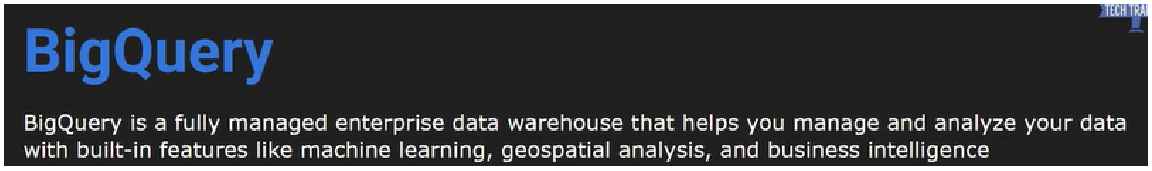
Introduction

1. WHat is bigquery?

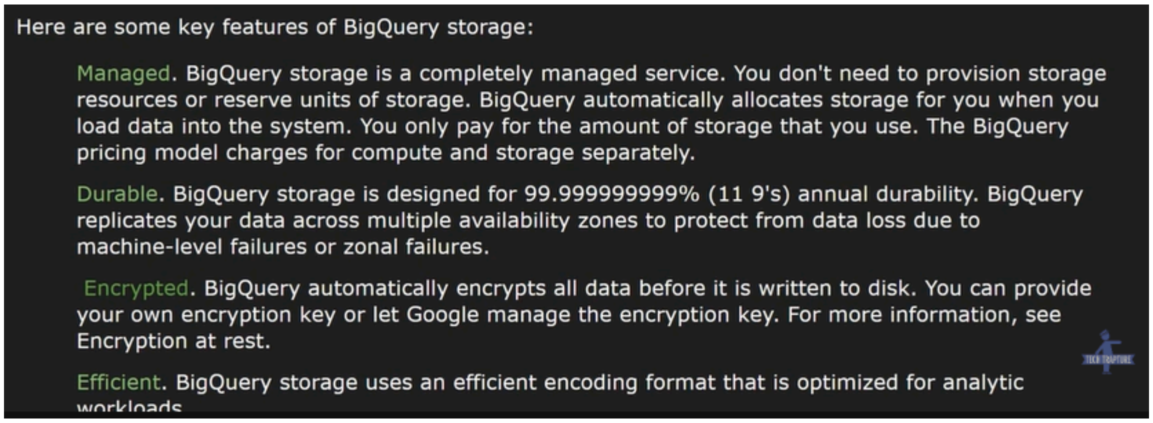


BigQuery

BigQuery is a fully managed enterprise data warehouse that helps you manage and analyze your data with built-in features like machine learning, geospatial analysis, and business intelligence

So here we dont need to manage any compute instance…we create the dataset and table…and load the data…and rest of infrastructure needs will be taken care by google

2. Key features



Here are some key features of BigQuery storage:

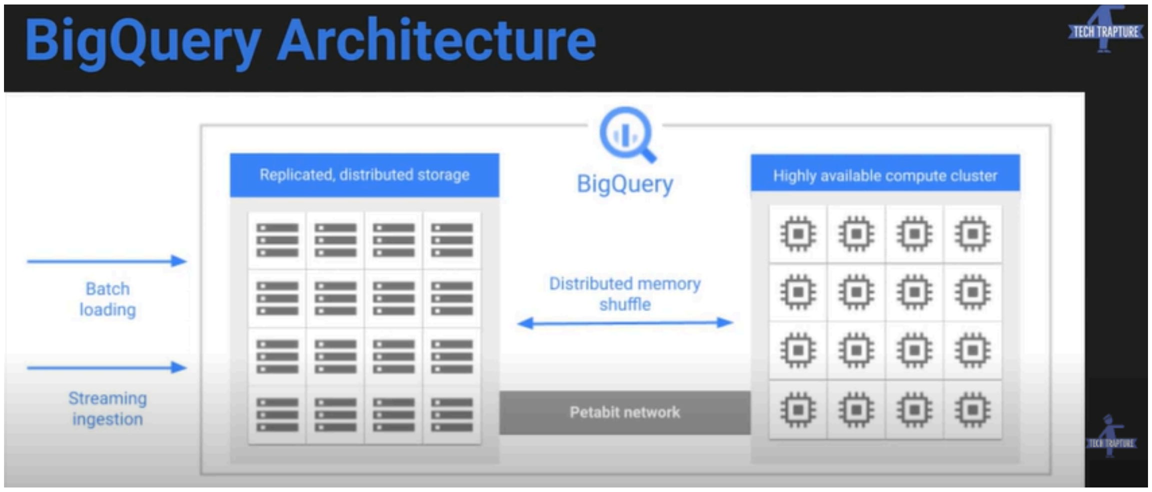
Managed. BigQuery storage is a completely managed service. You don't need to provision storage resources or reserve units of storage. BigQuery automatically allocates storage for you when you load data into the system. You only pay for the amount of storage that you use. The BigQuery pricing model charges for compute and storage separately.

Durable. BigQuery storage is designed for 99.999999999% (11 9's) annual durability. BigQuery replicates your data across multiple availability zones to protect from data loss due to machine-level failures or zonal failures.

Encrypted. BigQuery automatically encrypts all data before it is written to disk. You can provide your own encryption key or let Google manage the encryption key. For more information, see Encryption at rest.

Efficient. BigQuery storage uses an efficient encoding format that is optimized for analytic workloads

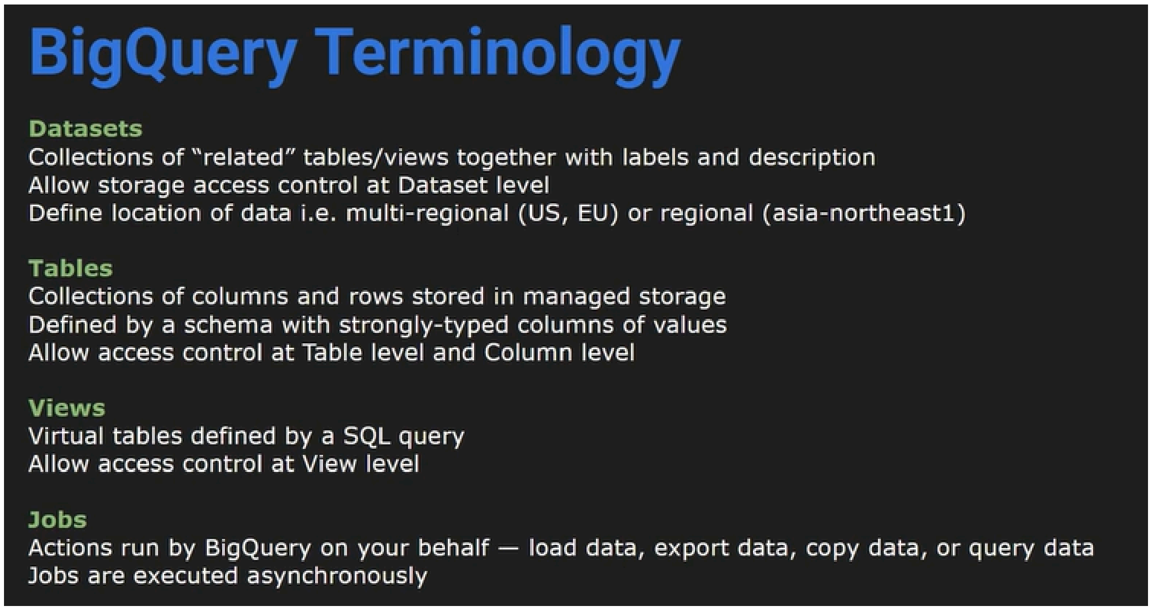
3. Architecture



BigQuery Architecture

Batch loading
Streaming ingestion
Replicated, distributed storage
BigQuery
Distributed memory shuffle
Petabit network
Highly available compute cluster

4. Terminology


BigQuery Terminology

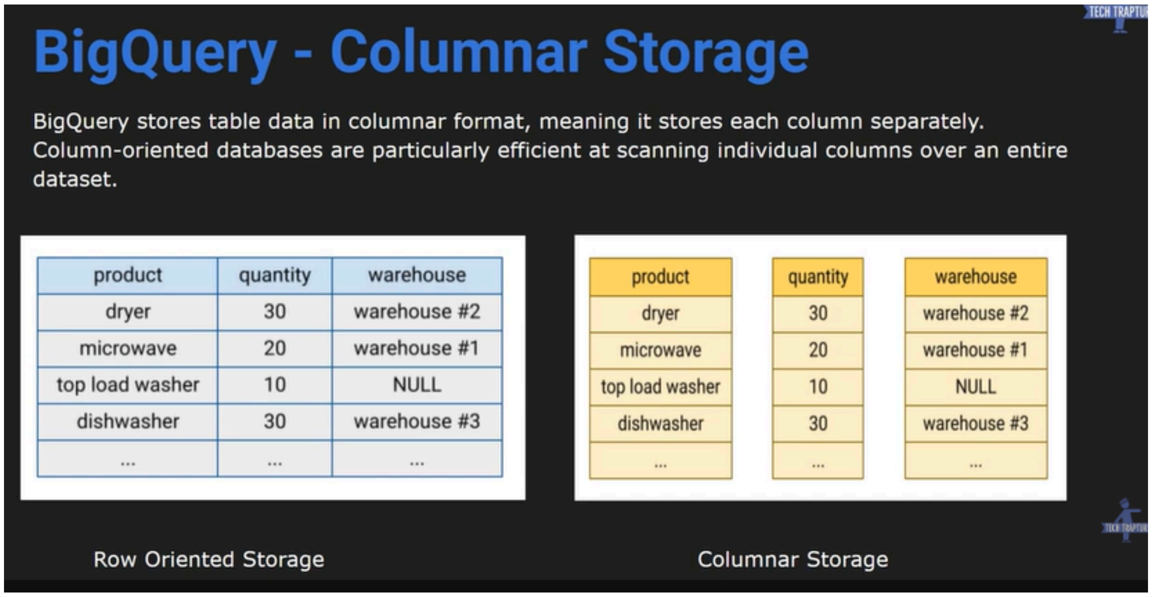
Datasets
Collections of "related" tables/views together with labels and description
Allow storage access control at Dataset level
Define location of data i.e. multi-regional (US, EU) or regional (asia-northeast1)

