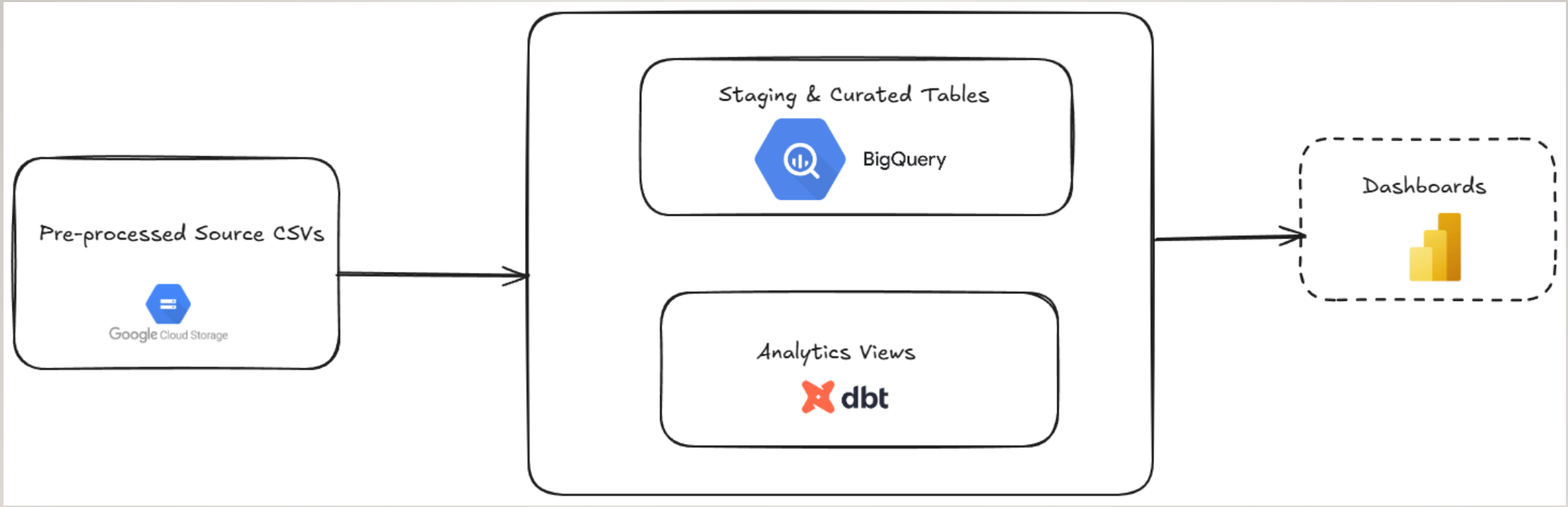
AMAZON REVIEWS DWH- JUST EAT

PRIYANKA KUPPUSWAMY

HIGH LEVEL ARCHITECTURE/TOOLS



EDA ON REVIEWS DATA

- reviewerID: Unique ID for each reviewer.
- asin: Amazon Standard Identification Number (product ID).
- reviewerName: Name of the reviewer (469 missing values).
- helpful: A list representing the helpfulness votes (e.g., [0,0] for no votes).
- reviewText: Full text of the review (24 missing values).
- overall: Rating given by the user (float, typically from 1 to 5).
- summary: Short review summary.
- unixReviewTime: Unix timestamp of the review.
- reviewTime: Formatted date of the review.

