A blue and black text

Description automatically generated

**Data Lake and Data Warehouse**

Year 2 (2025/26), Semester 4

## SCHOOL OF INFOCOMM TECHNOLOGY

Diploma in Data Science

**ASSIGNMENT**

**Part A**

|  |  |
| --- | --- |
| Module Group: | P01 |
| Team No: | Team 1 |
| Student Leader Name: | Ang Shao En |
| Student 2 Name: | Jayden Tee Rong Qing |
| Student 3 Name: | Tan Yu Xuan Eason |
| Student 4 Name: | Heng Jue Wei Tevin |
| Student 5 Name: | Ho Wei Xian |

Contents

[SCHOOL OF INFOCOMM TECHNOLOGY 1](#_Toc690374060)

[Scenario Overview 3](#_Toc1735854322)

[Scenario Description 3](#_Toc284963731)

[Purpose of the JSON files 3](#_Toc1654866168)

[Relationship Between Master Data and Transactional Data 5](#_Toc2039103539)

[JSON Data Files 6](#_Toc2060346561)

[Product Master (product\_master\_part01/02.json) 6](#_Toc1010096135)

[Sales Transactions (sales\_transactions\_part01/02.json) 13](#_Toc725397321)

[Customer Reviews (customer\_reviews\_part01–05.json) 21](#_Toc1983913308)

[Media Metadata (media\_metadata\_part01.json) 27](#_Toc1022226693)

[Returns / Issues (returns\_issues\_part01.json) 31](#_Toc1534550017)

[Proof of uploaded files 37](#_Toc776911636)

[i) GROUP1\_IMAGES\_STAGE 37](#_Toc1467331231)

[ii) GROUP1\_JSON\_STAGE 38](#_Toc1346686569)

[iii) GROUP1\_VIDEOS\_STAGE 39](#_Toc155691877)

[Proof of file formats 40](#_Toc40176018)

[File Compliance Table 41](#_Toc1988202634)

[Scripts used 42](#_Toc1399024047)

[Part 0- Session Setup and Working Context 42](#_Toc249838399)

[Part 1 - JSON File Format Definition (FF\_JSON) 42](#_Toc434163269)

[Part 2 - Enable Directory Listing for Media Stages 44](#_Toc1292562438)

[Part 3 - RAW Landing Tables (VARIANT Storage) 45](#_Toc787864373)

[Part 4 - Load JSON Files into RAW Tables (COPY INTO + PATTERN) 46](#_Toc1055250994)

[Part 5- RAW Load Validation Checks 48](#_Toc1597224812)

[Part 6- Stage Directory Tables and Scoped Media URLs 50](#_Toc559365153)

[Part 7 - Transform RAW JSON into Relational Staging Tables (STG\_) 51](#_Toc1703447196)

[Part 7- Build Final Curated Dataset (Review-Grain Integration) 58](#_Toc358076890)

[Part 8 - Curated Dataset Validation and Profiling 61](#_Toc1414627726)

[Final curated dataset 66](#_Toc1304988454)

[Sample Analytical Queries on the Final Curated Dataset 68](#_Toc1293828085)

[Data Lineage Diagram 71](#_Toc1207175475)

[Summary 72](#_Toc837060526)

# Scenario Overview

## Scenario Description

In Scenario A, the objective is to design and build a curated analytical dataset in Snowflake by integrating semi-structured JSON data with supporting media assets like images and videos. The scenario simulates a real retail environment where stakeholders need a single, reliable dataset to understand product performance and customer experience. In practice, business data is often fragmented across different sources: product reference data lives in “master” files, while operational events such as sales, reviews, media submissions, and returns are captured as transactional logs.

This assignment addresses the challenge of transforming nested JSON into structured tables and then integrating these tables into a curated dataset that is analysis ready. The final output supports common analytical questions such as:

* Which products generate higher revenue but receive weaker customer ratings?
* Do reviews that include photo/video evidence provide stronger insight into product quality issues?
* Are certain categories or models associated with higher return counts or recurring return reasons?
* How does post-purchase feedback relate to sales performance?

To achieve this, the solution implements a standard layered data pipeline:

(1) upload all raw JSON/media to stages for traceability

(2) load JSON into RAW tables as VARIANT to preserve the original structure

(3) flatten and standardise into STG tables for relational querying

(4) build a curated dataset that integrates master and transactional domains at a clearly defined grain.

## Purpose of the JSON files

Although the dataset is delivered in multiple “part” files, they represent four main data domains. Each domain contributes unique information to the final curated dataset.

**(1) Product Master (product\_master\_part01/02.json)**

**Purpose:**

Provides the authoritative product reference details used to enrich all other events. This includes identifiers and descriptive attributes such as product\_id, brand/manufacturer, model name, category/subcategory, and pricing fields like base price and currency.

**How it is used:**

The product master is transformed into a product dimension staging table and joined into the curated dataset using product\_id. This ensures every review and transaction can be analysed with consistent product context.

**(2) Sales Transactions (sales\_transactions\_part01/02.json)**

**Purpose:**

Captures purchase activity, including order headers and order line items that reference products purchased. The key measures include quantity, unit price, and line total.

**How it is used:**

Sales transactions are flattened into a line-item staging table, one row per transactional item. Metrics such as units\_sold and total\_revenue are aggregated by product\_id and attached to the curated dataset. This allows performance analysis of products alongside customer satisfaction signals.

**(3) Customer Reviews (customer\_reviews\_part01–05.json)**

**Purpose:**

Represents the voice of customer through review events. Each review contains rating, review date, title, full review text, reviewer identifier, and linking keys such as product\_id and feedback\_id.

**How it is used:**

Reviews define the primary analytical grain for the curated dataset. Each review event is parsed into a staging table and becomes a single row in the curated dataset, one row per review\_event\_id. This makes the curated dataset directly usable for sentiment/feedback and customer experience analysis.

**(4) Media Metadata + Returns/Issues (media\_metadata\_part01.json, returns\_issues\_part01.json)**

**Purpose:**

* **Media metadata** links review feedback to uploaded photos/videos through file information and identifiers including feedback\_id.
* **Returns/issues** captures post-purchase events and problems such as return counts and return reasons, typically nested in JSON structures.

**How it is used:**

Media metadata is used to connect reviews to staged images/videos, enabling “evidence-backed” feedback analysis. Returns/issues is flattened and aggregated to provide return signals that can be compared against product sales and review ratings.

## Relationship Between Master Data and Transactional Data

This scenario uses a classic master-transaction relationship model:

* **Master Data (Product Master):**

Provides stable descriptive attributes that define products. The primary key is product\_id.

* **Transactional/Event Data (Sales, Reviews, Media, Returns):**

Captures events that occur over time and reference the product master through product\_id and/or feedback\_id.

Key relationships implemented:

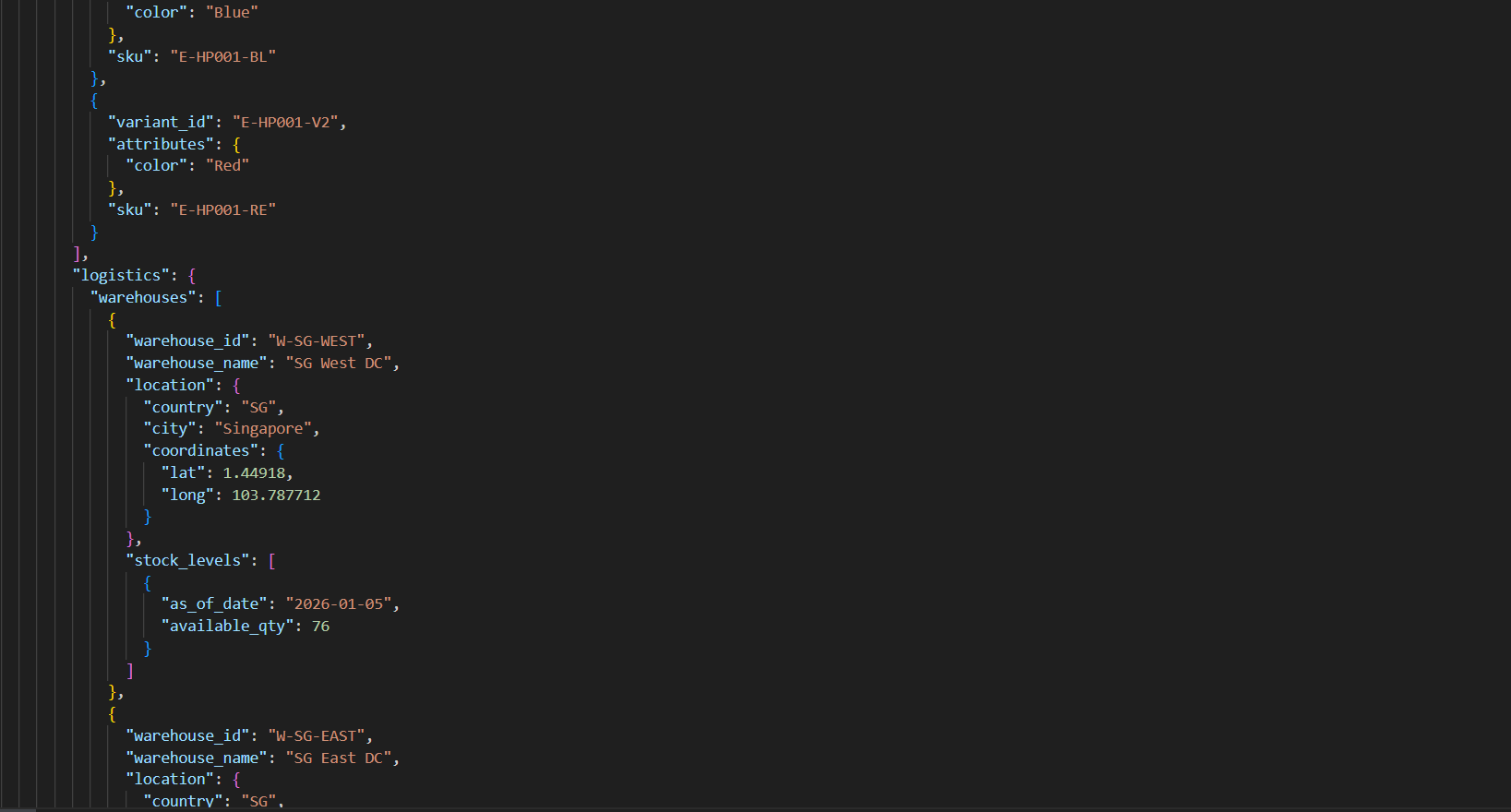
* product\_id links **Product Master ↔ Sales Items ↔ Reviews ↔ Returns**
* feedback\_id links **Reviews ↔ Media Metadata**, allowing media evidence to be associated with specific review feedback events
* Staged media files are resolved to secure, queryable URLs by matching media metadata file\_name with the stage directory listing

This design ensures that the curated dataset is both logically consistent and analytically useful. To ensure consistency and usefulness for analytics, we check that master attributes provide context, while transactional records provide measures and outcomes. The final curated dataset integrates these domains into a single, report-ready table that supports end-to-end analysis which is in a form of:

purchase → review → media evidence → returns.

