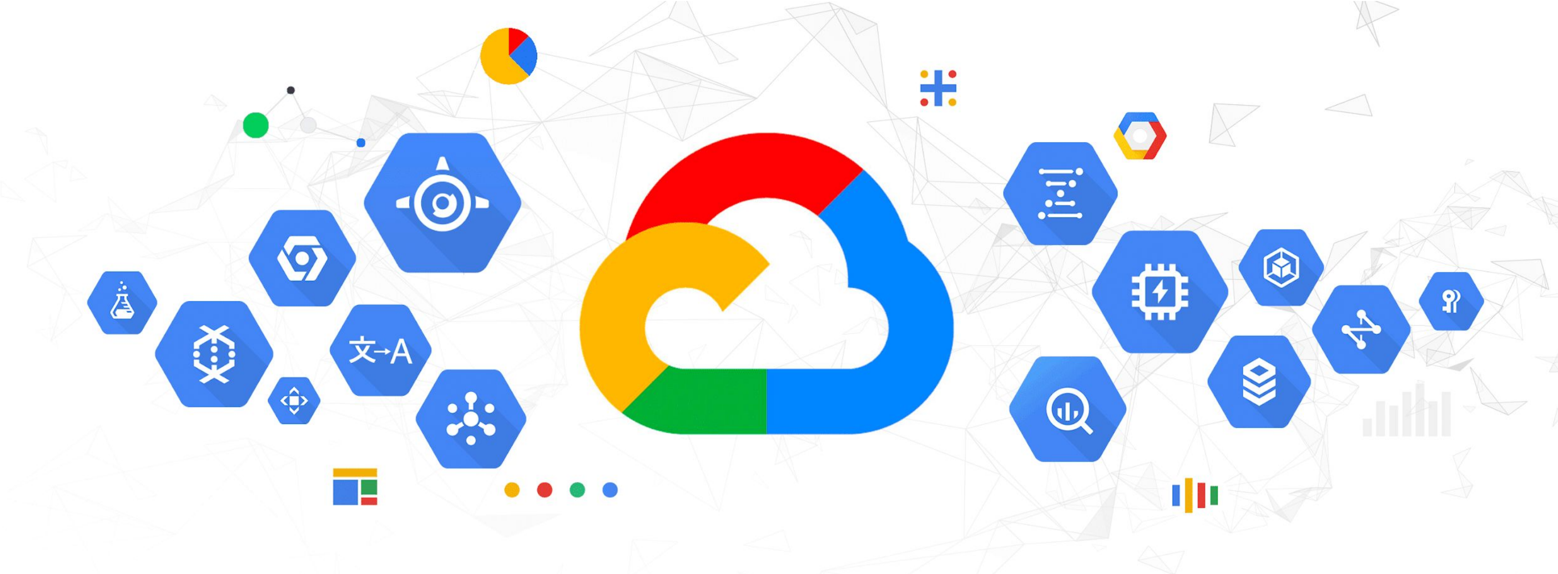
BigQuery

Optimization & Cost Efficiency



Highlight Content

1. BigQuery
2. Pricing Schema
3. Storage Cost optimization
4. Query Cost Optimization

01

Introduction

What is BigQuery?
Schema Pricing?

BigQuery ?

Cloud Based Analytical Database

Fully Managed Service

Highly Scalable

Cost Effective

SQL Compatible

Fast

Pricing Schema

STORAGE
Active Storage
Long Term Storage
Streaming Insert

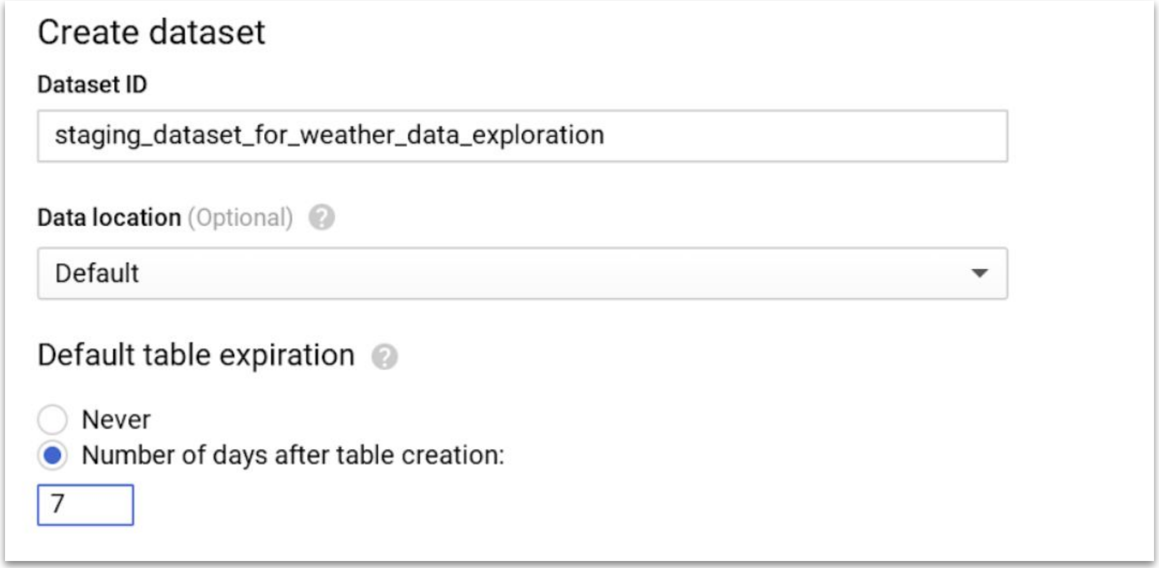
QUERY PROCESSING
On Demand
Flat Rate

02

Storage Cost Optimization

Set Expired Date For Temporary Data

1. Dataset Level



Create dataset

Dataset ID

staging_dataset_for_weather_data_exploration

Data location (Optional)

Default

Default table expiration

○ Never

● Number of days after table creation:

7

Set Expired Date For Temporary Data

2. Table Level

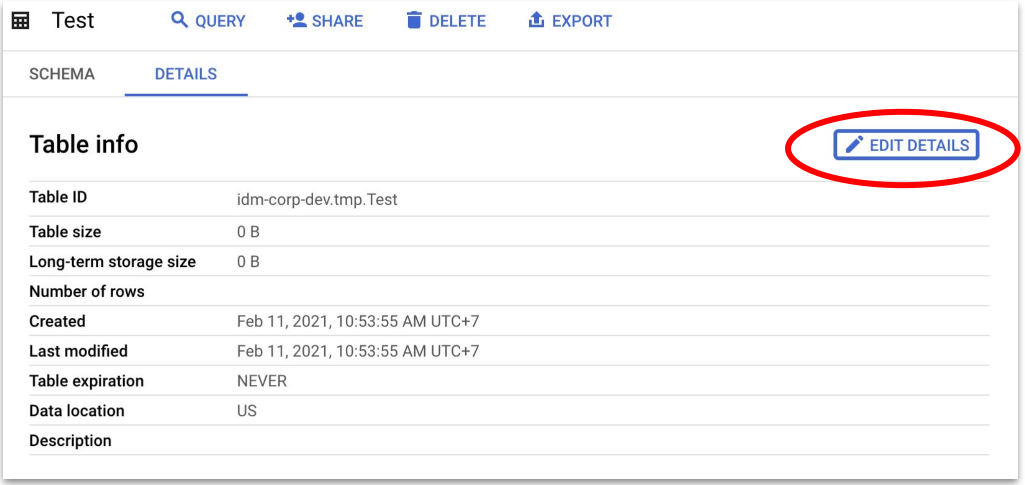
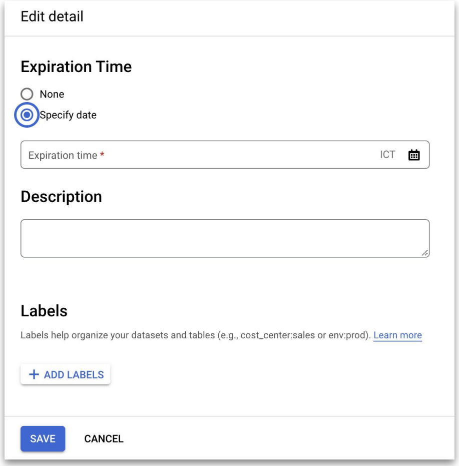


