End-to-End Data Pipeline & Analysis for Brazilian E-Commerce

Data Science and AI

By: Fabe, Jim, Wee Inn, Sheldon, Eugene, Jian Jin

20 June 2025, Friday

Agenda

- Challenges
- Our Strategy
- Solutions
- How it works
- Benefits

Challenges

Business Problem Statement

The core principles are to build a complete, ETL data pipeline that ensures data quality, models data using a star schema, and enables insightful analysis like Customer Lifetime Value across categories and regions:

1. E-commerce companies generate large volumes of raw data.
2. Lack of unified, clean, and validated data hampers decision making.
3. Business users struggle with fragmented, untrustworthy insights.

Our strategy

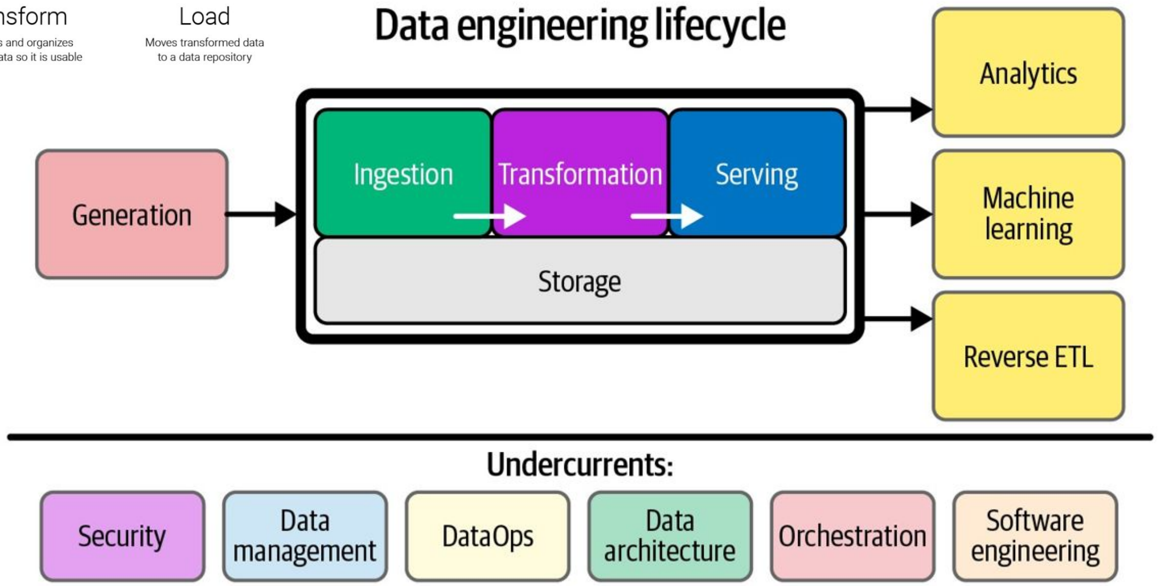
Project Objectives

1. Design and build an end-to-end data pipeline.
2. Implement a star schema data warehouse
3. Ecommerce Design Overview
4. Perform data quality checks and transformations.
5. Analyze CLV across product categories and regions.

Tools & Technologies Used

- VS Code (IDE)
- Meltano (Tools)
- dbt(Transformations)
- Data Quality Testing
- GCP Big Query (Data Warehouse)
- Dagster (Orchestration)
- Python, SQL

The ETL Process Explained



Customer Lifetime Value (CLV)

Pipeline: ETL – Extract, Transform, and Load (Extract from)

VS Code
Environment

Ingest
Meltano (Tools)

SQL/Python

Big Query
(GCP)