# JSON Data Files

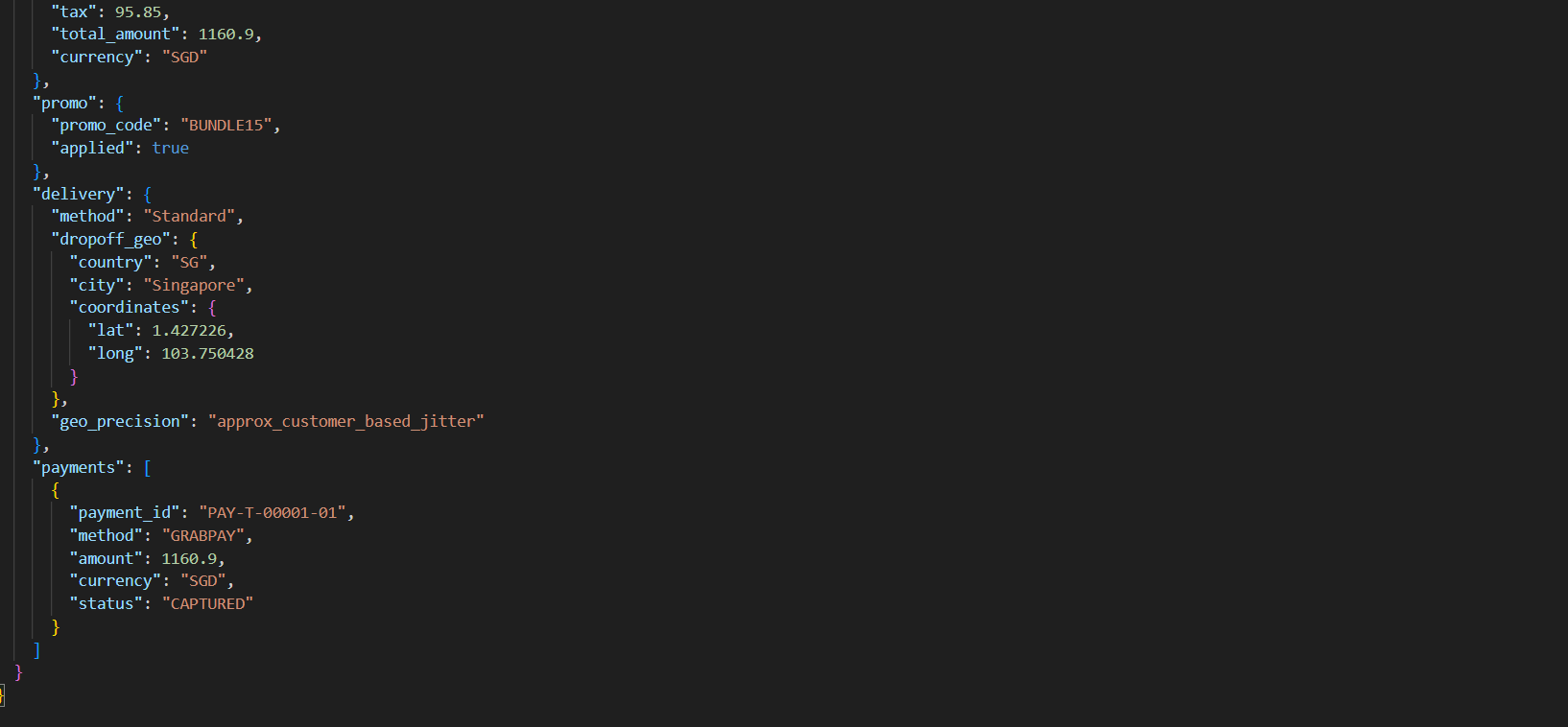
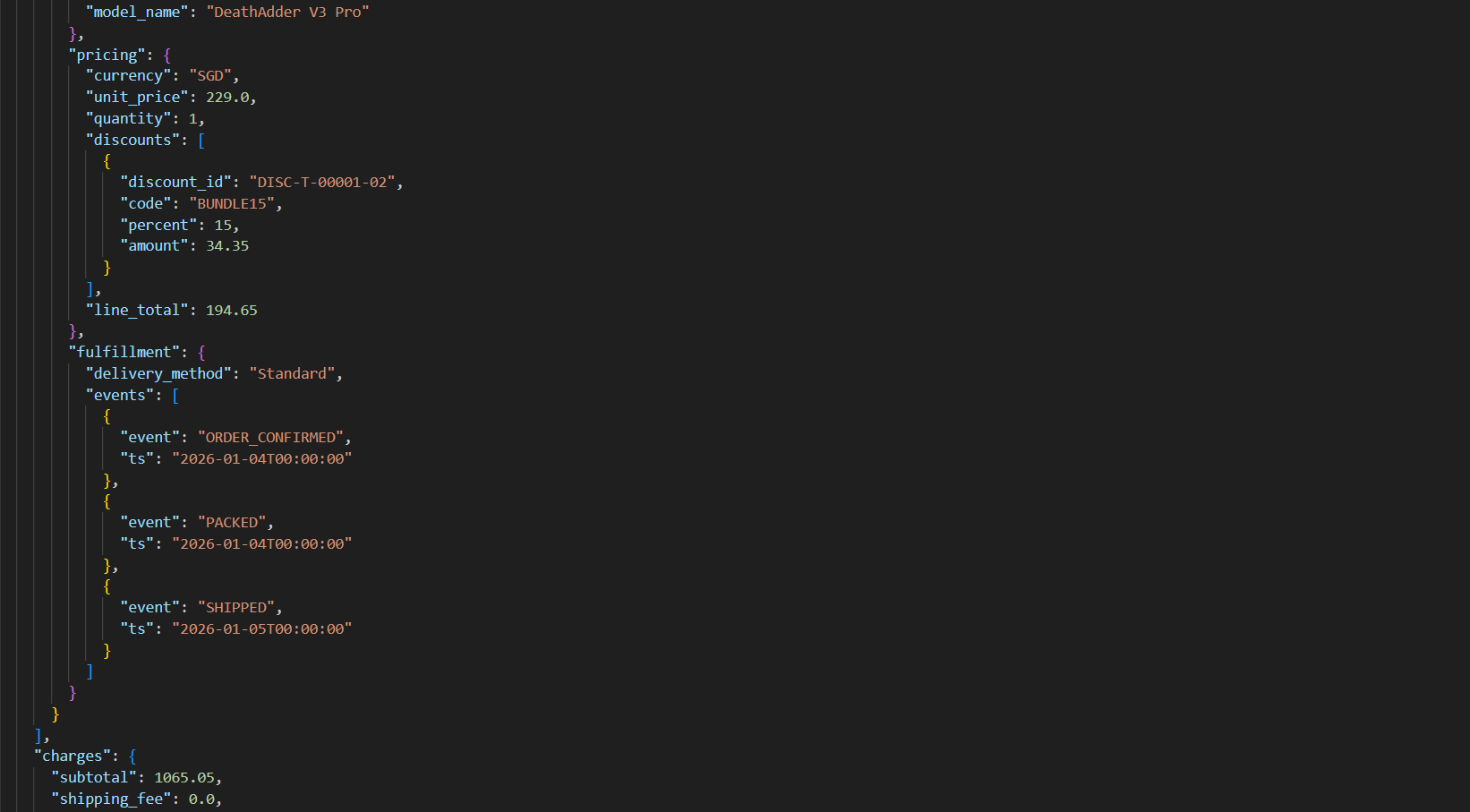
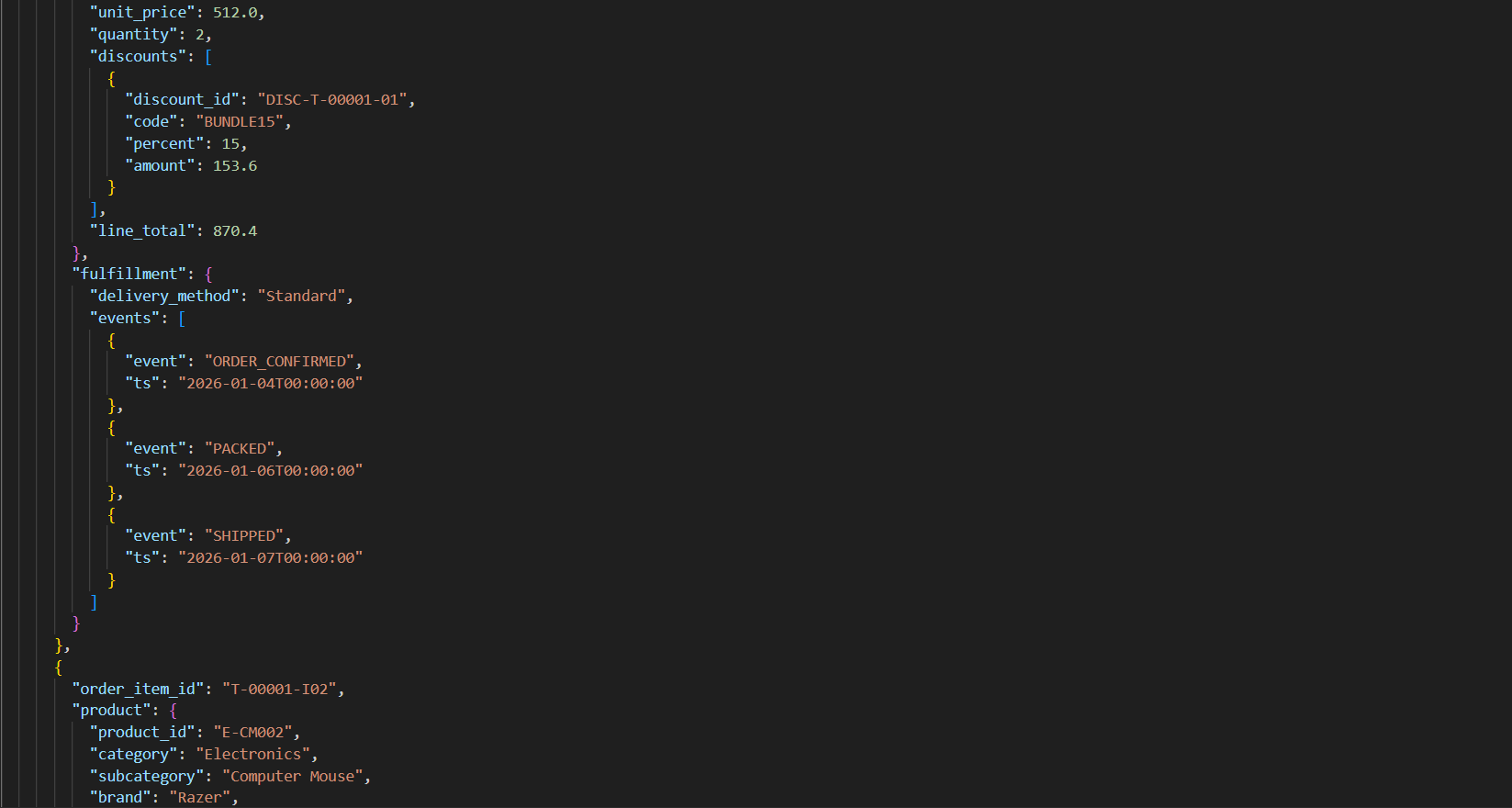
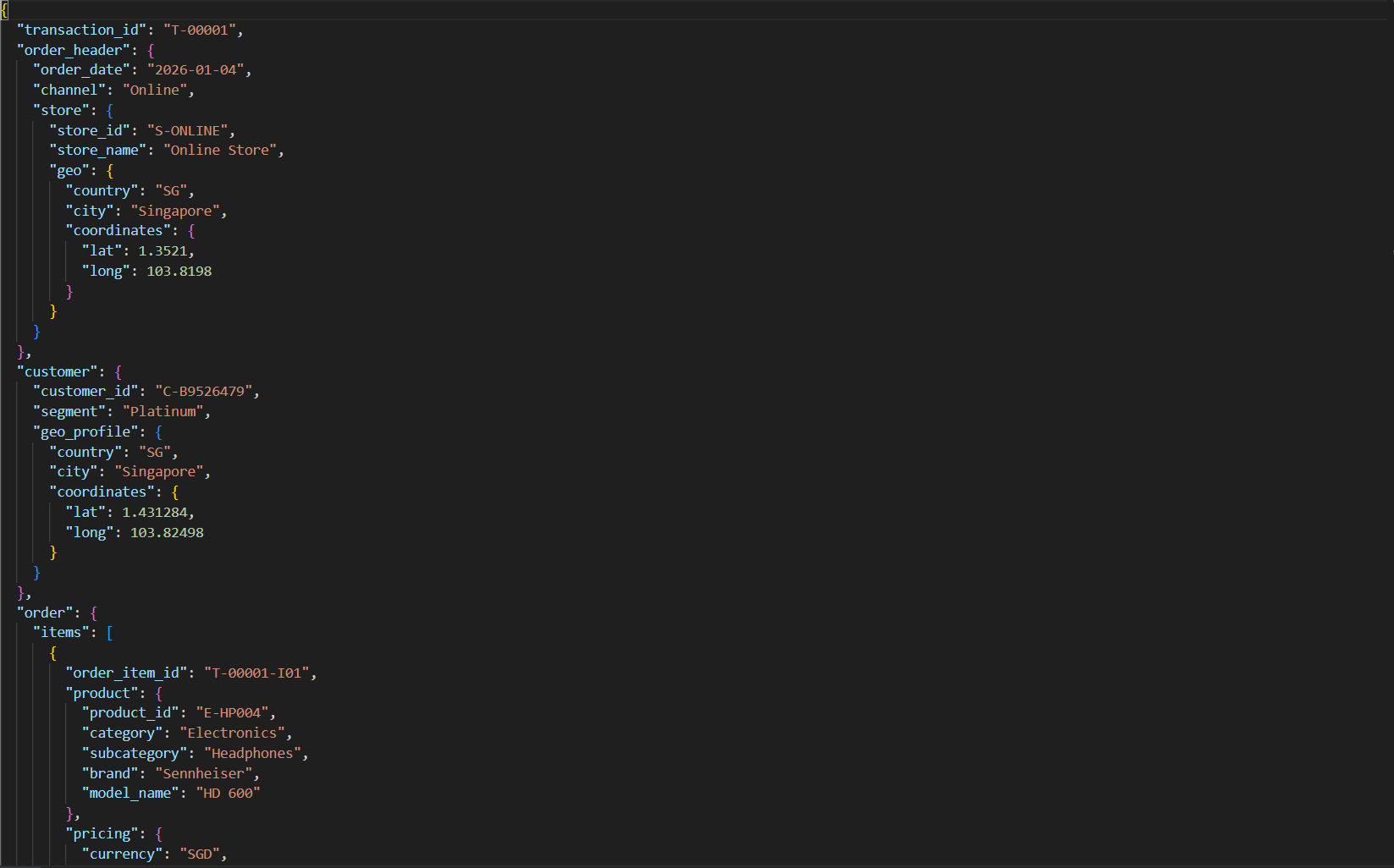
## Product Master (product\_master\_part01/02.json)



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Depth | JSON Key/ Path | Example Value | Type | Notes |
| 0 | product\_id | "E-HP001" | STRING | Unique product identifier (join key to sales/reviews/media) |
| 0 | product\_profile | {...} | OBJECT | Main container for all product attributes |
| 1 | product\_profile.identity | {...} | OBJECT | Product identity + manufacturer/model + external IDs |
| 2 | product\_profile.identity.manufacturer | {...} | OBJECT | Manufacturer details |
| 3 | product\_profile.identity.manufacturer.brand | "Sony" | STRING | Brand name (reporting / grouping) |
| 3 | product\_profile.identity.manufacturer.supplier | {...} | OBJECT | Supplier details |
| 4 | product\_profile.identity.manufacturer.supplier.supplier\_id | "SUP-SONY" | STRING | Supplier identifier |
| 4 | product\_profile.identity.manufacturer.supplier.hq\_location | {...} | OBJECT | Supplier HQ location |
| 5 | product\_profile.identity.manufacturer.supplier.hq\_location.country | "SG" | STRING | Country code |
| 5 | product\_profile.identity.manufacturer.supplier.hq\_location.city | "Singapore" | STRING | City |
| 5 | product\_profile.identity.manufacturer.supplier.hq\_location.coordinates | {...} | OBJECT | Geo coordinates |
| 6 | product\_profile.identity.manufacturer.supplier.hq\_location.coordinates.lat | 1.277964 | NUMBER | Latitude |
| 6 | product\_profile.identity.manufacturer.supplier.hq\_location.coordinates.long | 103.91028 | NUMBER | Longitude |
| 2 | product\_profile.identity.model | {...} | OBJECT | Model information |
| 3 | product\_profile.identity.model.model\_name | "WH-1000XM5" | STRING | Model name (useful product label) |
| 2 | product\_profile.identity.external\_keys | [...] | ARRAY | External identifiers (e.g., marketplace IDs) |
| 3 | product\_profile.identity.external\_keys[0] | {...} | OBJECT | One external key record |
| 4 | product\_profile.identity.external\_keys[0].key\_type | "ASIN" | STRING | Type of external key |
| 4 | product\_profile.identity.external\_keys[0].key\_value | "B09XS7JWHH" | STRING | External ID value |
| 1 | product\_profile.classification | {...} | OBJECT | Product category hierarchy |
| 2 | product\_profile.classification.category | "Electronics" | STRING | Category (dimension attribute) |
| 2 | product\_profile.classification.subcategory | "Headphones" | STRING | Subcategory (dimension attribute) |
| 2 | product\_profile.classification.taxonomy | {...} | OBJECT | Additional classification grouping |
| 3 | product\_profile.classification.taxonomy.product\_family | {...} | OBJECT | Family grouping for analytics |
| 4 | product\_profile.classification.taxonomy.product\_family.family\_name | "Headphones - Sony" | STRING | Family label |
| 4 | product\_profile.classification.taxonomy.product\_family.tags | [...] | ARRAY | Search/analytics tags |
| 5 | product\_profile.classification.taxonomy.product\_family.tags[0] | "electronics" | STRING | Tag (array element) |
| 5 | product\_profile.classification.taxonomy.product\_family.tags[1] | "headphones" | STRING | Tag (array element) |
| 5 | product\_profile.classification.taxonomy.product\_family.tags[2] | "sony" | STRING | Tag (array element) |
| 1 | product\_profile.commerce | {...} | OBJECT | Commercial info: pricing + channels |
| 2 | product\_profile.commerce.pricing | {...} | OBJECT | Pricing information |
| 3 | product\_profile.commerce.pricing.base\_price | 445.5 | NUMBER | Base selling price |
| 3 | product\_profile.commerce.pricing.currency | "SGD" | STRING | Currency code |
| 3 | product\_profile.commerce.pricing.discounts | [...] | ARRAY | Discount list applied to product |
| 4 | product\_profile.commerce.pricing.discounts[0] | {...} | OBJECT | One discount record |
| 5 | product\_profile.commerce.pricing.discounts[0].discount\_id | "DISC-E-HP001-001" | STRING | Discount ID |
| 5 | product\_profile.commerce.pricing.discounts[0].type | "Seasonal" | STRING | Discount type |
| 5 | product\_profile.commerce.pricing.discounts[0].percentage | 10 | NUMBER | Discount % |
| 4 | product\_profile.commerce.pricing.discounts[1] | {...} | OBJECT | Second discount record |
| 5 | product\_profile.commerce.pricing.discounts[1].type | "Member" | STRING | Example: member discount |
| 2 | product\_profile.commerce.channel | {...} | OBJECT | Channel listing info |
| 3 | product\_profile.commerce.channel.listings | [...] | ARRAY | Channels where product is sold |
| 4 | product\_profile.commerce.channel.listings[0] | {...} | OBJECT | One listing |
| 5 | product\_profile.commerce.channel.listings[0].channel | "Online" | STRING | Sales channel |
| 5 | product\_profile.commerce.channel.listings[0].status | "ACTIVE" | STRING | Channel availability |
| 4 | product\_profile.commerce.channel.listings[1].channel | "Retail" | STRING | Another channel |
| 1 | product\_profile.variants | [...] | ARRAY | Product variants (e.g., color) |
| 2 | product\_profile.variants[0] | {...} | OBJECT | Variant record |
| 3 | product\_profile.variants[0].variant\_id | "E-HP001-V1" | STRING | Variant ID |
| 3 | product\_profile.variants[0].attributes | {...} | OBJECT | Variant attributes |
| 4 | product\_profile.variants[0].attributes.color | "Blue" | STRING | Variant color |
| 3 | product\_profile.variants[0].sku | "E-HP001-BL" | STRING | SKU for inventory/sales |
| 2 | product\_profile.variants[1].attributes.color | "Red" | STRING | Another variant color |
| 1 | product\_profile.logistics | {...} | OBJECT | Logistics + warehouse inventory |
| 2 | product\_profile.logistics.warehouses | [...] | ARRAY | Warehouse list |
| 3 | product\_profile.logistics.warehouses[0] | {...} | OBJECT | One warehouse record |
| 4 | product\_profile.logistics.warehouses[0].warehouse\_id | "W-SG-WEST" | STRING | Warehouse ID |
| 4 | product\_profile.logistics.warehouses[0].warehouse\_name | "SG West DC" | STRING | Warehouse name |
| 4 | product\_profile.logistics.warehouses[0].location | {...} | OBJECT | Warehouse location |
| 5 | product\_profile.logistics.warehouses[0].location.country | "SG" | STRING | Country |
| 5 | product\_profile.logistics.warehouses[0].location.city | "Singapore" | STRING | City |
| 5 | product\_profile.logistics.warehouses[0].location.coordinates.lat | 1.44918 | NUMBER | Latitude |
| 4 | product\_profile.logistics.warehouses[0].stock\_levels | [...] | ARRAY | Stock snapshots |
| 5 | product\_profile.logistics.warehouses[0].stock\_levels[0] | {...} | OBJECT | One stock snapshot |
| 6 | product\_profile.logistics.warehouses[0].stock\_levels[0].as\_of\_date | "2026-01-05" | STRING (DATE) | Stock date |
| 6 | product\_profile.logistics.warehouses[0].stock\_levels[0].available\_qty | 76 | NUMBER | Available quantity |
| 3 | product\_profile.logistics.warehouses[1].warehouse\_id | "W-SG-EAST" | STRING | Another warehouse |
| 6 | product\_profile.logistics.warehouses[1].stock\_levels[0].available\_qty | 98 | NUMBER | East DC stock |
| 1 | product\_profile.content | {...} | OBJECT | Extra content/notes |
| 2 | product\_profile.content.notes | "Top tier noise canceling" | STRING | Unstructured product note |

This Product Master record is product centric. It starts with a unique product\_id, then stores all descriptive attributes inside product\_profile. The identity section captures brand, supplier, model, and external identifiers. The classification section provides category/subcategory and a taxonomy family with searchable tags. The commerce section defines pricing (base price, currency, discount rules) and selling channels (online/retail). Arrays such as variants[] and warehouses[] represent repeating entities: multiple SKUs/colors and multiple warehouse stock snapshots. This structure supports analytics by allowing the product master to act as a dimension table, enriching transactional datasets through product\_id.