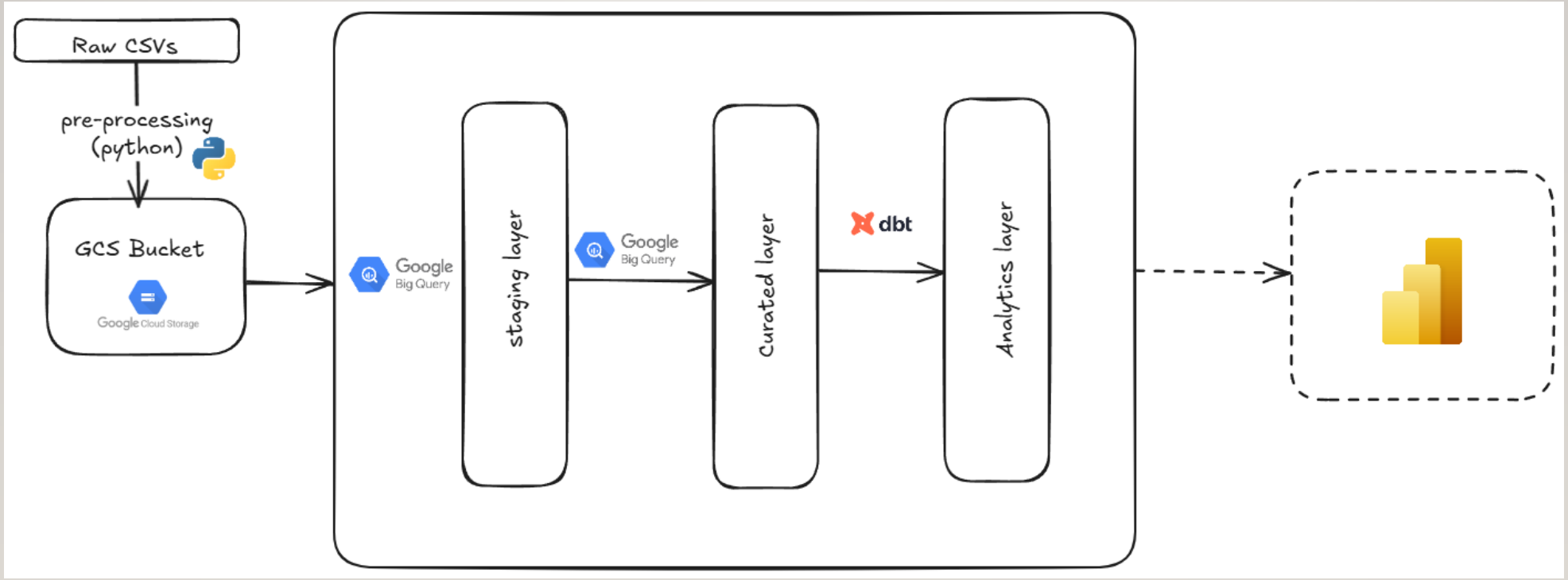
EDA ON METADATA

- **metadataid**: Unique identifier for each product metadata entry.
- **asin**: Amazon Standard Identification Number (product ID).
- **salesrank**: Dictionary-like structure storing the product's sales rank in categories.
- **imurl**: URL of the product image.
- **categories**: Category of the product (e.g., Clothing, Shoes & Jewelry).
- **title**: Product title (23 missing values).
- **description**: Product description.
- **price**: Product price.
- **related**: JSON-like structure listing related products (124 missing)
- **brand**: Product brand.

DATA QUALITY ISSUES - SOURCE

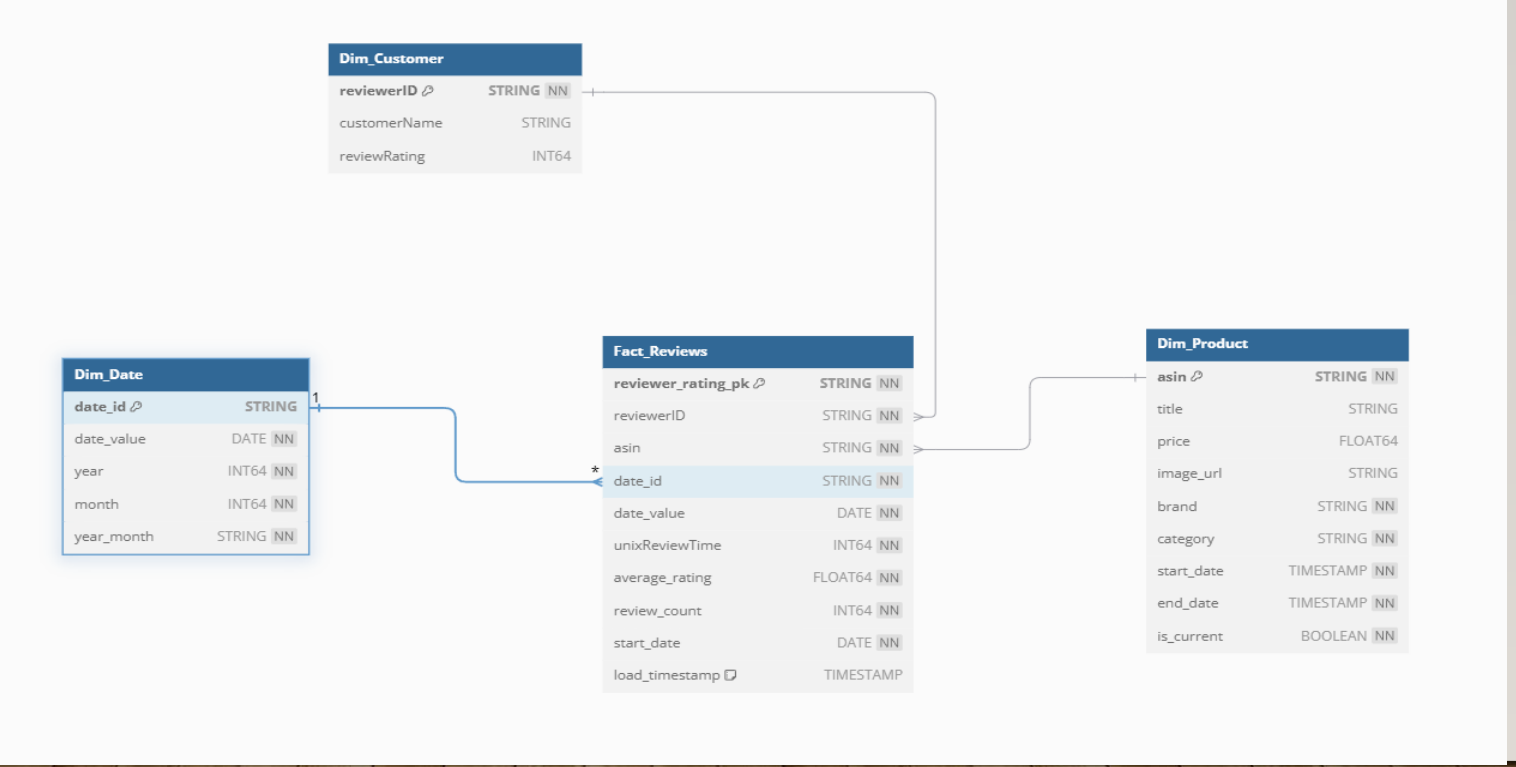
- **Malformed Rows:** Some rows did not have all columns, due to missing values or improper formatting(eg special characters: #).
- **Unescaped Quotes:** If a field contains unescaped quotes (“) or line breaks, it might cause misalignment. So, I escape the quotes (\”)
- **Delimiter Issues:** If a field contains a comma but is not enclosed in quotes so it is splitting into multiple columns incorrectly. So (,) inside fields was replaced with (;)
- **Preprocessing python scripts** help to fix these issues to successfully load in BigQuery
- Note : Most formatting problems are in the Description column of metadata table.

