BigQuery: Cost Optimization and Data Scan Restriction Tips

Thanusha Deepthi · Follow
Published in Searce · 7 min read · Mar 16, 2022

152

What is BigQuery :

BigQuery provides fully managed, serverless, and highly scalable cloud data warehouse designed for business agility. It is equipped with a built-in query engine that can run SQL queries on terabytes of data in seconds and petabytes in minutes.

Why BigQuery costs add up so quickly :

BigQuery is unpredictable when it comes to pricing**.** BigQuery mainly charges you for the storage and the queries but operations such as loading data, copying data, exporting data are free. This essentially means that if developers use unoptimised queries, it can result in thousands of dollars in cost. This case is quite common in large organisations, especially those utilising pay-as-you-go models. Identifying a cost spike can be challenging, but it's essential because if we don't, the costs are likely to rise. The more complex the query or the more data the query has to scan the faster the costs add up so, understanding the pricing architecture helps in controlling costs, optimising query performance and optimising storage.

Pricing Model :

BigQuery has two pricing models for running queries:

- **On-demand pricing:** You pay for the number of bytes processed by each query.
- **Flat-rate pricing**: You pay for dedicated query processing capacity, measured in slots.

The factors that govern Google BigQuery Pricing are Storage and Query Data Processed.

Factors deciding query performance and cost :

- How many bytes a query reads?
- How many bytes your query writes?
- How much CPU time does a query require?

Methods to restrict data scan :

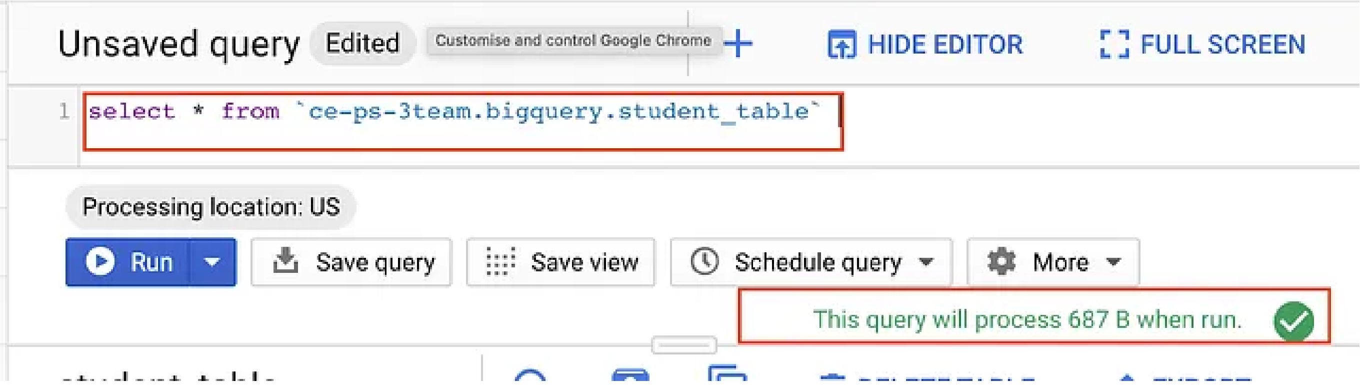
1. Avoid “ SELECT * “ statement select only required columns.
2. ‘LIMIT’ clause does not limit the data scanned. Use ‘Maximum bytes billed’ in query settings.
3. Use Partitioning and clustering of tables wherever possible.
4. De-normalize the data and take advantage of nested and repeated fields.
5. Create materialized views to keep aggregated data.
6. Use cached results of queries.
7. Use Query validator or dry run
8. Avoid external data sources if performance is a priority.
9. Use ‘ORDER BY’ wisely

Let’s get into the detailed explanation of these methods.

1. Avoid “ SELECT * “ statement :

You should avoid the select * statement and mention only those columns in the query that are needed. Actually **select *** is the most expensive way to query the data.

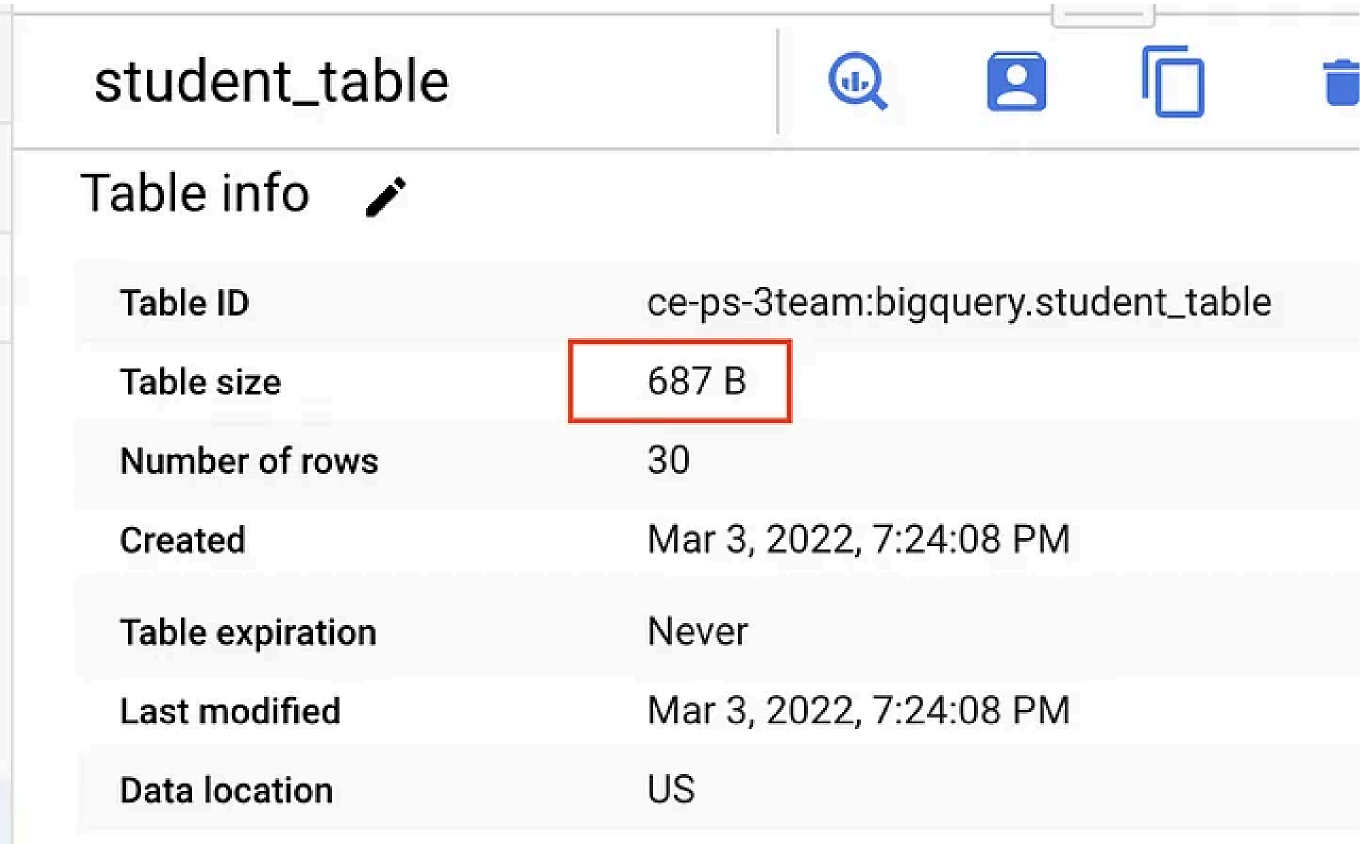
Bigquery storage is a columnar storage where individual columns are stored independently. Hence, instead of selecting all the columns, if we select only required columns then the other columns remain untouched and the query will scan only the required columns. This reduces the data read which also saves costs.



```
select * from `ce-ps-3team.bigquery.student_table`
```