Data
ware
house

Data: CSV

Star

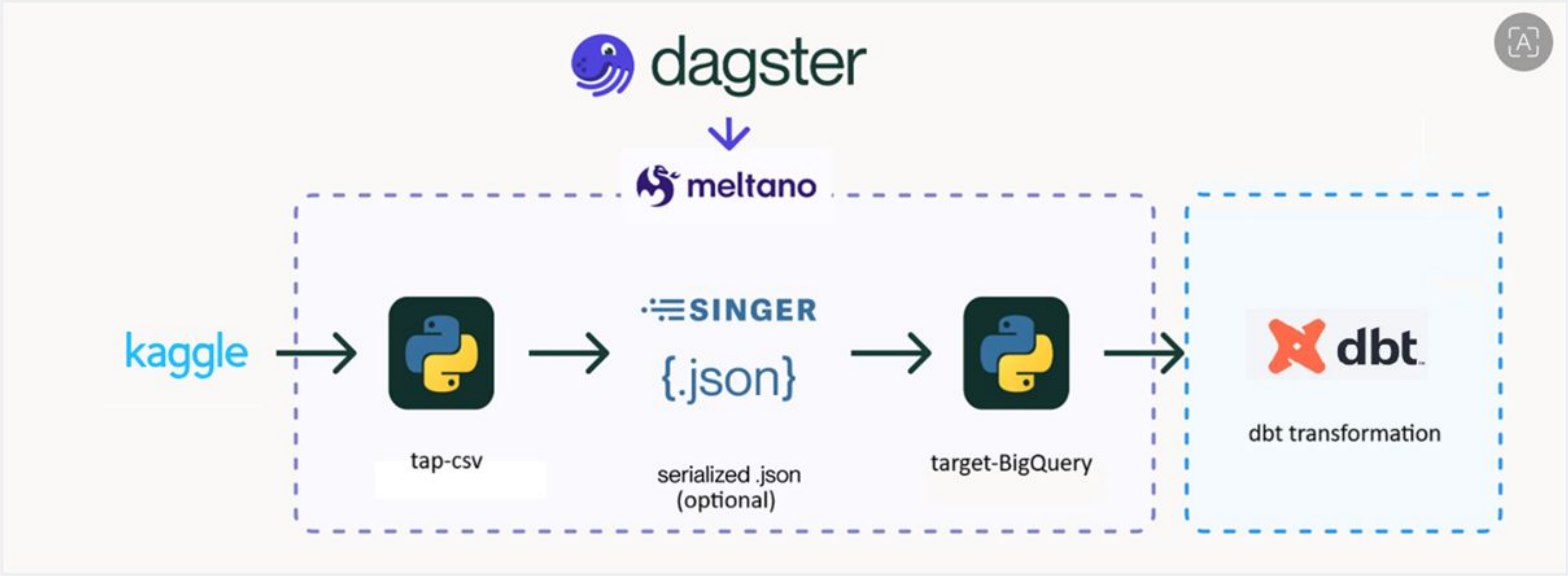
Automatic
Validation

Transformation
Dbt
Data Build Tool

Data Orchestrator - Dagster Platform

Challenges and Solutions

Building a ELT Pipeline



✅ Step 1: Set up your Meltano project

bash

```bash
meltano init my_csv_to_bq_project
cd my_csv_to_bq_project
```

✅ Step 2: Add necessary plugins

1. **Extractor:** `tap-csv` (for reading `.csv` files)
2. **Loader:** `target-bigquery` (for loading into BigQuery)

bash

```bash
# Add extractor for CSV files
meltano add extractor tap-csv

# Add loader for BigQuery
meltano add loader target-bigquery
```

✅ Step 3: Configure `tap-csv`

Create a directory for your CSV files, e.g., `./files/`.

Then configure `tap-csv` in `meltano.yml`:

yaml Copy Edit

```yaml
extractors:
- name: tap-csv
  variant: meltano
  config:
    files:
      - entity: my_data
        path: ./files/my_data.csv
        delimiter: ","
        encoding: utf-8
        keys: ["id"]  # replace with your unique identifier column(s)
        date_overrides: {}
```

Note: Delimiter, keys and data_overrides are optional.

✅ Step 4: Configure `target-bigquery`

You need a GCP service account key in JSON format. Save it as `bigquery_credentials.json`.

Then configure in `meltano.yml`:

yaml Copy Edit

```yaml
loaders:
- name: target-bigquery
  variant: transferwise
  config:
    project_id: your-gcp-project-id
    dataset_id: your_dataset_name
    credentials_path: bigquery_credentials.json
    location: asia-southeast1  # or your preferred location
```

Note: Make sure a GCP project for Meltano Ingestion & loading is created. Create an empty dataset in the GCP project in advance. Go to service accounts for the GCP project and create a JSON key for credentials_path. Keep the JSON key in the same project folder for easy reference.

✅ Step 5: Run the pipeline

bash

```bash
meltano run tap-csv target-bigquery
```

This command will:

- Read the CSV file(s)
- Load the data into BigQuery as a table named `my_data` (based on `entity` name)

Note: Do a quick manual check to ensure all rows of the 9 CSV files are successfully uploaded into BigQuery.

Dbt Staging

1 The importance of staging

-- > temporary view before transformation to final database

-- > ‘prep station’ for cleaning, organising or processing raw data without affecting the source or final system

2 7 Steps for the staging

3 Staging structure created

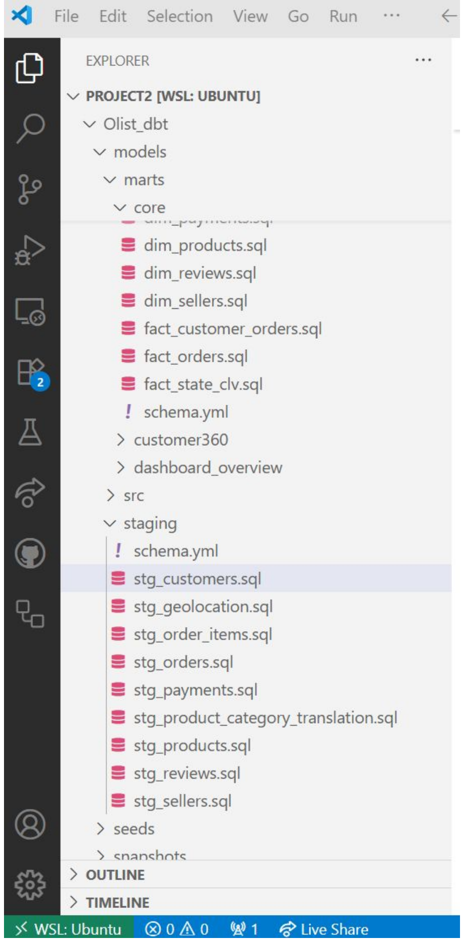
-- > from raw files -- > staging view -- > 9 staging view files