ETL SOLUTION DESIGN



ER DIAGRAM

- Dim_customer is part of the ETL Solution Design but **not implemented** in Data Warehouse.

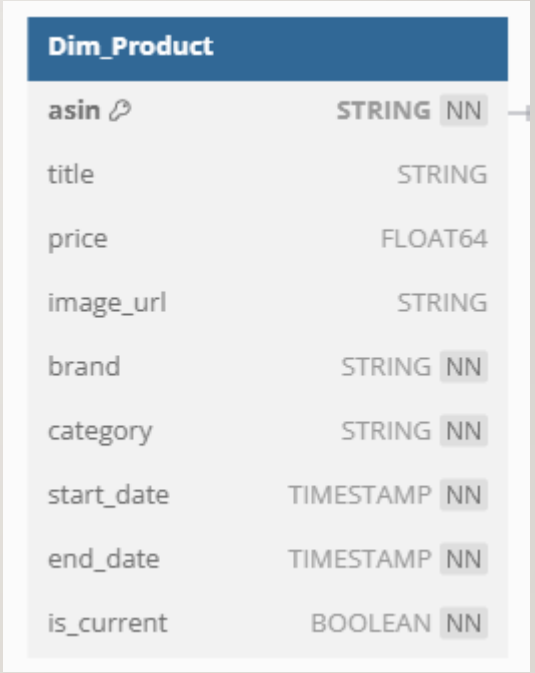


STAGING TABLES

- **Optimized CSV Integration:** configure parsing parameters (skip_leading_rows, field_delimiter, quote, encoding) to ensure accurate and efficient data ingestion.
- **Missing Data Management:** Use COALESCE and NULLIF functions to handle null or empty values, maintaining data consistency
- **Data Type Casting:** Apply CAST functions to enforce correct data types, enhancing data integrity.
- **Date Standardization:** Convert string dates into DATE format with PARSE_DATE, ensuring uniformity across datasets.