Processing location: US

This query will process 687 B when run.



student_table

Table info

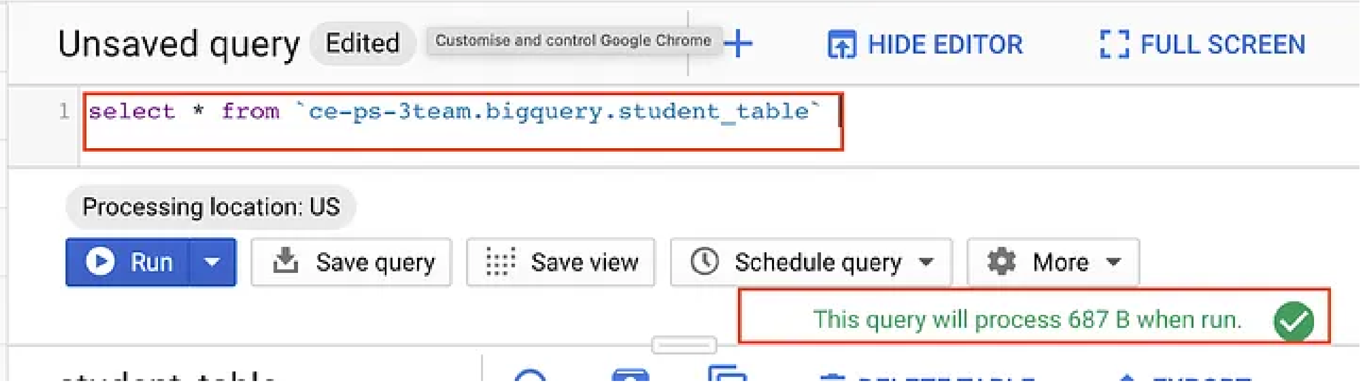
Table ID	ce-ps-3team:bigquery.student_table
Table size	687 B
Number of rows	30
Created	Mar 3, 2022, 7:24:08 PM
Table expiration	Never
Last modified	Mar 3, 2022, 7:24:08 PM
Data location	US

Select * scanned the whole table and hence the cost gets affected.

2. LIMIT clause does not limit the data scanned :

“ **Select * from table limit 100**“ is as costly as “ **Select * from table** “ Limit will have the full table scanned, it returns only the limited number of rows.

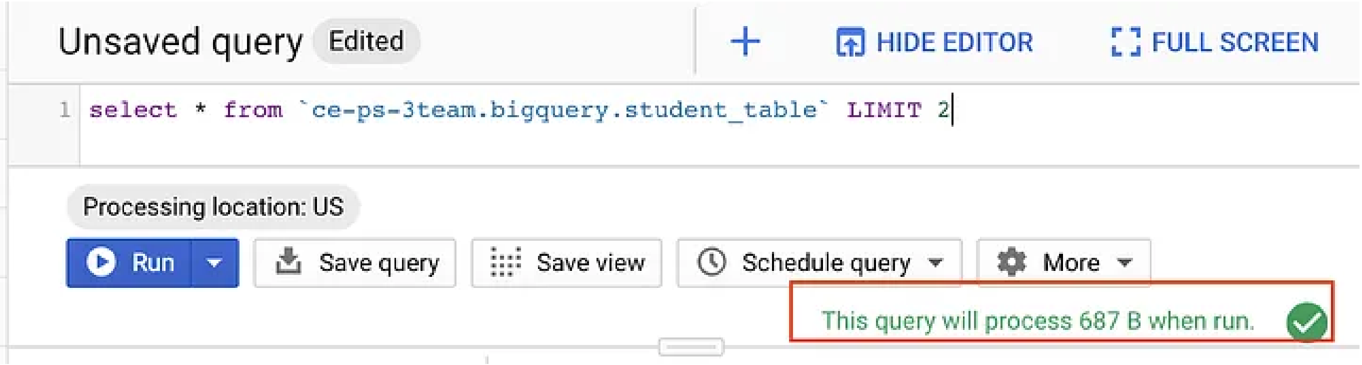
With **select ***



```
select * from `ce-ps-3team.bigquery.student_table`
```

This query will process 687 B when run.

With **LIMIT**



```
select * from `ce-ps-3team.bigquery.student_table` LIMIT 2
```

This query will process 687 B when run.

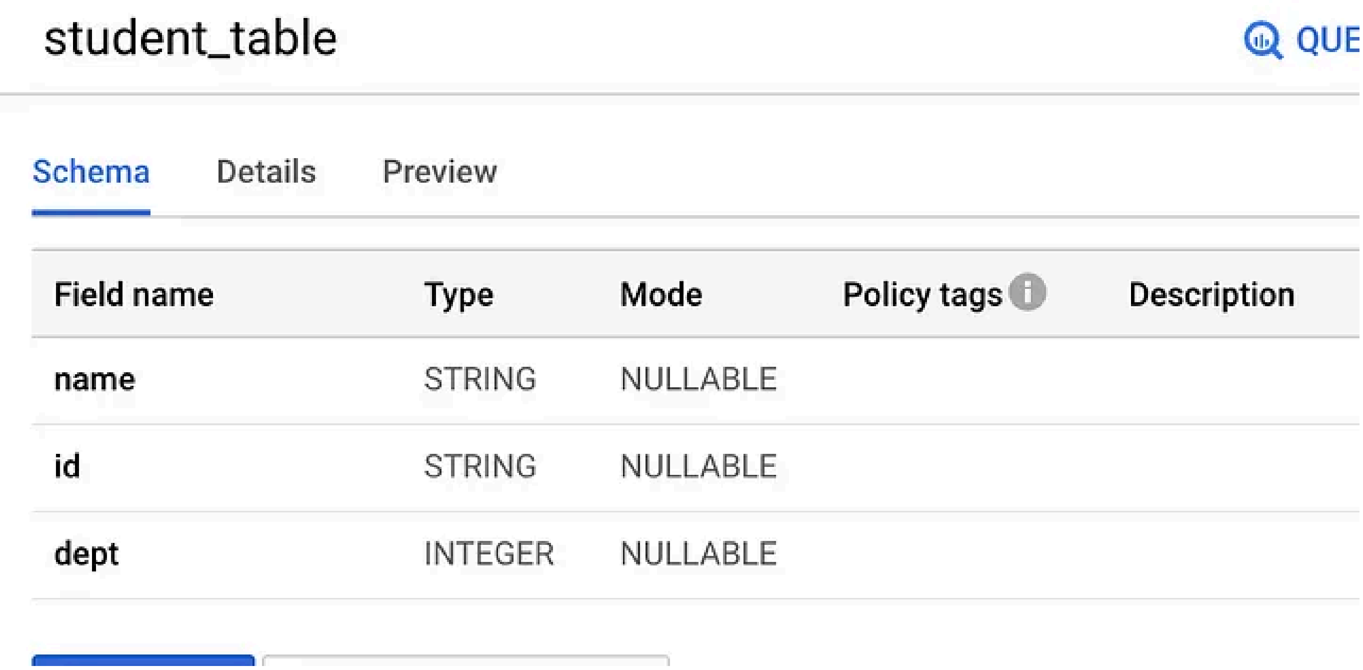
If you just want to explore the data, then do a preview on the table which is absolutely free. But, the LIMIT clause will not limit how much data can be read. It will scan the whole table and you will be charged for reading the bytes in the table. If you want to restrict the number of bytes to be billed in the table, use “ maximum bytes billed ”.