4 Staging Models

-- > refer to each staging .sql file

5 Perform Validation tests

-- > embedded validation tests into schema (in dimensions & fact)



Transforming Raw E-Commerce Data into **Strategic** Insights

✅ **Scalable transformation pipeline using dbt + BigQuery**
Modular SQL models with version control and in-warehouse execution

🧹 **Data cleaning and normalization in staging models**
Standardized timestamps, null handling, consistent naming

🏗️ **Modeled business entities using scalable star schema**
Fact table joined with clean, filterable dimension tables

🔍 **Built-in tests and validations with dbt-tests & dbt-expectations**
Enforced data integrity: nulls, uniqueness, referential links

📊 **Generated trusted, analytics-ready models (marts)**
Final curated models used directly by dashboards & direct analysis

📈 **Powered BI Tools & automated dashboards in Looker Studio**
Empowers business teams with **self-service analytics** powered by clean data

Scalable **Star Schema** Optimized for BI Tools & Deep Analysis

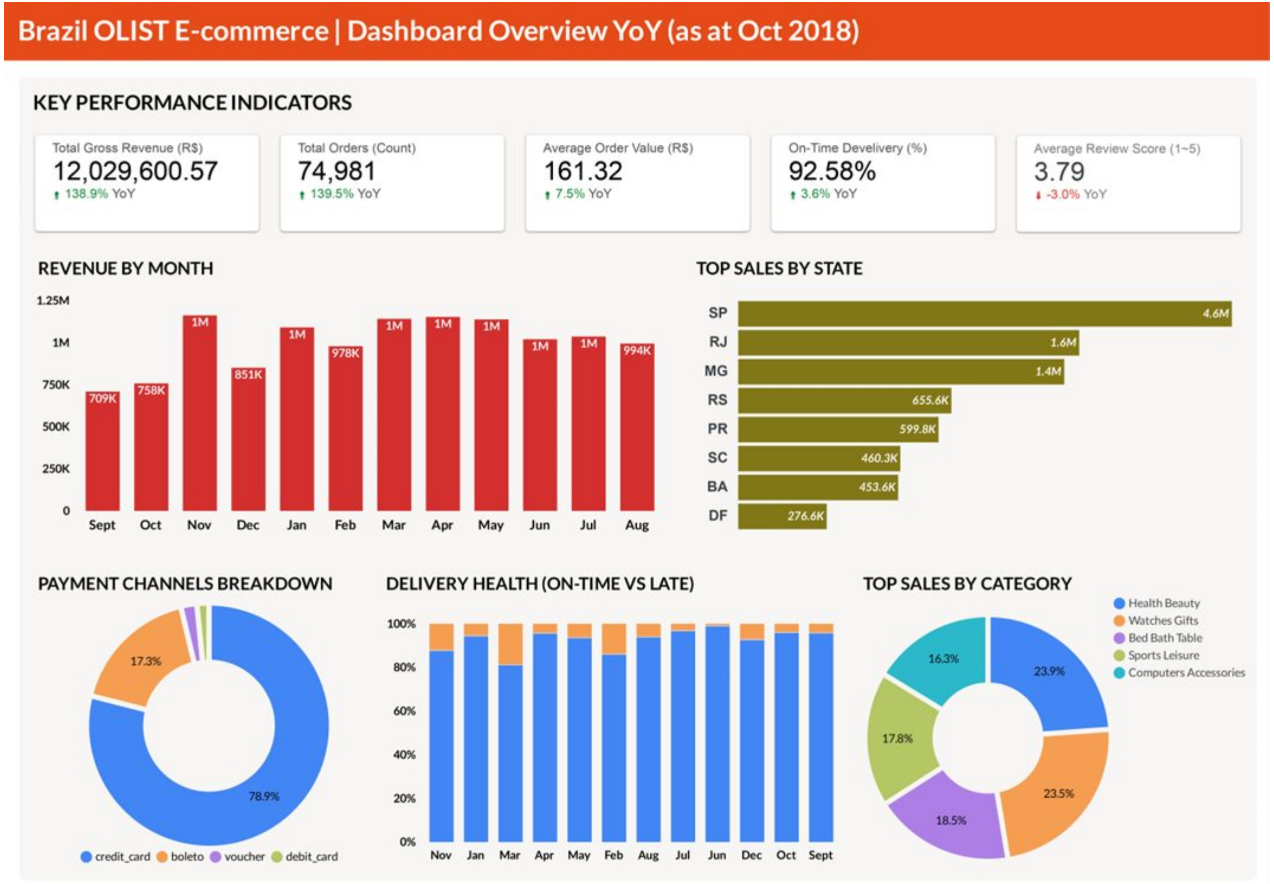
- Designed a **clear and performant** star schema to support analytics
- **Optimised** for dynamic business considerations of an E-Commerce marketplace
- Central **fact_orders** table connects to dim_customers, dim_products, dim_dates, dim_payments, dim_reviews, dim_sellers, dim_categories
- Enables **fast filtering** by: Time period, Product category, City/state, Customer type, Sellers, Reviews

→ Outcome: Enables BI Tools like **Looker Studio** to compute accurate KPIs like on-time delivery, AOV, CLV.