Tables
Collections of columns and rows stored in managed storage
Defined by a schema with strongly-typed columns of values
Allow access control at Table level and Column level

Views
Virtual tables defined by a SQL query
Allow access control at View level

Jobs
Actions run by BigQuery on your behalf — load data, export data, copy data, or query data
Jobs are executed asynchronously

5. BigQuery Columnar storage


BigQuery - Columnar Storage

BigQuery stores table data in columnar format, meaning it stores each column separately. Column-oriented databases are particularly efficient at scanning individual columns over an entire dataset.

product	quantity	warehouse
dryer	30	warehouse #2
microwave	20	warehouse #1
top load washer	10	NULL
dishwasher	30	warehouse #3
...	...	...

Row Oriented Storage

product
dryer
microwave
top load washer
dishwasher
...

quantity
30
20
10
30
...

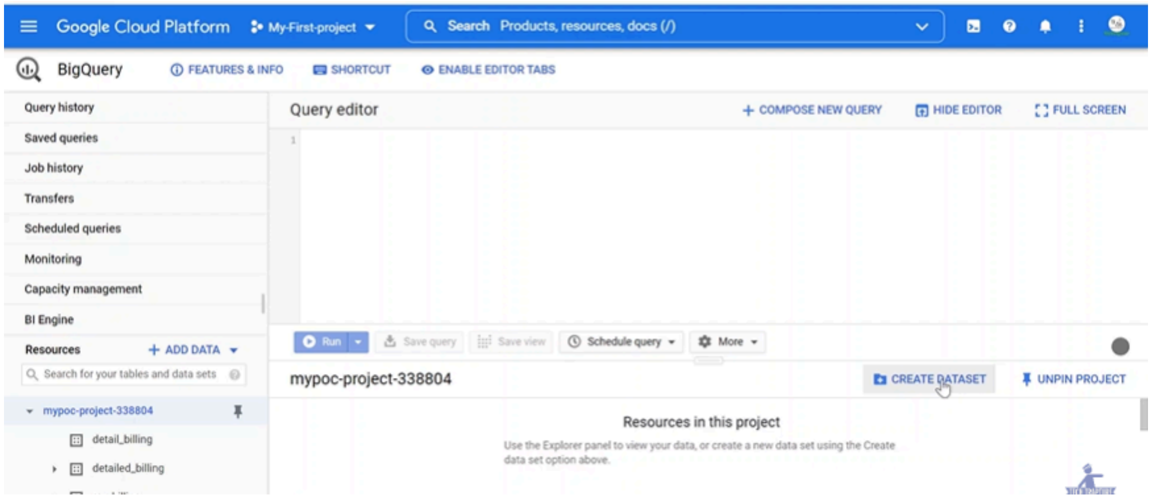
warehouse
warehouse #2
warehouse #1
NULL
warehouse #3
...

Columnar Storage

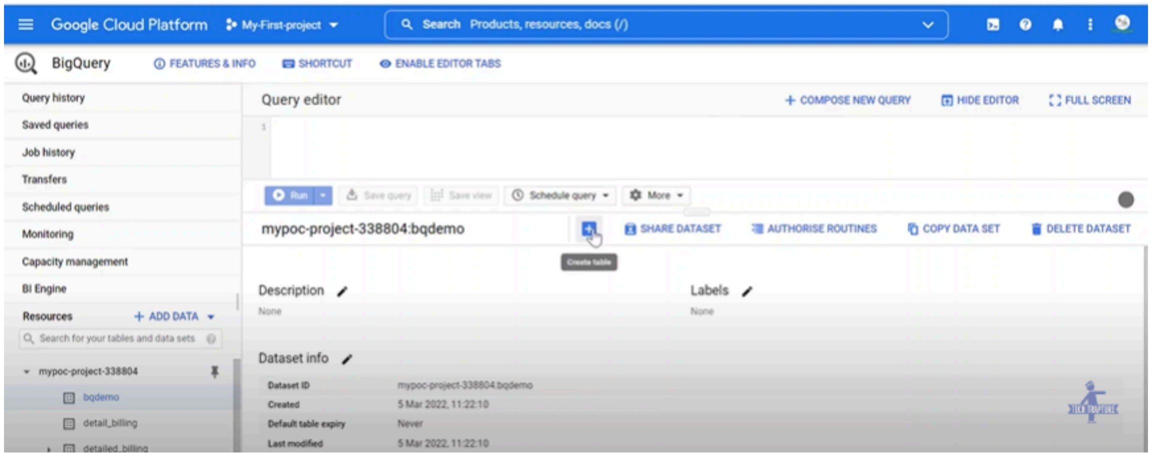
Dataset and table in BigQuery

1. Bigquery is a google manages cloud datawarehouse

2. Click on your project and then we can create dataset

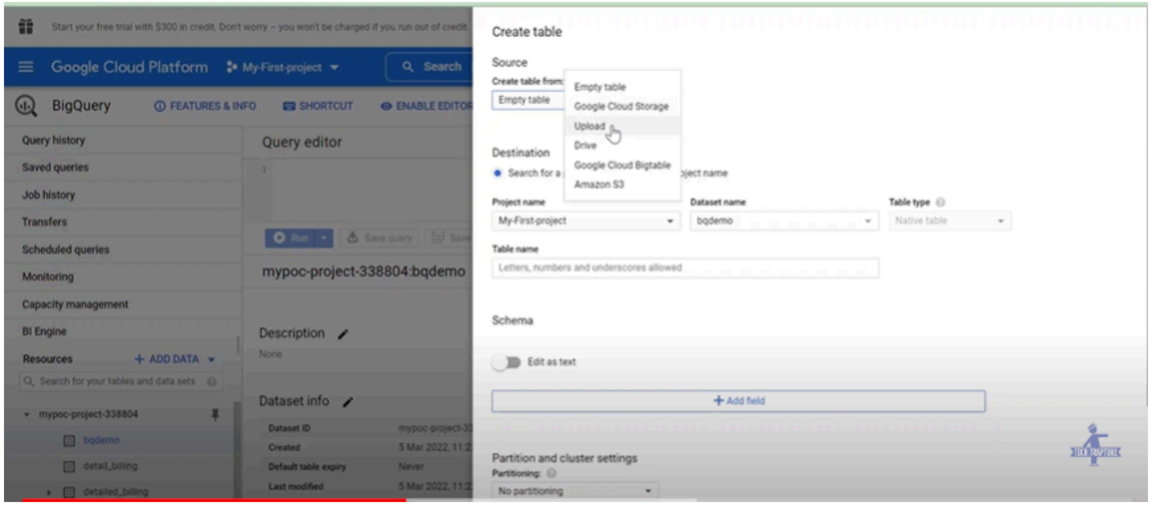


3. Click on create dataset..and give a name with default configs
4. After that we create a table in dataset

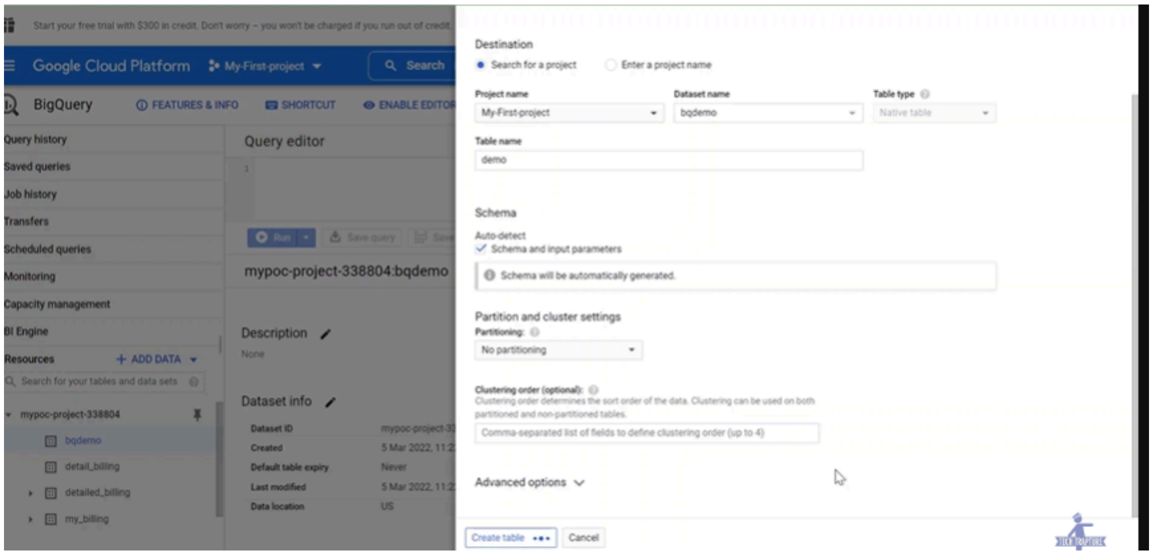


Dataset ID	mypoc-project-338804:bqdemo
Created	5 Mar 2022, 11:22:10
Default table expiry	Never
Last modified	5 Mar 2022, 11:22:10

