2.2: Justify the choice of the storage system:

**Google Cloud Storage (GCS)**: Stores the raw Olist CSV files.

**BigQuery**: Serves as the data warehouse.

**Airflow**: Schedules and orchestrates the loading and transformation steps.

**dbt**: Handles SQL-based transformation, testing, and documentation.

**1. Google Cloud Storage (GCS) for Raw Data Ingestion**

* GCS is a highly scalable, durable **object storage system**
* **Justification**:
  + GCS is suitable for storing the **raw CSV files** from Kaggle (object format).
  + It supports **batch ingestion** and is directly accessible by downstream tools (BigQuery).
  + Its **flat structure** with metadata support allows efficient organization by topic or date (/customers, /orders, etc.).

**2. BigQuery for Querying and Analytics**

* **Justification**:
  + Supports **SQL over large datasets**—perfect for Olist’s relational structure.
  + Automatically **scales compute independently** of storage: ideal for both light queries and heavy analytics
  + **No infrastructure management** needed → quick setup and experimentation.

**3. Security & Governance**

* GCP offers centralized IAM, encryption, and data governance
* **Justification**:
  + With multiple stakeholders (e.g., data engineers, analysts), Olist pipelines can define **role-based access**.
  + All data can be **encrypted in transit and at rest**, satisfying enterprise security expectations.
  + Tools like **Google Data Catalog** can enhance metadata and lineage tracking for the many interrelated tables such as our case with orders, sellers, geolocation, etc

**4. Cost Efficiency:**

* + With **pay-per-query** in BigQuery and **low-cost storage classes** in GCS, you only pay for what you use
  + BigQuery’s **automatic scaling and no idle costs** align well with the variable workloads common in exploratory analytics

**Apache Airflow – for Orchestration**

* A real-world e-commerce system generates **large volumes of data** continuously: orders, reviews, customer actions, logistics, payments, etc.
* **Airflow** provides:
  + **Scheduled workflows** (e.g., run ETL pipelines every hour or day).
  + **Task dependencies** (e.g., load customers → then orders → then order items).
  + **Monitoring and retries** for failures.
* Without Airflow, we’d have **manual or script-based workflows**, which quickly become unmanageable at scale.

**dbt – for Transformations**

* Once raw data lands in the data warehouse (like BigQuery), it needs to be cleaned, joined, and structured for analysis.
* **dbt** enables:
  + **Modular SQL transformations** (with Jinja templating).
  + **Version control and documentation**.
  + **Automated dependency resolution** between models.
  + **Materialization strategies** (views, tables, incremental models) to optimize cost and performance.
* This is crucial for **maintaining data quality** and enabling **self-service analytics** in a fast-moving e-commerce environment.

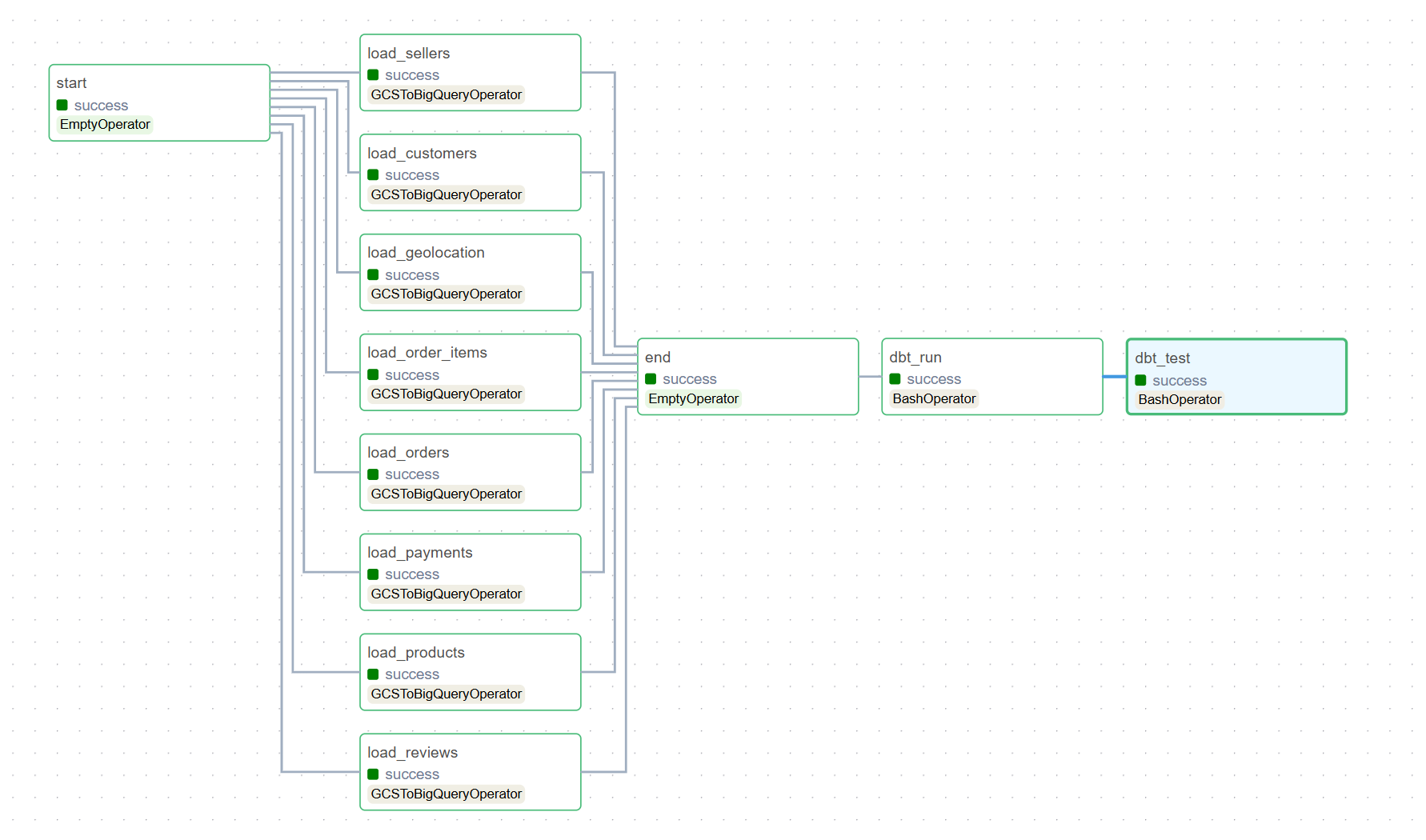


Figure 1: Pipeline Structure, loading, transforming, and testing data

| **Step** | **Task** | **Description** |
| --- | --- | --- |
| start | EmptyOperator | Dummy task to mark the beginning of the pipeline. |
| load\_\* | GCSToBigQueryOperator | Loads each CSV file from GCS into a corresponding BigQuery **staging table**: |
| end | EmptyOperator | Marks the end of the loading phase. |
| dbt\_run | BashOperator | Executes dbt run to transform staging tables into dimensional/fact models. |
| dbt\_test | BashOperator | Executes dbt test to validate data quality using assertions and expectations. |

Justification: A realistic e-commerce company generates **continuous, high-volume data**. Tools like Airflow and dbt are critical to:

* **Automate repetitive workflows** (no manual loading).
* **Ensure data integrity** (via dbt tests).
* **Enable modular, maintainable pipelines** (with clear separation of extraction, staging, and transformation).

This setup guarantees **scalability**, **reliability**, and **developer productivity**, especially when working across dozens of interlinked tables like those in the Olist dataset.