## Sales Transactions (sales\_transactions\_part01/02.json)



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Depth | JSON Key/Path | Example Value | Type | Notes |
| 0 | transaction\_id | "T-00001" | STRING | Primary transaction key |
| 0 | order\_header | {...} | OBJECT | High-level order details |
| 1 | order\_header.order\_date | "2026-01-04" | STRING (DATE) | Order date (can be cast to DATE) |
| 1 | order\_header.channel | "Online" | STRING | Sales channel |
| 1 | order\_header.store | {...} | OBJECT | Store context |
| 2 | order\_header.store.store\_id | "S-ONLINE" | STRING | Store identifier |
| 2 | order\_header.store.store\_name | "Online Store" | STRING | Store name |
| 2 | order\_header.store.geo | {...} | OBJECT | Store location |
| 3 | order\_header.store.geo.country | "SG" | STRING | Country |
| 3 | order\_header.store.geo.city | "Singapore" | STRING | City |
| 3 | order\_header.store.geo.coordinates | {...} | OBJECT | Store coordinates |
| 4 | order\_header.store.geo.coordinates.lat | 1.3521 | NUMBER | Latitude |
| 4 | order\_header.store.geo.coordinates.long | 103.8198 | NUMBER | Longitude |
| 0 | customer | {...} | OBJECT | Customer details for this transaction |
| 1 | customer.customer\_id | "C-B9526479" | STRING | Customer key (joinable across datasets) |
| 1 | customer.segment | "Platinum" | STRING | Customer segment |
| 1 | customer.geo\_profile | {...} | OBJECT | Customer location profile |
| 2 | customer.geo\_profile.country | "SG" | STRING | Country |
| 2 | customer.geo\_profile.city | "Singapore" | STRING | City |
| 2 | customer.geo\_profile.coordinates | {...} | OBJECT | Customer coordinates |
| 3 | customer.geo\_profile.coordinates.lat | 1.431284 | NUMBER | Latitude |
| 3 | customer.geo\_profile.coordinates.long | 103.82498 | NUMBER | Longitude |
| 0 | order | {...} | OBJECT | Main transaction body (items + charges + promo + delivery + payments) |
| 1 | order.items | [...] | ARRAY | Line items array (each item = one product purchase line) |
| 2 | order.items[0] | {...} | OBJECT | First line item |
| 3 | order.items[0].order\_item\_id | "T-00001-I01" | STRING | Line item key (granularity for sales fact table) |
| 3 | order.items[0].product | {...} | OBJECT | Product snapshot at purchase time |
| 4 | order.items[0].product.product\_id | "E-HP004" | STRING | Product key (join to product master) |
| 4 | order.items[0].product.category | "Electronics" | STRING | Product category (snapshot) |
| 4 | order.items[0].product.subcategory | "Headphones" | STRING | Product subcategory (snapshot) |
| 4 | order.items[0].product.brand | "Sennheiser" | STRING | Brand snapshot |
| 4 | order.items[0].product.model\_name | "HD 600" | STRING | Model name snapshot |
| 3 | order.items[0].pricing | {...} | OBJECT | Pricing at line-item level |
| 4 | order.items[0].pricing.currency | "SGD" | STRING | Currency |
| 4 | order.items[0].pricing.unit\_price | 512 | NUMBER | Unit price |
| 4 | order.items[0].pricing.quantity | 2 | NUMBER | Quantity purchased |
| 4 | order.items[0].pricing.discounts | [...] | ARRAY | Discount(s) applied to the line |
| 5 | order.items[0].pricing.discounts[0] | {...} | OBJECT | One discount record |
| 6 | order.items[0].pricing.discounts[0].discount\_id | "DISC-T-00001-01" | STRING | Discount ID |
| 6 | order.items[0].pricing.discounts[0].code | "BUNDLE15" | STRING | Discount code |
| 6 | order.items[0].pricing.discounts[0].percent | 15 | NUMBER | Discount percent |
| 6 | order.items[0].pricing.discounts[0].amount | 153.6 | NUMBER | Discount amount |
| 4 | order.items[0].pricing.line\_total | 870.4 | NUMBER | Final line total after discounts |
| 3 | order.items[0].fulfillment | {...} | OBJECT | Delivery/fulfillment details |
| 4 | order.items[0].fulfillment.delivery\_method | "Standard" | STRING | Delivery method |
| 4 | order.items[0].fulfillment.events | [...] | ARRAY | Fulfillment timeline |
| 5 | order.items[0].fulfillment.events[0] | {...} | OBJECT | One event |
| 6 | order.items[0].fulfillment.events[0].event | "ORDER\_CONFIRMED" | STRING | Event name |
| 6 | order.items[0].fulfillment.events[0].ts | "2026-01-04T00:00:00" | STRING (TIMESTAMP) | Event timestamp |
| 2 | order.items[1] | {...} | OBJECT | Second line item |
| 3 | order.items[1].order\_item\_id | "T-00001-I02" | STRING | Another line item key |
| 4 | order.items[1].product.product\_id | "E-CM002" | STRING | Another product key |
| 4 | order.items[1].product.subcategory | "Computer Mouse" | STRING | Subcategory snapshot |
| 4 | order.items[1].product.brand | "Razer" | STRING | Brand snapshot |
| 4 | order.items[1].pricing.unit\_price | 229 | NUMBER | Unit price |
| 4 | order.items[1].pricing.quantity | 1 | NUMBER | Quantity |
| 4 | order.items[1].pricing.line\_total | 194.65 | NUMBER | Line total |
| 1 | order.charges | {...} | OBJECT | Order-level totals |
| 2 | order.charges.subtotal | 1065.05 | NUMBER | Sum before tax/shipping |
| 2 | order.charges.shipping\_fee | 0 | NUMBER | Shipping fee |
| 2 | order.charges.tax | 95.85 | NUMBER | Tax amount |
| 2 | order.charges.total\_amount | 1160.9 | NUMBER | Final total |
| 2 | order.charges.currency | "SGD" | STRING | Currency |
| 1 | order.promo | {...} | OBJECT | Promotion applied at order level |
| 2 | order.promo.promo\_code | "BUNDLE15" | STRING | Promo code |
| 2 | order.promo.applied | TRUE | BOOLEAN | Whether promo applied |
| 1 | order.delivery | {...} | OBJECT | Delivery dropoff geo |
| 2 | order.delivery.method | "Standard" | STRING | Delivery method |
| 2 | order.delivery.dropoff\_geo | {...} | OBJECT | Dropoff location |
| 3 | order.delivery.dropoff\_geo.country | "SG" | STRING | Country |
| 3 | order.delivery.dropoff\_geo.city | "Singapore" | STRING | City |
| 3 | order.delivery.dropoff\_geo.coordinates.lat | 1.427226 | NUMBER | Latitude |
| 3 | order.delivery.dropoff\_geo.coordinates.long | 103.750428 | NUMBER | Longitude |
| 2 | order.delivery.geo\_precision | "approx\_customer\_based\_jitter" | STRING | Privacy/precision indicator |
| 1 | order.payments | [...] | ARRAY | Payment records |
| 2 | order.payments[0] | {...} | OBJECT | One payment record |
| 3 | order.payments[0].payment\_id | "PAY-T-00001-01" | STRING | Payment identifier |
| 3 | order.payments[0].method | "GRABPAY" | STRING | Payment method |
| 3 | order.payments[0].amount | 1160.9 | NUMBER | Amount paid |
| 3 | order.payments[0].currency | "SGD" | STRING | Currency |
| 3 | order.payments[0].status | "CAPTURED" | STRING | Payment status |

This Sales Transactions record is transaction centric. It begins with a single transaction\_id, then stores contextual attributes in order\_header (date, channel, store location) and customer (customer\_id, segment, geo profile). The core business content is inside order, where items[] is an array of purchased line items. Each items[i] contains an order\_item\_id, a product snapshot (product\_id, category, brand, model), pricing details (unit price, quantity, discounts, line\_total), and fulfillment events (order confirmed/packed/shipped timestamps). At the order level, charges summarizes subtotal, tax, and total\_amount, while promo stores campaign usage, delivery stores dropoff geo, and payments[] records payment method and status. This nesting supports a relational model by flattening order.items[] into a sales line-item fact table, one row per order\_item\_id for revenue and units-sold analysis.

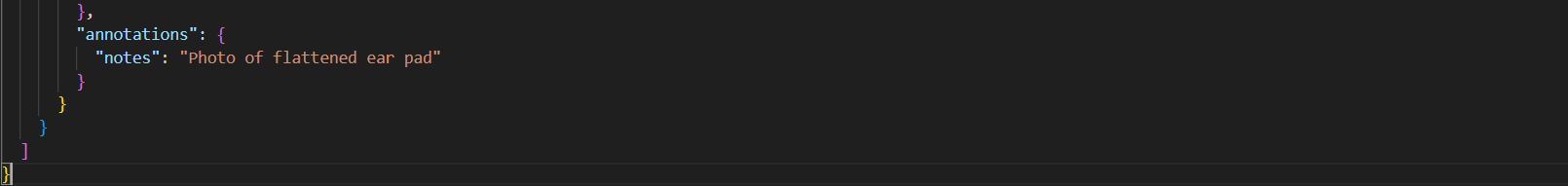
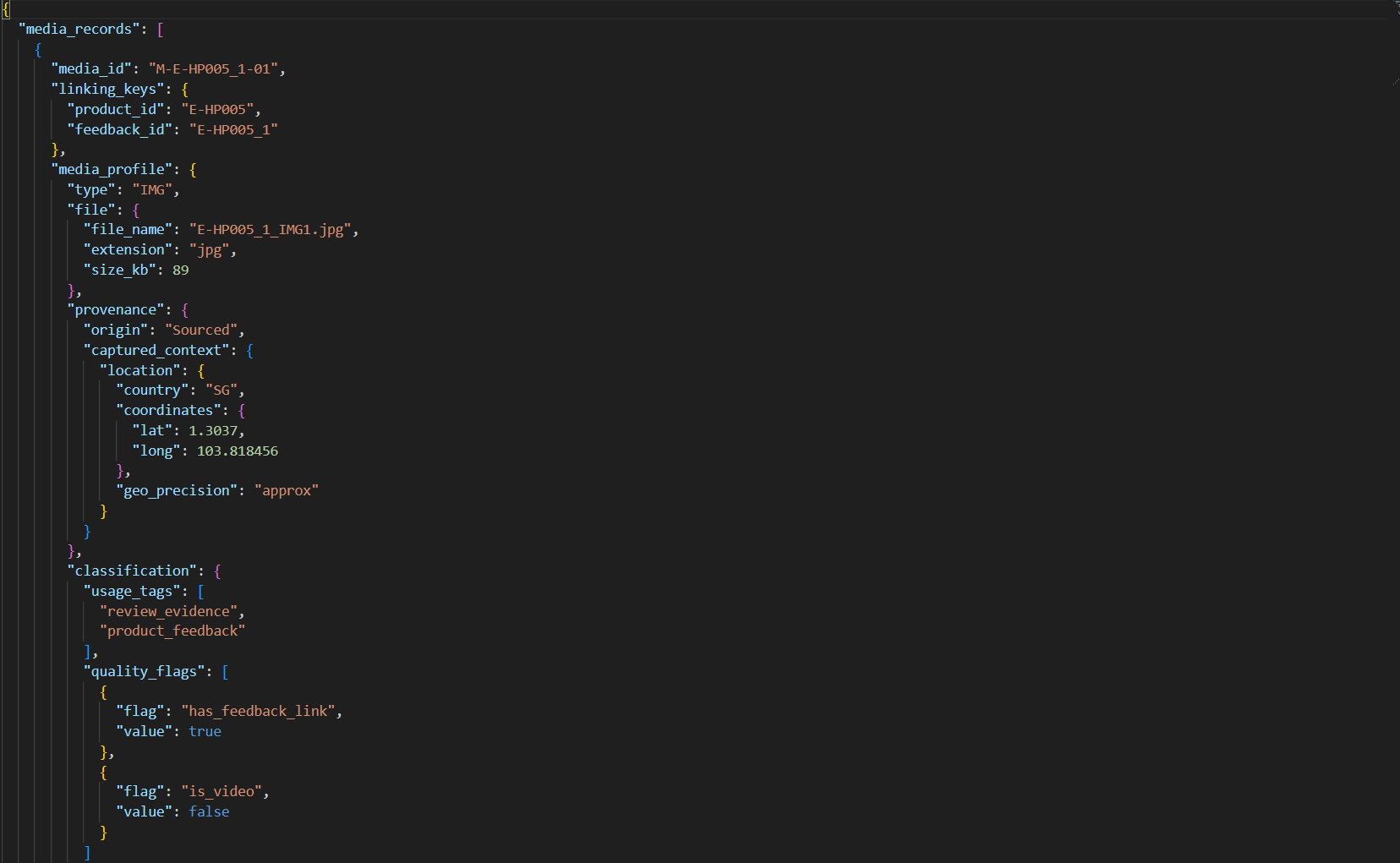
## Customer Reviews (customer\_reviews\_part01–05.json)



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Depth | JSON Key/Path | Example Value | Type | Notes |
| 0 | review\_events | [...] | ARRAY | Array of review event objects |
| 1 | review\_events[0] | {...} | OBJECT | One review event record |
| 2 | review\_events[0].review\_event\_id | "REV-00041" | STRING | Unique review event identifier (review grain key) |
| 2 | review\_events[0].linking\_keys | {...} | OBJECT | Keys used to join with other datasets |
| 3 | review\_events[0].linking\_keys.product\_id | "E-CM001" | STRING | Join to Product Master + Sales/Returns via product\_id |
| 3 | review\_events[0].linking\_keys.feedback\_id | "E-CM001\_7" | STRING | Join to Media Metadata via feedback\_id |
| 2 | review\_events[0].review\_context | {...} | OBJECT | Context about reviewer, purchase, product snapshot |
| 3 | review\_events[0].review\_context.collector | {...} | OBJECT | Collection information |
| 4 | review\_events[0].review\_context.collector.reviewer\_profile | {...} | OBJECT | Reviewer profile details |
| 5 | review\_events[0].review\_context.collector.reviewer\_profile.reviewer\_id | "U-D22036EA" | STRING | Reviewer identifier |
| 5 | review\_events[0].review\_context.collector.reviewer\_profile.geo\_profile | {...} | OBJECT | Reviewer geo profile |
| 6 | review\_events[0].review\_context.collector.reviewer\_profile.geo\_profile.reported\_location | "Malaysia" | STRING | Reported location |
| 6 | review\_events[0].review\_context.collector.reviewer\_profile.geo\_profile.coordinates | {...} | OBJECT | Coordinates object |
| 7 | review\_events[0].review\_context.collector.reviewer\_profile.geo\_profile.coordinates.lat | 3.409 | NUMBER | Latitude |
| 7 | review\_events[0].review\_context.collector.reviewer\_profile.geo\_profile.coordinates.long | 101.73954 | NUMBER | Longitude |
| 6 | review\_events[0].review\_context.collector.reviewer\_profile.geo\_profile.geo\_precision | "approx\_country\_capital\_plus\_jitter" | STRING | Precision / privacy indicator |
| 3 | review\_events[0].review\_context.purchase | {...} | OBJECT | Purchase validation |
| 4 | review\_events[0].review\_context.purchase.verified\_purchase | TRUE | BOOLEAN | Indicates verified purchase |
| 3 | review\_events[0].review\_context.product\_snapshot | {...} | OBJECT | Product category snapshot at review time |
| 4 | review\_events[0].review\_context.product\_snapshot.category | "Electronics" | STRING | Category snapshot |
| 4 | review\_events[0].review\_context.product\_snapshot.subcategory | "Computer Mouse" | STRING | Subcategory snapshot |
| 2 | review\_events[0].review\_payload | {...} | OBJECT | Main review content and sentiment fields |
| 3 | review\_events[0].review\_payload.rating | {...} | OBJECT | Rating details |
| 4 | review\_events[0].review\_payload.rating.overall | 5 | NUMBER | Overall rating score |
| 4 | review\_events[0].review\_payload.rating.scale | "1-5" | STRING | Rating scale |
| 3 | review\_events[0].review\_payload.review\_date | {...} | OBJECT | Date stored with format metadata |
| 4 | review\_events[0].review\_payload.review\_date.original\_field\_format | "DD/MM/YY" | STRING | Declared original format |
| 4 | review\_events[0].review\_payload.review\_date.value | "11/12/22" | STRING (DATE) | Date value stored as string |
| 3 | review\_events[0].review\_payload.text | {...} | OBJECT | Review text fields |
| 4 | review\_events[0].review\_payload.text.title | "Logi Options+ is great" | STRING | Review title |
| 4 | review\_events[0].review\_payload.text.full | "The software actually works well now..." | STRING | Full review text |
| 4 | review\_events[0].review\_payload.text.notes | null | NULL | Optional field (may be missing/NULL) |
| 3 | review\_events[0].review\_payload.analysis | {...} | OBJECT | NLP/analysis outputs (keywords, aspects, flags) |
| 4 | review\_events[0].review\_payload.analysis.keywords | [...] | ARRAY | Keyword list extracted from review text |
| 5 | review\_events[0].review\_payload.analysis.keywords[0] | "software" | STRING | Example keyword (array repeats) |
| 4 | review\_events[0].review\_payload.analysis.aspect\_sentiment | [...] | ARRAY | Aspect-based sentiment analysis |
| 5 | review\_events[0].review\_payload.analysis.aspect\_sentiment[0] | {...} | OBJECT | One aspect sentiment record |
| 6 | review\_events[0].review\_payload.analysis.aspect\_sentiment[0].aspect | "sound" | STRING | Aspect name |
| 6 | review\_events[0].review\_payload.analysis.aspect\_sentiment[0].sentiment | {...} | OBJECT | Sentiment object |
| 7 | review\_events[0].review\_payload.analysis.aspect\_sentiment[0].sentiment.label | "positive" | STRING | Sentiment label |
| 7 | review\_events[0].review\_payload.analysis.aspect\_sentiment[0].sentiment.score | 2 | NUMBER | Sentiment score |
| 4 | review\_events[0].review\_payload.analysis.quality\_flags | [...] | ARRAY | Quality/metadata flags |
| 5 | review\_events[0].review\_payload.analysis.quality\_flags[0] | {...} | OBJECT | One quality flag record |
| 6 | review\_events[0].review\_payload.analysis.quality\_flags[0].flag | "has\_media" | STRING | Flag name |
| 6 | review\_events[0].review\_payload.analysis.quality\_flags[0].value | FALSE | BOOLEAN | Flag value |
| 2 | review\_events[0].media\_bundle | {...} | OBJECT | Review media linkage (photos/videos references) |
| 3 | review\_events[0].media\_bundle.include\_media | FALSE | BOOLEAN | Whether review includes media |
| 3 | review\_events[0].media\_bundle.media\_refs | [….] | ARRAY | List of media references (empty here) |

