01. Big Query

03. BQ Partitioning code

Course: BigQuery Fundamentals for Redshift Professionals | Google Cloud Skills Boost

AVRO: https://sqream.com/blog/a-detailed-introduction-to-the-avro-data-format/
bq-load: https://cloud.google.com/bigquery/docs/reference/bq-cli-reference#bq_load

[BigQuery] ((https://cloud.google.com/bigquery/docs/introduction)

- Fully managed Data Warehouse, uses SQL, Supports AI ML via BQ ML, A serverless, analytical DB
- Features: BQ GIS-geospatial analysis,
  Automatic Backup- replicates data and keeps 7 day history of changes
- Has connectors for BigTable, GCS, Google Drive and can import from Datastore backups, CSV, JSON and ARVO
- ### BigQuery Hierarchy

```
├── Project
│    ├── Datasets
│    │    ├── Tables
│    │    │    ├── Data
│    │    │    ├── Jobs
```

- BQ is optimized for reads
- BigQuery BI Engine
- https://medium.com/@vutrinh274/lesson-learned-after-reading-the-bigquery-academic-paper-shuffle-operation-856ee8cdb51c
- Zero infrastructure management
- *Policy tags* give you fine-grained column level access control.
  https://cloud.google.com/bigquery/docs/column-level-security-intro
- Can query TB of data in seconds and PB of data in minutes
- Columnar storage
- Not for transactional purposes

- Can also query from external data sources like SQL, Bigtable
- Supports CSV, JSON, Avro, SQL, Parquet
- Querying is very expensive here
- Command line tool = `bq`
- Tables here have schema
- Jobs are async tasks that work on the top of table
- Used for load, query, extract or query data
- No join is preferred in BigQuery
- The data should be denormalized
- Denormalizing in BigQuery can be done with **nested and repeated columns**
- Security can be applied at a project level or dataset or at table level.
- *Authorized views* allows to share query results without giving access to the underlying data
- https://nileshk611.medium.com/bigquery-authorized-views-and-why-you-should-be-using-them-f10d8f3ae73a
- https://medium.com/google-cloud/hidden-gems-of-bigquery-part-4-five-types-of-views-ddfbc05118fb
- Medium parser - Authorized Views in Big Query. Authorized views and access control are… | by VIKRANT SINGH | Medium (googleusercontent.com)
- Caching time = 24 hrs
  - Not charged for data fetched from cache.
- Can directly run query on cloud storage. No need for dataflow.
- **BigQuery Data Transfer Service** can only transfer data into BigQuery, not out of it
  Sources: S3, Redshift, Azure Blob Storage, Google Ads, Salesforce, Teradata, YouTube Channel
  What is BigQuery Data Transfer Service? | Google Cloud
- You can load data into BigQuery via two options: batch loading (free) and streaming (costly).
- When using the Cloud Console, files loaded from a local data source cannot exceed 10 MB.
- BigQuery maintains a 7 day history of changes so that you can query a point-in time snapshot of data

- Wildcard tables - Used if you want to union all similar tables with similar names. '' *(e.g. project.dataset.Table*)

- **Partitioning > Table Sharding**: When you have multiple wildcard tables, best option is to shard it into single partitioned table. Time and cost efficient

- Types of queries -
  - Interactive: query is executed immediately
  - Batch: Batches of queries are queued and the query starts when idle resources are available