5. Here we can upload the tables from our local computer or from google cloud storage etc



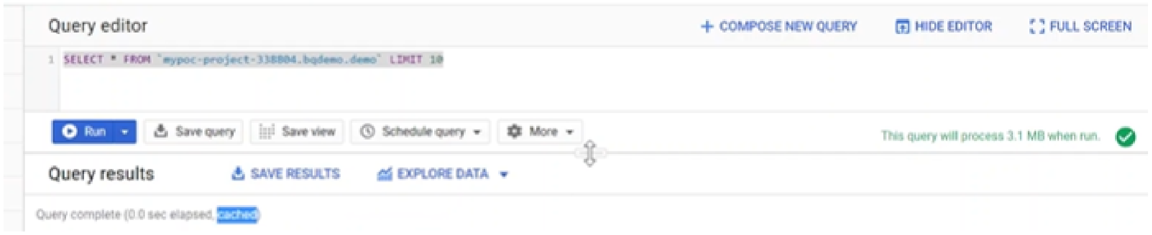
6. Next we upload the file and give a new table name



Project name: My-First-project
Dataset name: bqdemo
Table type: Native table
Table name: demo
Auto-detect: ✓ Schema and input parameters
Schema will be automatically generated.
Partitioning: No partitioning
Clustering order (optional): Clustering order determines the sort order of the data. Clustering can be used on both partitioned and non-partitioned tables.
Comma-separated list of fields to define clustering order (up to 4)

and create it

7. After that we can query our data
8. Here if we run the same query again..then it retrieve the output from the cache



```
SELECT * FROM `mypoc-project-338804.bqdemo.demo` LIMIT 10
```

This query will process 3.1 MB when run.

Query complete (0.0 sec elapsed, cached)

Partitioning in Bigquery

1. Here we see …how we can optimie the performance using partitioning and clustering
2. Inside our project here..we have multiple datasets and tables

3. Here we'll work with techforum table

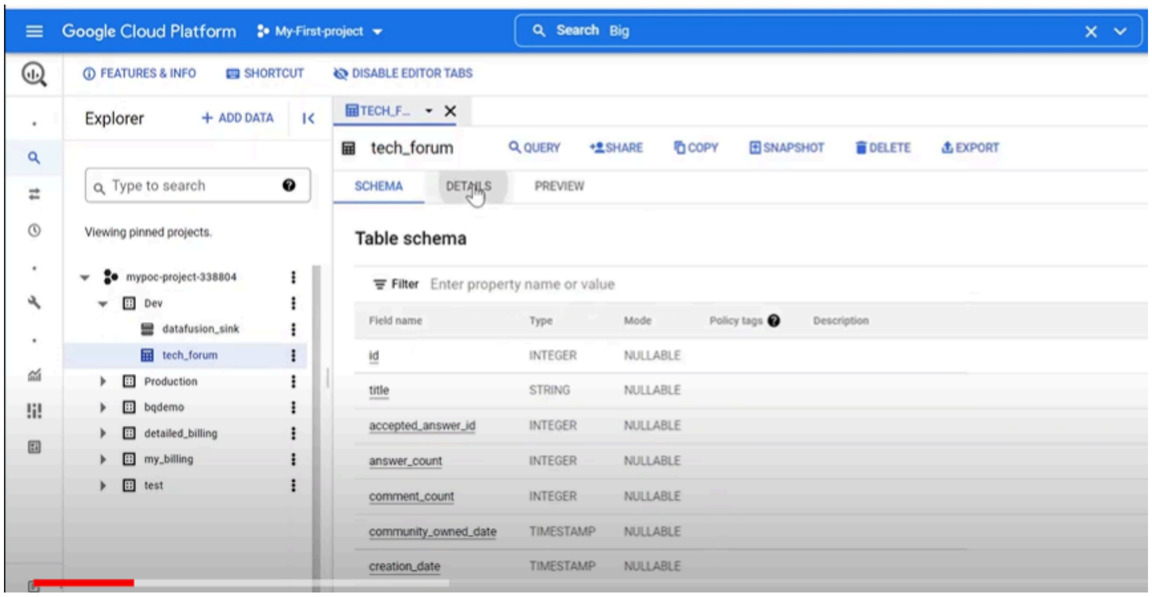


Table schema

Field name	Type	Mode	Policy tags	Description
id	INTEGER	NULLABLE		
title	STRING	NULLABLE		
accepted_answer_id	INTEGER	NULLABLE		
answer_count	INTEGER	NULLABLE		
comment_count	INTEGER	NULLABLE		
community_owned_date	TIMESTAMP	NULLABLE		
creation_date	TIMESTAMP	NULLABLE		

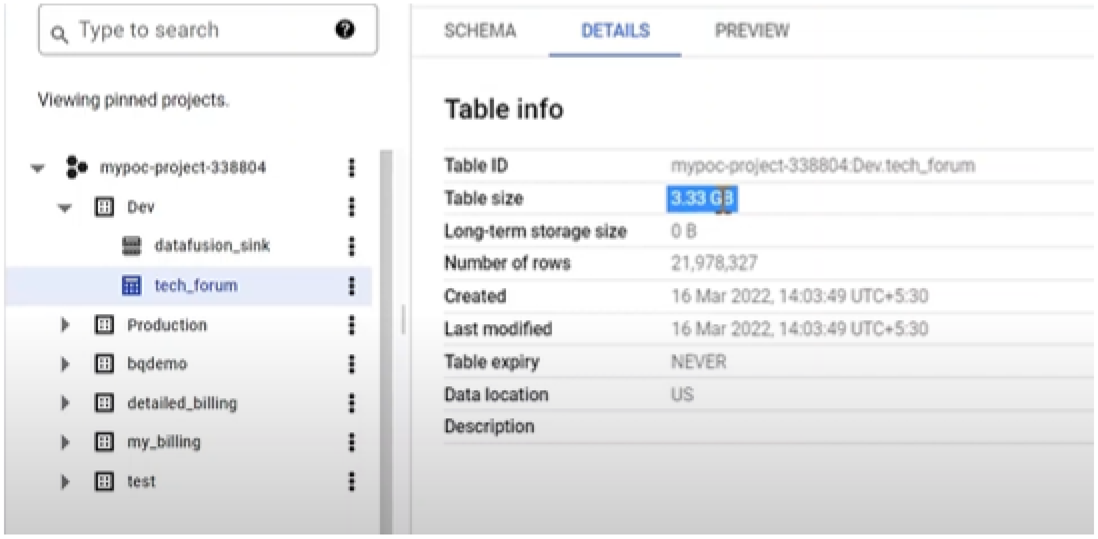
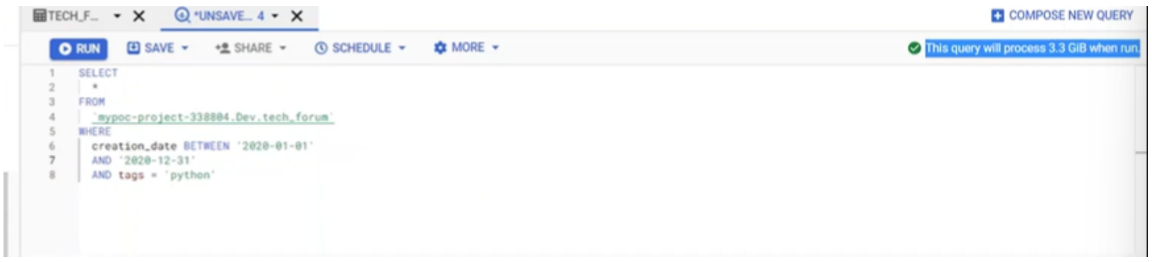


Table info

Table ID	mypoc-project-338804:Dev.tech_forum
Table size	3.33 GB
Long-term storage size	0 B
Number of rows	21,978,327
Created	16 Mar 2022, 14:03:49 UTC+5:30
Last modified	16 Mar 2022, 14:03:49 UTC+5:30
Table expiry	NEVER
Data location	US
Description	

Table size is 3.3GB

4. Now lets query this table
5. As the data is stored in columnar format in bigquery…if we use select * from table limit 10….it read the entire file…even though we need only 10rows
6. Practical example
7. In this query