> Note : If you strictly want to limit the number of bytes billed for a query then use the maximum bytes billed in “Query settings “ to limit query costs.

3. Partitioning and clustering :

Use partitioning or clustering or both on the table wherever possible. Partitioning and clustering are great methods to restrict the data scanned. Partitions could be created on the basis of Ingestion Time, DATE/TIMESTAMP column, or an INTEGER RANGE column. There can be a maximum of 4000 partitions per partitioned table.

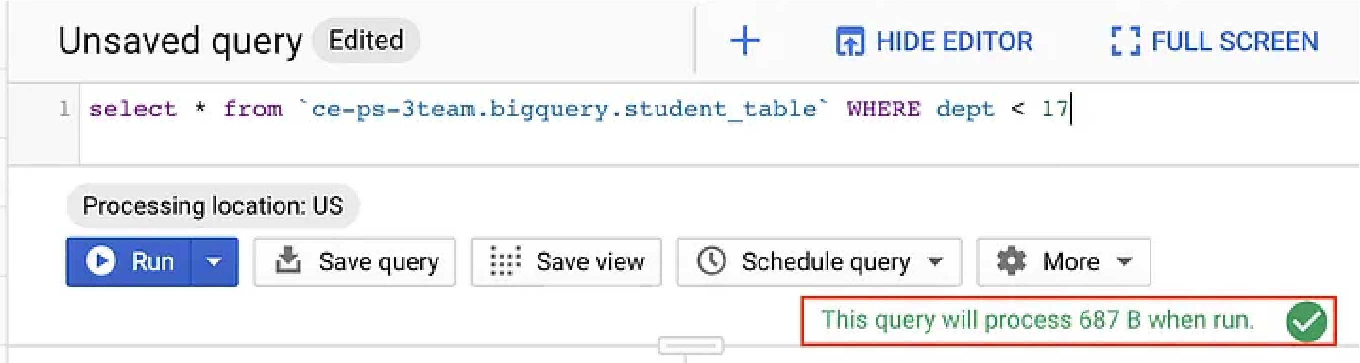
Table schema we are using here :



Field name	Type	Mode	Policy tags	Description
name	STRING	NULLABLE		
id	STRING	NULLABLE		
dept	INTEGER	NULLABLE		

Table schema

Without Partitioning

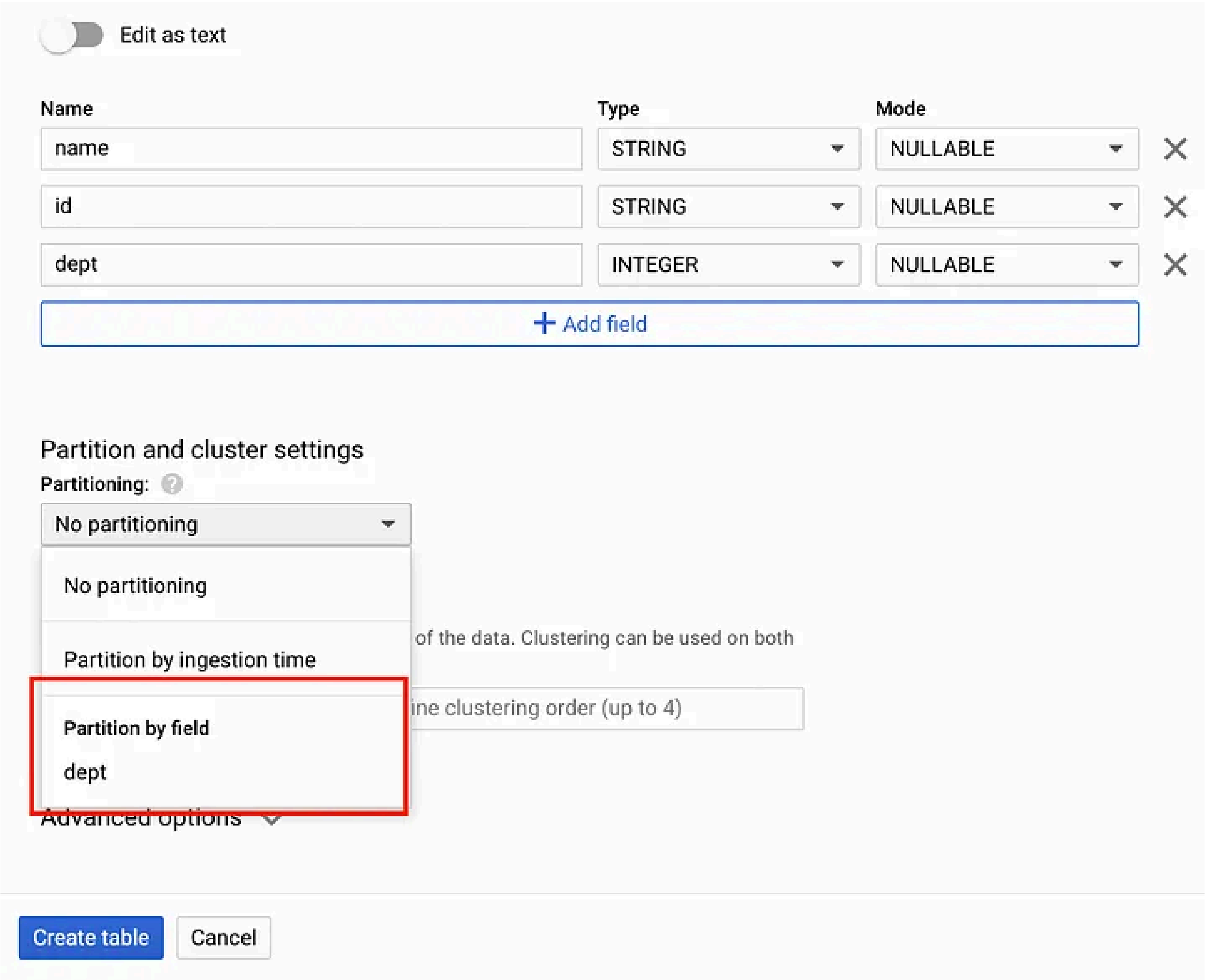


```
select * from `ce-ps-3team.bigquery.student_table` WHERE dept < 17
```

This query will process 687 B when run.