- Link for Migrating data warehouses to BigQuery

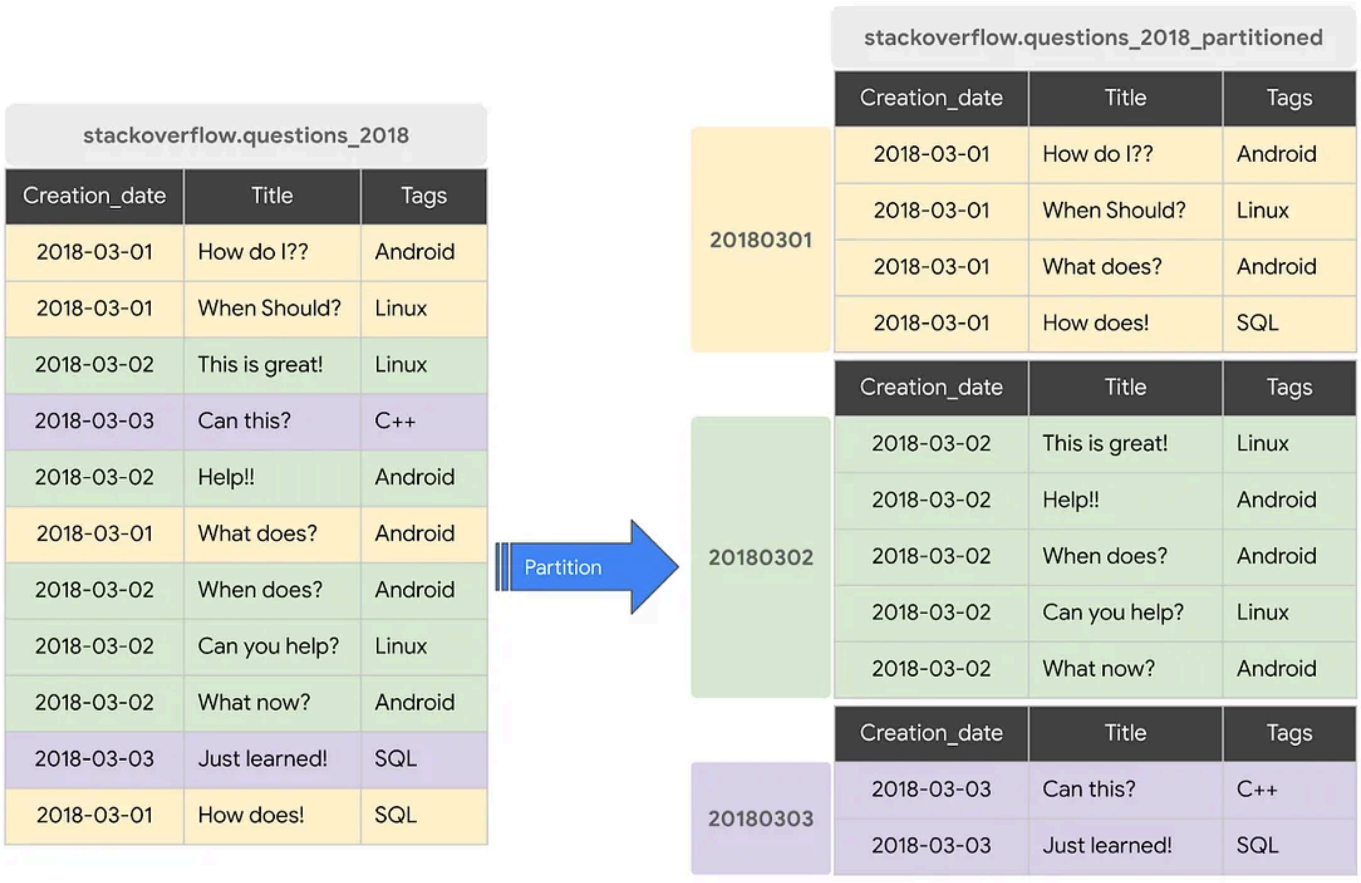
- Link for Best practices for performance

Partitioning in BQ

Partitioning in BigQuery involves dividing large tables into smaller, manageable segments based on the values of one or more columns.

> First partition and then cluster

- Dividing a large table into **segments** to make it easier to manage and query data. This saves cost and time.
- Partition is nothing but a newly created table.
- Helps in pruning other partitions if data is found in one partition
- You can partition on hourly or daily.
- Once table is created, you cannot change it partitioned



stackoverflow.questions_2018

Creation_date	Title	Tags
2018-03-01	How do I??	Android
2018-03-01	When Should?	Linux
2018-03-02	This is great!	Linux
2018-03-03	Can this?	C++
2018-03-02	Help!!	Android
2018-03-01	What does?	Android
2018-03-02	When does?	Android
2018-03-02	Can you help?	Linux
2018-03-02	What now?	Android
2018-03-03	Just learned!	SQL
2018-03-01	How does!	SQL