Table info

Table ID	idm-corp-dev.tmp.Test
Table size	0 B
Long-term storage size	0 B
Number of rows	
Created	Feb 11, 2021, 10:53:55 AM UTC+7
Last modified	Feb 11, 2021, 10:53:55 AM UTC+7
Table expiration	NEVER
Data location	US
Description	



Expiration Time

○ None

● Specify date

Expiration time * ICT

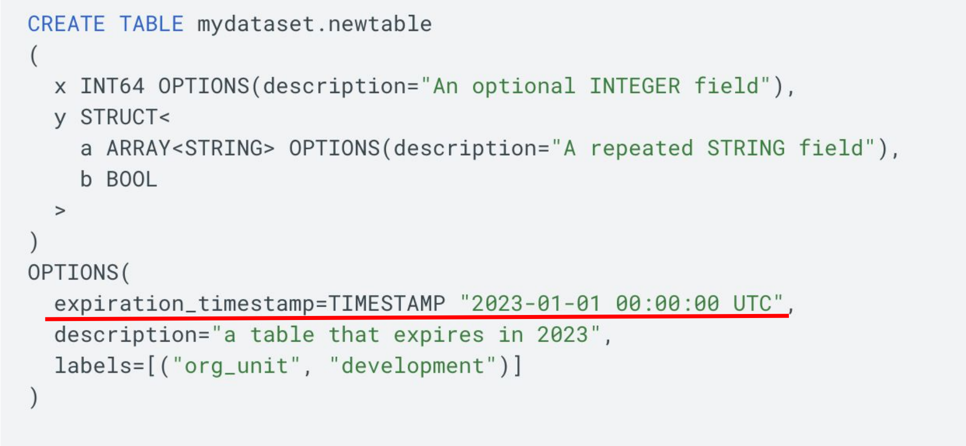
Description

Labels

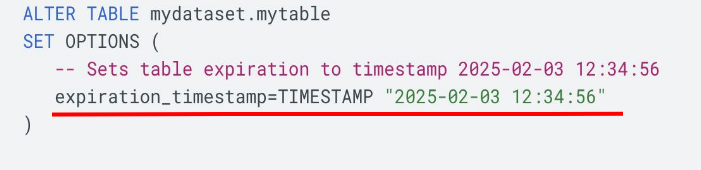
Labels help organize your datasets and tables (e.g., cost_center:sales or env:prod). Learn more

Set Expired Date For Temporary Data

2. Table Level



```
CREATE TABLE mydataset.newtable
(
  x INT64 OPTIONS(description="An optional INTEGER field"),
  y STRUCT<
    a ARRAY<STRING> OPTIONS(description="A repeated STRING field"),
    b BOOL
  >
)
OPTIONS(
  expiration_timestamp=TIMESTAMP "2023-01-01 00:00:00 UTC",
  description="a table that expires in 2023",
  labels=[("org_unit", "development")]
)
```



```
ALTER TABLE mydataset.mytable
SET OPTIONS (
  -- Sets table expiration to timestamp 2025-02-03 12:34:56
  expiration_timestamp=TIMESTAMP "2025-02-03 12:34:56"
)
```

Be Mindful of Editing Table Data

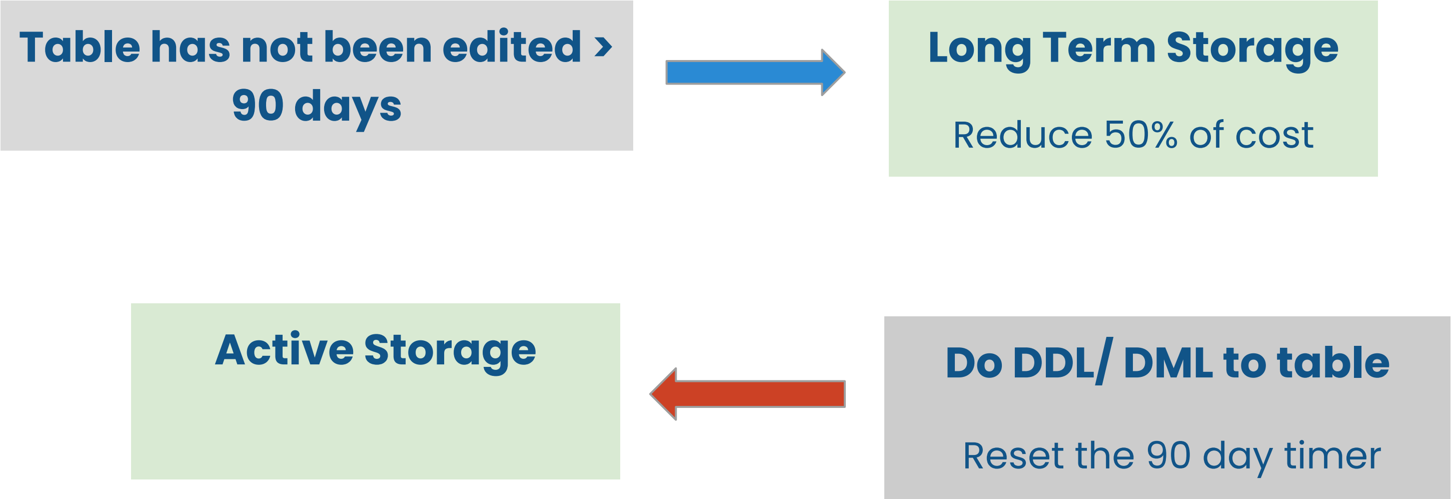


Table has not been edited > 90 days

Long Term Storage

Reduce 50% of cost

Active Storage

Do DDL/ DML to table

Reset the 90 day timer

Avoid Duplicate Copies of Data

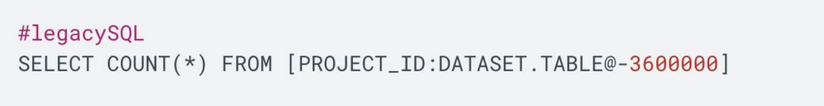
Query directly from external source*

- ❏ External Table
- ❏ Federated Queries (External Query)

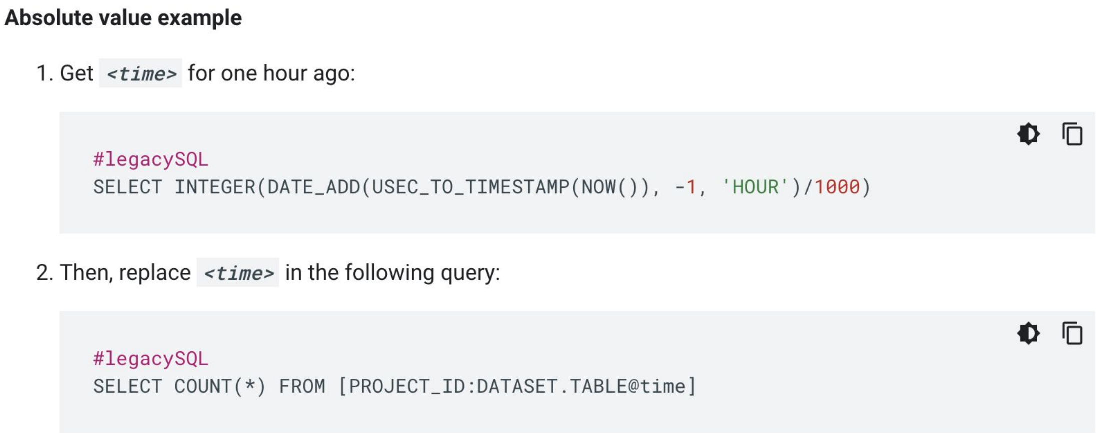
**query don’t perform as well compared to query executed on same data stored on BigQuery,*