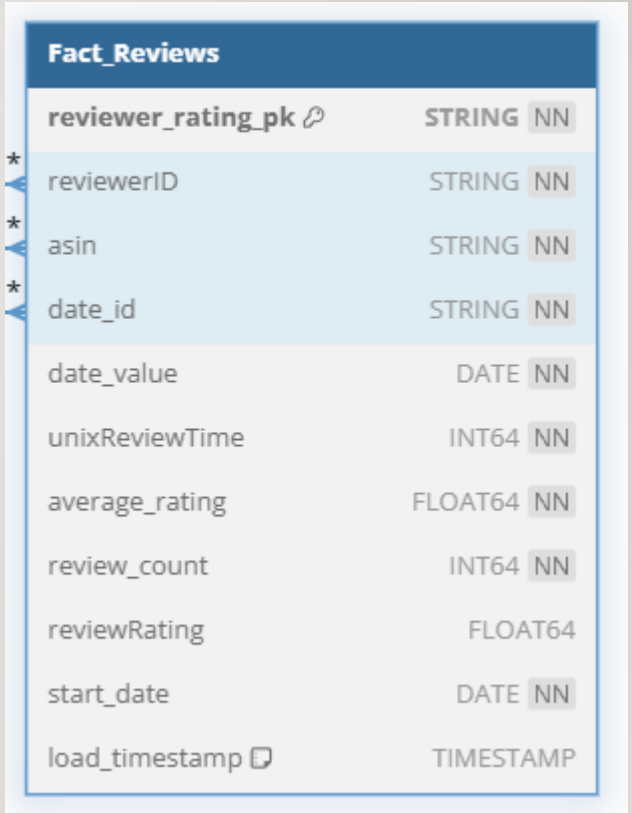
DIM TABLES – SCD TYPE 2

- **Historical Tracking:** Utilized a MERGE statement to handle updates and insertions (upsert), ensuring changes in product attributes are captured over time.
- **Record Updates:** Set end_date to the current timestamp and is_current to FALSE for existing records when changes are detected.
- **New Record Insertions:** Added new records with start_date as the current timestamp and end_date set to a distant future timestamp (9999-12-31 23:59:59 UTC), marking them as current (is_current = TRUE).