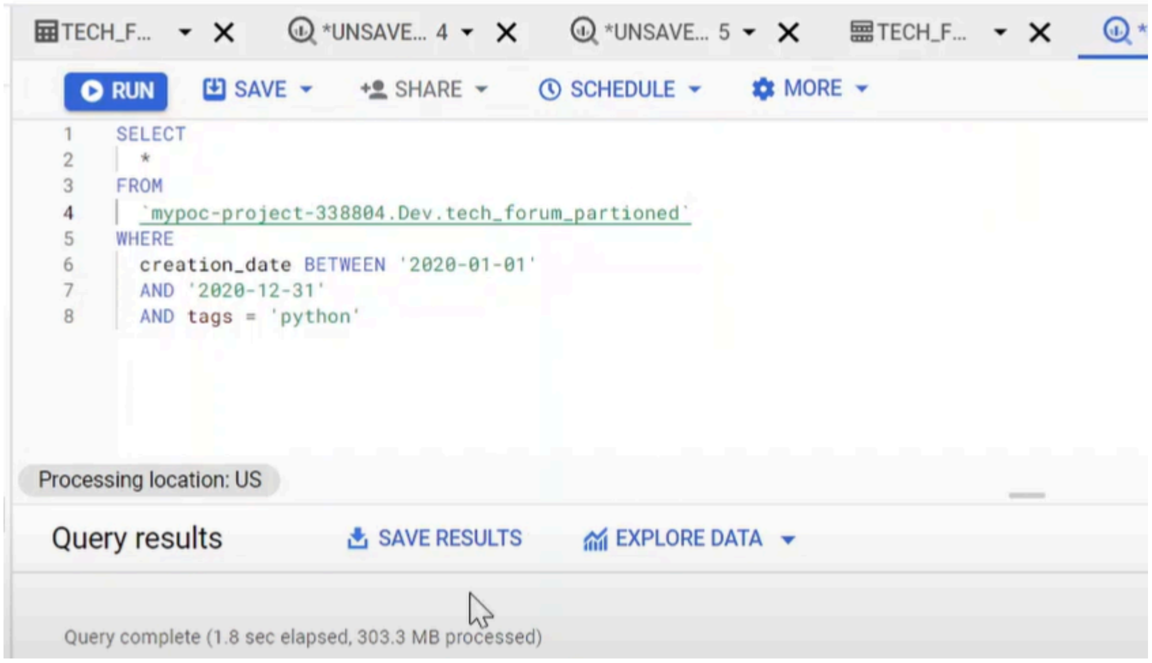


This query will process 3.3 GiB when run.

```
SELECT
  *
FROM
  `mypoc-project-338804.Dev.tech_forum`
WHERE
  creation_date BETWEEN '2020-01-01'
  AND '2020-12-31'
  AND tags = 'python'
```

even if we filtered the data…it is reading complete table(3.3GB)

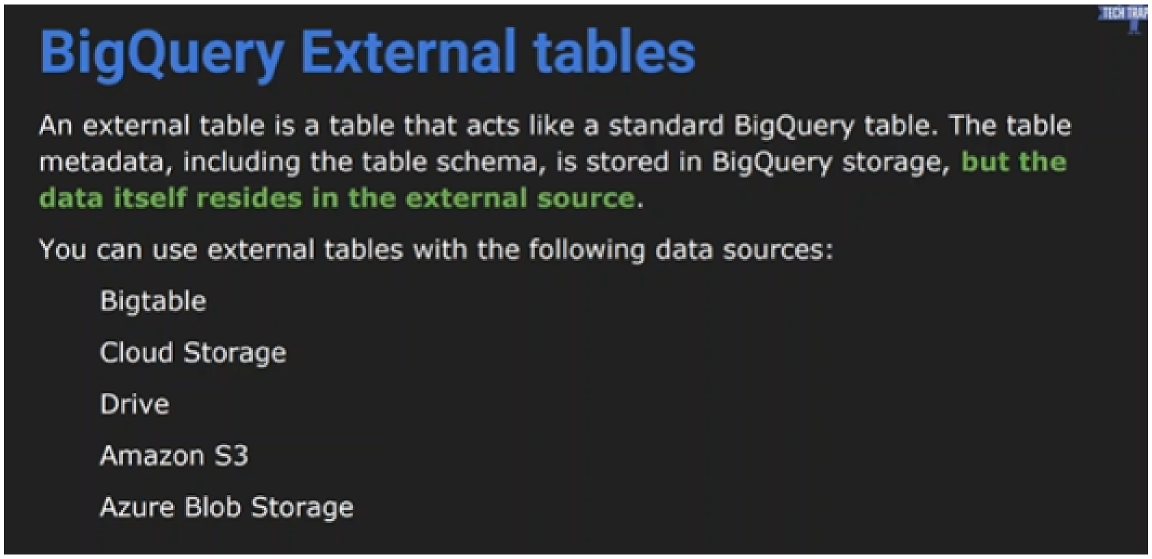
8. SO to avoid this..we can use partitions
9. Now here we partitioned the table by day
10. Now if run the same query with partition..then time complexity would br



```
SELECT
  *
FROM
  `mypoc-project-338804.Dev.tech_forum_partioned`
WHERE
  creation_date BETWEEN '2020-01-01'
  AND '2020-12-31'
  AND tags = 'python'
```

Query complete (1.8 sec elapsed, 303.3 MB processed)

BigQuery Federated Queries

1. 

BigQuery External tables

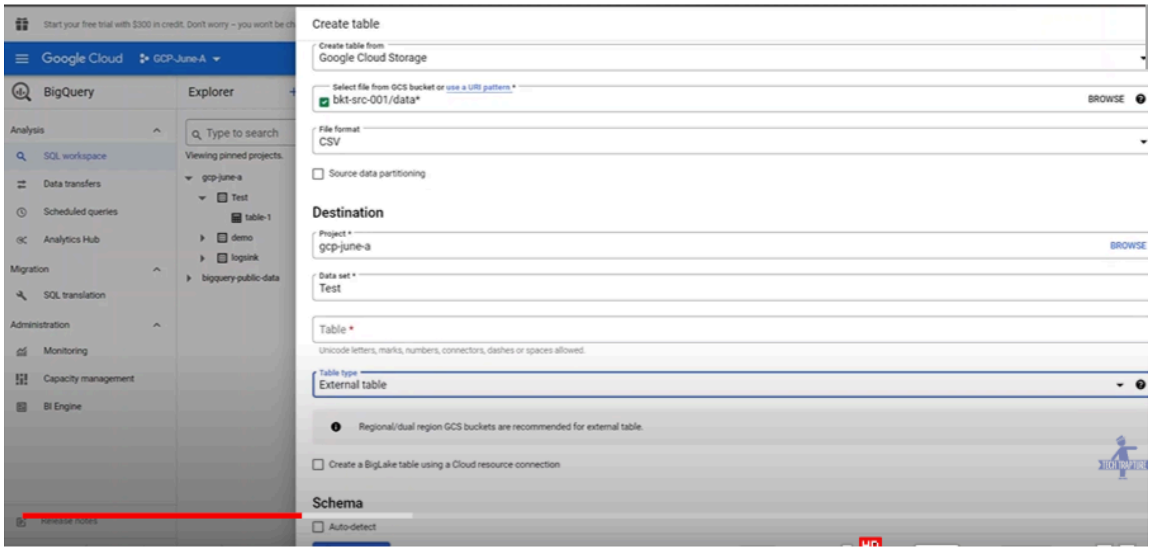
An external table is a table that acts like a standard BigQuery table. The table metadata, including the table schema, is stored in BigQuery storage, **but the data itself resides in the external source.**

You can use external tables with the following data sources:

- Bigtable
- Cloud Storage
- Drive
- Amazon S3
- Azure Blob Storage

2. Next we'll open a bigquery…and create a external tables

3. Here we'll create a external table from GCS



Create table

Create table from: Google Cloud Storage

Select file from GCS bucket or use a URI pattern *: bkt-src-001/data*

File format: CSV

Source data partitioning

Destination

Project *: gcp-june-a

Data set *: Test

Table *

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

Table type: External table

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

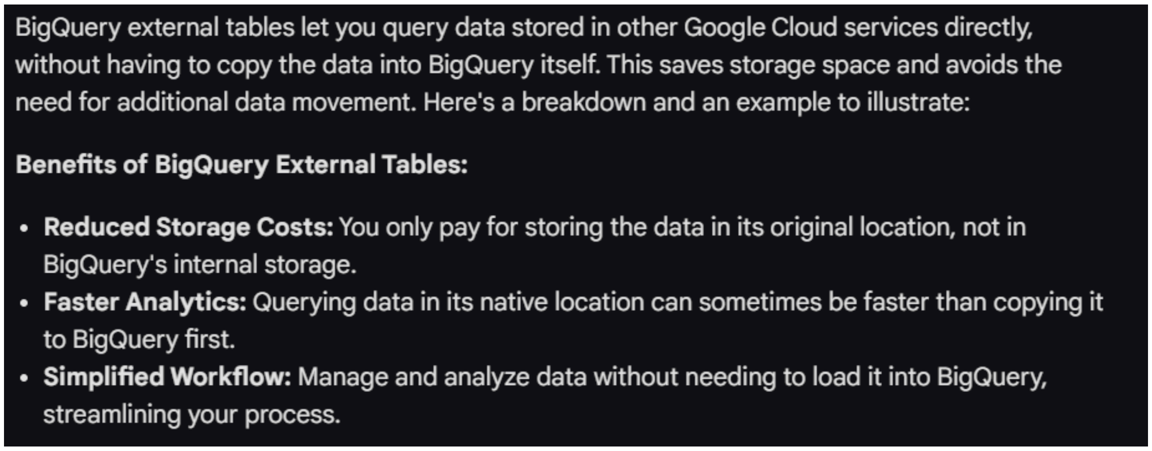
Create a BigLake table using a Cloud resource connection