Understand BQ’s Backup & DR Process

- BigQuery maintains a seven-day history of changes
- Can query a point-in-time snapshot of table



```
#legacySQL
SELECT COUNT(*) FROM [PROJECT_ID:DATASET.TABLE@-3600000]
```



Absolute value example

1. Get `<time>` for one hour ago:

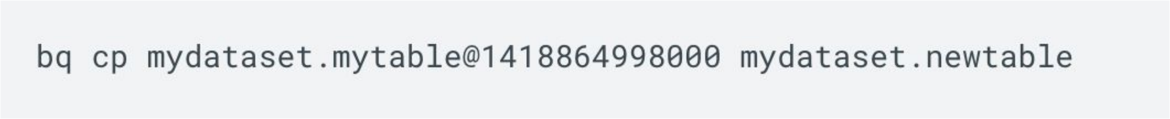
```
#legacySQL
SELECT INTEGER(DATE_ADD(USEC_TO_TIMESTAMP(NOW()), -1, 'HOUR')/1000)
```

2. Then, replace `<time>` in the following query:

```
#legacySQL
SELECT COUNT(*) FROM [PROJECT_ID:DATASET.TABLE@time]
```

Understand BQ’s Backup & DR Process

- Can Restore deleted tables within 7 days of deletion

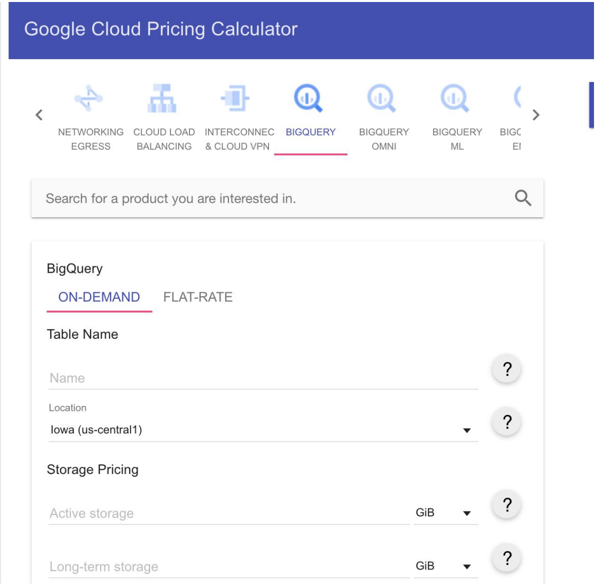


```
bq cp mydataset.mytable@1418864998000 mydataset.newtable
```

**using epoch time/ unix time (in milliseconds)*

Estimate storage cost
Google Cloud Pricing Calculator

- https://cloud.google.com/products/calculator



Google Cloud Pricing Calculator

Search for a product you are interested in.

BigQuery

ON-DEMAND FLAT-RATE

Table Name

Name ?

Location
Iowa (us-central1) ?

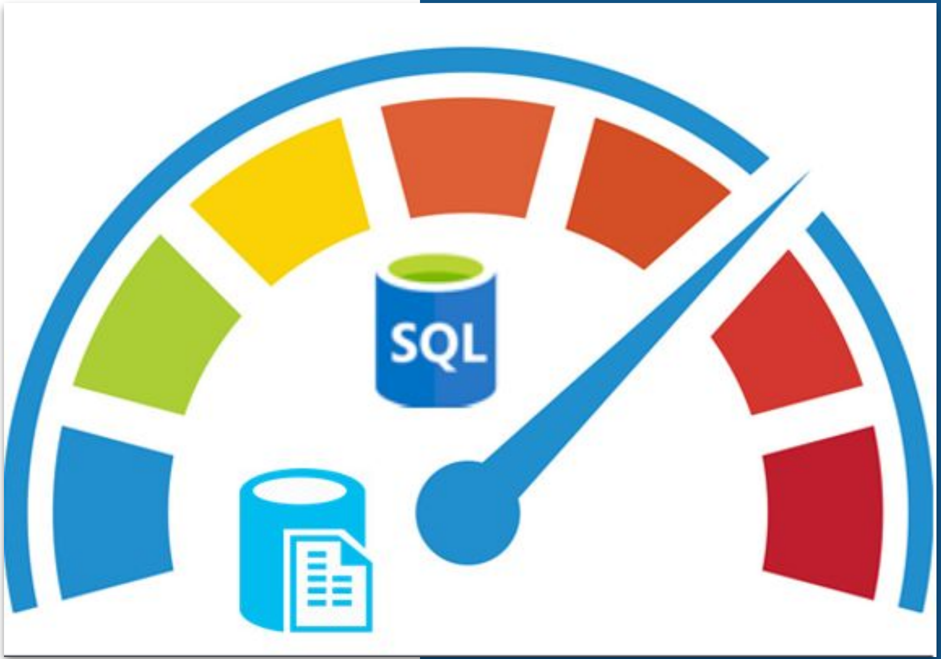
Storage Pricing

Active storage GiB ?

Long-term storage GiB ?

