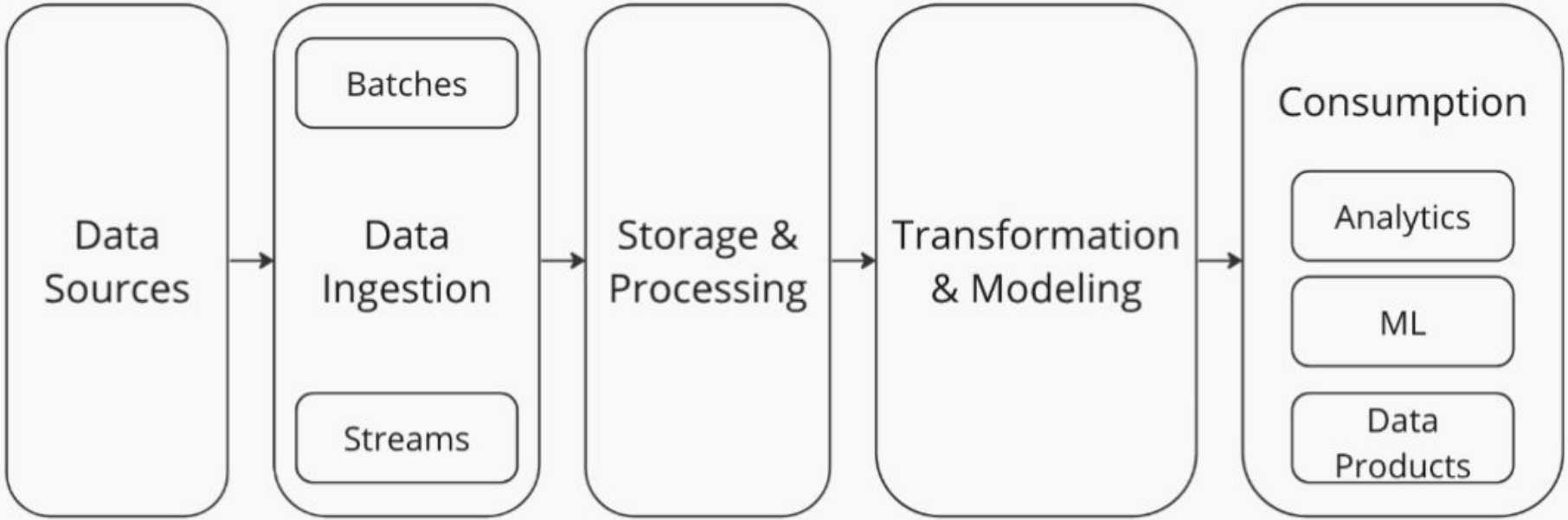
Case of troubleshooting

Inspiration


Data Sources
Data Ingestion
Batches
Streams
Storage & Processing
Transformation & Modeling
Consumption
Analytics
ML
Data Products

Dbt, the T in ELT

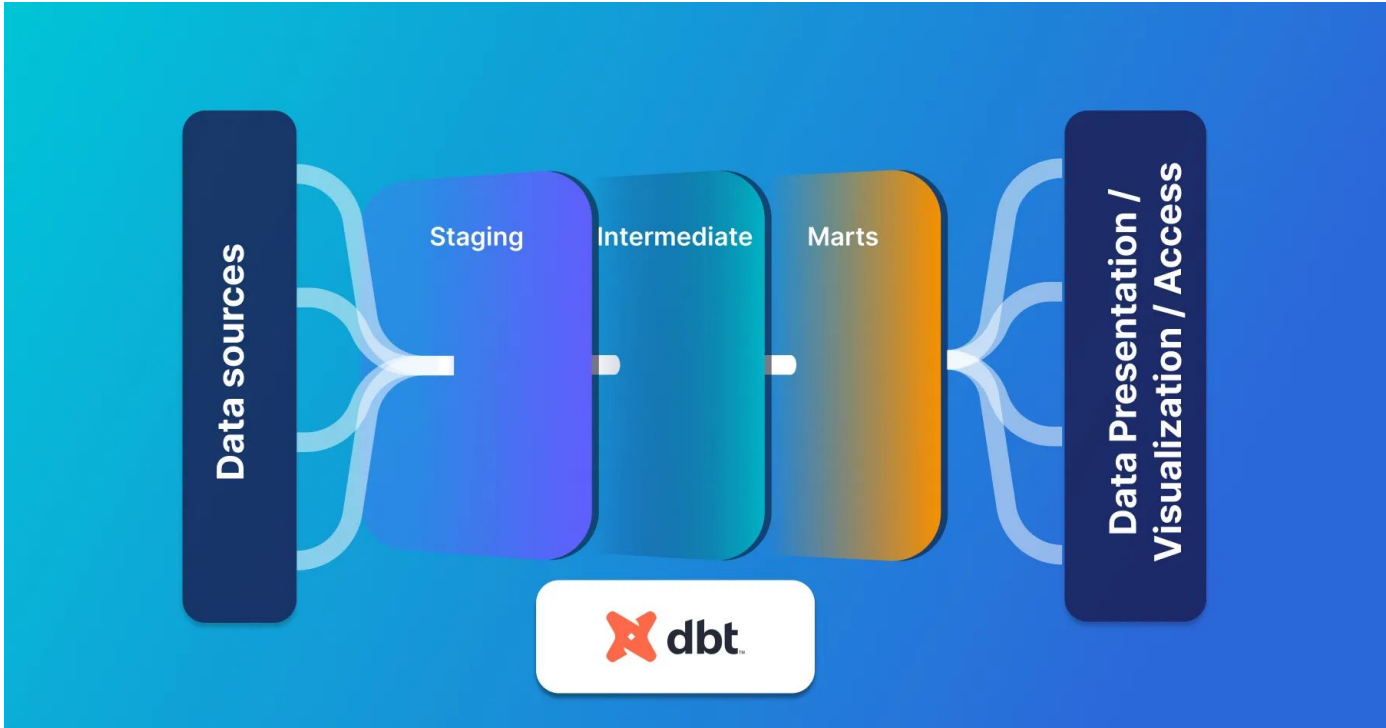
In an ELT pipeline, the raw data is loaded(EL) into the data warehouse. Then the raw data is transformed into usable tables, using SQL queries run on the data warehouse.