Schema

Auto-detect

and we create table

4. So here the bigquery will only contain the metadata…but the actual data will be in the external source itself

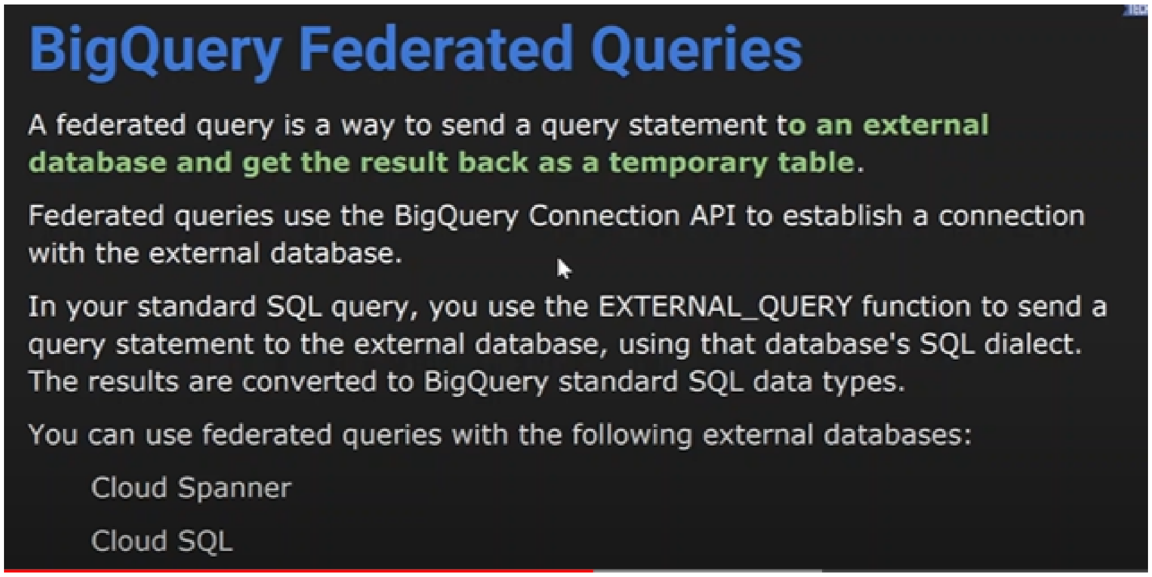


BigQuery external tables let you query data stored in other Google Cloud services directly, without having to copy the data into BigQuery itself. This saves storage space and avoids the need for additional data movement. Here's a breakdown and an example to illustrate:

Benefits of BigQuery External Tables:

- **Reduced Storage Costs:** You only pay for storing the data in its original location, not in BigQuery's internal storage.
- **Faster Analytics:** Querying data in its native location can sometimes be faster than copying it to BigQuery first.
- **Simplified Workflow:** Manage and analyze data without needing to load it into BigQuery, streamlining your process.

5. Federated tables



BigQuery Federated Queries

A federated query is a way to send a query statement t**o an external database and get the result back as a temporary table**.

Federated queries use the BigQuery Connection API to establish a connection with the external database.

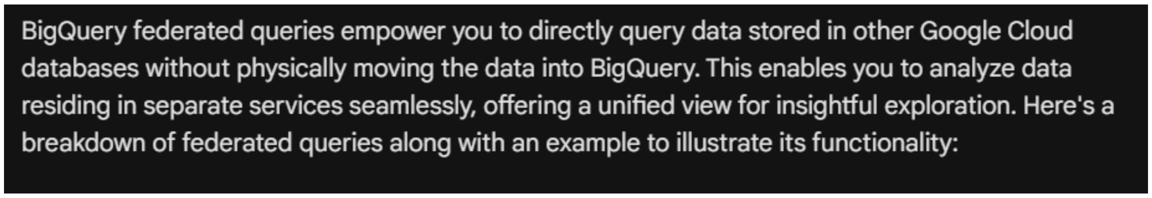
In your standard SQL query, you use the EXTERNAL_QUERY function to send a query statement to the external database, using that database's SQL dialect. The results are converted to BigQuery standard SQL data types.

You can use federated queries with the following external databases:

Cloud Spanner

Cloud SQL

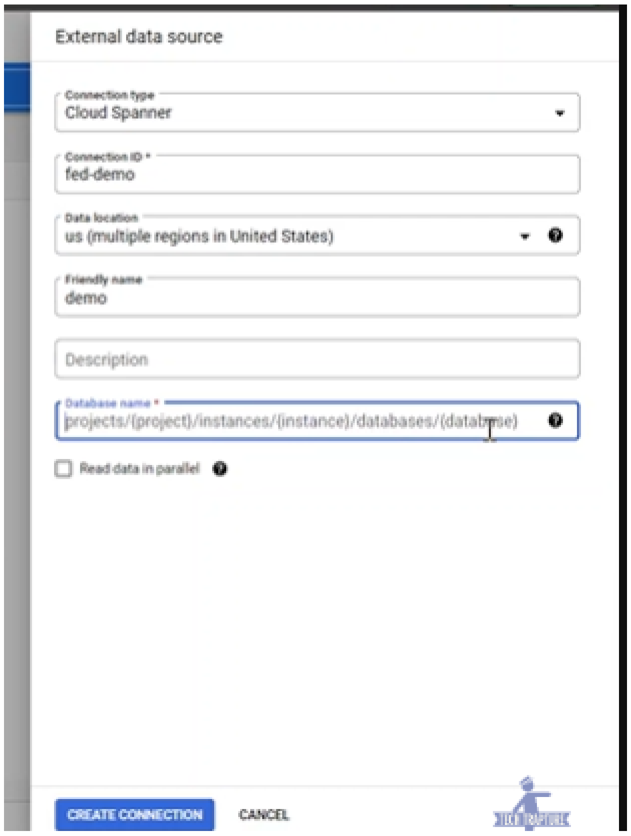
practical



BigQuery federated queries empower you to directly query data stored in other Google Cloud databases without physically moving the data into BigQuery. This enables you to analyze data residing in separate services seamlessly, offering a unified view for insightful exploration. Here's a breakdown of federated queries along with an example to illustrate its functionality:

6.

7. To establish a connection..we click on add data…and create an external connection…and give details



External data source

Connection type: Cloud Spanner

Connection ID *: fed-demo

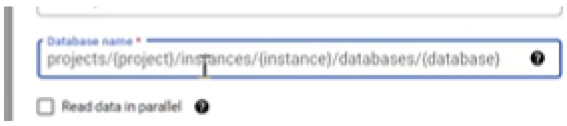
Data location: us (multiple regions in United States)