This Customer Reviews JSON file is event-based, where each record is stored inside the review\_events[] array. Each review event contains (1) linking keys (product\_id, feedback\_id) that connect the review to product master and to review media, (2) review context such as reviewer profile and verified purchase indicator, and (3) the review payload containing rating, review date (stored as a string with its original format), and unstructured review text. An additional nested analysis object stores derived features such as keywords, aspect-level sentiment, and quality flags like whether media exists or not. Because the review events and many fields are stored as arrays/objects, Snowflake uses LATERAL FLATTEN() to convert review\_events[] into relational rows, one row per review\_event\_id for downstream joins and curated dataset creation.

## Media Metadata (media\_metadata\_part01.json)



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Depth | JSON Key / Path |  | Example Value | Type | Notes |
| 0 | media\_records | | [...] | ARRAY | Array of media metadata records |
| 1 | media\_records[0] | | {...} | OBJECT | One media record (photo/video) |
| 2 | media\_records[0].media\_id | | "M-E-HP005\_1-01" | STRING | Unique media identifier (media grain key) |
| 2 | media\_records[0].linking\_keys | | {...} | OBJECT | Keys used to join to other datasets |
| 3 | media\_records[0].linking\_keys.product\_id | | "E-HP005" | STRING | Join to Product Master via product\_id |
| 3 | media\_records[0].linking\_keys.feedback\_id | | "E-HP005\_1" | STRING | Join to Reviews via feedback\_id |
| 2 | media\_records[0].media\_profile | | {...} | OBJECT | Main media information container |
| 3 | media\_records[0].media\_profile.type | | "IMG" | STRING | Media type (IMG/VID) |
| 3 | media\_records[0].media\_profile.file | | {...} | OBJECT | File metadata |
| 4 | media\_records[0].media\_profile.file.file\_name | | "E-HP005\_1\_IMG1.jpg" | STRING | Filename used to match staged files |
| 4 | media\_records[0].media\_profile.file.extension | | "jpg" | STRING | File extension |
| 4 | media\_records[0].media\_profile.file.size\_kb | | 89 | NUMBER | File size in KB |
| 3 | media\_records[0].media\_profile.provenance | | {...} | OBJECT | Source/origin tracking for media |
| 4 | media\_records[0].media\_profile.provenance.origin | | "Sourced" | STRING | Where the media came from |
| 4 | media\_records[0].media\_profile.provenance.captured\_context | | {...} | OBJECT | Captured context metadata |
| 5 | media\_records[0].media\_profile.provenance.captured\_context.location | | {...} | OBJECT | Location where media was captured |
| 6 | media\_records[0].media\_profile.provenance.captured\_context.location.country | | "SG" | STRING | Country |
| 6 | media\_records[0].media\_profile.provenance.captured\_context.location.coordinates | | {...} | OBJECT | Geo coordinates |
| 7 | media\_records[0].media\_profile.provenance.captured\_context.location.coordinates.lat | | 1.3037 | NUMBER | Latitude |
| 7 | media\_records[0].media\_profile.provenance.captured\_context.location.coordinates.long | | 103.818456 | NUMBER | Longitude |
| 6 | media\_records[0].media\_profile.provenance.captured\_context.location.geo\_precision | | "approx" | STRING | Precision indicator |
| 3 | media\_records[0].media\_profile.classification | | {...} | OBJECT | Classification and quality tags |
| 4 | media\_records[0].media\_profile.classification.usage\_tags | | [...] | ARRAY | Tags describing how media is used |
| 5 | media\_records[0].media\_profile.classification.usage\_tags[0] | | "review\_evidence" | STRING | Tag example |
| 5 | media\_records[0].media\_profile.classification.usage\_tags[1] | | "product\_feedback" | STRING | Tag example |
| 4 | media\_records[0].media\_profile.classification.quality\_flags | | [...] | ARRAY | Quality flags stored as objects |
| 5 | media\_records[0].media\_profile.classification.quality\_flags[0] | | {...} | OBJECT | One quality flag record |
| 6 | media\_records[0].media\_profile.classification.quality\_flags[0].flag | | "has\_feedback\_link" | STRING | Flag name |
| 6 | media\_records[0].media\_profile.classification.quality\_flags[0].value | | TRUE | BOOLEAN | Flag value |
| 5 | media\_records[0].media\_profile.classification.quality\_flags[1] | | {...} | OBJECT | Another flag record |
| 6 | media\_records[0].media\_profile.classification.quality\_flags[1].flag | | "is\_video" | STRING | Flag name |
| 6 | media\_records[0].media\_profile.classification.quality\_flags[1].value | | FALSE | BOOLEAN | Flag value |
| 3 | media\_records[0].media\_profile.annotations | | {...} | OBJECT | Human-entered notes/annotations |
| 4 | media\_records[0].media\_profile.annotations.notes | | "Photo of flattened ear pad" | STRING | Description of media content |

This Media Metadata JSON file stores media evidence records (photos/videos) inside the media\_records[] array. Each media record is uniquely identified by media\_id and includes linking\_keys such as product\_id and feedback\_id, which enable joining the media to both the product master dataset and the specific customer review event. The media\_profile object contains the media type (IMG/VID), file information (file name, extension, size), provenance details (origin and capture location/precision), and classification fields such as usage tags and quality flags (e.g., whether it is linked to feedback and whether it is a video). Because filenames are stored as strings, this dataset can be integrated with Snowflake stages by matching file\_name to the staged files and generating scoped URLs for the curated dataset.

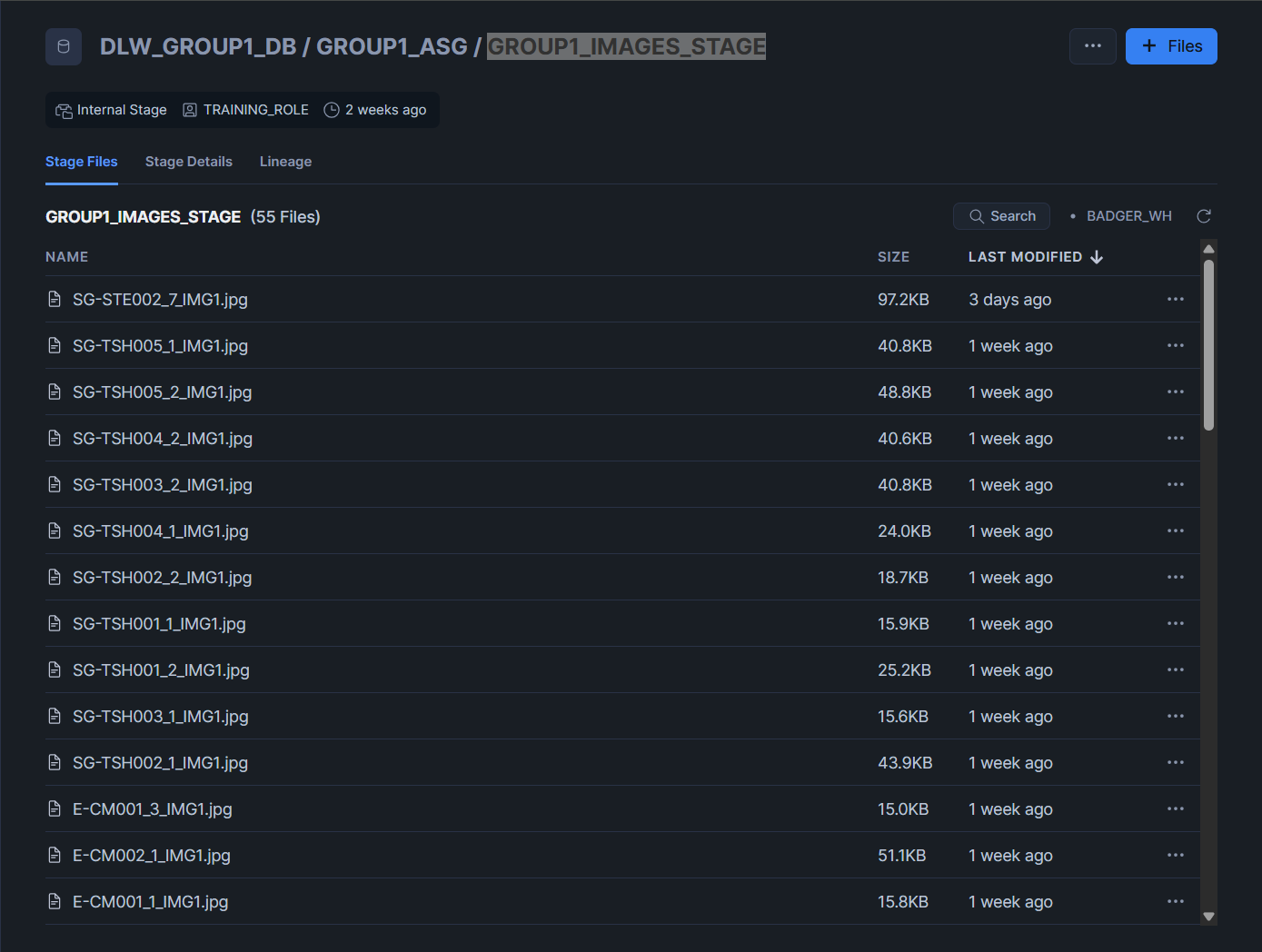
## Returns / Issues (returns\_issues\_part01.json)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Depth | JSON Key/Path | Example Value | Type | Notes |
| 0 | return\_event\_id | "RET-00001" | STRING | Unique return event identifier (return grain key) |
| 0 | linking\_keys | {...} | OBJECT | Join keys linking return to transaction, item, product, customer, feedback |
| 1 | linking\_keys.transaction\_id | "T-00006" | STRING | Join to sales transaction |
| 1 | linking\_keys.order\_item\_id | "T-00006-I01" | STRING | Join to sales line-item grain |
| 1 | linking\_keys.product\_id | "SG-STE001" | STRING | Join to product master + sales product metrics |
| 1 | linking\_keys.customer\_id | "C-D22036EA" | STRING | Customer identifier |
| 1 | linking\_keys.feedback\_id | "SG-STE001\_6" | STRING | Join to review/media via feedback\_id (if applicable) |
| 0 | return\_context | {...} | OBJECT | Context about timing + channel/store |
| 1 | return\_context.dates | {...} | OBJECT | Date fields and derived metrics |
| 2 | return\_context.dates.order\_date | "2025-11-29" | STRING (DATE-like) | Original order date |
| 2 | return\_context.dates.return\_requested\_date | "2025-12-05" | STRING (DATE-like) | Return request date |
| 2 | return\_context.dates.days\_since\_purchase | 6 | NUMBER | Derived days since purchase |
| 1 | return\_context.channel | {...} | OBJECT | Sales channel + store information |
| 2 | return\_context.channel.sales\_channel | "Retail" | STRING | Channel where purchase/return occurred |
| 2 | return\_context.channel.store\_id | "S-TAMPINES" | STRING | Store identifier |
| 2 | return\_context.channel.store\_geo | {...} | OBJECT | Store geographic info |
| 3 | return\_context.channel.store\_geo.country | "SG" | STRING | Country |
| 3 | return\_context.channel.store\_geo.city | "Singapore" | STRING | City |
| 3 | return\_context.channel.store\_geo.coordinates | {...} | OBJECT | Geo coordinates |
| 4 | return\_context.channel.store\_geo.coordinates.lat | 1.3526 | NUMBER | Latitude |
| 4 | return\_context.channel.store\_geo.coordinates.long | 103.944 | NUMBER | Longitude |
| 0 | product\_snapshot | {...} | OBJECT | Product snapshot captured at return time |
| 1 | product\_snapshot.category | "Sporting Goods" | STRING | Snapshot category |
| 1 | product\_snapshot.subcategory | "Strength Training Equipment" | STRING | Snapshot subcategory |
| 1 | product\_snapshot.brand | "Fostoy" | STRING | Brand snapshot |
| 1 | product\_snapshot.model\_name | "Fostoy Adjustable Weight Bench Set" | STRING | Model snapshot |
| 1 | product\_snapshot.product\_id | "SG-STE001" | STRING | Product ID (repeats linking key) |
| 0 | return\_payload | {...} | OBJECT | Main return details: reason, issues, resolution, timeline |
| 1 | return\_payload.reason | {...} | OBJECT | Nested reason object (important for parsing) |
| 2 | return\_payload.reason.reason\_code | "COMFORT\_FIT" | STRING | Reason code |
| 2 | return\_payload.reason.reason\_description | "Comfort / fit issue" | STRING | Human-readable reason (kept in curated dataset) |
| 1 | return\_payload.issue\_details | [...] | ARRAY | List of issue records linked to the return |
| 2 | return\_payload.issue\_details[0] | {...} | OBJECT | One issue record |
| 3 | return\_payload.issue\_details[0].issue\_id | "RET-00001-ISS-01" | STRING | Issue identifier |
| 3 | return\_payload.issue\_details[0].category | "COMPATIBILITY" | STRING | Issue category |
| 3 | return\_payload.issue\_details[0].reason\_code | "COMFORT\_FIT" | STRING | Issue reason code |
| 3 | return\_payload.issue\_details[0].description | "Comfort / fit issue" | STRING | Issue description |
| 3 | return\_payload.issue\_details[0].severity | "LOW" | STRING | Severity rating |
| 3 | return\_payload.issue\_details[0].detected\_in | "CustomerReport" | STRING | Detection source |
| 1 | return\_payload.resolution | {...} | OBJECT | Return outcome/resolution actions |
| 2 | return\_payload.resolution.resolution\_type | "REPLACEMENT" | STRING | Resolution type |
| 2 | return\_payload.resolution.resolution\_steps | [...] | ARRAY | Step-by-step process tracking |
| 3 | return\_payload.resolution.resolution\_steps[0] | {...} | OBJECT | One resolution step |
| 4 | return\_payload.resolution.resolution\_steps[0].step | "VERIFY\_ELIGIBILITY" | STRING | Step name |
| 4 | return\_payload.resolution.resolution\_steps[0].status | "DONE" | STRING | Step status |
| 3 | return\_payload.resolution.resolution\_steps[3].step | "PROCESS\_REFUND\_OR\_REPLACEMENT" | STRING | Final step example |
| 1 | return\_payload.timeline\_events | [...] | ARRAY | Timeline audit trail of return process |
| 2 | return\_payload.timeline\_events[0] | {...} | OBJECT | One timeline event |
| 3 | return\_payload.timeline\_events[0].event | "RETURN\_REQUESTED" | STRING | Event label |
| 3 | return\_payload.timeline\_events[0].ts | "2025-12-05T00:00:00" | STRING (TIMESTAMP-like) | Event timestamp |
| 3 | return\_payload.timeline\_events[0].channel | "Retail" | STRING | Channel (when present) |
| 2 | return\_payload.timeline\_events[1].warehouse\_id | "W-SG-WEST" | STRING | Warehouse (when present) |
| 2 | return\_payload.timeline\_events[2].outcome | "APPROVED" | STRING | Outcome (when present) |
| 2 | return\_payload.timeline\_events[3].refund\_method | "StoreCredit" | STRING | Refund method (when present) |
| 0 | evidence\_media | {...} | OBJECT | Evidence attachments for returns |
| 1 | evidence\_media.has\_media | TRUE | BOOLEAN | Whether evidence media exists |
| 1 | evidence\_media.media\_refs | [...] | ARRAY | Media references array |
| 2 | evidence\_media.media\_refs[0] | {...} | OBJECT | One media reference record |
| 3 | evidence\_media.media\_refs[0].type | "IMG" | STRING | Media type |
| 3 | evidence\_media.media\_refs[0].file\_name | "SG-STE001\_3\_IMG9.jpg" | STRING | File name for matching staged image |
| 3 | evidence\_media.media\_refs[0].notes | "" | STRING | Optional notes (empty string here) |
| 3 | evidence\_media.media\_refs[0].linked\_feedback\_id | "SG-STE001\_6" | STRING | Links evidence to feedback\_id |