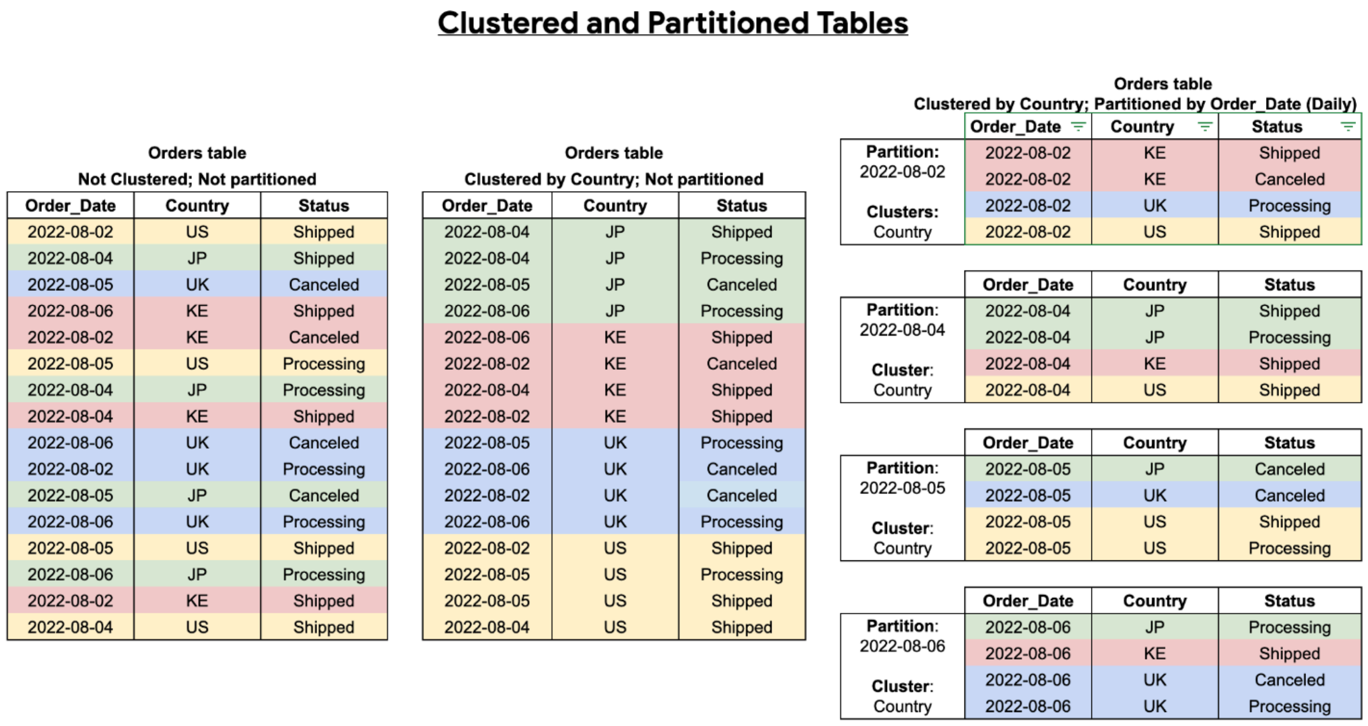
stackoverflow.questions_2018_partitioned

	Creation_date	Title	Tags
20180301	2018-03-01	How do I??	Android
	2018-03-01	When Should?	Linux
	2018-03-01	What does?	Android
	2018-03-01	How does!	SQL

	Creation_date	Title	Tags
20180302	2018-03-02	This is great!	Linux
	2018-03-02	Help!!	Android
	2018-03-02	When does?	Android
	2018-03-02	Can you help?	Linux
	2018-03-02	What now?	Android

	Creation_date	Title	Tags
20180303	2018-03-03	Can this?	C++
	2018-03-03	Just learned!	SQL

Comparing tables that are not clustered or partitioned to tables that are clustered and partitioned.