Scalable **Star Schema** Optimized for BI Tools & Deep Analysis

“When the stars are aligned, every dashboard tells the full story — instantly.”

Data Driven Dashboards — Driving **Real-Time** Business Decisions

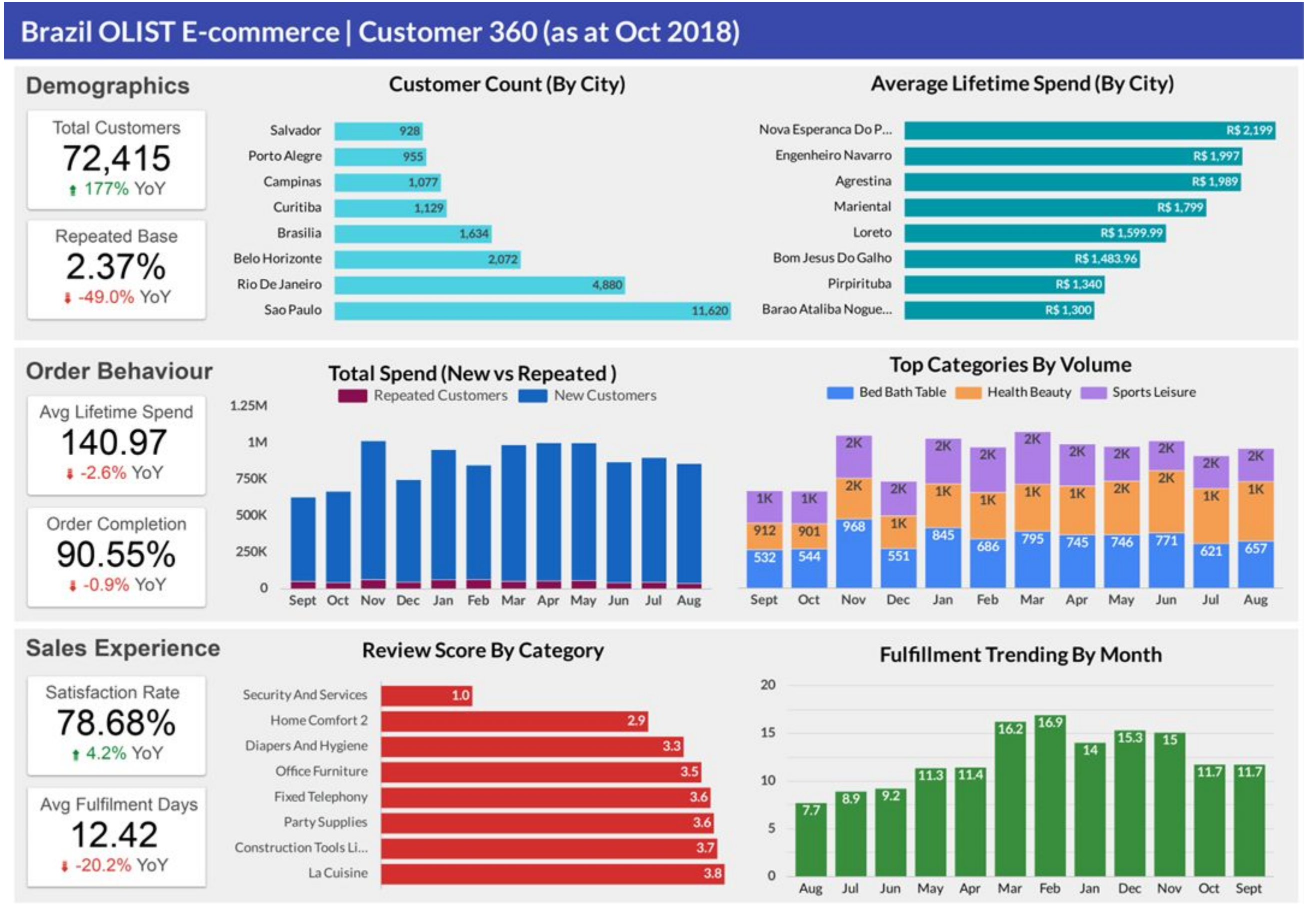


🛒 ***E-Commerce Overview***: Business-wide operational health

[Link: https://lookerstudio.google.com/reporting/d4ae5458-656c-4df2-8329-34a251bea08e](https://lookerstudio.google.com/reporting/d4ae5458-656c-4df2-8329-34a251bea08e)

- ➔ Business KPIs
- ➔ Growth Rate
- ➔ Payment Behaviour
- ➔ Fulfilment Health
- ➔ Trending Categories
- ➔ Geolocation Breakdown