This Returns/Issues record is return-event-centric, identified by return\_event\_id. It includes a linking\_keys object that connects the return to the original purchase (transaction\_id, order\_item\_id) and to the product/customer (product\_id, customer\_id). The return\_context section provides time-related fields (order date, return request date, days since purchase) and channel/store information with geographic coordinates. A product\_snapshot captures product classification at the time of return. The return\_payload object stores the most important operational details: the return reason (nested as an object containing reason\_code and reason\_description), detailed issue records, resolution steps, and a timeline audit trail. Finally, evidence\_media indicates whether supporting images/videos exist and includes media\_refs[] with filenames for linking to staged media files. This structure contains multiple nested arrays (issue\_details, resolution\_steps, timeline\_events, media\_refs), which requires FLATTEN() during transformation into relational staging tables.

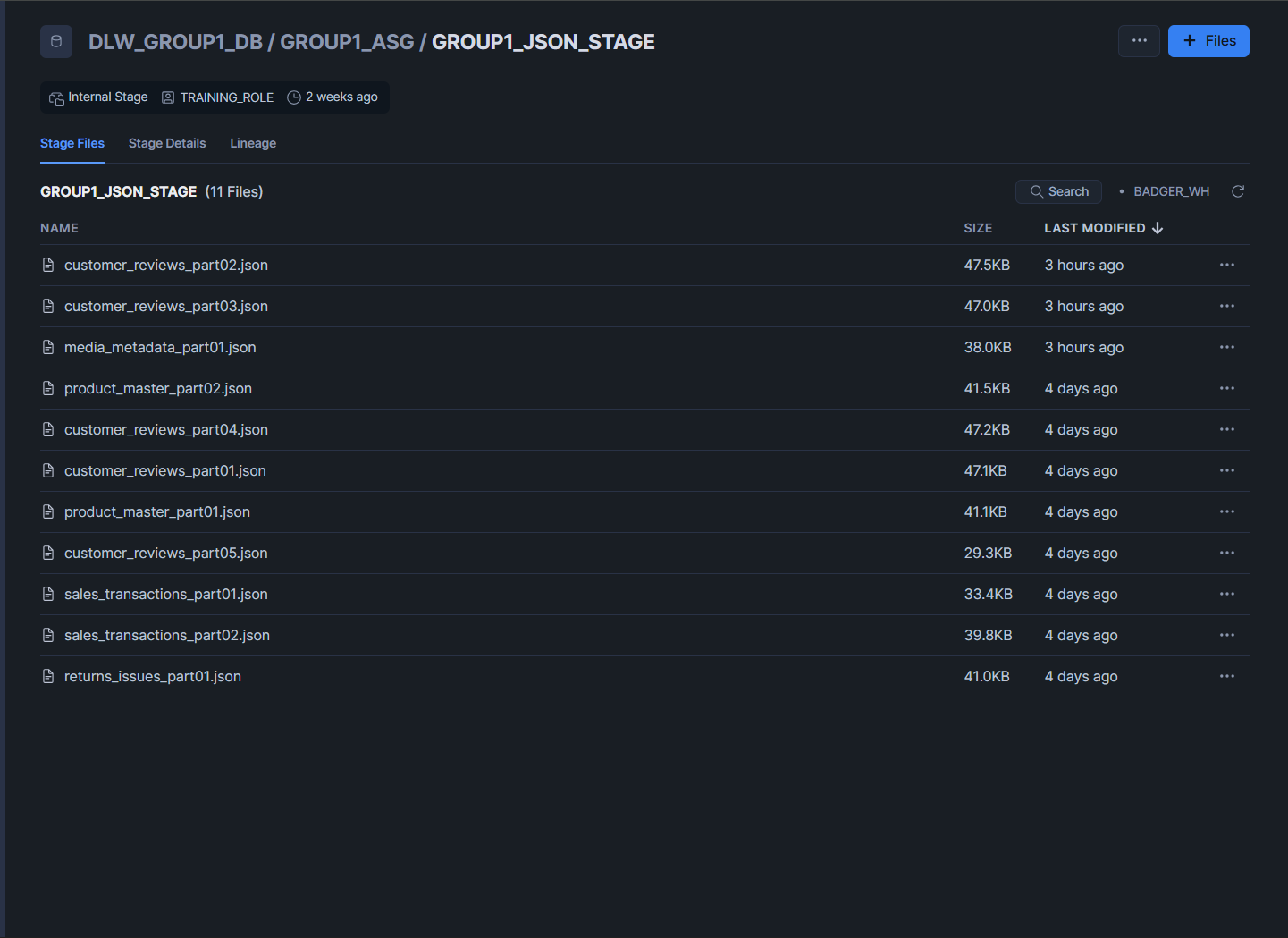
# Proof of uploaded files

## GROUP1\_IMAGES\_STAGE



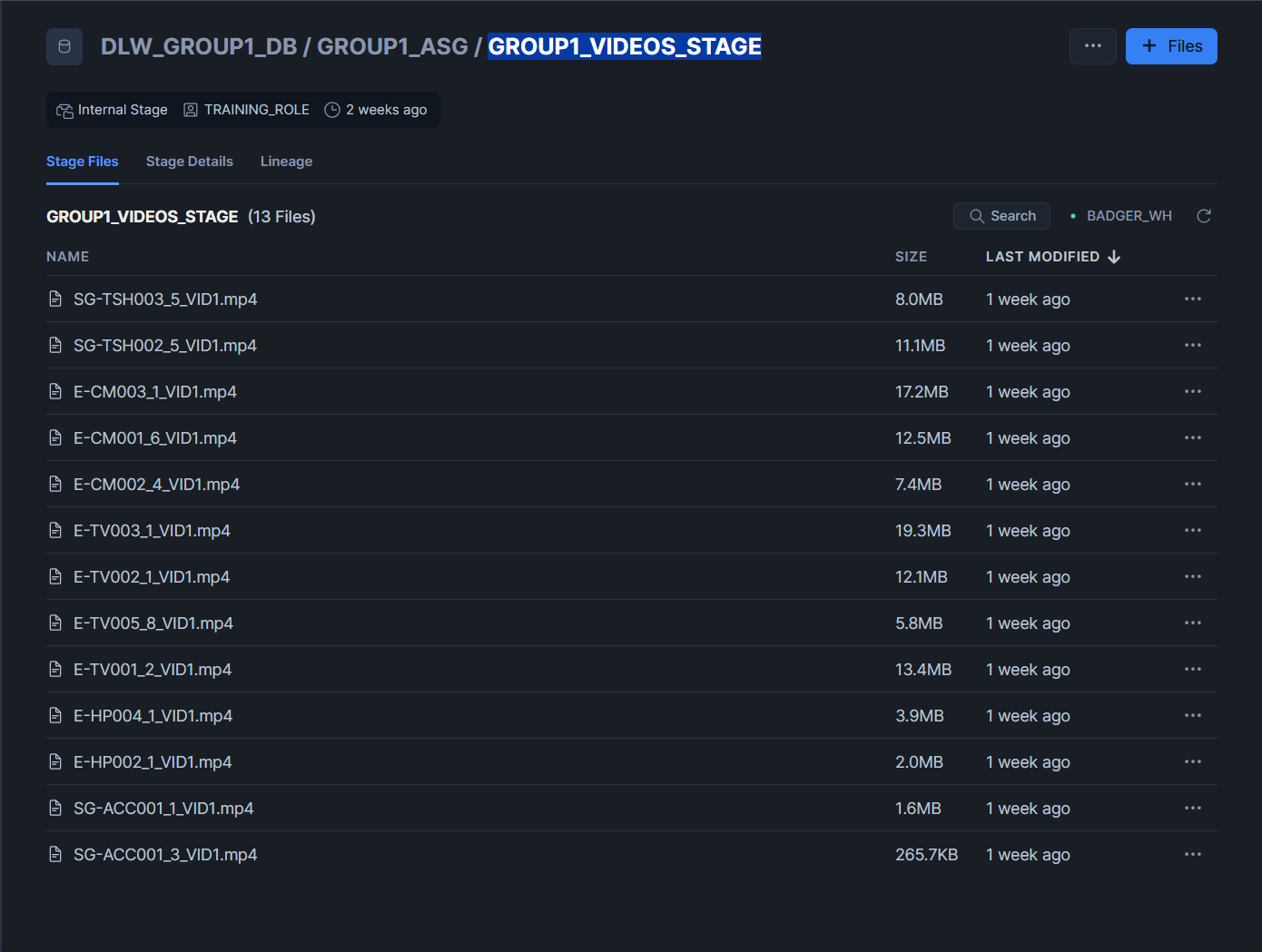
The screenshot shows the contents of the Snowflake internal stage @GROUP1\_IMAGES\_STAGE, which contains 55 image files uploaded as media evidence attached to customer reviews. The stage listing confirms the images were successfully uploaded, displaying key metadata such as file name, file size, and last modified timestamp. These image files are later referenced using DIRECTORY(@GROUP1\_IMAGES\_STAGE) and matched to media metadata to generate scoped image URLs in the curated dataset.

## GROUP1\_JSON\_STAGE



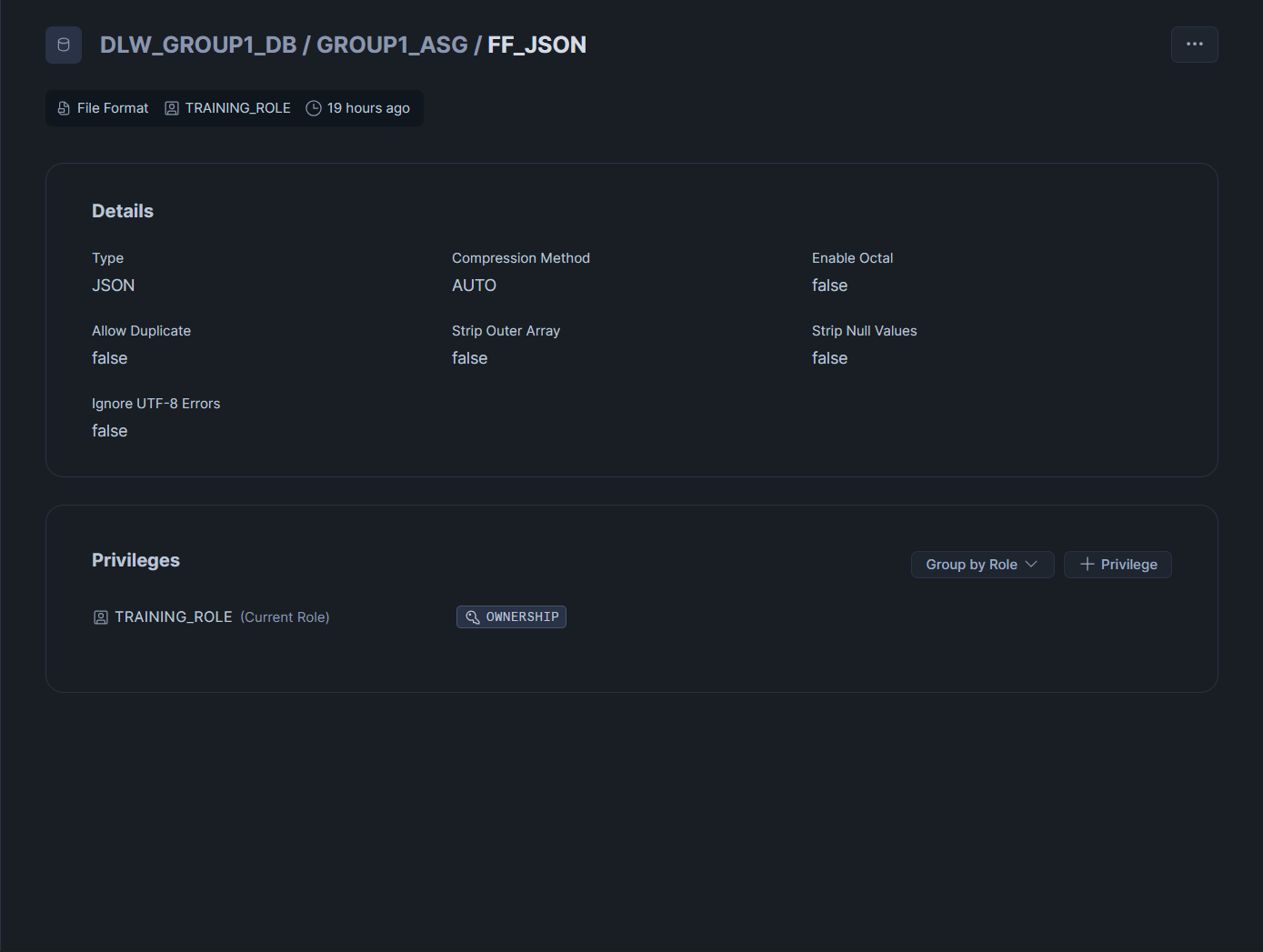
The screenshot shows the contents of @GROUP1\_JSON\_STAGE, containing 11 JSON files uploaded successfully. These files represent the four core analytical datasets required to build the curated view: (1) Product Master (reference/master data), (2) Sales Transactions (transactional performance), (3) Customer Reviews (customer sentiment/feedback), and (4) Returns/Issues (post-sale quality outcomes). In addition, a supporting Media Metadata JSON file is included to link the staged image/video assets to specific feedback via feedback\_id and filenames. This stage listing provides evidence that all JSON sources are present and ready to be ingested into RAW tables as VARIANT before being flattened into structured staging tables and integrated into the final curated dataset.

## GROUP1\_VIDEOS\_STAGE



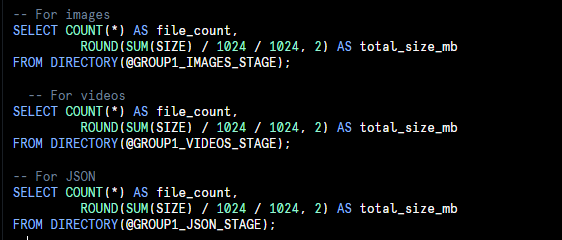
The screenshot shows the contents of @GROUP1\_VIDEOS\_STAGE, which contains 13 video files uploaded as review media evidence. The listing confirms successful upload and provides file metadata like file name, size, last modified. These staged video assets are later retrieved through DIRECTORY(@GROUP1\_VIDEOS\_STAGE) and linked to review feedback records so the curated dataset can include scoped video URLs where applicable.