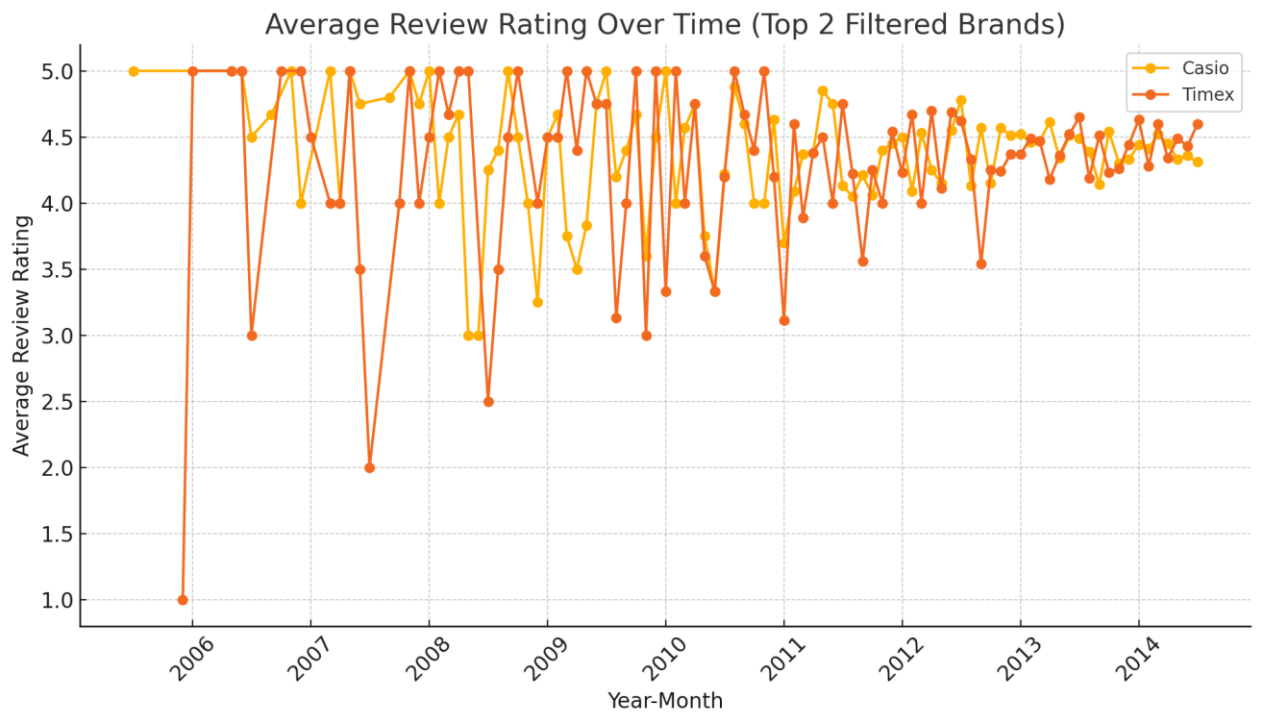


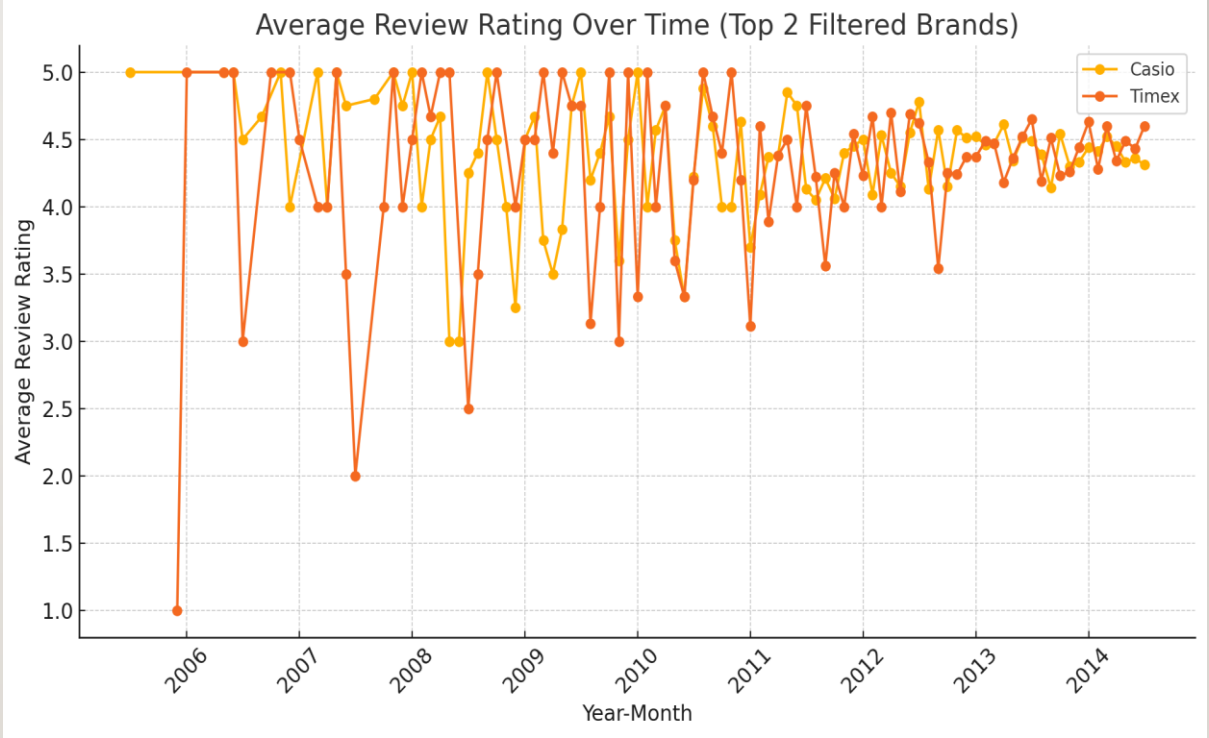
FACT TABLE

- **Composite Primary Key:** Generated a unique reviewer_rating_pk using a hash of key fields to accurately identify and merge records.
- **Review Metrics Calculation:** Computed average_rating and review_count for each product on a given date
- **Date Dimension Integration:** Joined with the Dim_Date table to enrich review data with standardized date attributes, supporting consistent time-based analysis.



ANALYTICS VIEWS





DATA QUALITY CHECKS - IMPLEMENTED

- Total Record Count Check
- Null Value Check
- Duplicate Check
- MINUS Check (Data Mismatch Between External & Staged)
- Check for Default Value Assignments
- Referential Integrity
- Data Range Check

MERGE – CODE QUALITY

- Better readability and code maintainability
- MERGE ensures INSERT, UPDATE, and DELETE operations into a single action

OPTIMIZATION TECHNIQUES

Considerations for High-Volume, High-Frequency Data Loads:

- **Partitioning and Clustering:** Implemented partitioning by date_value and clustering by asin to enhance query performance in BigQuery.
- **Staging Tables:** Load new data into staging tables and then use selective queries to update or append only the necessary data.

WHAT COULD BE BETTER & FUTURE STEPS

- Create a Dim_Customer table to hold customer information & reviewRating
- Implement staging & curated layers in DBT (Currently only analytics views are built in BigQuery using DBT)
- Scheduling jobs using cron scripts

DEMO

THANK YOU!