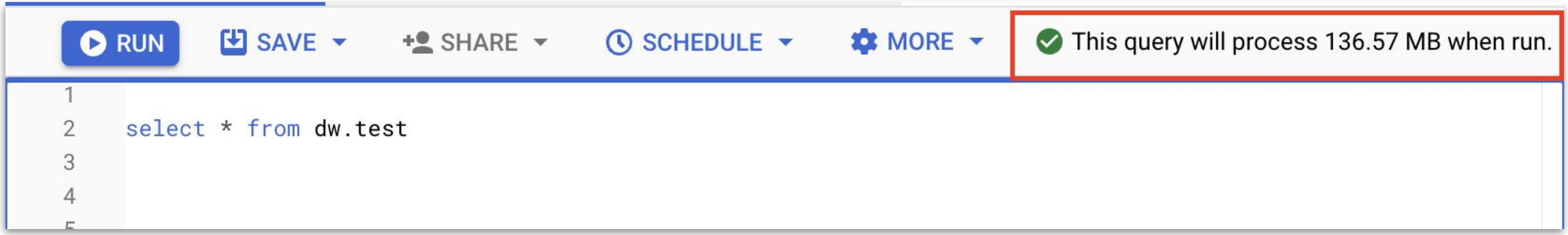
03

Query Optimization



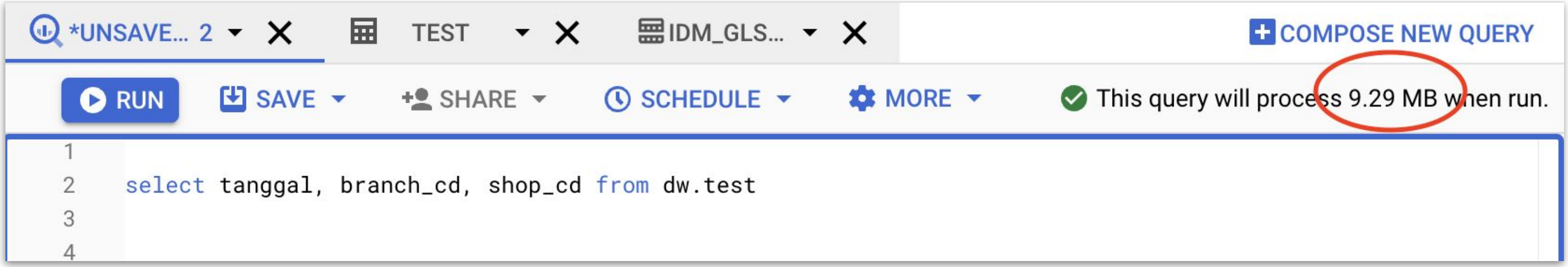
Limit bytes to scan

- Select necessary columns only



This query will process 136.57 MB when run.

```
select * from dw.test
```

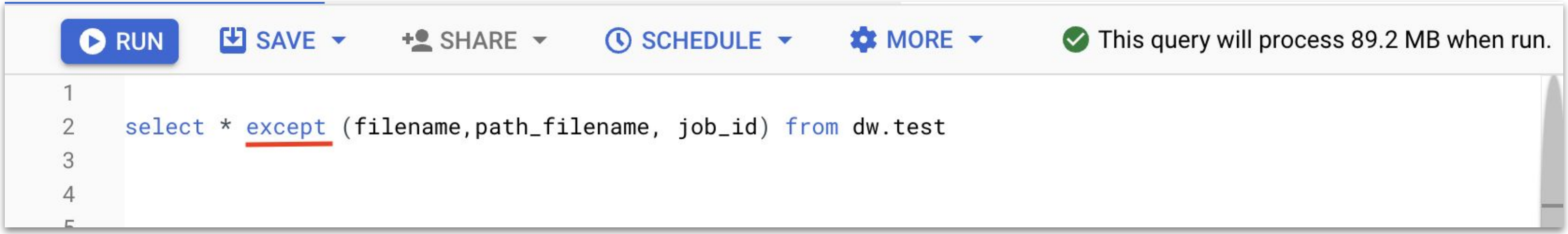


This query will process 9.29 MB when run.

```
select tanggal, branch_cd, shop_cd from dw.test
```

Limit bytes to scan

- Use except



This query will process 89.2 MB when run.

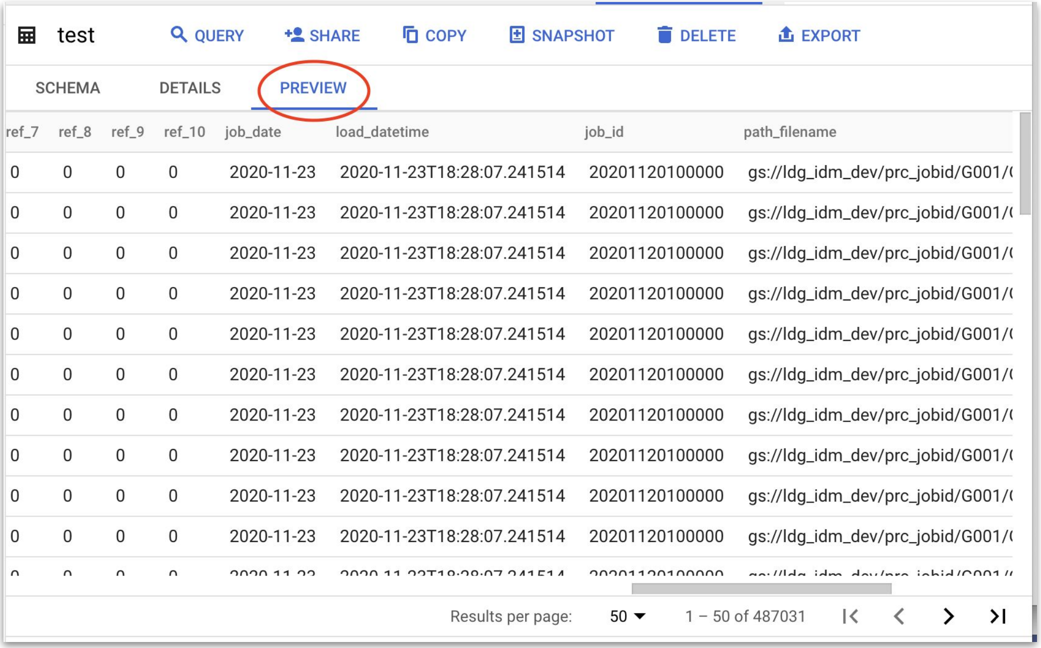
```

select * except (filename,path_filename, job_id) from dw.test


```

Limit bytes to scan

- If need to explore all columns, use Preview table

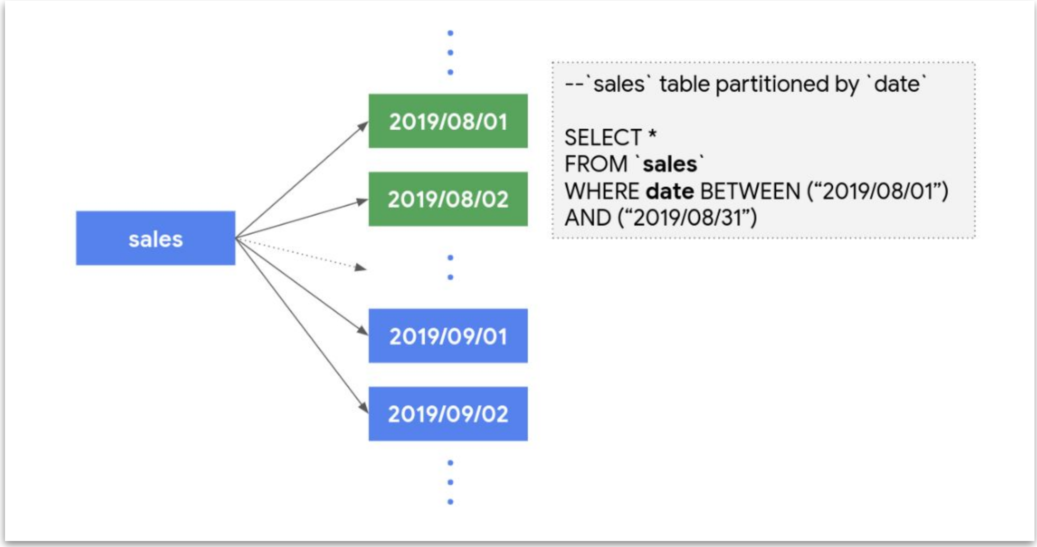


ref_7	ref_8	ref_9	ref_10	job_date	load_datetime	job_id	path_filename
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/
0	0	0	0	2020-11-23	2020-11-23T18:28:07.241514	20201120100000	gs://ldg_idm_dev/prc_jobid/G001/

Results per page: 50 ▾ 1 – 50 of 487031 |< < > >|

Limit bytes to scan

❑ Create Partition table (whenever possible)



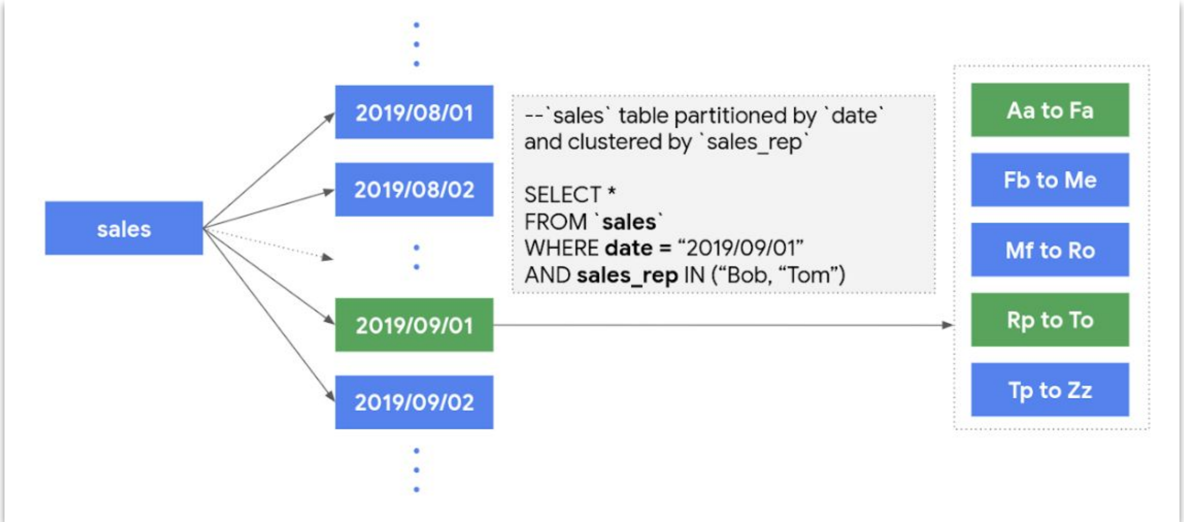
```
--`sales` table partitioned by `date`

SELECT *
FROM `sales`
WHERE date BETWEEN (“2019/08/01”)
AND (“2019/08/31”)
```