# Proof of file formats



To prove the file format configuration, I executed SHOW FILE FORMATS; (and/or SHOW FILE FORMATS LIKE 'FF\_%';) to display all available file formats in the schema, followed by DESC FILE FORMAT FF\_JSON; to confirm the exact settings. The screenshot shows that **FF\_JSON** is configured with TYPE = JSON to correctly interpret semi-structured JSON during ingestion, and STRIP\_OUTER\_ARRAY = FALSE so Snowflake preserves the JSON document structure exactly as stored (including top-level arrays such as products[], review\_events[], and media\_records[]). This is important because the pipeline intentionally loads the raw JSON into VARIANT first and then uses LATERAL FLATTEN() during staging to convert nested arrays into relational rows. The file format is referenced directly in every COPY INTO … (FILE\_FORMAT => FF\_JSON) command, so verifying FF\_JSON ensures all raw loads and downstream transformations are based on consistent parsing rules.

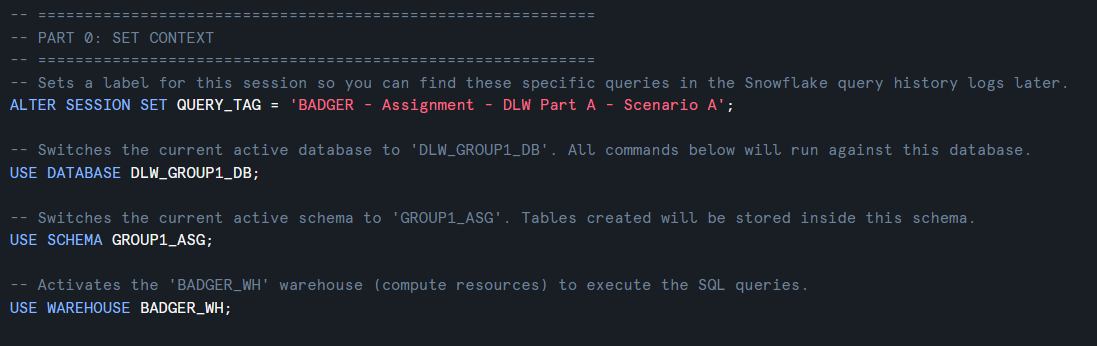
# File Compliance Table



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stage | File Type | File Count | Total Size | Compliance Check |
| GROUP1\_JSON\_STAGE | JSON | 11 files | 0.43 MB | ✓ All >50KB |
| GROUP1\_IMAGES\_STAGE | JPG | 55 files | 3.66 MB | ✓ All >100KB |
| GROUP1\_VIDEOS STAGES | MP4 | 13 files | 114.59 MB | ✓ All >100MB |

# Scripts used

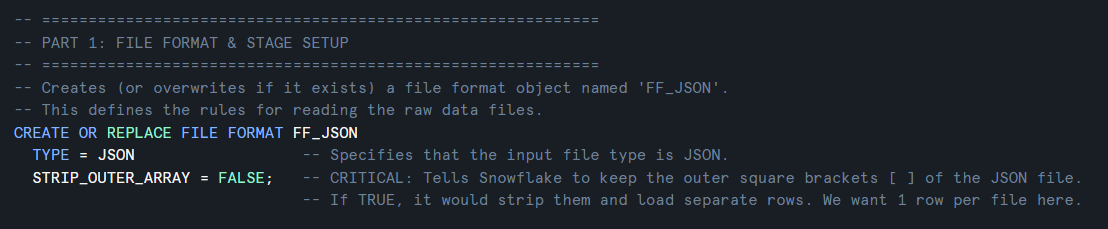
## Part 0- Session Setup and Working Context



This section sets the execution context for the entire workflow:

* **ALTER SESSION SET QUERY\_TAG = ...**:  
  Labels every query executed in this session with a tag. This is useful for tracking and proving work in Snowflake query history.
* **USE DATABASE ...:**  
  Ensures all objects like tables, views, file formats are created inside the correct database.
* **USE SCHEMA ...:**  
  Ensures objects are created inside the correct schema, so that our work does not accidentally go into another schema.
* **USE WAREHOUSE ...:**Specifies the compute warehouse that runs the queries. Without selecting a warehouse, many operations such as COPY INTO, table creation, queries won’t execute.

## Part 1 - JSON File Format Definition (FF\_JSON)

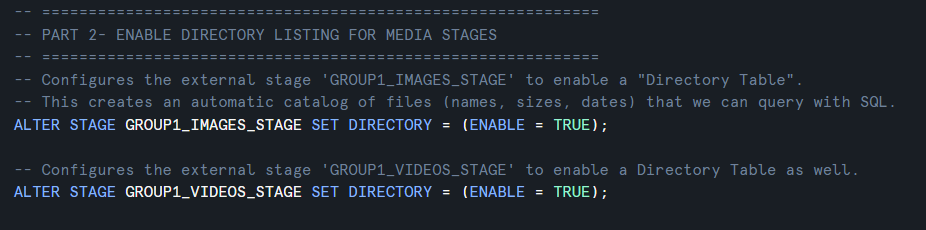


Snowflake needs a file format object to understand how to read staged files. Here we define a JSON file format:

* **TYPE = JSON**:  
  Tells Snowflake the staged files are JSON.
* **STRIP\_OUTER\_ARRAY = FALSE**:   
  Means Snowflake will not automatically remove an outer [...] array wrapper.  
  This is important because many JSON datasets are stored as an array of objects. We load each JSON document into a VARIANT column and later use FLATTEN() to explode arrays into rows.

Using a file format keeps the COPY commands consistent, readable, and reusable.

## Part 2 - Enable Directory Listing for Media Stages



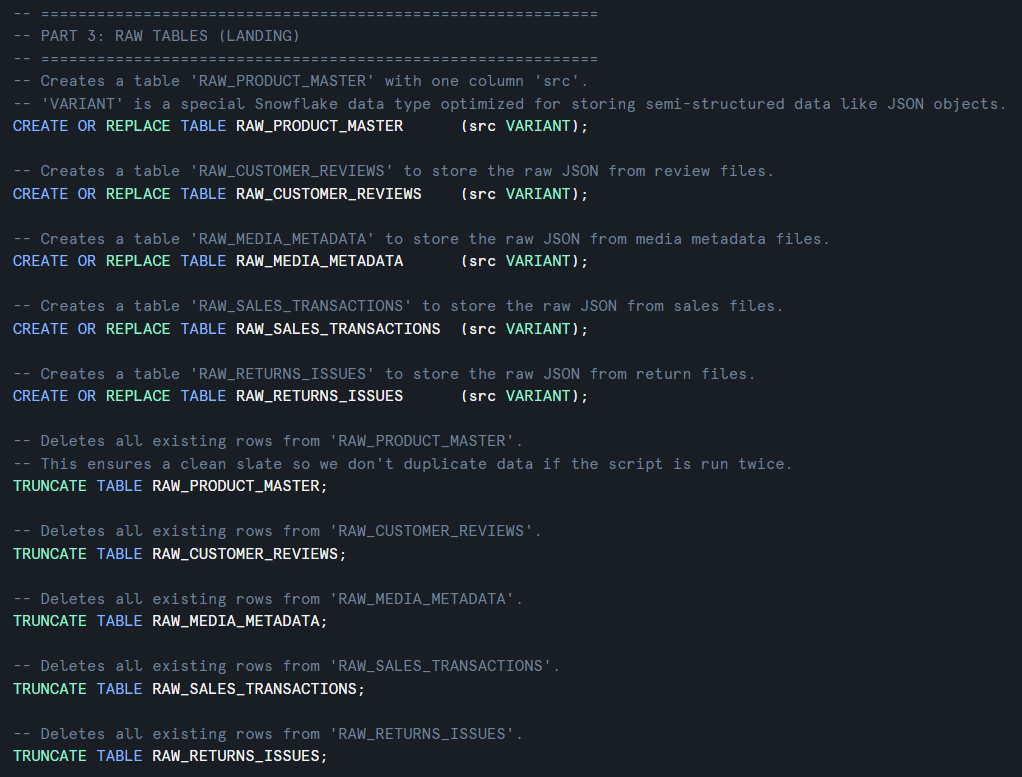
This step enables **directory tables** on the image/video stages. It is required to use:

* **DIRECTORY(@stage)** which returns a structured list of staged files which includes the relative\_path, size, last\_modified, etc.

Why we need this:

* We uploaded **review media** (photos/videos) into stages.
* Later, we need to generate **scoped URLs** so that the curated dataset can reference the correct media files.
* To generate URLs reliably, we must first list files and their paths from the stage.

## Part 3 - RAW Landing Tables (VARIANT Storage)



These are the RAW (landing) tables. Each table has a single column:

* **src VARIANT:**  
  Stores the entire JSON document in Snowflake’s semi-structured format.
* **TRUNCATE**  
  Removes all rows from a table by deallocating the data pages, effectively resetting the table while keeping its structure and columns intact.

Why we do this:

* JSON files often have nested structures and arrays.
* Loading into VARIANT preserves the original structure without forcing a relational schema too early.
* This supports best practice “ELT”: load raw first, then transform.
* In the case where there is duplicate data, TRUNCATE ensures that the all rows are unique and clean.

How this helps map back to assignment:

* RAW tables correspond to the key datasets: master data + transactional data + reviews/media.

## Part 4 - Load JSON Files into RAW Tables (COPY INTO + PATTERN)



This step ingests staged JSON files into RAW tables.

Key code syntax used:

* **COPY INTO <table>:**Loads data from a stage into a table.
* **SELECT $1:**Reads the entire JSON document as a single column value.
* **(FILE\_FORMAT => FF\_JSON):**Uses the JSON parsing rules defined earlier.
* **PATTERN = ...:**Ensures only the intended files are loaded (especially important because multiple JSON files exist in the same stage).

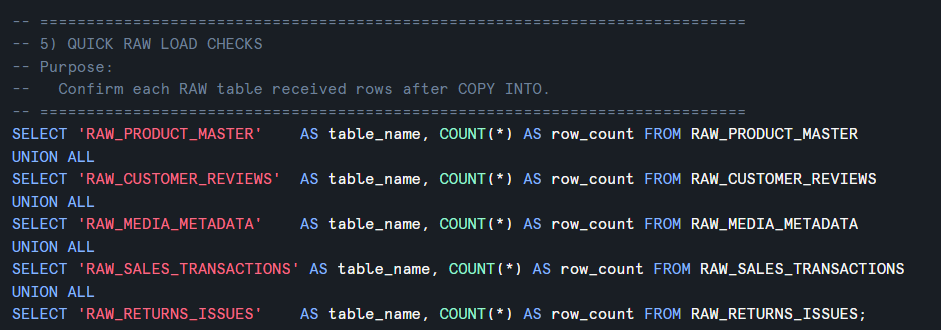
Why pattern matters:

* Your stage contains multiple files such as product\_master, sales\_transactions, reviews, etc.  
  If we don’t filter using PATTERN, we might load wrong files into the wrong RAW tables.

This step is repeated for:

* Product master (part01 + part02)
* Customer reviews (part01–part05)
* Media metadata (part01)
* Sales transactions (part01 + part02)
* Returns/issues (part01)

## Part 5- RAW Load Validation Checks



This produces a single output table showing how many rows were loaded into each RAW table.

Why we include this:

* It is a data ingestion validation step.
* It proves COPY INTO worked successfully.
* If any RAW table has row\_count = 0, it indicates an issue like:
  + wrong filename pattern
  + wrong stage location
  + file format mismatch



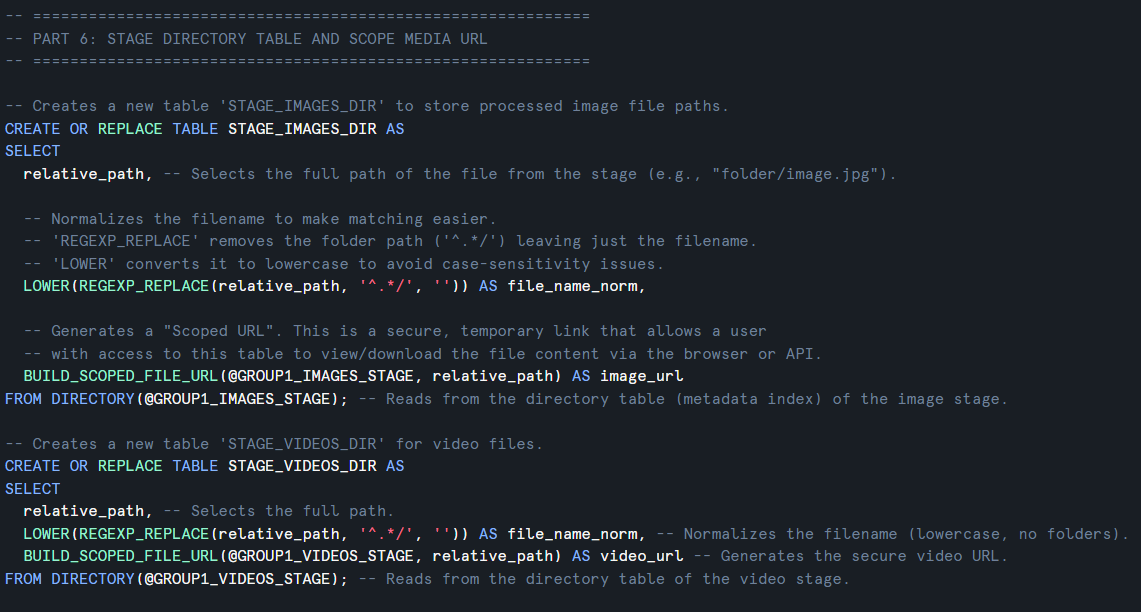
Running the RAW load validation query produced a consolidated row-count output for all RAW tables. Because the COPY INTO statements use SELECT $1, each JSON file is loaded as a single VARIANT “document row”, so the expected counts match the number of JSON part files staged for each dataset. The result confirmed that every RAW table had a non-zero row\_count, meaning the stage path, filename PATTERN filters, and FF\_JSON parsing rules were correct.

As shown in the query result screenshot, the RAW tables loaded successfully as follows:

* RAW\_PRODUCT\_MASTER: 2 rows (product\_master\_part01–part02)
* RAW\_CUSTOMER\_REVIEWS: 5 rows (customer\_reviews\_part01–part05)
* RAW\_MEDIA\_METADATA: 1 row (media\_metadata\_part01)
* RAW\_SALES\_TRANSACTIONS: 2 rows (sales\_transactions\_part01–part02)
* RAW\_RETURNS\_ISSUES: 1 row (returns\_issues\_part01)

This validation is important because a row\_count of 0 would immediately indicate ingestion failure (wrong PATTERN, wrong stage, or file format mismatch). After confirming counts, I also performed a “peek” query (SELECT src … LIMIT 1) to visually verify that each RAW table stored the correct JSON document structure in VARIANT before transformation.

## Part 6- Stage Directory Tables and Scoped Media URLs



This creates a relational lookup table of the actual staged media files.

* **DIRECTORY(@GROUP1\_IMAGES\_STAGE):**Outputs one row per file, including file path and metadata.
* **REGEXP\_REPLACE(relative\_path, '^.\*/', ''):**Removes folders, keeping only the filename.
* **LOWER(...):**Standardizes filenames, making joins reliable (case differences are common).
* **BUILD\_SCOPED\_FILE\_URL(@stage, relative\_path):**Generates a secure scoped URL that can be used to access the file.