What is dbt

https://www.youtube.com/watch?v=8FZZivIfJVo&ab_channel=SeattleDataGuy

Key concepts

- models
- staging
- marts
- dim
- facts
- macros



Models

dbt models are SQL or Python files containing SQL `SELECT` statements that define transformations, acting as blueprints for creating new tables or views in a data warehouse.

Staging

Staging in dbt is the first transformation layer where you clean up and standardize your raw source data. Think of it as the "data prep" step before doing any business logic.

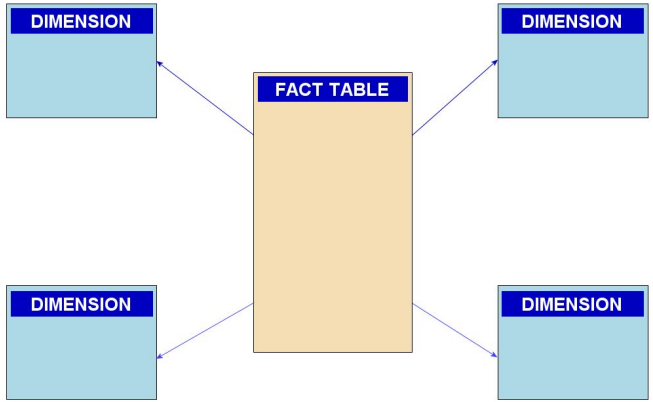
Only light cleanup: rename columns, cast types, standardize timestamps, trim text, de-duplicate obvious dupes

Marts = Business-ready data models

Think of marts as your "data products" - clean, well-organized tables that answer specific business questions.

Fact/Dimension tables = *types* of data structures

Marts = the *layer* where business-ready models live



Messy raw data with nested arrays?

“Exploding array into rows”

“CROSS JOIN UNNEST” awesome BQ function to help us “unpack” a json

Top-level array in the file → UNNEST once to get rows.

Arrays inside each row → UNNEST once per nested array level.

Plain objects with only primitives → no UNNEST.

Table vs view

Table = Actual Storage

A **table** physically stores data on disk. When you create a table, BigQuery actually writes the data and keeps it there.

Data is stored - takes up storage space (costs money)

Fast to query - data is right there, no computation needed

Static - data doesn't change unless you explicitly update it

Costs storage but queries are cheaper since no computation needed

View

A **view** is just a saved SQL query. No data is stored - it's like a "recipe" that runs every time someone queries it.

No data stored - it's just the SQL definition

Always fresh - runs the underlying query each time, so data is current

Can be slower - has to execute the full query every time

No storage cost but queries cost more (compute cost)

Quick rule of thumb: Staging = views, Marts = tables (usually).

Jinja

Jinja is a templating language that makes dbt models dynamic and reusable by allowing you to write code that generates SQL at compile time.

Variables: Use `{{ var('variable_name') }}` to make models configurable

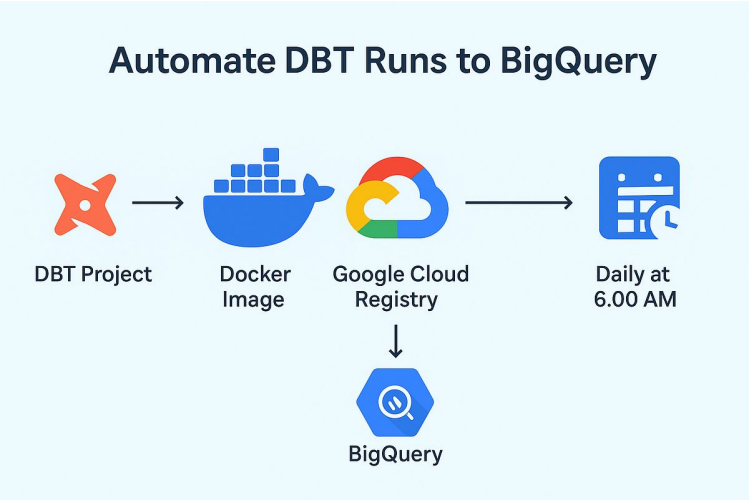
Macros: Create reusable SQL snippets with `{{ macro_name() }}`

Control structures: Use if/else logic and for loops to conditionally build SQL

Built-in functions: Access dbt's built-in functions like `{{ ref('model_name') }}` and `{{ source('schema', 'table') }}`

how to automate the runs?

dbt run in a docker container + cloud run jobs + scheduler



Idempotency/deduping

- Within a query

  You can deduplicate data in dbt by adding the `DISTINCT` keyword to your `SELECT` statements, using the `ROW_NUMBER()` window function within a Common Table Expression (CTE) and filtering for `row_number = 1`

- Macros (requires additional setup)

Are there any other way?

Yes, you can do all your transformation in python by loading the data from the BQ to the