Clustered and Partitioned Tables

Orders table
Not Clustered; Not partitioned

Order_Date	Country	Status
2022-08-02	US	Shipped
2022-08-04	JP	Shipped
2022-08-05	UK	Canceled
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-05	US	Processing
2022-08-04	JP	Processing
2022-08-04	KE	Shipped
2022-08-06	UK	Canceled
2022-08-02	UK	Processing
2022-08-05	JP	Canceled
2022-08-06	UK	Processing
2022-08-05	US	Shipped
2022-08-06	JP	Processing
2022-08-02	KE	Shipped
2022-08-04	US	Shipped

Orders table
Clustered by Country; Not partitioned

Order_Date	Country	Status
2022-08-04	JP	Shipped
2022-08-04	JP	Processing
2022-08-05	JP	Canceled
2022-08-06	JP	Processing
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-04	KE	Shipped
2022-08-02	KE	Shipped
2022-08-05	UK	Processing
2022-08-06	UK	Canceled
2022-08-02	UK	Canceled
2022-08-06	UK	Processing
2022-08-02	US	Shipped
2022-08-05	US	Processing
2022-08-05	US	Shipped
2022-08-04	US	Shipped

Orders table
Clustered by Country; Partitioned by Order_Date (Daily)

	Order_Date	Country	Status
Partition: 2022-08-02 **Clusters:** Country	2022-08-02	KE	Shipped
	2022-08-02	KE	Canceled
	2022-08-02	UK	Processing
	2022-08-02	US	Shipped

	Order_Date	Country	Status
Partition: 2022-08-04 **Cluster:** Country	2022-08-04	JP	Shipped
	2022-08-04	JP	Processing
	2022-08-04	KE	Shipped
	2022-08-04	US	Shipped

	Order_Date	Country	Status
Partition: 2022-08-05 **Cluster:** Country	2022-08-05	JP	Canceled
	2022-08-05	UK	Canceled
	2022-08-05	US	Shipped
	2022-08-05	US	Processing

	Order_Date	Country	Status
Partition: 2022-08-06 **Cluster:** Country	2022-08-06	JP	Processing
	2022-08-06	KE	Shipped
	2022-08-06	UK	Canceled
	2022-08-06	UK	Processing

Types: https://cloud.google.com/bigquery/docs/partitioned-tables#types_of_partitioning

1. Date-partitioned tables (Date column)

```
CREATE TABLE dataset.table_name (
...
event_date DATE
)
PARTITION BY DATE(event_date)
```

2. Integer-range partitioned (Customer ID)

```
CREATE TABLE dataset.table_name (
...
customer_id INT64
)
PARTITION BY RANGE(customer_id BETWEEN 1 AND 1000, 1001 AND 2000, ...);
```

3. Ingestion time:
   Tables are partitioned based on the timestamp when BigQuery ingests the data.

Considerations

- Partition Expiration (save storage)
- Ingestion Time partitioning (for)
  Partitioning is preferred over Sharding
- **Sharding:** Horizontally splitting a large dataset across multiple independent databases or shards. Each shard contains a subset of the data based on a defined sharding key (e.g., hash of customer ID, geographic location)

Database Sharding

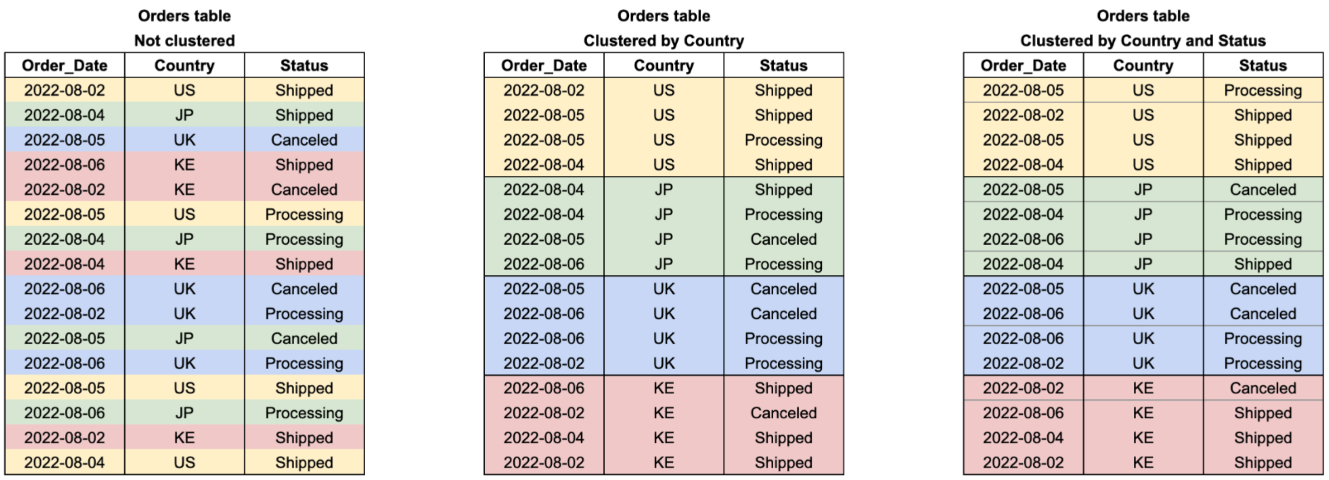
```
-- Suppose you have a partitioned table sales_data partitioned by order_date.
-- Query:
SELECT * FROM sales_data WHERE order_date ≥ '2023-01-01' AND order_date <
'2023-02-01';
-- In this query, order_date ≥ '2023-01-01' AND order_date < '2023-02-01' is
the qualifying filter.
-- BigQuery will only scan partitions that contain data within the date range
'2023-01-01' to '2023-01-31', ignoring other partitions, thus optimizing
performance and cost.
```