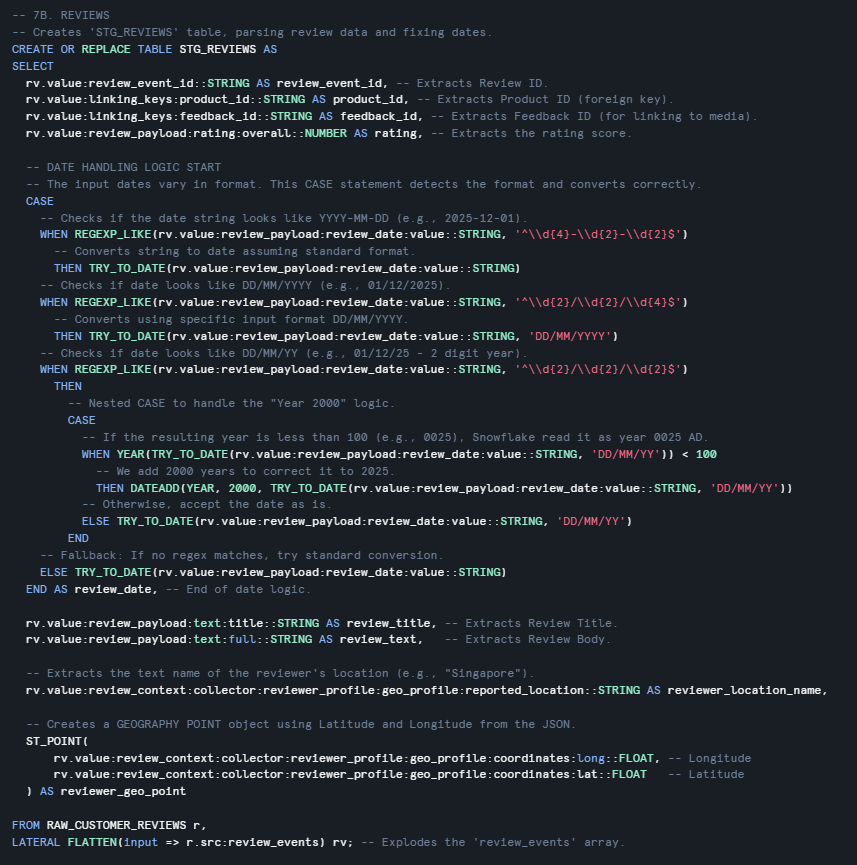
Why we need this:

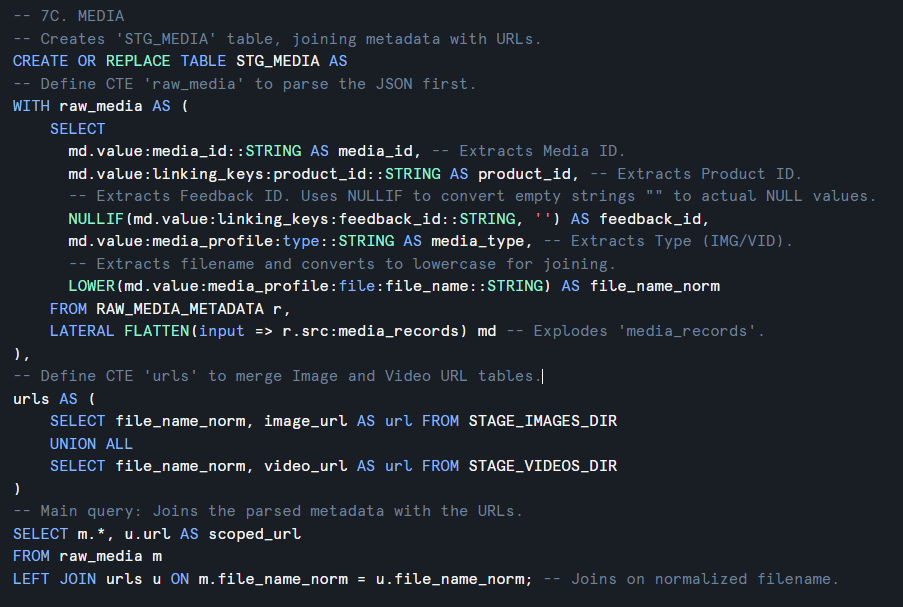
* Media metadata JSON gives file names.
* Stage directory gives actual uploaded files.
* We join them together later to attach the correct URL into the curated dataset.

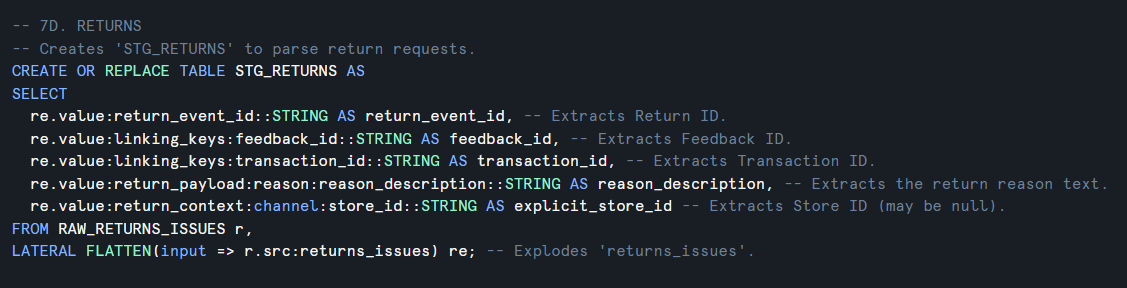
Same logic is applied for videos in STAGE\_VIDEOS\_DIR.

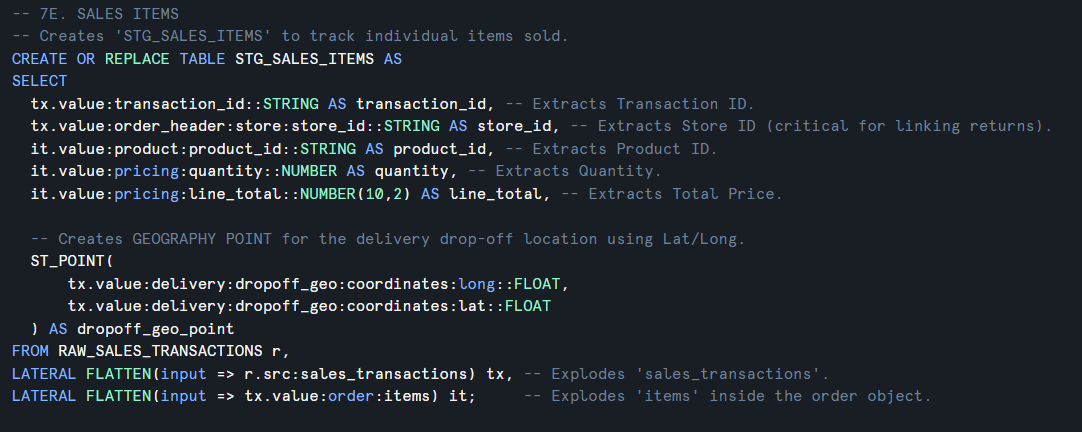
## Part 7 - Transform RAW JSON into Relational Staging Tables (STG\_)

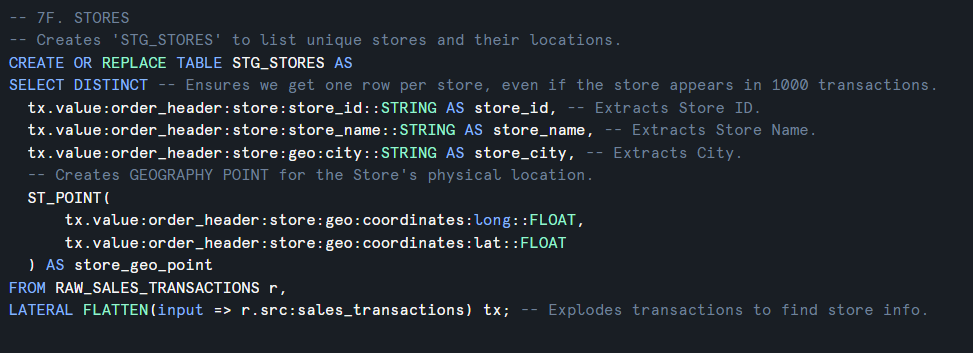












This section converts nested semi-structured JSON (VARIANT) into clean relational staging tables. These STG tables standardise the data like the IDs, datatypes, and grain so that the final curated dataset can accurately integrate product master data + transactions + customer sentiment + review media.

**7A) Product Master → STG\_PRODUCTS**

**Purpose:**   
Create a master reference table that defines each product’s identity and commercial attributes. This table is later used to enrich reviews and sales records.

* The product master JSON contains a top-level array where each element represents a single product.
* **LATERAL FLATTEN():**Is applied to convert each nested product object into **one relational row**.
* **Row grain:**   
  1 row per product\_id
* Key attributes extracted include:
  + product\_id used for primary reference key used throughout the dataset
  + brand + model\_name used for product identity for reporting
  + category/subcategory used for product segmentation and grouping
  + base\_price + currency which is used for revenue context and comparisons

**Outcome:**   
STG\_PRODUCTS becomes the dimension-style reference table that ensures all transaction and feedback data can be tied back to consistent product definitions.

**7B) Customer Reviews → STG\_REVIEWS**

**Purpose:**   
Convert review events into structured rows so customer sentiment can be analysed and linked to sales performance and review media.

* The customer reviews JSON contains an array such as review\_events.
* Each element of review\_events represents one submitted review event.
* Using **LATERAL FLATTEN()**, each review event becomes a row.

**Row grain:**   
**1 row per review\_event\_id**.

**Fields extracted and why they matter:**

* review\_event\_id: unique identifier for each review event (prevents duplicates and supports auditing).
* product\_id: links review to product master and sales.
* feedback\_id: the key used to link reviews to media attachments from the media metadata JSON.
* rating + title + full review text: core sentiment fields used for customer satisfaction analysis.
* reviewer\_id: supports behavioural analysis (e.g., repeat reviewers).
* reviewer\_location\_name: The text name of the reviewer's location (e.g., "Singapore").
* reviewer\_geo\_point: Uses ST\_POINT() to convert raw Latitude and Longitude into a GEOGRAPHY object, enabling distance-based analysis or heat-mapping.

**Robust date parsing:**

* Review dates may appear in multiple formats (e.g., DD/MM/YYYY and DD/MM/YY).
* TRY\_TO\_DATE() is used so invalid formats return NULL instead of failing the pipeline.
* COALESCE() attempts multiple formats to maximise successful parsing and preserve as many records as possible.

**Outcome:** STG\_REVIEWS provides reliable review-level structure and clean linking keys for integration.

**7C) Media Metadata → STG\_MEDIA**

**Purpose:**   
Transform media metadata into a structured table that connects reviews to their attached photos/videos.

* The media metadata JSON contains an array such as **media\_records**.
* Each element represents one media item (image or video).
* LATERAL FLATTEN() converts each record into one row.

**Row grain:**   
**1 row per media\_id**.

**Key fields extracted:**

* media\_id: unique media record identifier.
* product\_id: allows product-level media analysis.
* feedback\_id: enables direct join to reviews.
* media\_type (IMG/VID): supports separating photos vs videos.
* file\_name + file extension: needed to match physical files in Snowflake stages.

**URL merging and common table expressions (CTE):**

* raw\_media CTE**:** Handles the initial parsing and flattening of the JSON metadata.
* urls CTE**:** Consolidates file paths by performing a UNION ALL between STAGE\_IMAGES\_DIR and STAGE\_VIDEOS\_DIR. This creates a single source for all media locations regardless of type.

**File matching preparation:**

* file\_name\_norm is created by lowercasing the filename.
* This standardises filenames for joins against stage directory tables.

**Outcome:**   
STG\_MEDIA is the bridge between structured reviews and unstructured evidence (photos/videos).

**7D) Returns/Issues → STG\_RETURNS**

**Purpose:**   
Transform return events into a structured format to incorporate “post-sale outcome signals” into the final dataset (e.g., return counts and return reasons).

* The returns/issues JSON contains an array such as returns\_issues.
* Each element represents one return or issue event.
* LATERAL FLATTEN() converts each return event into one row.

**Row grain:**   
**1 row per return\_event\_id**

**Key Fields Extracted:**

* return\_event\_id: Unique identifier for the return event.
* feedback\_id: Used to link the return back to specific customer feedback or reviews.
* transaction\_id: The critical link to the original sales transaction to analyze return rates.
* reason\_description: The text explanation for why the item was returned, essential for identifying product defects or quality issues.
* explicit\_store\_id: Extracts the specific channel or store ID where the return was processed, allowing for location-based performance analysis.

**Handling nested reason fields:**

* Reason may appear as a nested OBJECT rather than plain text.
* To prevent missing values and schema mismatch errors:
  + reason\_context is extracted from reason:reason\_context

**Outcome:**   
STG\_RETURNS enables return counts and reason summaries to be integrated into curated analysis (helpful for quality/issues insights).

**7E) Sales Transactions → STG\_SALES\_ITEMS**

**Purpose:**   
Convert nested transaction JSON into line-item rows, which is the most accurate level for measuring units sold and revenue.

Sales JSON is hierarchical:

* A transaction contains header-level information (date, channel, customer,etc)
* It also contains an array of order items (products purchased in that transaction)

**Double flatten strategy:**

1. First flatten **sales\_transactions** → transaction-level objects
2. Then flatten **order:items** → item-level objects

**Row grain:**

**1 row per transaction\_id**

**Key Fields Extracted:**

* transaction\_id: Unique identifier for the overall sale.
* store\_id: Extracted from the order\_header, critical for linking returns and analyzing performance by location.
* product\_id: Identifies the specific item sold.
* quantity + line\_total: Numeric values for units sold and revenue, with line\_total cast to a specific decimal precision (10,2).
* dropoff\_geo\_point: Uses ST\_POINT() to convert delivery longitude and latitude coordinates into a GEOGRAPHY object.

**Why line-item grain is best:**

* Prevents double counting when multiple products exist within one transaction.
* Enables accurate calculation of:
  + units sold (SUM(quantity))
  + total revenue (SUM(line\_total))
* Supports correct joins to:
  + product master via product\_id
  + returns via order\_item\_id (when applicable)
  + feedback via product/review linkage

**Outcome:**  
STG\_SALES\_ITEMS forms the transactional foundation for product performance metrics.

**7F) Stores → STG\_STORES**

**Purpose:** Create a reference list of unique stores and their physical locations by extracting store data from transaction headers.

**Row Grain:** 1 row per unique store\_id.

**Key Fields Extracted:**

* store\_id: Unique identifier for the store.
* store\_name: The descriptive name of the retail location.
* store\_city: The city where the store is located.
* store\_geo\_point: Uses ST\_POINT() to convert raw longitude and latitude into a GEOGRAPHY object for geospatial analysis.

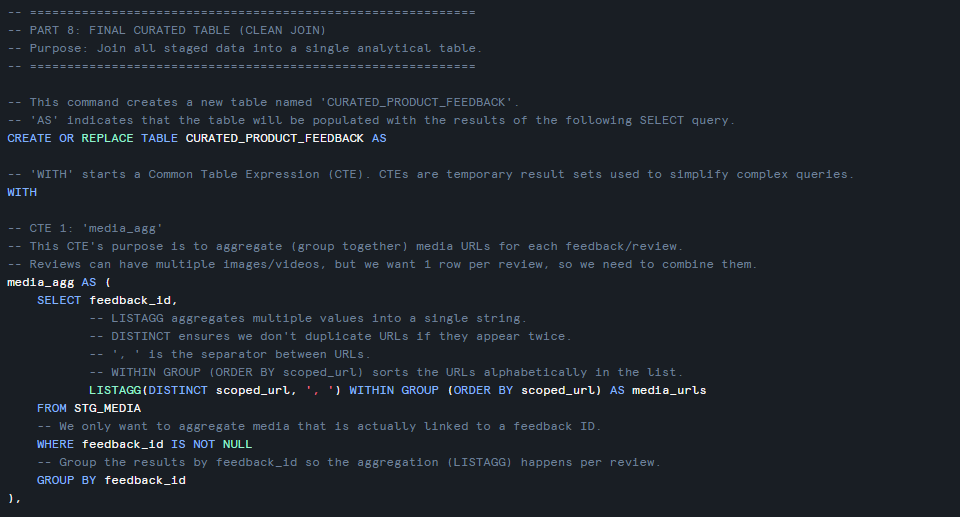
**Logic:**

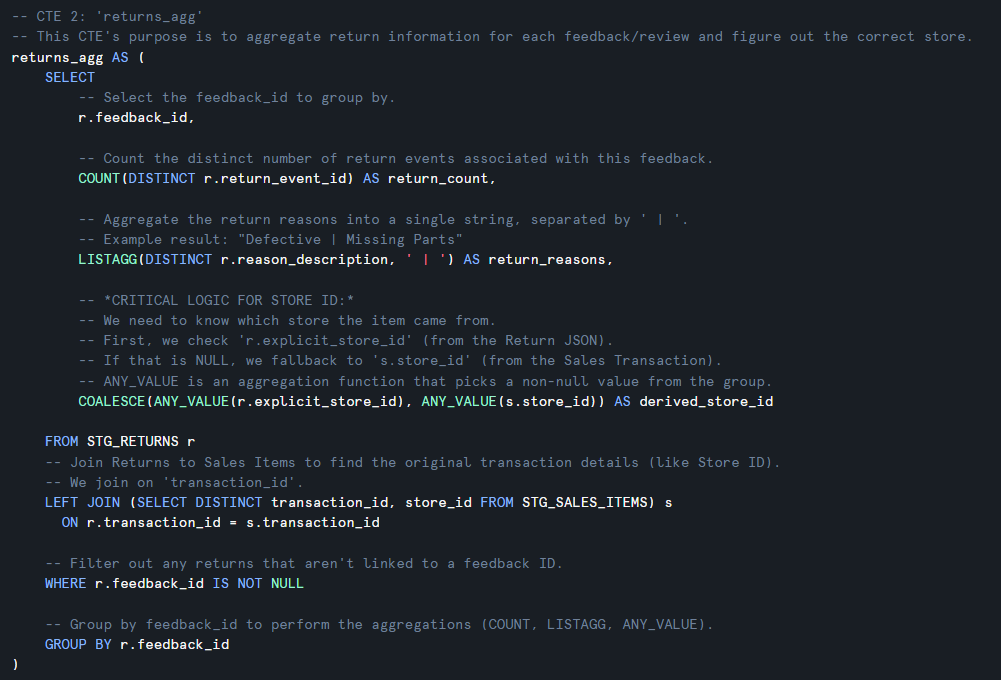
Uses LATERAL FLATTEN() on sales\_transactions to reach the store information nested within the order header.