Data Driven Dashboards — Driving **Real-Time** Business Decisions



👥 ***Customer 360*:**

Deep dive into customer behaviour and value

Link: https://lookerstudio.google.com/reporting/6bc1d14d-ad4d-4c80-8fdb-4185c7009335

- ➔ Geolocation Targeting
- ➔ Churn Rate
- ➔ Satisfaction Level
- ➔ Lifetime Value
- ➔ Delivery Performance
- ➔ Growth Stimulant

Automated ELT Pipeline with Dagster

Objective: Automate transformation and validation of Brazilian E-Commerce data

Stack:

- **Dagster** – Orchestration & scheduling
- **dbt** – SQL-based data transformations
- **BigQuery** – Cloud data warehouse
- **dbt tests** – Data quality (nulls, uniqueness)

Pipeline Flow:

1. `run_dbt_staging`: Clean raw data into staging models
2. `run_dbt_tests`: Validate staging outputs via schema.yml
3. `run_dbt_marts`: Build analytics-ready star schema

Scheduling, Monitoring & Benefits

Daily Automation:

- Scheduled at 9:00 AM (SGT) via Dagster's ScheduleDefinition
- Dagster UI shows logs, lineage, and failure tracing

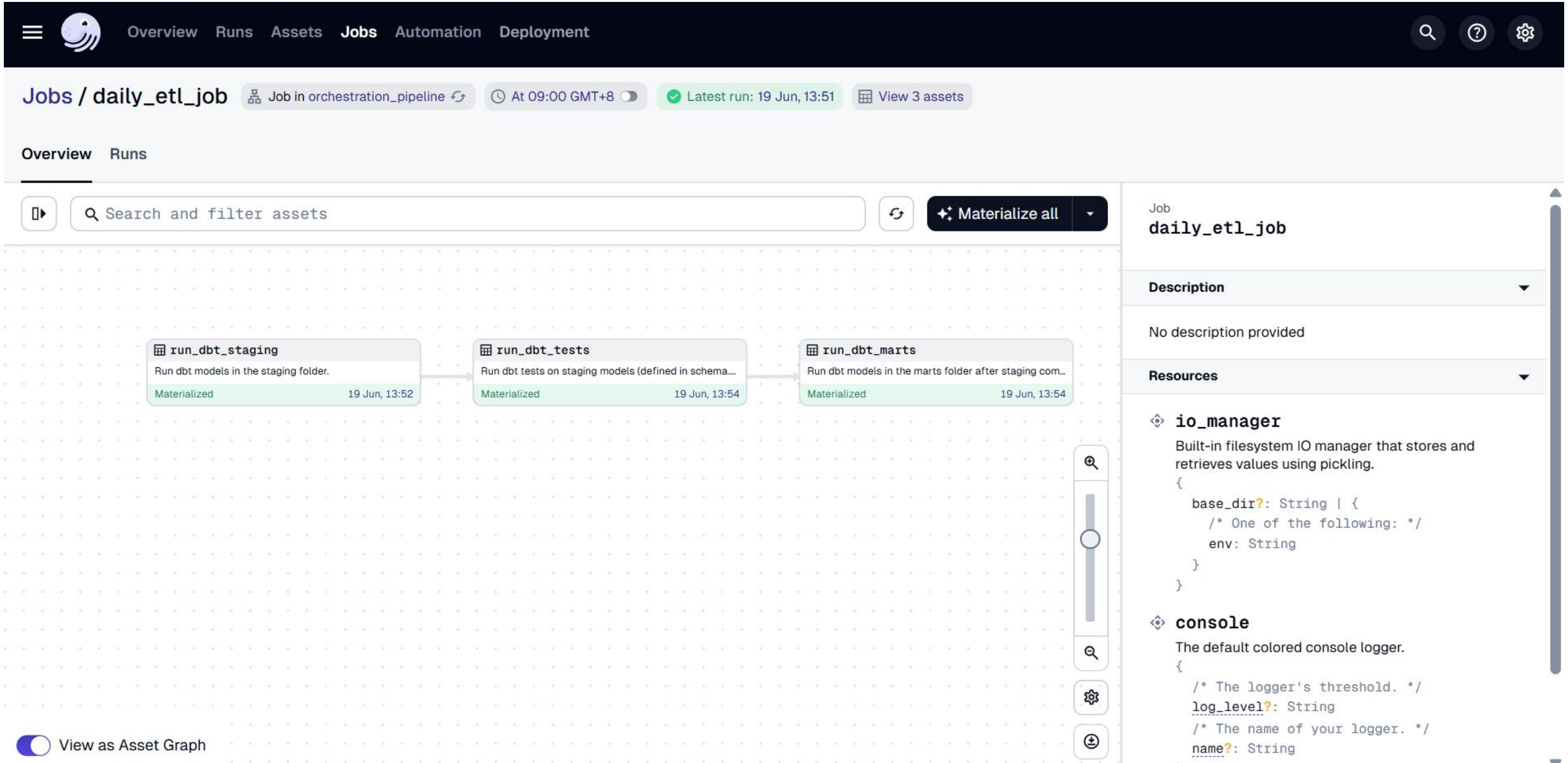
Failure Handling:

- Pipeline halts if any dbt test fails
- Logs show exact failed model and column

Key Benefits:

- Zero manual intervention after setup
- Fast issue visibility via Dagster UI
- Clean modular design, easy to extend or debug

Scheduling, Monitoring & Benefits



Data Analysis with Python

EDA or exploratory data analysis, using pandas:

One of the tables from BQ:

- States, Customers, 4 ratios

Customer Lifetime Value (CLV), a business metric

- Estimates total revenue a customer will generate over ‘a lifetime’
- A higher CLV value means **more valuable** over time

To Calculate CLV?