Clustering in BQ- how a table is stored

- Data in clustered tables are **sorted** based on values in one or more columns
- Clustering is supported only on partitioned tables
- Clustered tables can improve performance of aggregate queries.

- BigQuery sorts data in clustered tables to improve query performance.

Clustered Tables



Orders table
Not clustered

Order_Date	Country	Status
2022-08-02	US	Shipped
2022-08-04	JP	Shipped
2022-08-05	UK	Canceled
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-05	US	Processing
2022-08-04	JP	Processing
2022-08-04	KE	Shipped
2022-08-06	UK	Canceled
2022-08-02	UK	Processing
2022-08-05	JP	Canceled
2022-08-06	UK	Processing
2022-08-05	US	Shipped
2022-08-06	JP	Processing
2022-08-02	KE	Shipped
2022-08-04	US	Shipped

Orders table
Clustered by Country

Order_Date	Country	Status
2022-08-02	US	Shipped
2022-08-05	US	Shipped
2022-08-05	US	Processing
2022-08-04	US	Shipped
2022-08-04	JP	Shipped
2022-08-04	JP	Processing
2022-08-05	JP	Canceled
2022-08-06	JP	Processing
2022-08-05	UK	Canceled
2022-08-06	UK	Canceled
2022-08-06	UK	Processing
2022-08-02	UK	Processing
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-04	KE	Shipped
2022-08-02	KE	Shipped

Orders table
Clustered by Country and Status

Order_Date	Country	Status
2022-08-05	US	Processing
2022-08-02	US	Shipped
2022-08-05	US	Shipped
2022-08-04	US	Shipped
2022-08-05	JP	Canceled
2022-08-04	JP	Processing
2022-08-06	JP	Processing
2022-08-04	JP	Shipped
2022-08-05	UK	Canceled
2022-08-06	UK	Canceled
2022-08-06	UK	Processing
2022-08-02	UK	Processing
2022-08-02	KE	Canceled
2022-08-06	KE	Shipped
2022-08-04	KE	Shipped
2022-08-02	KE	Shipped

When to use clustering?

- Unpartitioned tables > 64MB and Table partitions >64MB
- Commonly filter on particular columns

Types of Tables in BQ

1. Standard BigQuery tables

- *Permanent tables* are those that are created in a dataset and linked to an external source. Dataset-level access controls can be applied to these tables.
- tables contain structured data and are stored in BigQuery storage in a columnar format.
- Table clones, which are lightweight, writeable copies of BigQuery tables. BigQuery only stores the delta between a table clone and its base table.
- Table snapshots, which are point-in-time copies of tables. They are read-only, but you can restore a table from a table snapshot. BigQuery stores bytes that are different between a snapshot and its base table, so a table snapshot typically uses less storage than a full copy of the table.

2. External tables

External tables are stored outside out of BigQuery storage and refer to data that's stored outside of BigQuery. For more information, see Introduction to external data sources. External tables include the following types:

- BigLake tables, which reference structured data stored in data stores such as Cloud Storage, Amazon Simple Storage Service (Amazon S3), and Azure Blob Storage. These tables let you enforce fine-grained security at the table level.