With Partitioning

Partitioning can be done by selecting the partitioning field while creating the table.

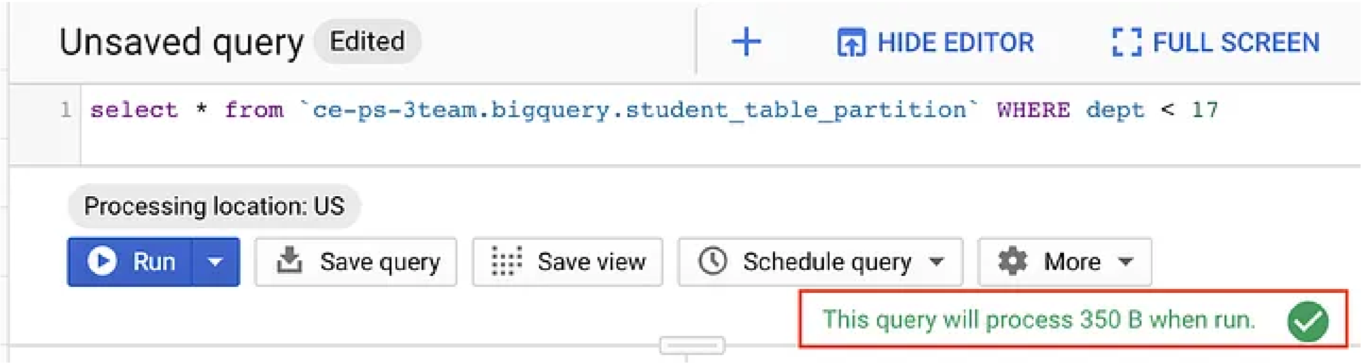


Edit as text
Add field
Partition and cluster settings
Advanced options
Create table
Cancel

Name	Type	Mode
name	STRING	NULLABLE
id	STRING	NULLABLE
dept	INTEGER	NULLABLE

Partitioning: No partitioning

No partitioning
Partition by ingestion time
Partition by field
dept
of the data. Clustering can be used on both
ine clustering order (up to 4)



```
select * from `ce-ps-3team.bigquery.student_table_partition` WHERE dept < 17
```

Processing location: US

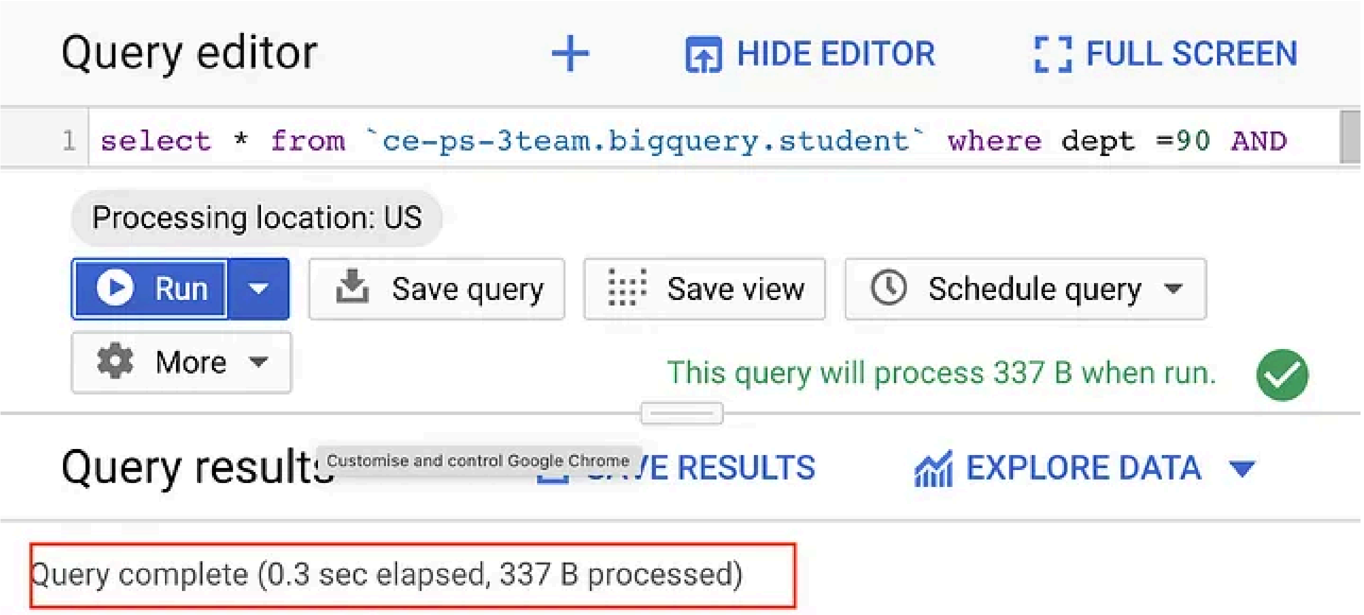
This query will process 350 B when run.

with partitioning

Clustering:

Clustering could be done on fields of different types such as DATE, NUMERIC, BOOL, STRING, TIMESTAMP, etc. The WHERE clause should have an additional condition on the clustered field.