- *$100 / purchase*
- *12 purchases / year*
- *5-year lifespan*

Customer Lifetime Value (CLV) is $6,000 = revenue for the business

Data Visualization

… a Jupyter notebook demo

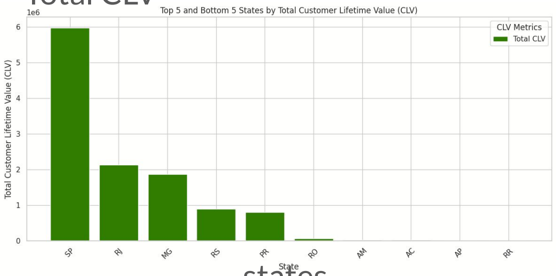
```python
df.head()
```
✓ 0.0s

	state	total_customers	total_clv	avg_clv	max_clv	min_clv
0	PB	517	140441.490000000	271.646982592	4681.780000000	29.740000000
1	RO	241	60349.150000000	250.411410788	2452.120000000	34.340000000
2	AL	399	96907.670000000	242.876365915	2269.980000000	32.390000000
3	AP	67	16262.800000000	242.728358209	1482.420000000	34.800000000
4	PA	943	217369.800000000	230.508801697	4042.740000000	26.130000000

```python
df.describe()
```
✓ 0.0s

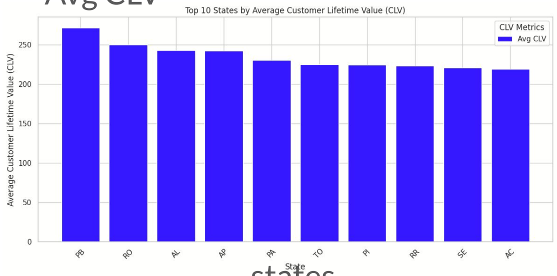
	total_customers	total_clv	avg_clv	max_clv	min_clv
count	27.0	2.700000e+01	27.000000	27.000000	27.000000
mean	3545.518519	5.898910e+05	203.813704	3922.470741	25.038889
std	7964.860414	1.197894e+06	31.028133	2586.067883	7.742155
min	45.0	1.006462e+04	148.690000	994.770000	9.590000
25%	369.0	8.589804e+04	173.680000	2252.320000	19.345000
50%	869.0	1.856102e+05	209.860000	3242.840000	25.430000
75%	2611.5	4.822725e+05	224.250000	4668.845000	31.410000
max	40208.0	5.978630e+06	271.650000	13664.080000	39.030000