- ➔ only pay for related partitions
- ➔ each partition is separately considered for long-term storage
- ➔ require_partition_filter

Limit bytes to scan

❑ Clustering



```
--`sales` table partitioned by `date`
and clustered by `sales_rep`

SELECT *
FROM `sales`
WHERE date = “2019/09/01”
AND sales_rep IN (“Bob, “Tom”)
```

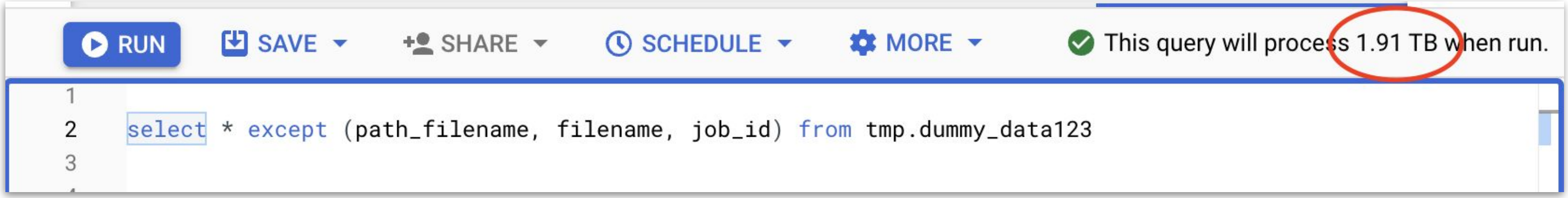
➔ only scan/pay for relevant blocks

➔ Cluster is only exist in partition table

➔ Use fake partition, if need cluster on unpartitioned table

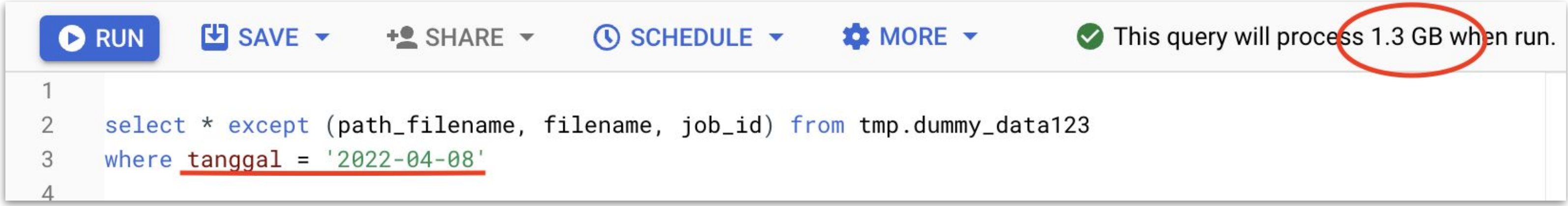
Limit bytes to scan

❑ Filter data with Partitions and Clusters



This query will process 1.91 TB when run.

```
select * except (path_filename, filename, job_id) from tmp.dummy_data123
```

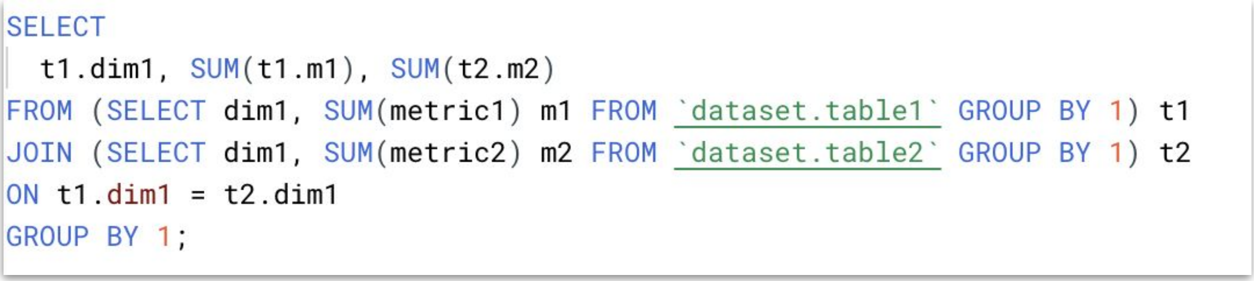


This query will process 1.3 GB when run.

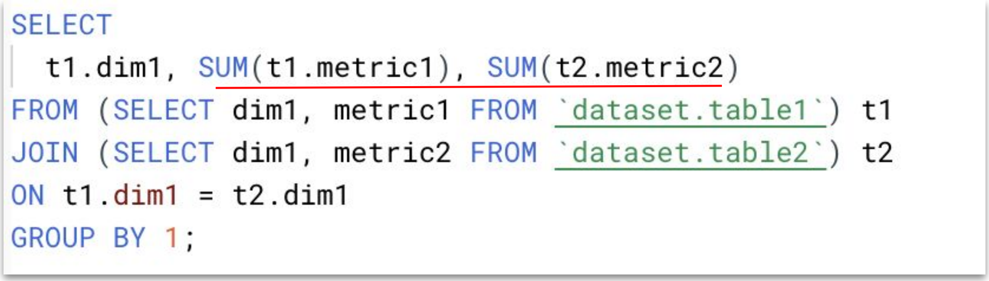
```
select * except (path_filename, filename, job_id) from tmp.dummy_data123
where tanggal = '2022-04-08'
```

Aggregation

- Late Aggregation



```
SELECT
  t1.dim1, SUM(t1.m1), SUM(t2.m2)
FROM (SELECT dim1, SUM(metric1) m1 FROM `dataset.table1` GROUP BY 1) t1
JOIN (SELECT dim1, SUM(metric2) m2 FROM `dataset.table2` GROUP BY 1) t2
ON t1.dim1 = t2.dim1
GROUP BY 1;
```



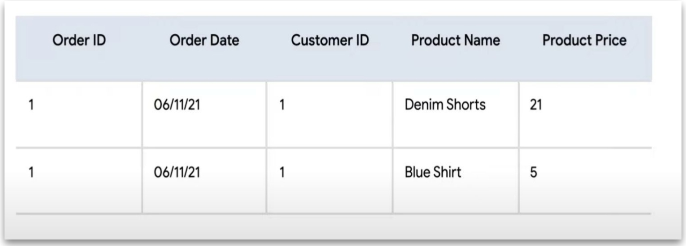
```
SELECT
  t1.dim1, SUM(t1.metric1), SUM(t2.metric2)
FROM (SELECT dim1, metric1 FROM `dataset.table1`) t1
JOIN (SELECT dim1, metric2 FROM `dataset.table2`) t2
ON t1.dim1 = t2.dim1
GROUP BY 1;
```

optimized

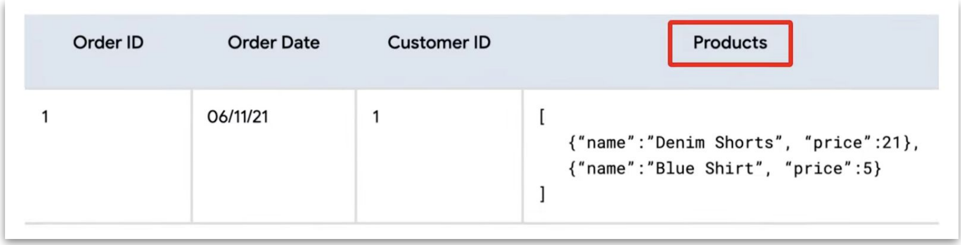
**__The exception__ is if a table can be reduced drastically by aggregation in preparation for a join*