Without Clustering

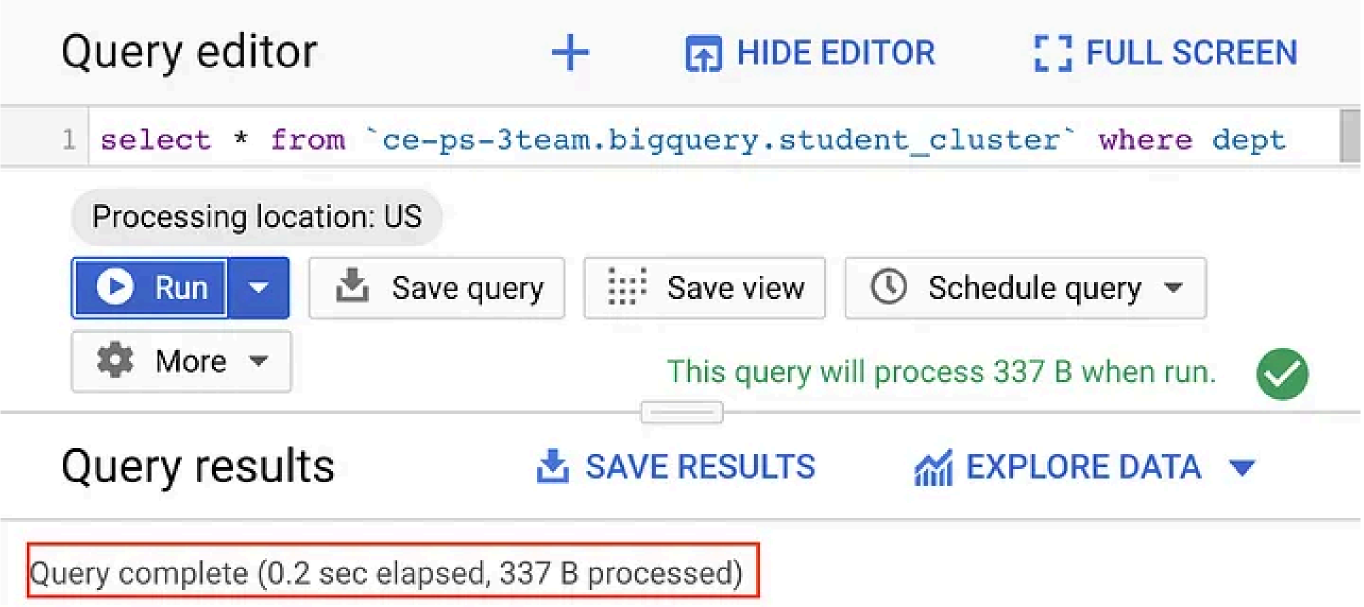


```
select * from `ce-ps-3team.bigquery.student` where dept =90 AND
```

This query will process 337 B when run.

Query complete (0.3 sec elapsed, 337 B processed)

With Clustering



```
select * from `ce-ps-3team.bigquery.student_cluster` where dept
```

This query will process 337 B when run.

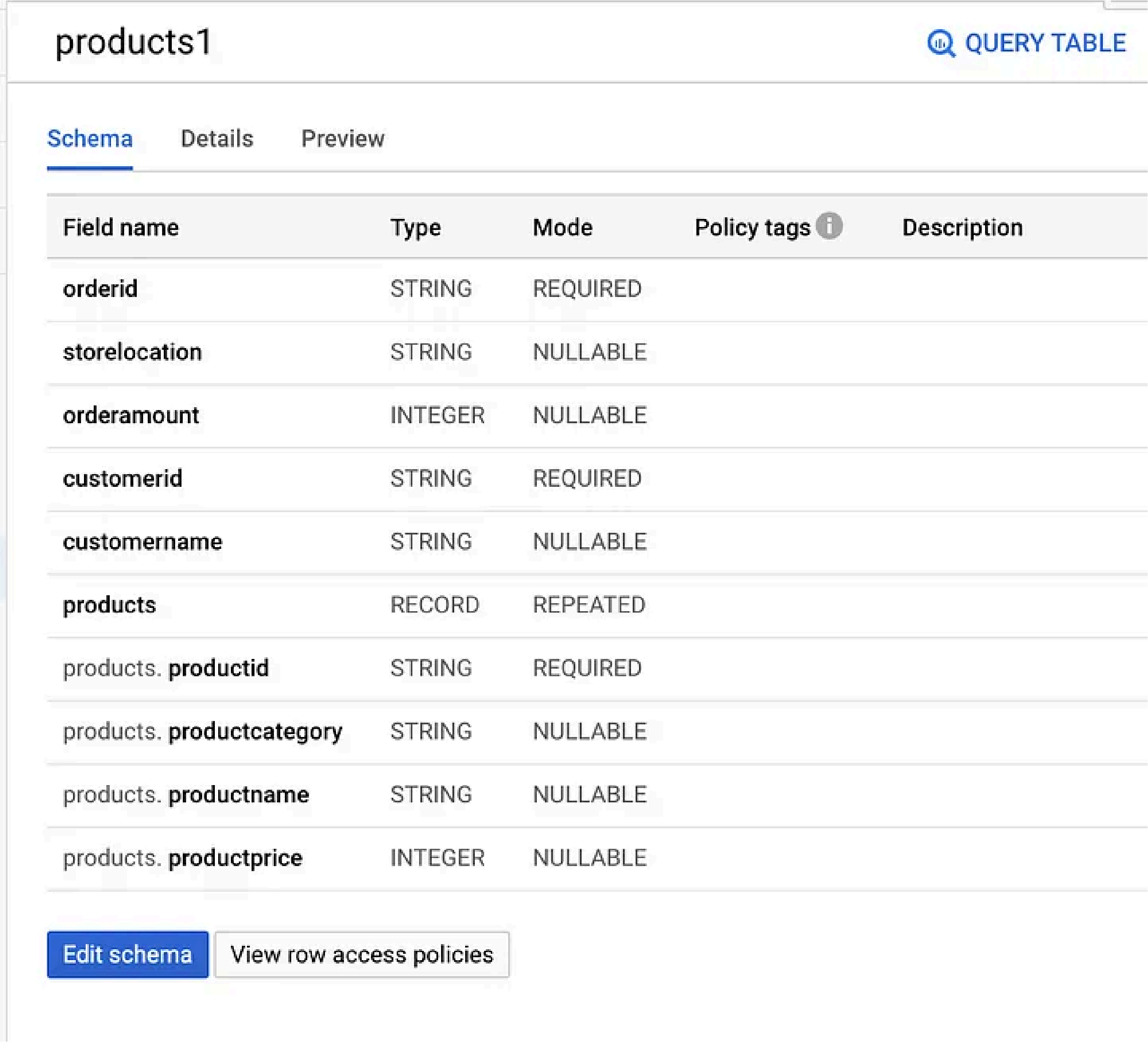
Query complete (0.2 sec elapsed, 337 B processed)

It took only 0.2 seconds to return the results with Clustering. For large tables the time factor is significant and helpful. Clustering with Partitioning will give you good performance along with cost benefits.