Friendly name: demo

Description

Database name *: projects/{project}/instances/{instance}/databases/{database}

☐ Read data in parallel

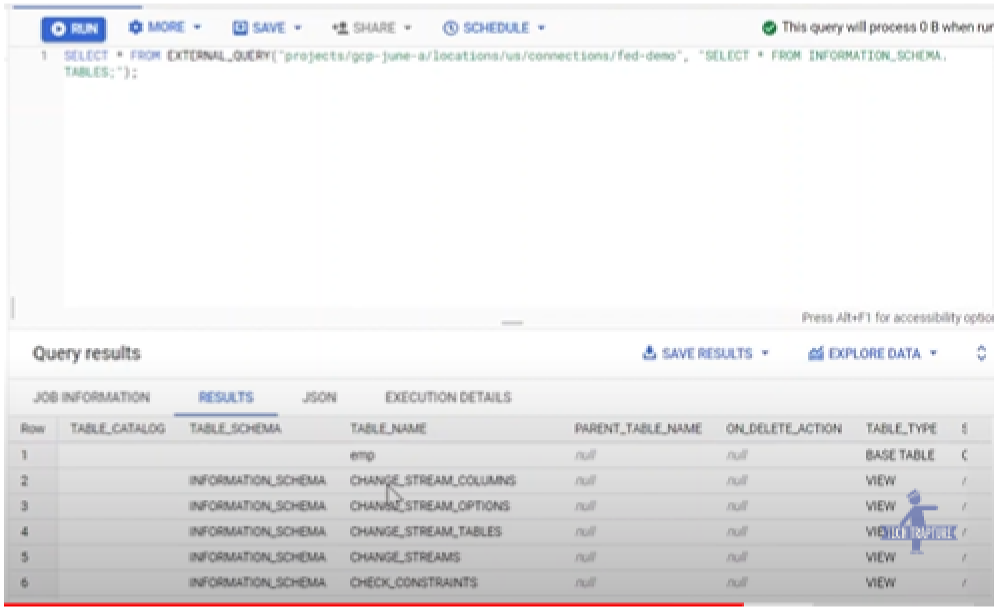


Database name *: projects/{project}/instances/{instance}/databases/{database}

☐ Read data in parallel

8. Database name should be in this format click on create connection

9. After that we can query the table using external query



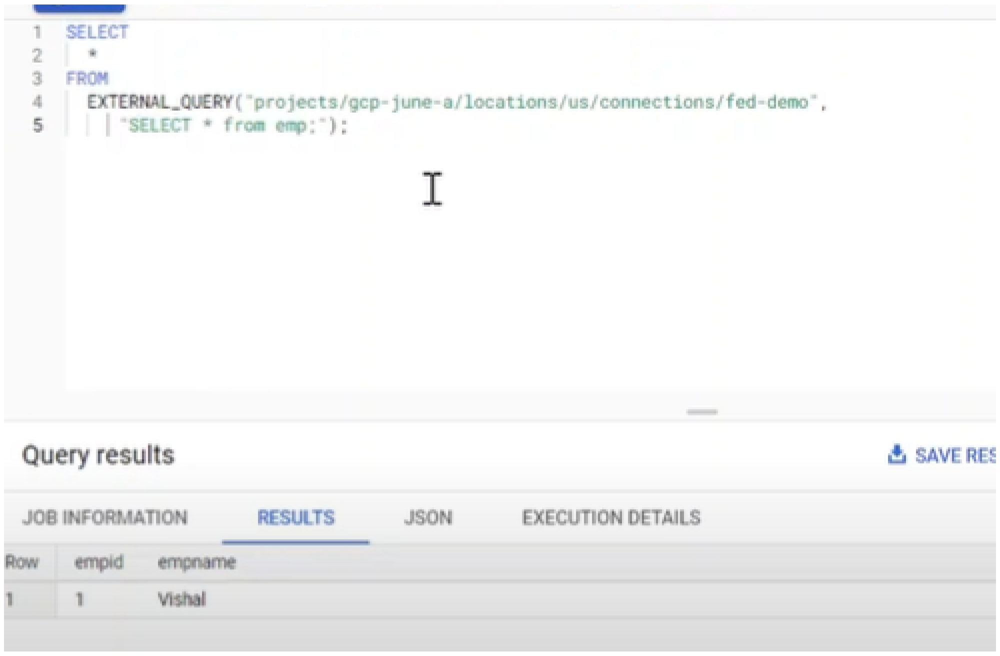
```
SELECT * FROM EXTERNAL_QUERY("projects/gcp-june-a/locations/us/connections/fed-demo", "SELECT * FROM INFORMATION_SCHEMA.
TABLES;");
```

Query results

Row	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	PARENT_TABLE_NAME	ON_DELETE_ACTION	TABLE_TYPE
1			emp	null	null	BASE TABLE
2		INFORMATION_SCHEMA	CHANGE_STREAM_COLUMNS	null	null	VIEW
3		INFORMATION_SCHEMA	CHANGE_STREAM_OPTIONS	null	null	VIEW
4		INFORMATION_SCHEMA	CHANGE_STREAM_TABLES	null	null	VIEW
5		INFORMATION_SCHEMA	CHANGE_STREAMS	null	null	VIEW
6		INFORMATION_SCHEMA	CHECK_CONSTRAINTS	null	null	VIEW

it gives all the tables in external DB

10. Using federated query we have executed queries of external DB



```
SELECT
  *
FROM
  EXTERNAL_QUERY("projects/gcp-june-a/locations/us/connections/fed-demo",
    "SELECT * from emp;");
```

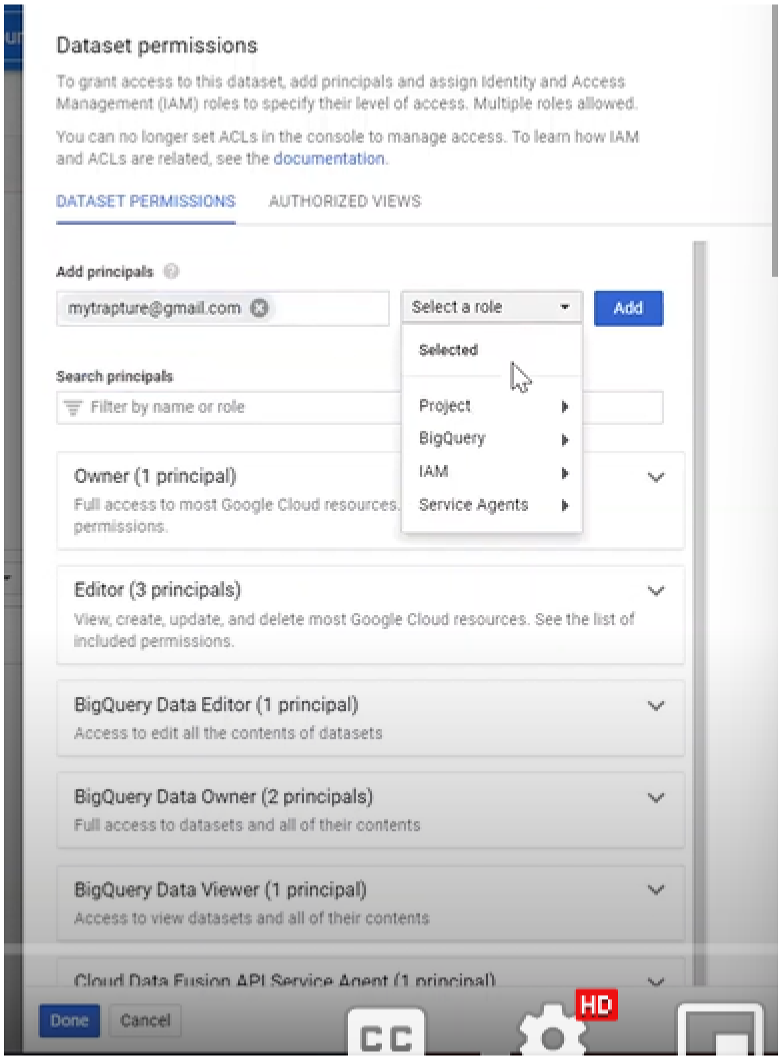
Query results

Row	empid	empname
1	1	Vishal

BigQuery Access control

1. Here we'll learn access controls
2. Lets suppose we have 2 accounts in GCP…first account has is owner and have access to all the files including production

3. 2nd user is employee and has only access to dev
4. Now to grant the access to 2nd user…we have to add dataset permissions and add principals(2nd user email) and select a role



Dataset permissions

To grant access to this dataset, add principals and assign Identity and Access Management (IAM) roles to specify their level of access. Multiple roles allowed.

You can no longer set ACLs in the console to manage access. To learn how IAM and ACLs are related, see the documentation.

DATASET PERMISSIONS AUTHORIZED VIEWS

Add principals

mytrapture@gmail.com

Select a role

Add

Selected

Project

BigQuery

IAM

Service Agents

Search principals

Filter by name or role

Owner (1 principal)

Full access to most Google Cloud resources. permissions.

Editor (3 principals)

View, create, update, and delete most Google Cloud resources. See the list of included permissions.

BigQuery Data Editor (1 principal)

Access to edit all the contents of datasets

BigQuery Data Owner (2 principals)

Full access to datasets and all of their contents

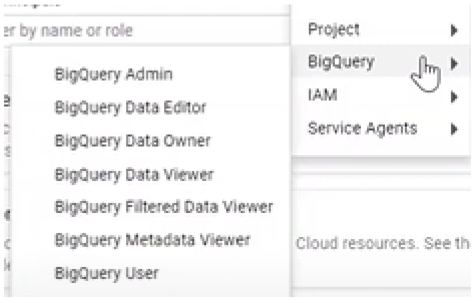
BigQuery Data Viewer (1 principal)

Access to view datasets and all of their contents

Cloud Data Fusion API Service Agent (1 principal)

Done Cancel

5. Here we can assign access on project,bigquery etc…we gave big query(data viewer) access



6. We can also give access levels on tables too

7. For that we go to the tables in our database and select the table and click on share table..then add principal and select role

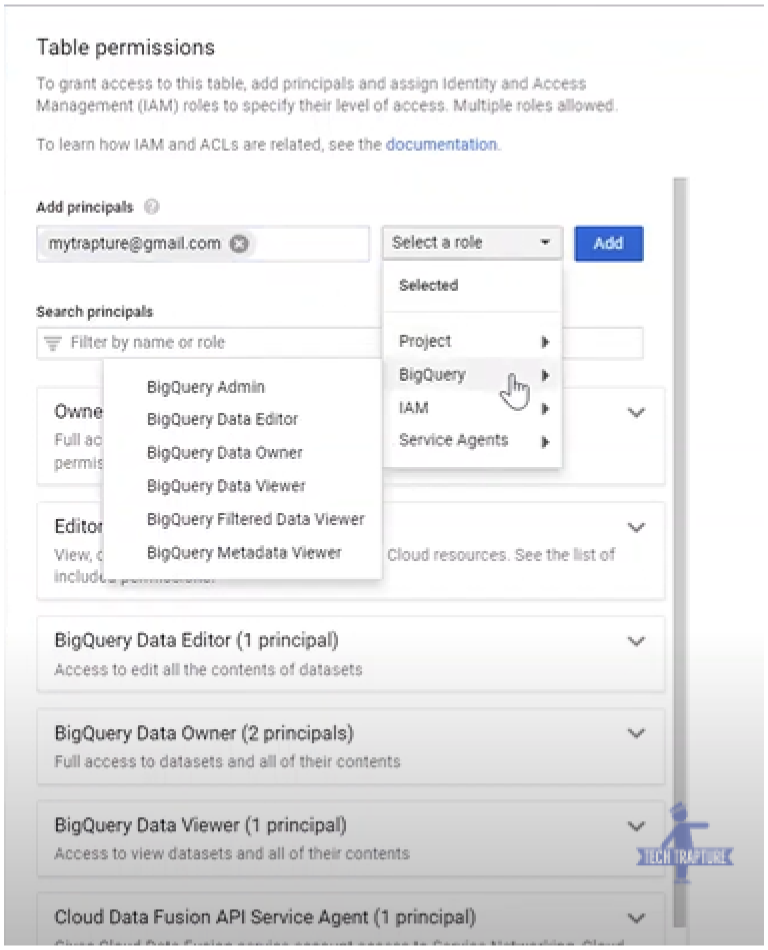


Table permissions

To grant access to this table, add principals and assign Identity and Access Management (IAM) roles to specify their level of access. Multiple roles allowed.

