned table**
- 582.63 MB for non-partitioned table and 0 MB for the partitioned table
- 646.25 MB for non-partitioned table and 646.25 MB for the partitioned table

Question 5 Solution

The correct answer is **647.87 MB for non-partitioned table and 23.06 MB for the partitioned table.**

First, I created the partitioned table


```
-- Create partition table

CREATE OR REPLACE TABLE `<project id>.dezoomcamp.fhv_partitioned_trip_data_2019`
PARTITION BY DATE(pickup_datetime)
CLUSTER BY Affiliated_base_number
AS SELECT * FROM `<project id>.dezoomcamp.fhv_trip_data`;
```

Then, I selected the table that is neither partitioned nor clustered, which gives 647.87 MB Bytes processed

```
-- WITH partitioning on pickup_datetime and clustering on Affiliated_base_number

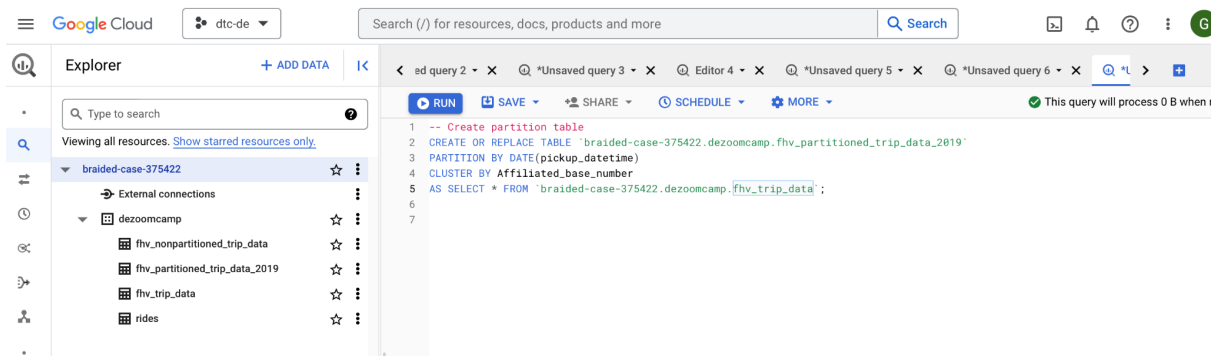
SELECT DISTINCT Affiliated_base_number, FROM `<project
id>.dezoomcamp.fhv_nonpartitioned_trip_data`
WHERE DATE(pickup_datetime) BETWEEN '2019-03-01' AND '2019-03-31';
```

The screenshot displays the Google Cloud BigQuery interface. On the left, the 'Explorer' pane shows a project named 'braided-case-375422' with a dataset 'dezoomcamp' containing tables: 'fhv_nonpartitioned_trip_data', 'fhv_partitioned_trip_data_2019', 'fhv_trip_data', and 'rides'. The main editor shows a SQL query:


```
1 -- select on NON partitioned and non clustered table
2 SELECT DISTINCT Affiliated_base_number, FROM `braided-case-375422.dezoomcamp.fhv_nonpartitioned_trip_data`
3 WHERE DATE(pickup_datetime) BETWEEN '2019-03-01' AND '2019-03-31';
4
```

 The 'Query results' pane at the bottom shows the execution details for the job 'braided-case-375422:EU.bqjob_57c2eb4d_1864c55eee5'. The job status is 'COMPLETED'. The 'Bytes processed' is 647.87 MB, and the 'Bytes billed' is 648 MB. The job priority is 'INTERACTIVE'.

Job ID	braided-case-375422:EU.bqjob_57c2eb4d_1864c55eee5
User	g.v.grigoriou@gmail.com
Location	EU
Creation time	13 Feb 2023, 19:52:25 UTC
Start time	13 Feb 2023, 19:52:25 UTC
End time	13 Feb 2023, 19:52:26 UTC
Duration	0 sec
Bytes processed	647.87 MB
Bytes billed	648 MB
Job priority	INTERACTIVE



Then, I selected the table with partitioning on pickup_datetime and clustering on Affiliated_base_number, which gives 23.05 MB Bytes processed

```

-- WITH partitioning on pickup_datetime and clustering on Affiliated_base_number
SELECT DISTINCT Affiliated_base_number, FROM `<project
id>.dezoomcamp.fhv_nonpartitioned_trip_data`
WHERE DATE(pickup_datetime) BETWEEN '2019-03-01' AND '2019-03-31';

```

The screenshot shows the Google Cloud BigQuery Explorer interface with the same project and dataset. The SQL query is updated to:

```

1 -- WITH partitioning on pickup_datetime and clustering on Affiliated_base_number
2 SELECT DISTINCT Affiliated_base_number, FROM `braided-case-375422.dezoomcamp.fhv_partitioned_trip_data_2019`
3 WHERE DATE(pickup_datetime) BETWEEN '2019-03-01' AND '2019-03-31';
4

```

Below the query editor, the 'Query results' section is visible, showing job information:

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Job ID	braided-case-375422:EU.bqxjob_4d1262_1864c5a4bba				
User	g.v.grigoriou@gmail.com				
Location	EU				
Creation time	13 Feb 2023, 19:57:11 UTC				
Start time	13 Feb 2023, 19:57:11 UTC				
End time	13 Feb 2023, 19:57:12 UTC				
Duration	0 sec				
Bytes processed	23.05 MB				
Bytes billed	24 MB				
Job priority	INTERACTIVE				

Question 6:

Where is the data stored in the External Table you created?

- Big Query
- **GCP Bucket**
- Container Registry
- Big Table

Question 6 Solution

The correct answer is **GCP Bucket**. The data in an External Table in BigQuery is stored in a Google Cloud Storage (GCS) bucket or a Cloud Storage file system.

Question 7:

It is best practice in Big Query to always cluster your data:

- True
- **False**

Question 7 Solution

The correct answer is **False**. Clustering is a performance optimization in BigQuery that rearranges the physical storage of table data to optimize query performance by reducing the amount of data that needs to be read. While clustering can improve query performance, it is not always necessary, and it depends on the nature of your data and the types of queries you run.

For example, if your data is already sorted in the order that you need to query it, then clustering is not necessary because the data is already well organized for efficient querying. If your queries tend to filter data based on a specific set of columns, then clustering on those columns may provide performance benefits.

In general, whether or not clustering is necessary depends on the specific use case and data patterns. Before enabling clustering, it is important to understand the trade-offs and performance implications. If you're unsure, you can experiment with clustering to see if it provides performance benefits for your specific queries and data.