4. De-normalize the data :

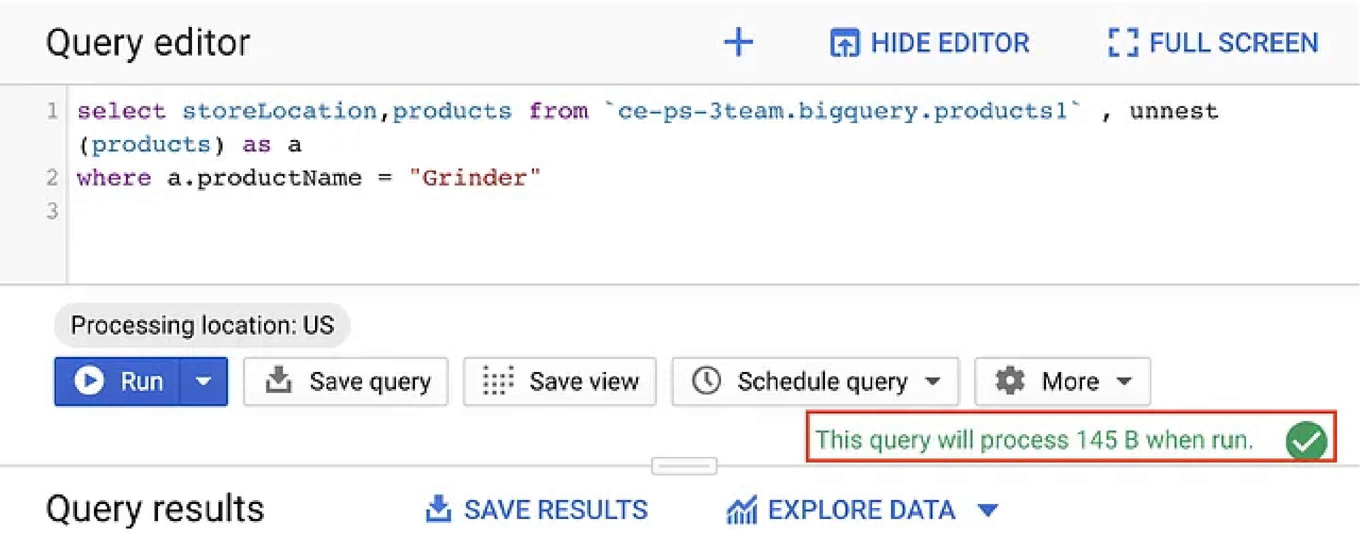
De-normalizing is a strategy of allowing duplicate field values for a column in a table in the data to gain processing performance. Bigquery performs best when the data is denormalized. Rather than preserving a relational schema, you should try to denormalize the data and take advantage of nested and repeated fields. Nested and repeated columns can maintain relationships without impacting the performance by preserving a normalized schema.

Table schema we are using here :



Field name	Type	Mode	Policy tags	Description
orderid	STRING	REQUIRED		
storelocation	STRING	NULLABLE		
orderamount	INTEGER	NULLABLE		
customerid	STRING	REQUIRED		
customername	STRING	NULLABLE		
products	RECORD	REPEATED		
products. productid	STRING	REQUIRED		
products. productcategory	STRING	NULLABLE		
products. productname	STRING	NULLABLE		
products. productprice	INTEGER	NULLABLE		

Table schema

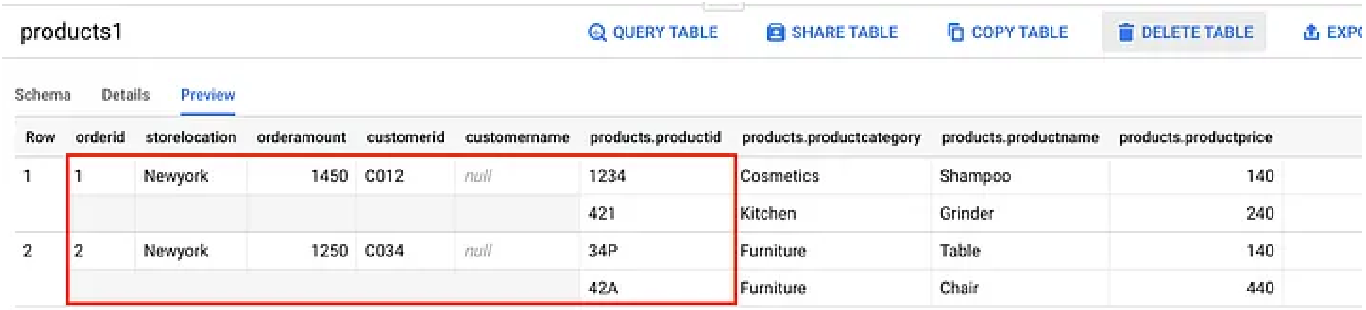


```
select storeLocation,products from `ce-ps-3team.bigquery.products1` , unnest
(products) as a
where a.productName = "Grinder"
```

Processing location: US

This query will process 145 B when run.

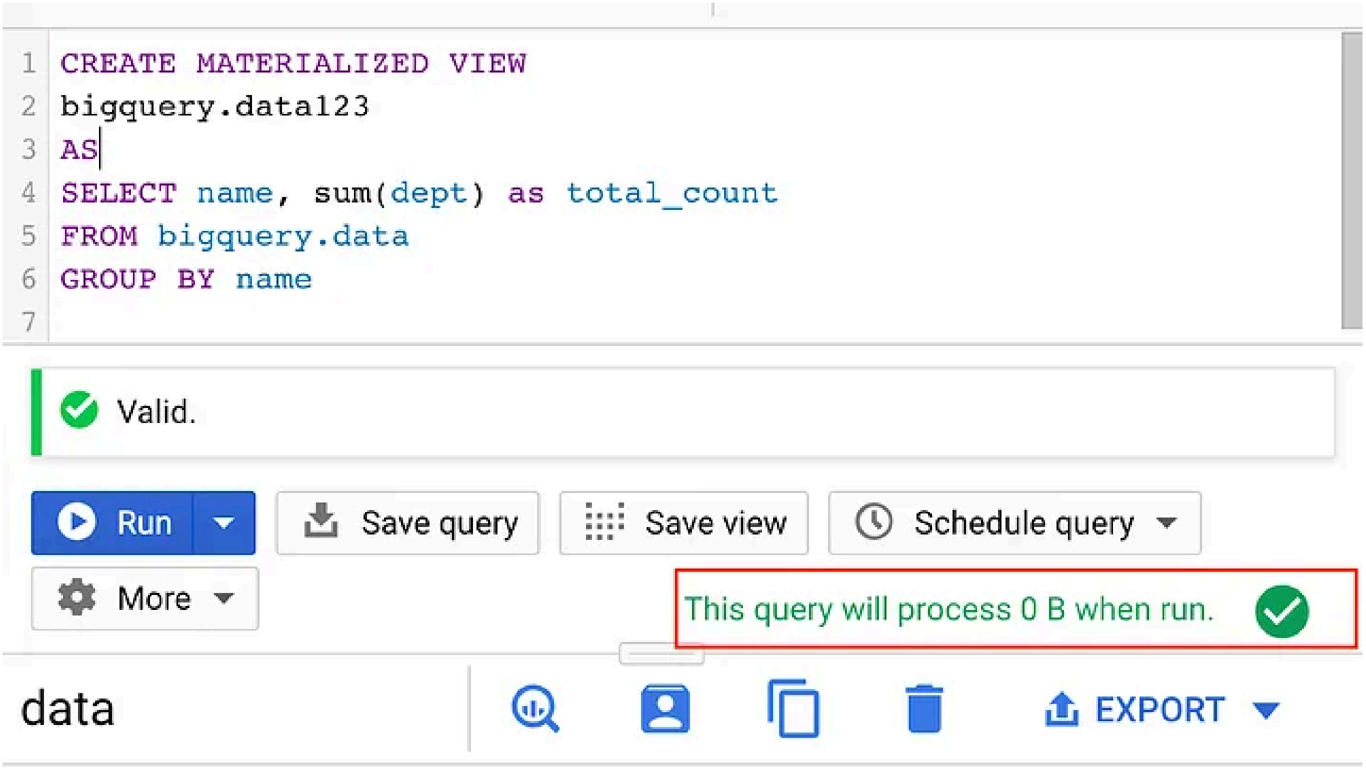
Structure of the table for denormalised data:



Row	orderid	storelocation	orderamount	customerid	customername	products.productid	products.productcategory	products.productname	products.productprice
1	1	Newyork	1450	C012	null	1234	Cosmetics	Shampoo	140
						421	Kitchen	Grinder	240
2	2	Newyork	1250	C034	null	34P	Furniture	Table	140
						42A	Furniture	Chair	440

5. Materialised Views :

Take advantage of materialised views wherever you can. Queries that use materialised views are generally faster and consume fewer resources than queries that retrieve the same data only from the base tables. Materialised views can significantly improve the performance of workloads that have the characteristic of common and repeated queries.