To learn how IAM and ACLs are related, see the documentation.

Add principals: mytrapture@gmail.com

Select a role

Search principals: Filter by name or role

BigQuery Data Editor (1 principal)
Access to edit all the contents of datasets

BigQuery Data Owner (2 principals)
Full access to datasets and all of their contents

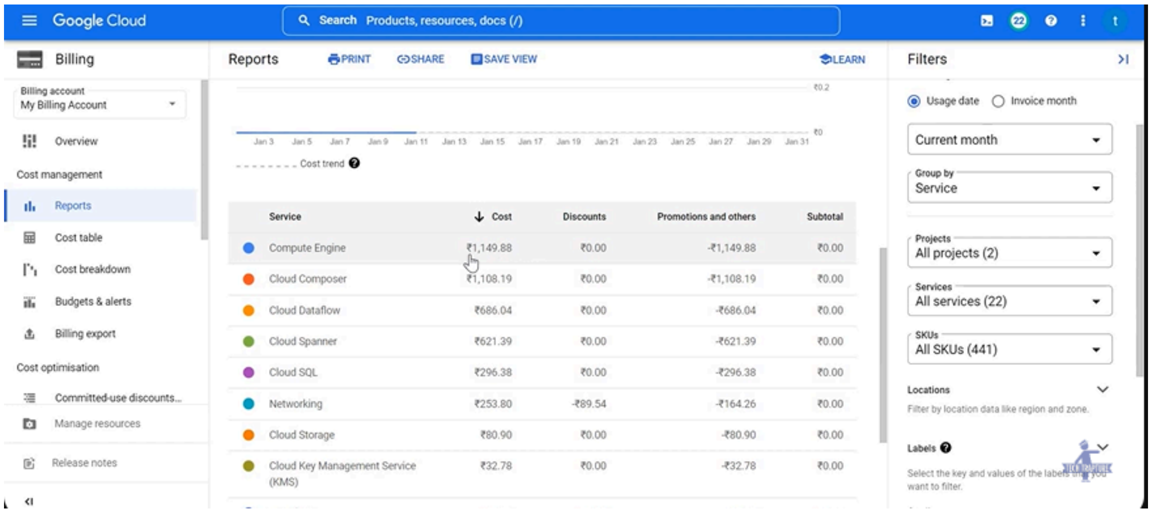
BigQuery Data Viewer (1 principal)
Access to view datasets and all of their contents

Cloud Data Fusion API Service Agent (1 principal)

Export Billing data to Bigquery

1. We'll learn how can we export billings data from big query and create some dashboards
2. The billings permissions will only have who recreate the account

3. Billings of our services



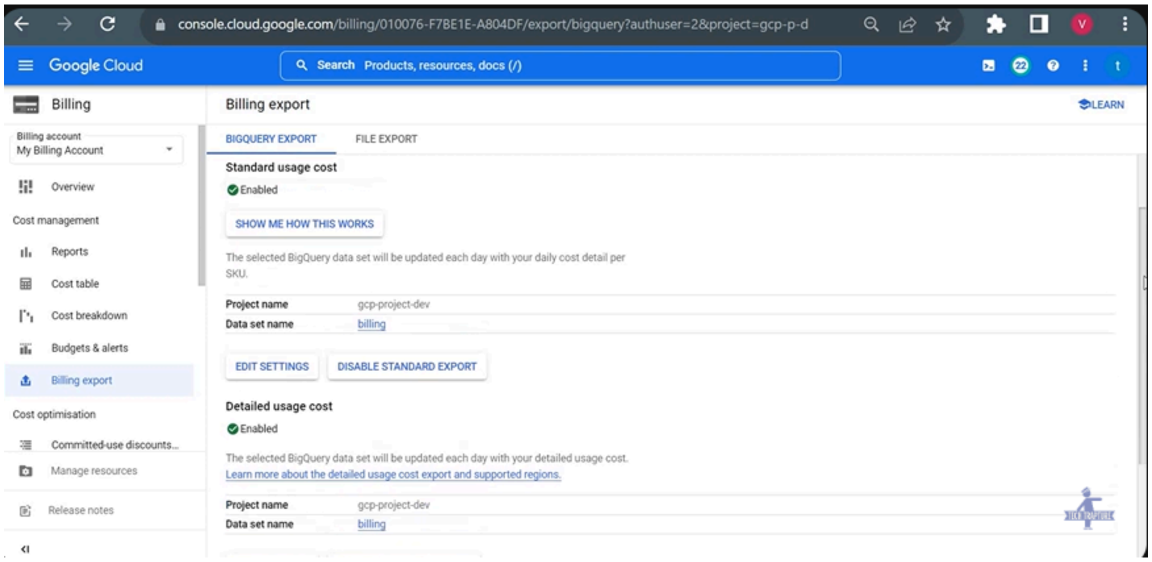
Billing account: My Billing Account

Reports

Service	Cost	Discounts	Promotions and others	Subtotal
Compute Engine	₹1,149.88	₹0.00	-₹1,149.88	₹0.00
Cloud Composer	₹1,108.19	₹0.00	-₹1,108.19	₹0.00
Cloud Dataflow	₹686.04	₹0.00	-₹686.04	₹0.00
Cloud Spanner	₹621.39	₹0.00	-₹621.39	₹0.00
Cloud SQL	₹296.38	₹0.00	-₹296.38	₹0.00
Networking	₹253.80	-₹89.54	-₹164.26	₹0.00
Cloud Storage	₹80.90	₹0.00	-₹80.90	₹0.00
Cloud Key Management Service (KMS)	₹32.78	₹0.00	-₹32.78	₹0.00

Filters: [✓ Usage date] Invoice month
Current month
Group by: Service
Projects: All projects (2)
Services: All services (22)
SKUs: All SKUs (441)
Locations: Filter by location data like region and zone.
Labels: Select the key and values of the labels you want to filter.

4. And we want our team to analyze the data…to have them access this data..we have export this data to bigquery
5. Go to billings export



console.cloud.google.com/billing/010076-F7BE1E-A804DF/export/bigquery?authuser=2&project=gcp-p-d

Billing account: My Billing Account

Billing export

Standard usage cost
Enabled
The selected BigQuery data set will be updated each day with your daily cost detail per SKU.

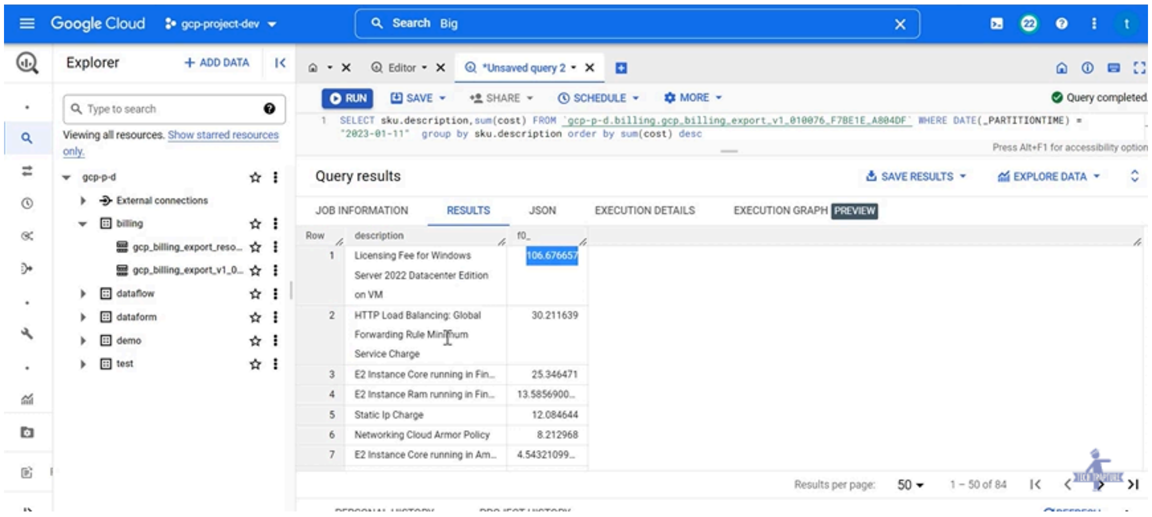
Project name	gcp-project-dev
Data set name	billing

Detailed usage cost
Enabled
The selected BigQuery data set will be updated each day with your detailed usage cost.
Learn more about the detailed usage cost export and supported regions.