Total CLV



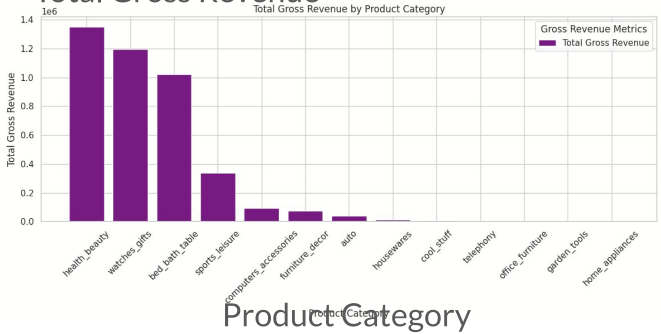
states

Avg CLV

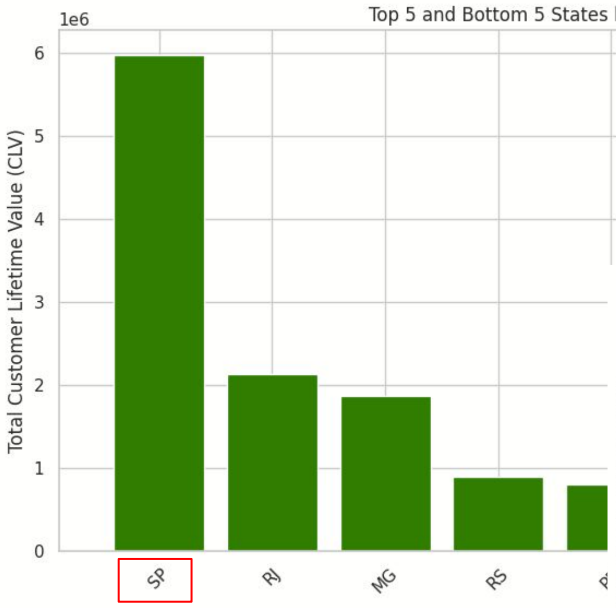


states

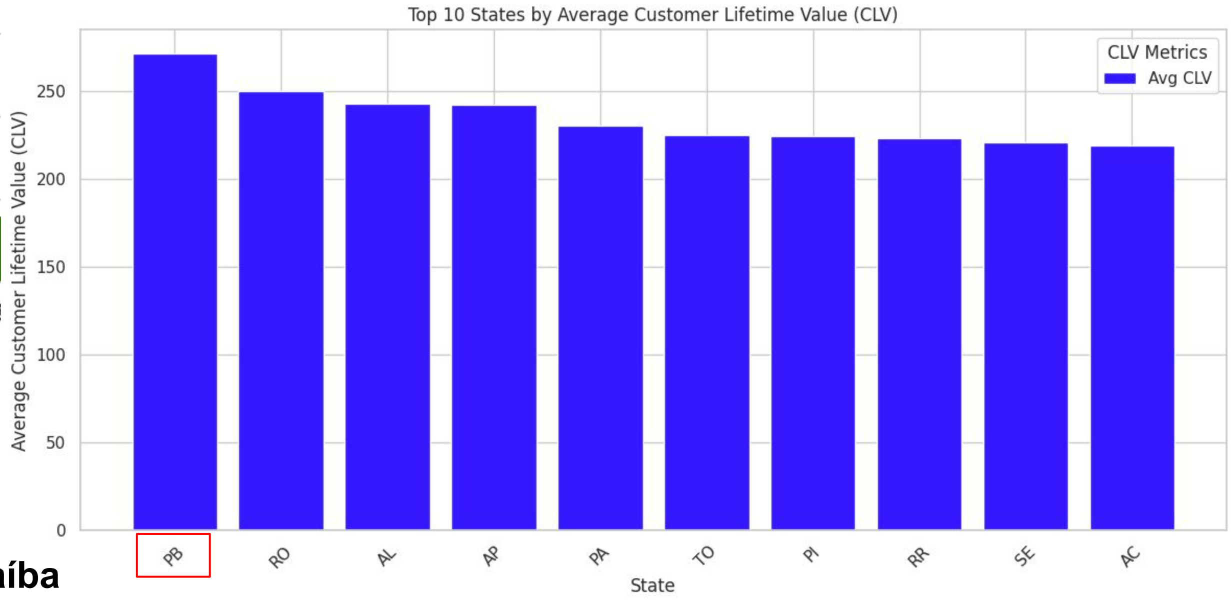
Total Gross Revenue



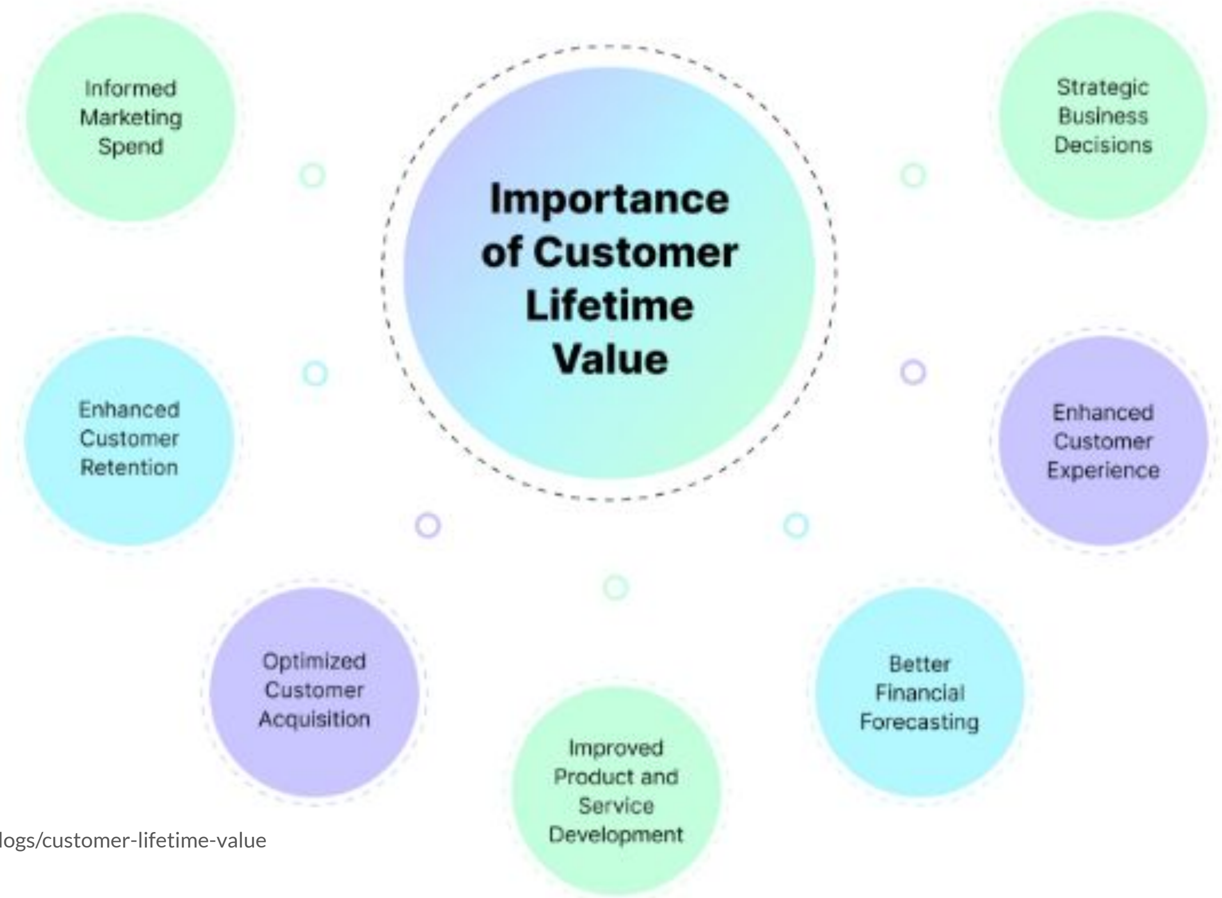
Product Category