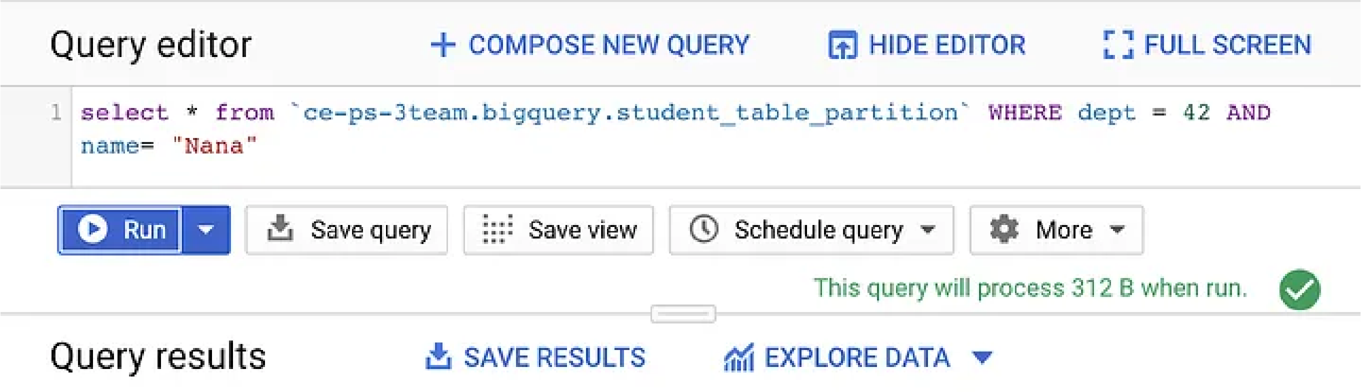
```
CREATE MATERIALIZED VIEW
bigquery.data123
AS
SELECT name, sum(dept) as total_count
FROM bigquery.data
GROUP BY name

```

Valid.

This query will process 0 B when run.

6. Cache the query results :

Caching query results can reduce the load on your BigQuery and boost your performance. Keep the caching results to ON if your team is repeatedly firing the same set of queries again and again.There is no need to limit the data scan during this approach as the cached results are totally free. You also won't be charged for the results retrieved from the cached tables. By default, cache preference is turned on for 24 hours in BigQuery but you can customize it depending on your use case.



```
select * from `ce-ps-3team.bigquery.student_table_partition` WHERE dept = 42 AND
name= "Nana"
```

This query will process 312 B when run.

To estimate the costs before running a query you can use the following methods.

- Query validator in the Cloud Console
- `— dry_run` flag in the `bq `command-line tool

8. Avoid external data sources if performance is a priority :

Use external data sources appropriately. Querying data which is present in Bigquery managed storage itself is typically much faster than querying the data which sits externally like cloud storage, BigTable and all. If query performance is a top priority for you then try to avoid external data sources as much as possible. Usually we use external data sources for use cases like

Performing an ETL load, storing frequently changing data and Periodic loads. For any other types of workloads, try to ingest the data in Bigquery.

9. Use 'ORDER BY' wisely :

Order by clause is also a costly operation, it requires sorting at the whole data level so you need to use it very carefully. Use "**ORDER BY**" only in the outermost query or within the window clause as that would render the final data on which ordering is to be performed to be filtered and reduced. Hence, you would be sorting on a subset of data and not the unnecessary data that is already filtered. Try placing the order by, regular expressions or any other complex expressions at the end of the query for the best performance and cost savings. This results in the query to perform better.Use "**LIMIT**" whenever you are using an order by clause as order by sorts the entire dataset. Hence, it must be done on a single slot and if you are attempting to order a very large result set, the final sorting can overwhelm the slot that is processing the data which may result in a **"Resources exceeded"** error and such errors are returned when your query uses too many resources. If you are using an ORDER BY clause, it is recommended to use a LIMIT clause along with it for optimized performance and cost savings.

Conclusion :

BigQuery can run analytical queries at lightning-fast speeds, but as the data warehouse grows in size, the costs can increase significantly. The techniques used above can result in huge cost savings for recurring queries run each time by the team. The benefits of a highly scalable data architecture can be obtained without spending millions of dollars.

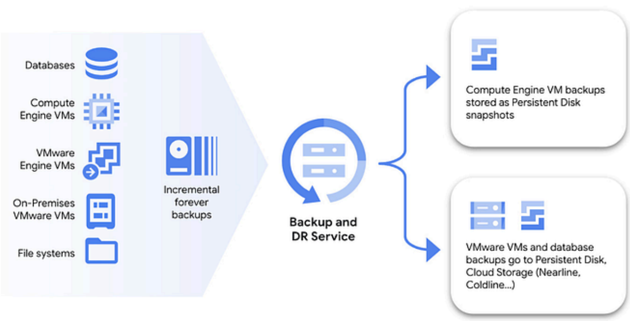
Bigquery Gcp Cost Analysis Cost Optimization Partitioning

Written by Thanusha Deepthi

Follow

23 Followers · Writer for Searce

More from Thanusha Deepthi and Searce

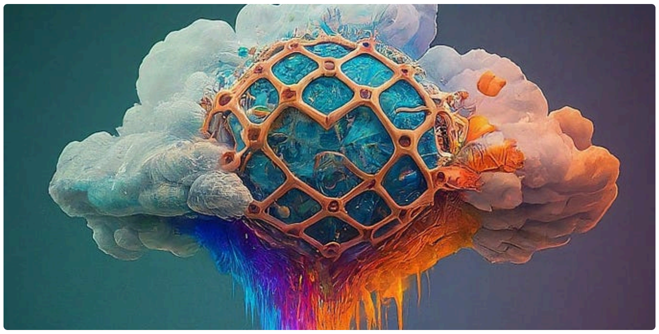

Databases
Compute Engine VMs
VMware Engine VMs
On-Premises VMware VMs
File systems
Incremental forever backups
Backup and DR Service
Compute Engine VM backups stored as Persistent Disk snapshots
VMware VMs and database backups go to Persistent Disk, Cloud Storage (Nearline, Coldline...)