Project name	gcp-project-dev
Data set name	billing

and enable export to GCP

6. Lets go to bigquery and we can query the billing data…to see more accurate details



```
SELECT sku.description,sum(cost) FROM `gcp-p-d.billing.gcp_billing_export_v1_010076_F7BE1E_A804DF` WHERE DATE(_PARTITIONTIME) =
"2023-01-11"  group by sku.description order by sum(cost) desc
```

Query results

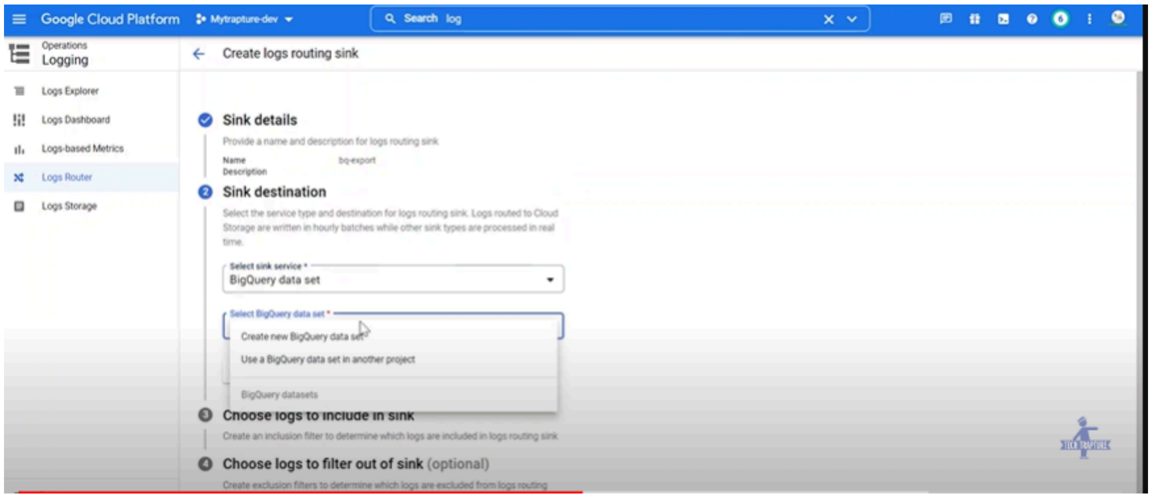
Row	description	f0_
1	Licensing Fee for Windows Server 2022 Datacenter Edition on VM	106.676657
2	HTTP Load Balancing: Global Forwarding Rule Minimum Service Charge	30.211639
3	E2 Instance Core running in Fin...	25.346471
4	E2 Instance Ram running in Fin...	13.5856900...
5	Static Ip Charge	12.084644
6	Networking Cloud Armor Policy	8.212968
7	E2 Instance Core running in Am...	4.54321099...

Results per page: 50 1 – 50 of 84

7. Now we can create dashboards in the looker studio…well connect to GCP and share the datasets

Export GCP logs to BigQuery

1. Search logs and go to logs router
2. Then we have to create sink…but before that we have to create a dataset in the bigquery to store this log tables/data
3. Next we have to create the sink details in log router



Create logs routing sink

Sink details
Provide a name and description for logs routing sink
Name bq-export
Description

Sink destination
Select the service type and destination for logs routing sink. Logs routed to Cloud Storage are written in hourly batches while other sink types are processed in real time.
Select sink service *: BigQuery data set
Select BigQuery data set *
- Create new BigQuery data set
- Use a BigQuery data set in another project
- BigQuery datasets

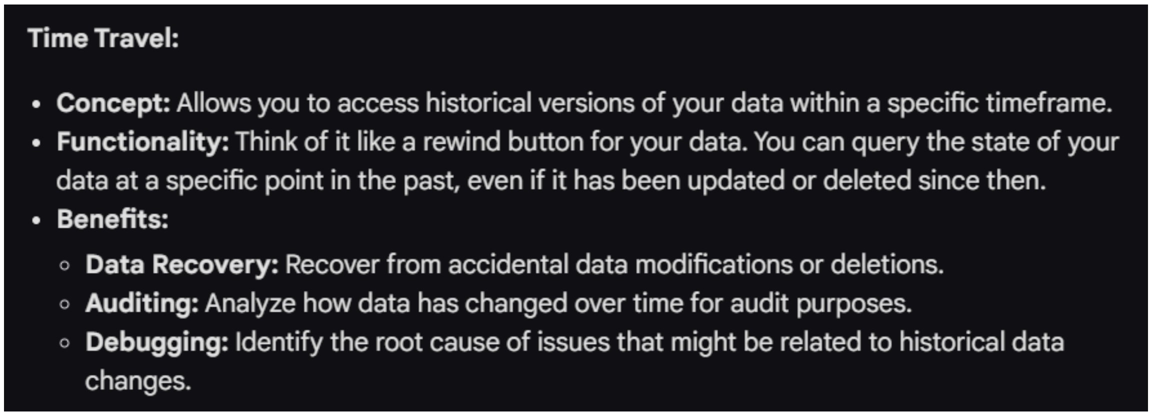
Choose logs to include in sink
Create an inclusion filter to determine which logs are included in logs routing sink

Choose logs to filter out of sink (optional)
Create exclusion filters to determine which logs are excluded from logs routing

select the dataset for logs..and create log sink

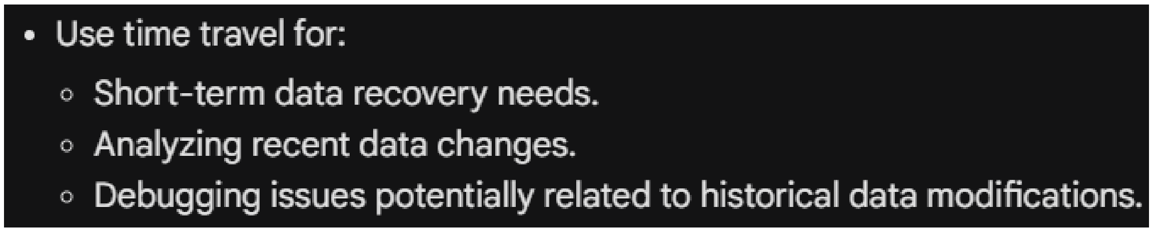
4. Now we can logs in our bigquery

Restore deleted data in Bigquery

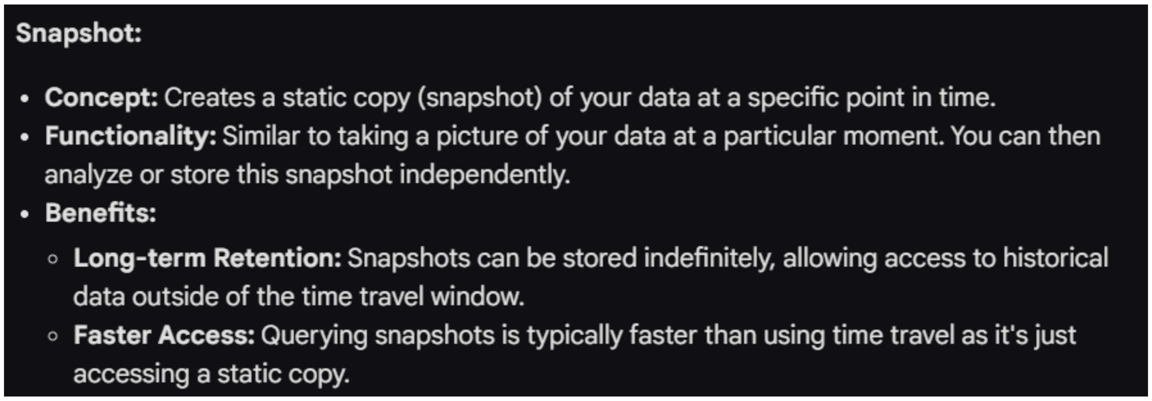
1. We have time travel features…where we can see our data back in time at particular moment…we can only go backwards upto 7 days
2. 

Time Travel:

- **Concept:** Allows you to access historical versions of your data within a specific timeframe.
- **Functionality:** Think of it like a rewind button for your data. You can query the state of your data at a specific point in the past, even if it has been updated or deleted since then.
- **Benefits:**
  - **Data Recovery:** Recover from accidental data modifications or deletions.
  - **Auditing:** Analyze how data has changed over time for audit purposes.
  - **Debugging:** Identify the root cause of issues that might be related to historical data changes.

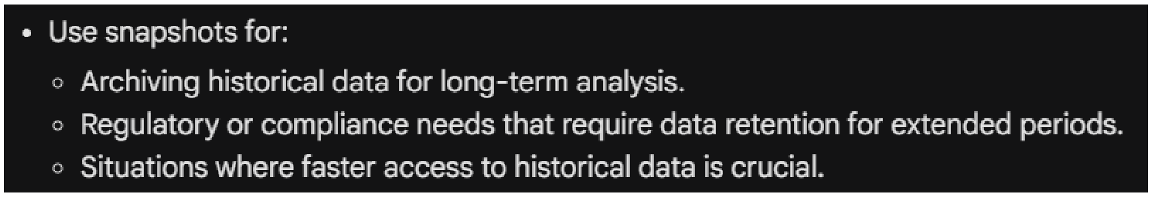


- Use time travel for:
  - Short-term data recovery needs.
  - Analyzing recent data changes.
  - Debugging issues potentially related to historical data modifications.



Snapshot:

- **Concept:** Creates a static copy (snapshot) of your data at a specific point in time.
- **Functionality:** Similar to taking a picture of your data at a particular moment. You can then analyze or store this snapshot independently.
- **Benefits:**
  - **Long-term Retention:** Snapshots can be stored indefinitely, allowing access to historical data outside of the time travel window.
  - **Faster Access:** Querying snapshots is typically faster than using time travel as it's just accessing a static copy.



- Use snapshots for:
  - Archiving historical data for long-term analysis.
  - Regulatory or compliance needs that require data retention for extended periods.
  - Situations where faster access to historical data is crucial.

3.