Aggregation

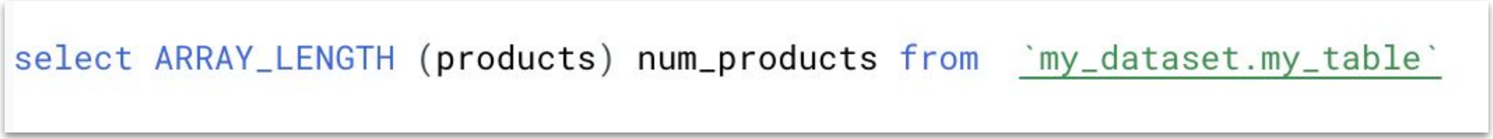
- Nest Repeated Data



Order ID	Order Date	Customer ID	Product Name	Product Price
1	06/11/21	1	Denim Shorts	21
1	06/11/21	1	Blue Shirt	5



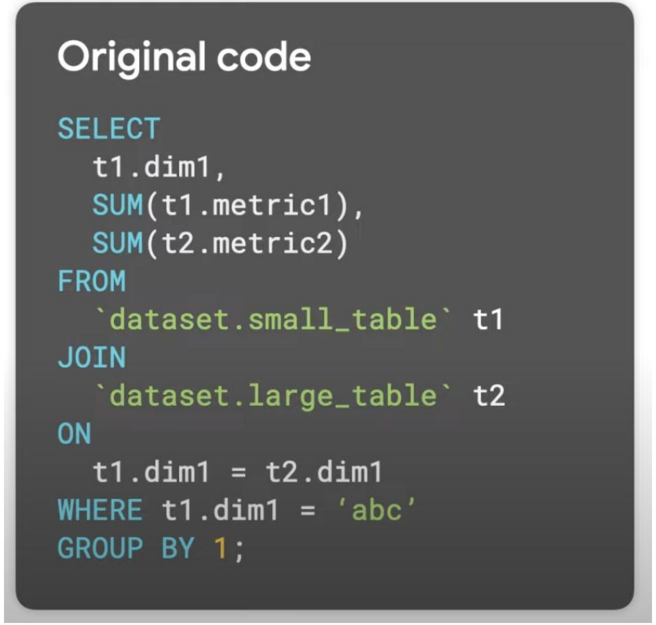
Order ID	Order Date	Customer ID	Products
1	06/11/21	1	[{“name”:”Denim Shorts”, “price”:21}, {“name”:”Blue Shirt”, “price”:5}]



```
select ARRAY_LENGTH (products) num_products from  `my_dataset.my_table`
```

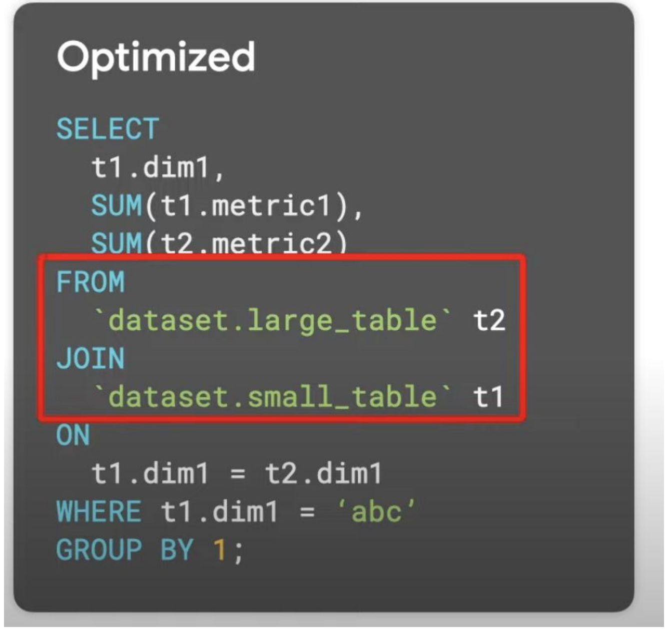
Joins

❑ Largest Table First



Original code

```
SELECT
  t1.dim1,
  SUM(t1.metric1),
  SUM(t2.metric2)
FROM
  `dataset.small_table` t1
JOIN
  `dataset.large_table` t2
ON
  t1.dim1 = t2.dim1
WHERE t1.dim1 = 'abc'
GROUP BY 1;
```

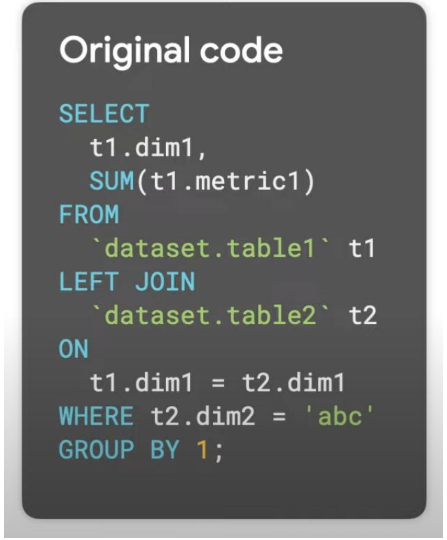


Optimized

```
SELECT
  t1.dim1,
  SUM(t1.metric1),
  SUM(t2.metric2)
FROM
  `dataset.large_table` t2
JOIN
  `dataset.small_table` t1
ON
  t1.dim1 = t2.dim1
WHERE t1.dim1 = 'abc'
GROUP BY 1;
```

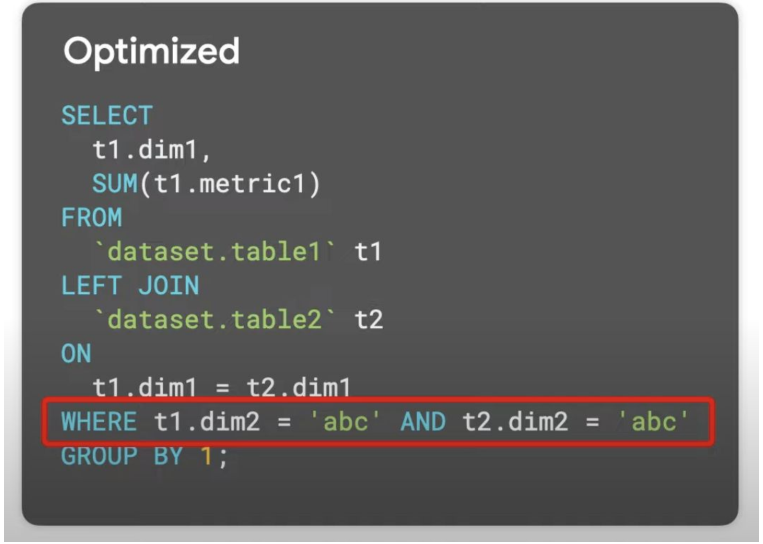
Joins

❑ Filter Before Join



Original code

```
SELECT
  t1.dim1,
  SUM(t1.metric1)
FROM
  `dataset.table1` t1
LEFT JOIN
  `dataset.table2` t2
ON
  t1.dim1 = t2.dim1
WHERE t2.dim2 = 'abc'
GROUP BY 1;
```



Optimized

```
SELECT
  t1.dim1,
  SUM(t1.metric1)
FROM
  `dataset.table1` t1
LEFT JOIN
  `dataset.table2` t2
ON
  t1.dim1 = t2.dim1
WHERE t1.dim2 = 'abc' AND t2.dim2 = 'abc'
GROUP BY 1;
```