Thanusha Deepthi

Backup & DR

What is Backup and Disaster Recovery:

Apr 21, 2023 52

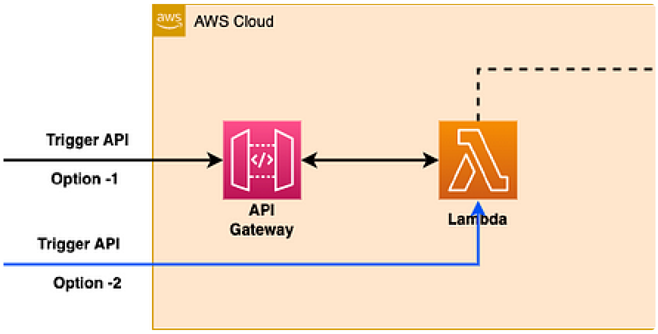


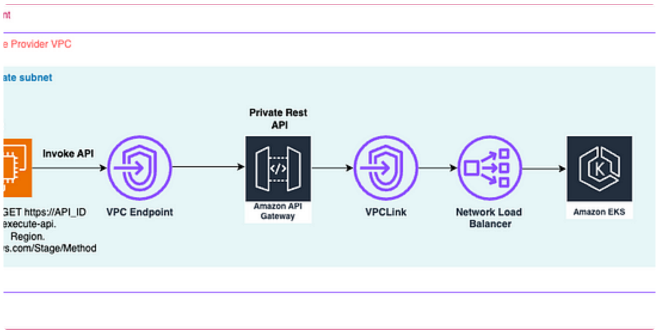
Udesh Udayakumar in Searce

Why Google Kubernetes Engine (GKE) Leads the Pack: A...

This article discusses potential issues companies may encounter when using...

Aug 10 148





Prasad Midde in Searce

FastAPI App Deployment Using AWS Lambda And API Gateway

FastAPI is a modern fast (high-performance) web framework for building APIs with Python.

Jun 5, 2023 166 6

Prasad Midde in Searce

A Step-by-Step Guide for Private API Gateway and EKS Integration

API Gateway private endpoints allows us for building private API–based services inside...

Jun 28 8 1

See all from Thanusha Deepthi

See all from Searce

Recommended from Medium

Hugo Lu (Orchestra) 9:04 AM
Does anyone have incremental models that are
bigquery? dbt cite this as a limitation on th
when I compile my code the code d
(calling an unknown "string_func"
in the meantime, but curious to
2 replies Last reply today

Hugo Lu (Orchestra) 9:06 AM
Like I am pretty sure you can pa
2
6 replies Last r

vogoybrand.info 10:43 AM
Can we load multiple files with
a single table model in dbt
1 reply 23 hours ago

soliev 11:48 AM
Hello, I would like to add description, tags to
doing it with `dbt_project.yml` , but i think i'm doing
1 reply 22 hours ago

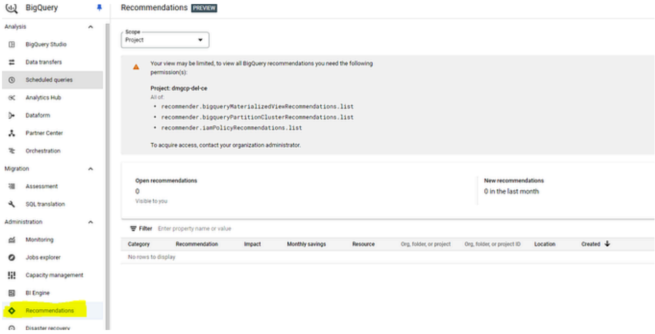
slothkong 1 day ago
I think you mean to say cluster by whatever you
want, as opposed to partition by, or?

1 day ago
your string column is unique, you can
`_by='column_name'` within a `config`
r in the sql or the yaml file

ago
@Hugo Lu (Orchestra) I just
ed this recently. Clustering also has also
formance boost. clustering also allows
multiple columns

rchestra) 22 hours ago
I mean partitioning

(Orchestra) 22 hours ago
s it would be nice if dbt threw an error
because you can run something from dbt where
the config specifies to partition on string. Given
I've tried that and got a regular un-partitioned
table, I guess the solution is indeed to partition by



Recommendations PREVIEW

Scope: Project

Your view may be limited, to view all BigQuery recommendations you need the following permission(s):

Project: dmgcp-del-ce

All of:

- `recommender.bigqueryMaterializedViewRecommendations.list`
- `recommender.bigqueryPartitionClusterRecommendations.list`
- `recommender.iamPolicyRecommendations.list`

To acquire access, contact your organization administrator.

Open recommendations
0
Visible to you

New recommendations
0 in the last month

Filter Enter property name or value

Category	Recommendation	Impact	Monthly savings	Resource	Org, folder, or project	Org, folder, or project ID	Location	Created ↓
No rows to display								

Hugo Lu

BigQuery Table Partitioning and clustering Tips with dbt

Partitioning and clustering in BigQuery using different Column Types

Apr 4 95

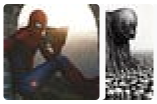
Praveen Bhushan

Optimize Cost Using BigQuery Recommenders

BigQuery works with Active Assist and provides recommendations to optimize your…

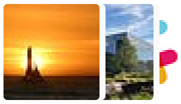
Aug 20 5

Lists



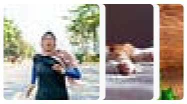
Staff Picks

727 stories · 1275 saves



Stories to Help You Level-Up at Work

19 stories · 780 saves

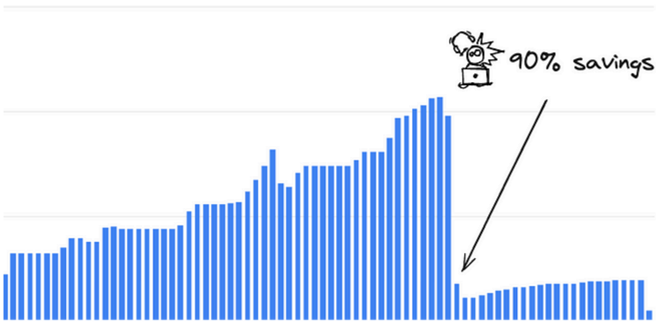


Self-Improvement 101

20 stories · 2677 saves

Productivity 101

20 stories · 2297 saves



Yerachmiel Feltzman in Israeli Tech Radar

How to save 90% on BigQuery storage

Hey man, stop with the click-bait titles—I can listen to you thinking. ;)

May 12 1K 2

Xiaoxu Gao in Towards Data Science

FinOps: Four Ways to Reduce Your BigQuery Storage Cost

Don't overlook the cloud storage cost

Jan 30, 2023 678 8

Akshay Bagal

Mastering Google BigQuery with Python: A Comprehensive Guide t…

Introduction: In today's data-centric era, the ability to harness vast amounts of data…

Apr 3 👏 6

Tom Ellyatt

What is Farm_Fingerprint in BigQuery, and Why Do I Love It?

Joins are one of the most resource-intensive operations in BigQuery, especially when…

✦ Jul 31 👏 30 💬 3

See more recommendations