  For information about how to create BigLake tables, see the following topics:
  - Cloud Storage

    - Amazon S3
    - Blob Storage

- Object tables, which reference unstructured data stored in data stores such as Cloud Storage.

  For information about how to create object tables, see Create object tables.

- Non-BigLake external tables, which reference structured data stored in data stores such as Cloud Storage, Google Drive, and Bigtable. Unlike BigLake tables, these tables don't let you enforce fine-grained security at the table level.

  For information about how to create non-BigLake external tables, see the following topics:
    - Cloud Storage
    - Google Drive
    - Bigtable

- External tables (from external sources, also know as **federated** sources). Used when tables are small. As an external data source, the frequently changing data does not need to be reloaded every time it is updated.

3. Views

Views are logical tables that are defined by using a SQL query. These include the following types:

- Views, which are logical tables that are defined by using SQL queries. These queries define the view that is run each time the view is queried.
  For information about how to create views, see Create views.
- Materialized views, which are precomputed views that periodically cache the results of the view query. The cached results are stored in BigQuery storage.
  For information about how to create materialized views, see Create materialized views.

Use Cases of BQ

- When workload is analytical
- When the data does not change much in DB, since caching is also used in BigQuery
- When complex queries needs to be run
- Offload some work from primary DB to run time taking/complex queries

Import/Export in BQ

- Data Import -
    - Batch import - web console (local files), GCS, GDS
    - Stream - data with CDF, Cloud logging or POST calls. This is a paid service

    - Raw files - federated data source, CSV/JSON/Avro on GCS, Google sheets. Default file format is csv
    - By default, the BigQuery service expects all source data to be UTF-8 encoded. BQ also supports ISO-8859-1
    - To support (occasionally) schema changing we can use 'Automatically detect' for schema changes. Automatically detect is not default selected
- Data Export -
    - Data can only be exported in JSON / CSV / Avro
    - To export more than 1 GB of data, you need to put a wildcard in the destination filename. (up to 1 GB of table data to a single file)

Controlling Cost in BigQuery

- Avoid SELECT *(full scan), select only columns needed (SELECT* EXCEPT)
- Sample data using preview options for free
- Preview queries to estimate costs (dryrun)
- Use max bytes billed to limit query costs
- Don't use LIMIT clause to limit costs (still full scan)
- Monitor costs using dashboards and audit logs
- Partition data by date
- Break query results into stages
- Use default table expiration to delete unneeded data
- Use streaming inserts wisely
- Set hard limit on bytes (members) processed per day

Query Performance in BigQuery

- Generally, queries that do less work perform better
- Input Data/Data Sources
    - Avoid SELECT *
    - Prune partitioned queries (for time-partitioned table, use PARTITIONTIME pseudo column to filter partitions)
    - Denormalize data (use nested and repeated fields)
    - Use external data sources appropriately
    - Avoid excessive wildcard tables
- SQL Anti-Patterns
    - **Avoid self-joins.**, use window functions (perform calculations across many table rows related to current row)
    - Partition/Skew: avoid unequally sized partitions, or when a value occurs more often than any other value
    - Cross-Join: avoid joins that generate more outputs than inputs (pre-aggregate data or use window function) Update/Insert Single Row/Column: avoid point-specific DML, **instead**

batch updates and inserts.

- Optimizing Query Computation
  - Avoid repeatedly transforming data via SQL queries
  - Avoid JavaScript user-defined functions
  - Use approximate aggregation functions (approx count) Approximate aggregate functions | BigQuery | Google Cloud
  - Order query operations to maximize performance. Use ORDER BY only in outermost query, push complex operations to end of the query.
  - For queries that join data from multiple tables, optimize join patterns. Start with the largest table.

Pricing in BigQuery

BigQuery offers a choice of two pricing models for running queries -

1. **On-demand pricing**: With this pricing model, you are **charged for the number of bytes processed** by each query. The first 1 TB of query data processed per month is free.
2. **Flat-rate pricing**. With this pricing model, you purchase slots, which are**virtual CPUs**. When you buy slots, you are buying dedicated processing capacity that you can use to run queries. Slots are available in the following commitment plans:
   - *Flex slots*: You commit to an initial 60 seconds.
     Pre-purchase query processing capacity in BigQuery at a discounted rate compared to on-demand pricing.
     -> For the on-demand pricing model, a maximum of 2000 concurrent slots are allowed per BigQuery project.
   - *Monthly*: You commit to an initial 30 days.
   - *Annual*: You commit to 365 days.