State of São Paulo



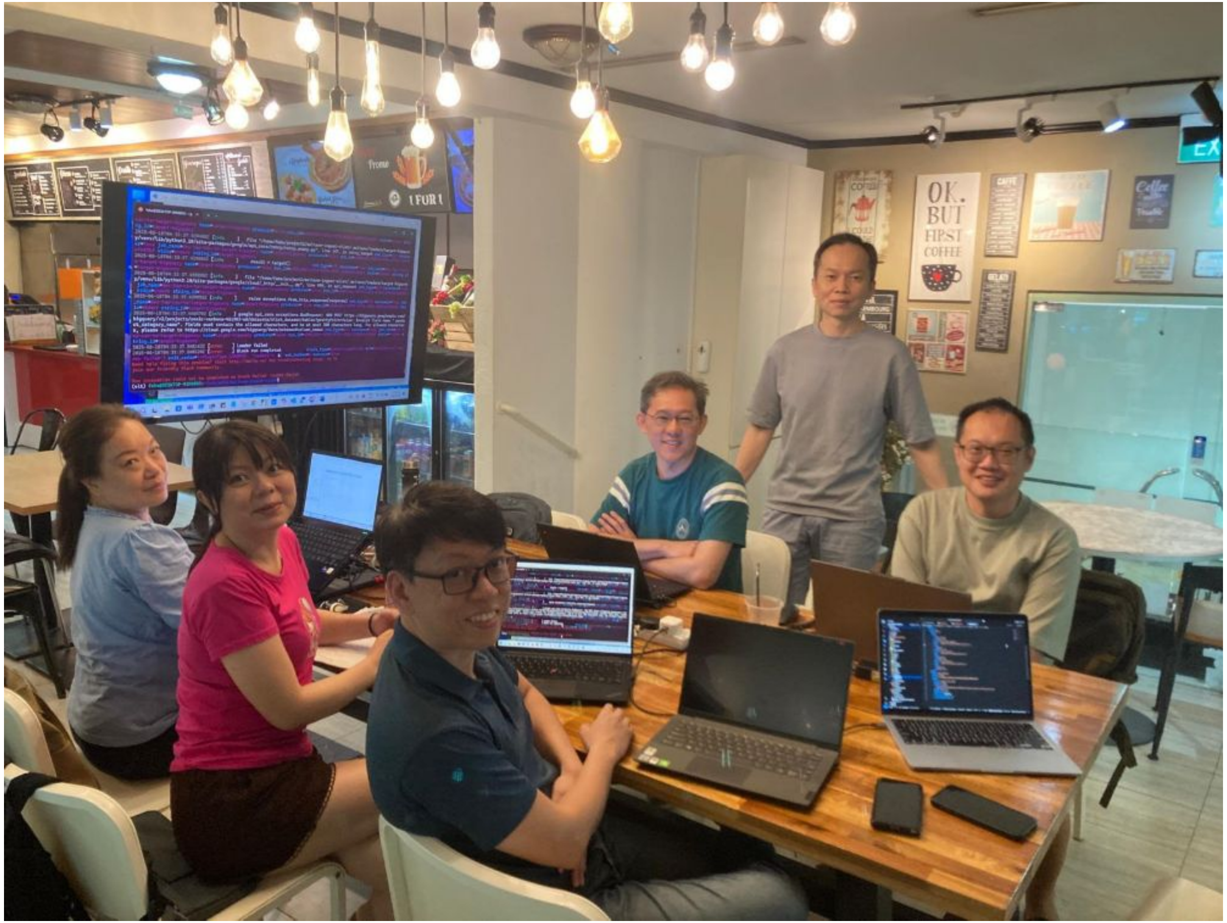
State of Paraíba



Source: https://www.entropik.io/blogs/customer-lifetime-value

Confidential

Our Communications!



Questions?

GitHub: https://github.com/fabel99/project2

Thank you!