**Objective : Tables to be joined are as small as possible*
**May use subquery to filter in advance*

Joins

❏ Clustering on Join Keys

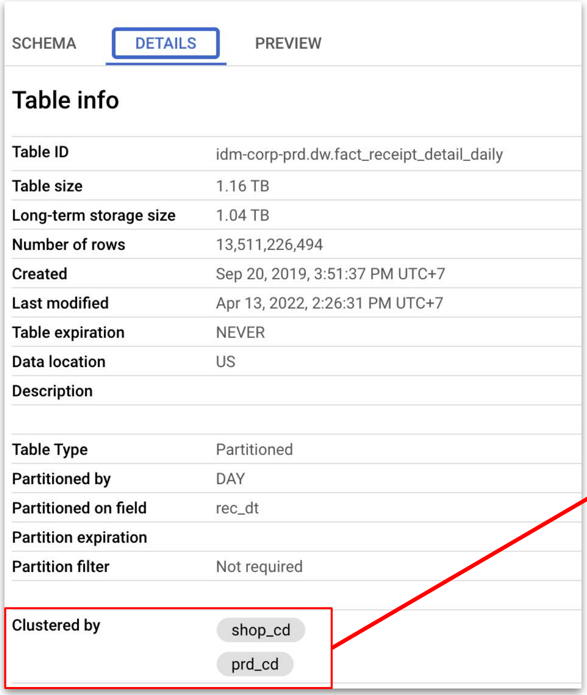


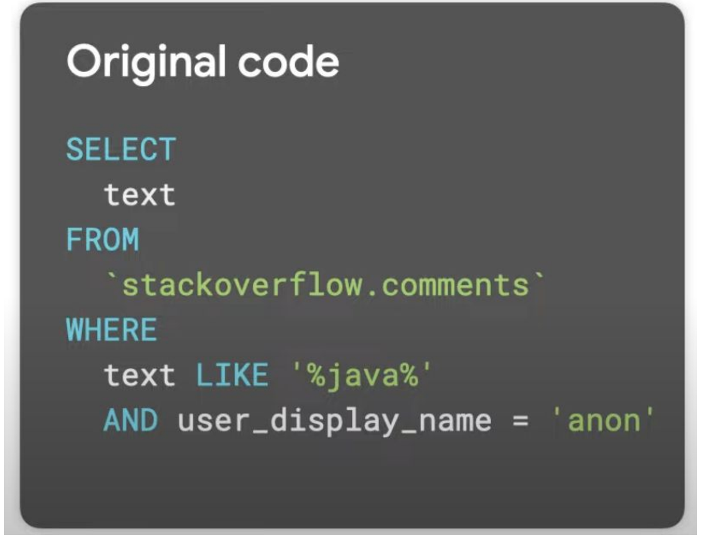
Table info

Table ID	idm-corp-prd.dw.fact_receipt_detail_daily
Table size	1.16 TB
Long-term storage size	1.04 TB
Number of rows	13,511,226,494
Created	Sep 20, 2019, 3:51:37 PM UTC+7
Last modified	Apr 13, 2022, 2:26:31 PM UTC+7
Table expiration	NEVER
Data location	US
Description	
Table Type	Partitioned
Partitioned by	DAY
Partitioned on field	rec_dt
Partition expiration	
Partition filter	Not required
Clustered by	shop_cd prd_cd

Column shop_cd & prd_cd :
key columns that is used to join

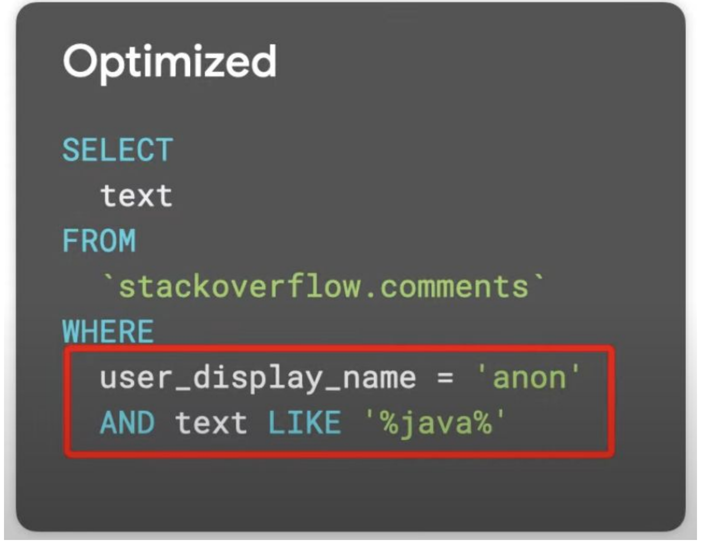
Where Clause

- Where clause order matters



Original code

```
SELECT
  text
FROM
  `stackoverflow.comments`
WHERE
  text LIKE '%java%'
  AND user_display_name = 'anon'
```



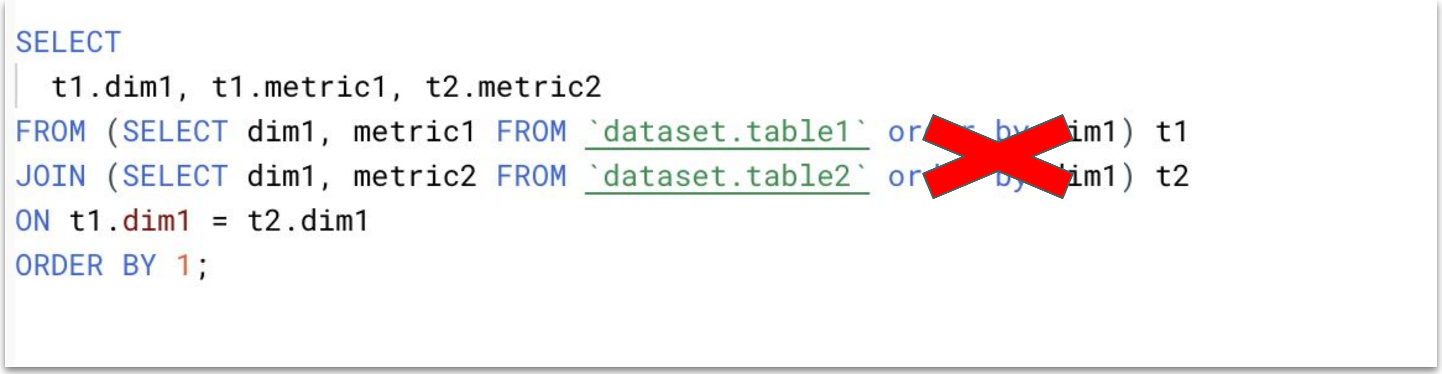
Optimized

```
SELECT
  text
FROM
  `stackoverflow.comments`
WHERE
  user_display_name = 'anon'
  AND text LIKE '%java%'
```

The first filter should eliminated the most data

Order By

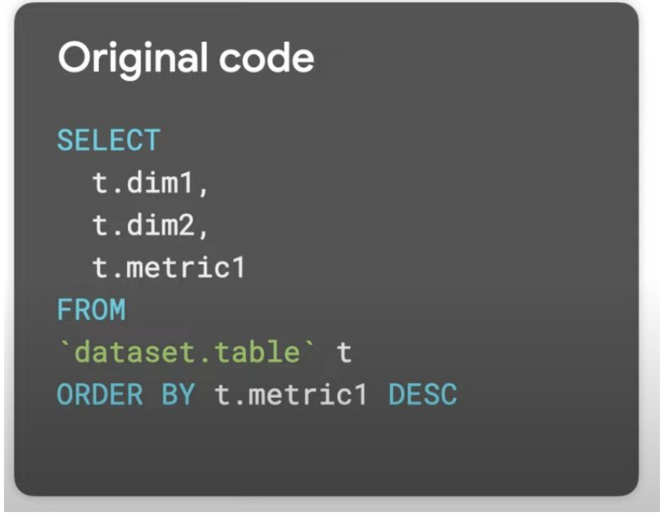
- Don’t put order by in subquery



```
SELECT
  t1.dim1, t1.metric1, t2.metric2
FROM (SELECT dim1, metric1 FROM `dataset.table1` order by dim1) t1
JOIN (SELECT dim1, metric2 FROM `dataset.table2` order by dim1) t2
ON t1.dim1 = t2.dim1
ORDER BY 1;
```