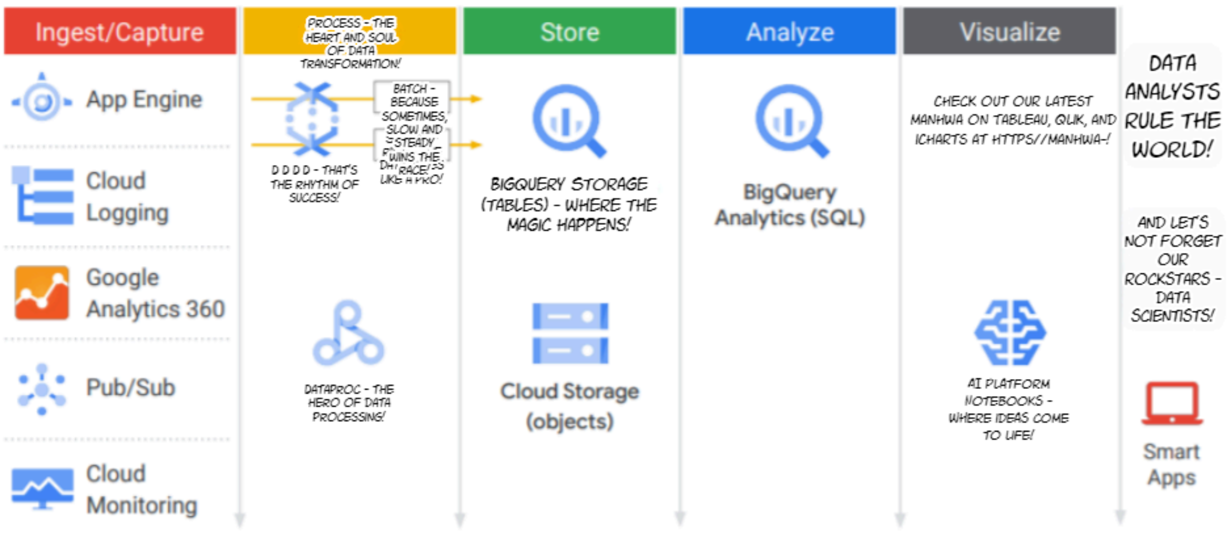
Dry Run

A dry run in BigQuery provides the following information:
estimate of charges in on-demand mode
validation of your query
approximate size and complexity of your query in capacity mode

```
bq query \
--use_legacy_sql=false \
--dry_run \
'SELECT
   COUNTRY,
   AIRPORT,
   IATA
 FROM
   `project_id`.dataset.airports
```

```
LIMIT
  1000'
```

Data processing solutions


Ingest/Capture
PROCESS - THE HEART AND SOUL OF DATA TRANSFORMATION!
Store
Analyze
Visualize
App Engine
Cloud Logging
Google Analytics 360
Pub/Sub
Cloud Monitoring
BATCH - BECAUSE SOMETIMES, SLOW AND STEADY WINS THE RACE!
LIKE A PRO!
D D D D - THAT'S THE RHYTHM OF SUCCESS!
DATAPROC - THE HERO OF DATA PROCESSING!
BIGQUERY STORAGE (TABLES) - WHERE THE MAGIC HAPPENS!
Cloud Storage (objects)
BigQuery Analytics (SQL)
CHECK OUT OUR LATEST MANHWA ON TABLEAU, QLIK, AND ICHARTS AT HTTPS//MANHWA-!
AI PLATFORM NOTEBOOKS - WHERE IDEAS COME TO LIFE!
DATA ANALYSTS RULE THE WORLD!
AND LET'S NOT FORGET OUR ROCKSTARS - DATA SCIENTISTS!
Smart Apps

Scaling streaming beyond BigQuery

- `Cloud Bigtable` : **Low latency/high-throughput**
  - 100,000 QPS at 6ms latency for a 10-node cluster
  - **More Cost-Efficient than** `Cloud Spanner` (~150 nodes)
- `BigQuery` : Easy, **inexpensive**
  - latency in order of seconds
  - 100k rows/second streaming