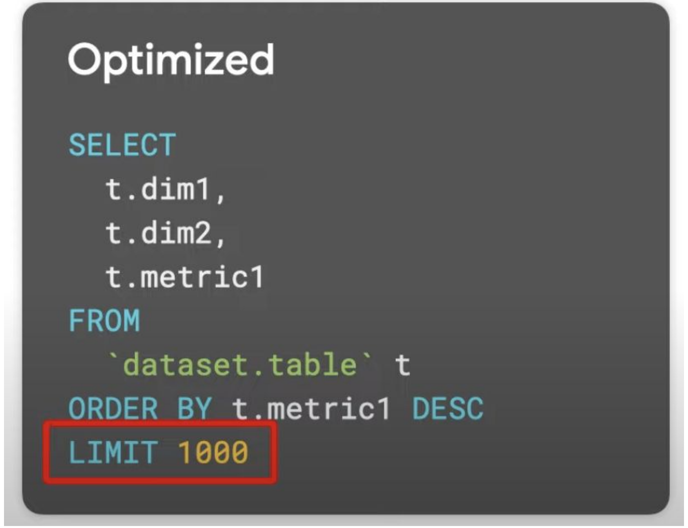
Order By

- ❑ Order by with Limit



Original code

```
SELECT
  t.dim1,
  t.dim2,
  t.metric1
FROM
`dataset.table` t
ORDER BY t.metric1 DESC
```



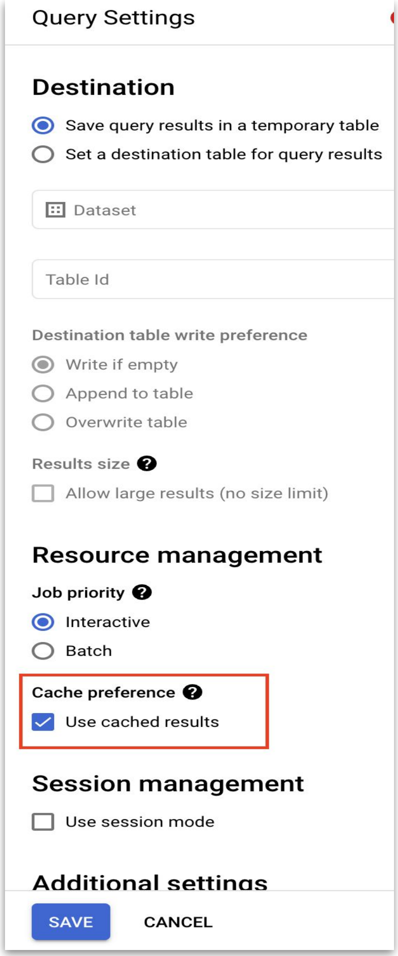
Optimized

```
SELECT
  t.dim1,
  t.dim2,
  t.metric1
FROM
  `dataset.table` t
ORDER BY t.metric1 DESC
LIMIT 1000
```

Other

❏ Use Cached Result option

- Caching is per user per project
- Results are cache for approximately 24 hours
- If cached results are return, it won’t be billed for any usage



Query Settings

Destination

● Save query results in a temporary table
○ Set a destination table for query results

Dataset

Table Id

Destination table write preference

● Write if empty
○ Append to table
○ Overwrite table

Results size

□ Allow large results (no size limit)

Resource management

Job priority

● Interactive
○ Batch

Cache preference

☑ Use cached results

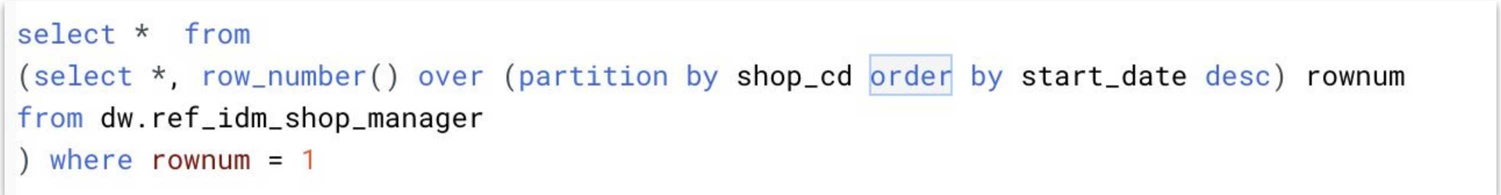
Session management

□ Use session mode

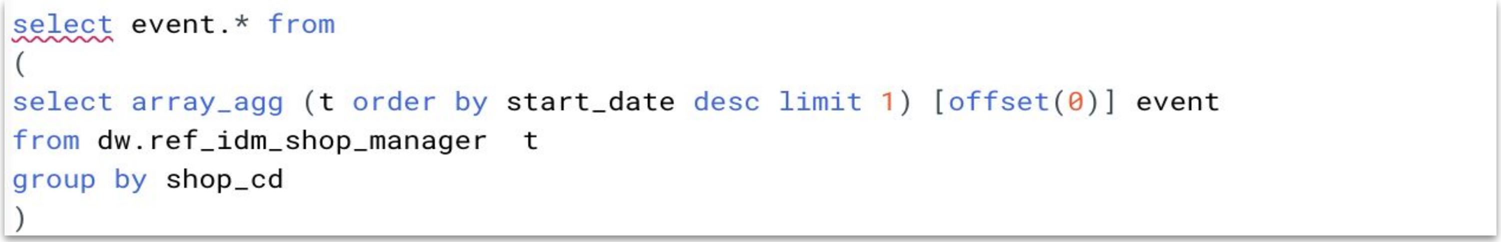
Additional settings

Other

❑ To get First/ Last Record use ARRAY_AGG() instead of ROW_NUMBER()



```
select *  from
(select *, row_number() over (partition by shop_cd order by start_date desc) rownum
from dw.ref_idm_shop_manager
) where rownum = 1
```



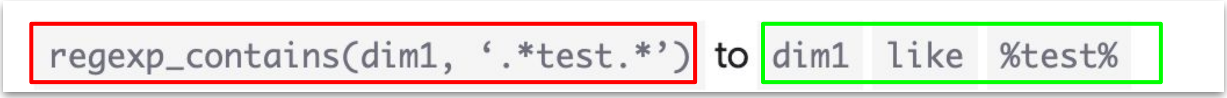
```
select event.* from
(
select array_agg (t order by start_date desc limit 1) [offset(0)] event
from dw.ref_idm_shop_manager  t
group by shop_cd
)
```

**Solution for “Resources exceeded” error*
**Array_agg() might perform a little slower*

Other

- REGEXP_CONTAINS is slower than LIKE

Use LIKE when the full power of regex is not needed (e.g. wildcard matching)



```
regexp_contains(dim1, ‘.*test.*’) to dim1 like %test%
```

THANK YOU!

Reference Links

❑ https://cloud.google.com/blog/products/data-analytics/cost-optimization-best-practices-for-bigquery

❑ https://medium.com/analytics-vidhya/write-efficient-queries-on-bigquery-42686c72d81e

❑ https://cloud.google.com/blog/topics/developers-practitioners/bigquery-admin-reference-guide-query-optimization

❑ https://cloud.google.com/bigquery/table-decorators

❑ https://cloud.google.com/bigquery/docs/querying-partitioned-tables#querying_partitioned_tables_2

❑

THANK YOU!

Office

Wisma 46, 39th Floor
Jl. Jend. Sudirman Kav. 1
Central Jakarta, 10220

Phone

(+62)21 39731122

Email

info@sg-edts.com

edts_sg

SG-EDTS

SG-EDTS