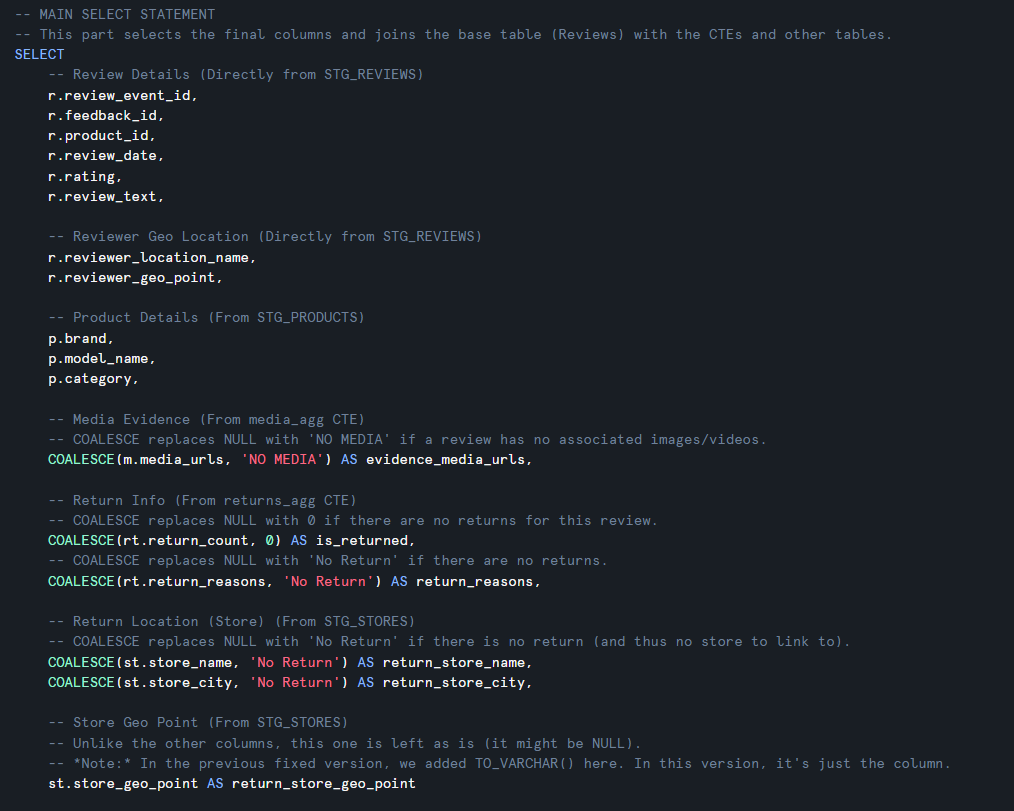
* SELECT DISTINCT ensures that even if a store appears in thousands of transactions, it only occupies one row in this dimension-style table.

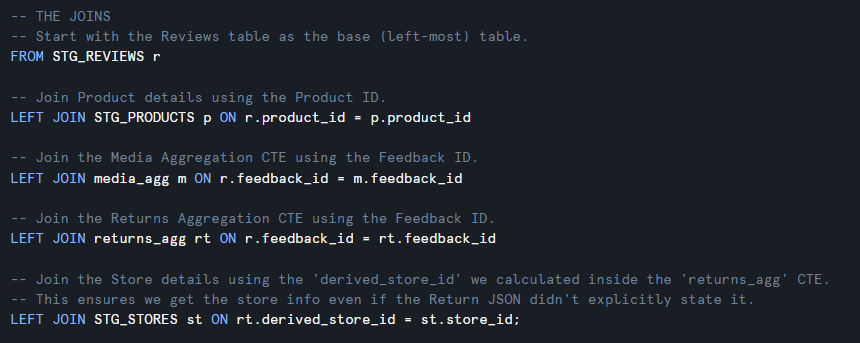
**Outcome:** STG\_STORES provides a clean master list for location-based performance analysis and mapping store distributions.

## Part 8 - Build Final Curated Dataset (Review-Grain Integration)









This step produces the final curated dataset used for analysis of product sales performance vs customer sentiment and media evidence. The curated table is intentionally designed at review-grain (1 row per review\_event\_id) so each record represents a single customer feedback event enriched with product context, sales outcomes, media evidence, and return signals.

**Why Pre-aggregation is Required**

Directly joining STG\_REVIEWS to STG\_MEDIA or STG\_RETURNS would create row duplication (e.g., one review × multiple media files or multiple return events). To prevent double counting and maintain the review-level grain, these data points are aggregated into Common Table Expressions (CTEs) before the final join.

**CTE 1: media\_agg (one row per feedback\_id)**

* Purpose: Groups media URLs by feedback\_id so they can be joined back to a single review without duplicating rows.
* Logic: Uses LISTAGG(DISTINCT scoped\_url, ', ') to concatenate all valid, unique media links into a single comma-separated string (media\_urls).
* Filtering: Specifically filters for records where feedback\_id IS NOT NULL to ensure the link to the review remains valid.

**CTE 2: returns\_agg (one row per feedback\_id)**

* Purpose: Aggregates return information for each unique feedback/review and identifies the correct store of origin.
* Metrics: Calculates return\_count using COUNT(DISTINCT return\_event\_id) and combines multiple return reasons into a single string using LISTAGG.
* Critical Store Logic: Uses COALESCE(ANY\_VALUE(r.explicit\_store\_id), ANY\_VALUE(s.store\_id)) to determine the derived\_store\_id.
* It first checks the return metadata, if missing, it falls back to the original sales transaction record.
* This is achieved by joining STG\_RETURNS with STG\_SALES\_ITEMS on transaction\_id.

**Final Integration (LEFT JOIN Strategy)**

The final SELECT statement uses a LEFT JOIN strategy starting with STG\_REVIEWS as the base table to ensure no customer feedback is lost.

Join Chain: Reviews are joined to STG\_PRODUCTS (on product\_id), the media\_agg CTE, and the returns\_agg CTE (both on feedback\_id).

Store Dimension: A final join to STG\_STORES is made using the derived\_store\_id calculated in the returns CTE to bring in store names and locations.

Geospatial Data: Includes both the reviewer's location (reviewer\_geo\_point) and the return store's location (return\_store\_geo\_point) for mapping analysis.

**Reporting-Friendly Null Handling**

To improve dashboard readability and prevent misinterpretation, COALESCE is used to replace NULL values with meaningful labels:

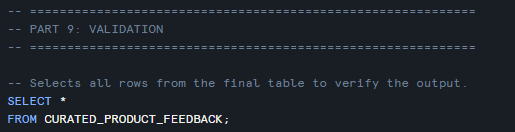
* Media: Missing links are labeled 'NO MEDIA'.
* Returns: If no return exists, the record shows 0 for count and 'No Return' for reasons.
* Location: Missing store data is labeled 'No Return' (since no return implies no return store).

**Outcome:** CURATED\_PRODUCT\_FEEDBACK forms a comprehensive, clean, and analysis-ready tabl that bridges the gap between what customers say (Reviews/Media) and what they do (Sales/Returns).

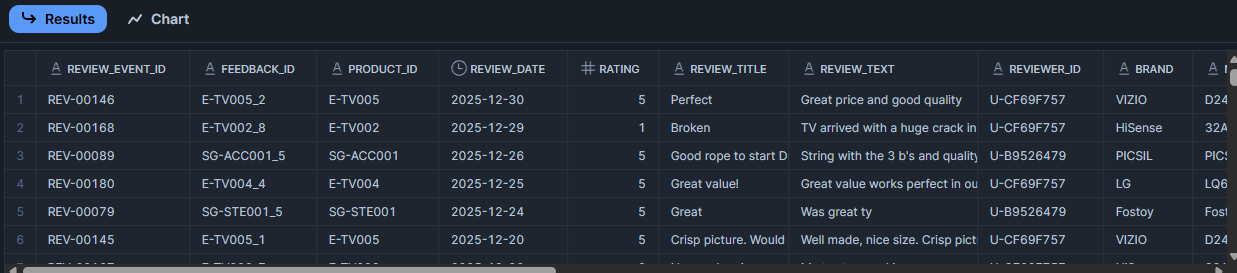
## Part 9 - Curated Dataset Validation and Profiling

**1) Preview:**  
Shows final curated results sorted by latest review\_date.

**SQL Code**

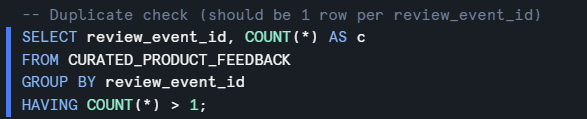


**Output**

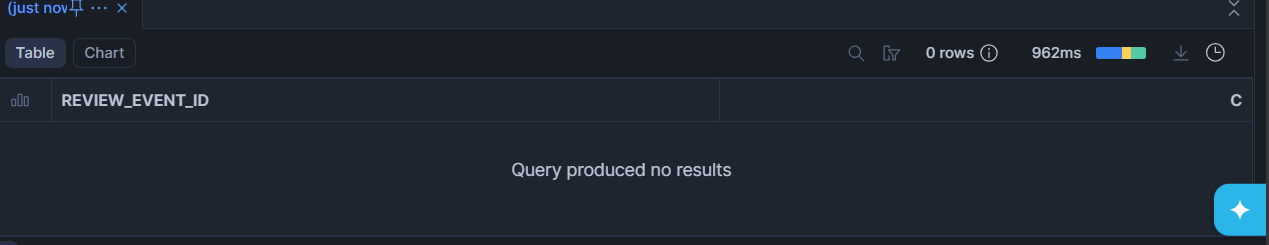


**2) Duplicate check:**  
Ensures one row per review\_event\_id.  
If duplicates exist, it means the grain was broken.

**SQL Code**



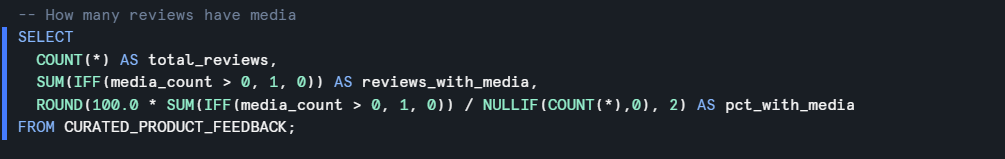
**Output**



**3) Media coverage summary:**  
Counts how many reviews have media evidence.

These checks strengthen the report because they prove the dataset is correct and well-designed.

**SQL Code**



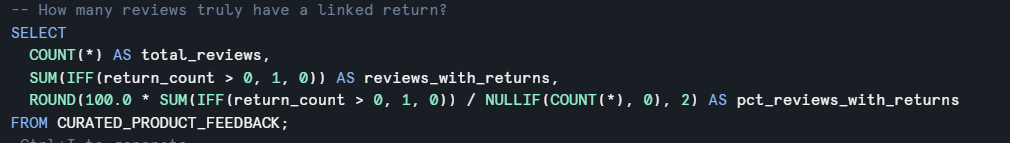
**Output**



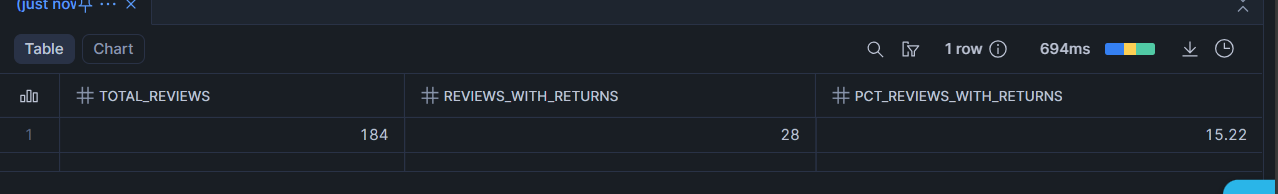
**4) Double checking returns count**

Validates that the number of reviews linked to a return in the final curated layer matches the expected volume. This ensures no data was lost or duplicated during the transformation process.

**SQL Code**



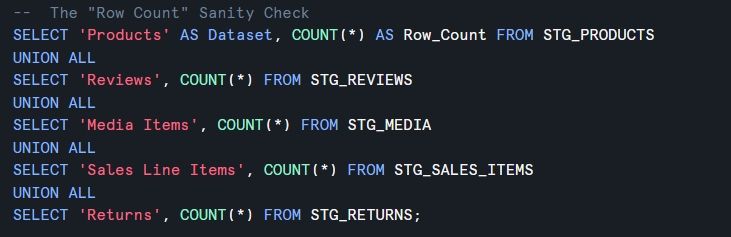
**Output**



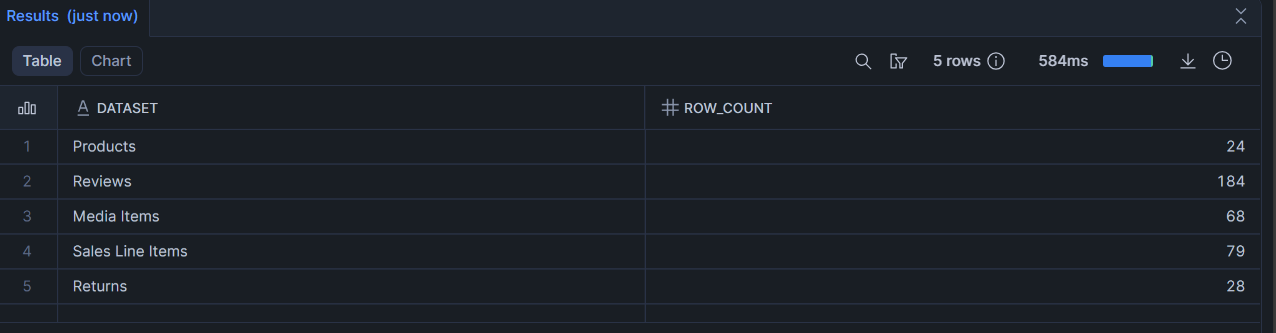
**5) Row Count Sanity Check**

This code provides the number of rows across all staging staging tables such as Products, Reviews, Media, Sales and Returns so that we can confirm that fact check the curated\_product\_feedback with the record counts.

**SQL Code**



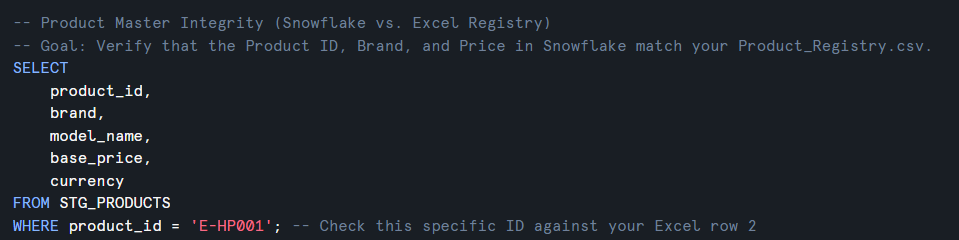
**Output**



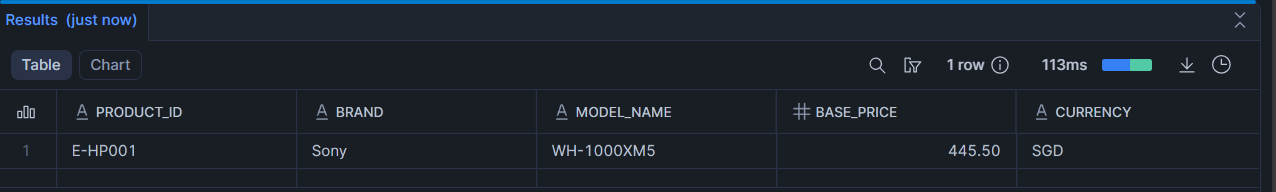
**6) Product Master Integrity**

Verifies that key attributes such as Product ID, Brand, Model Name, and Price in the Snowflake database align perfectly with the external Product\_Registry.csv. This spot-check (e.g., for product 'E-HP001') ensures that the source data was ingested accurately without any corruption of core product details.

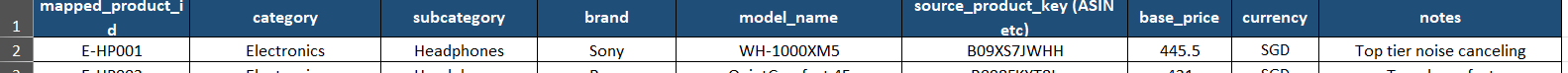
**SQL Code**



**Output**



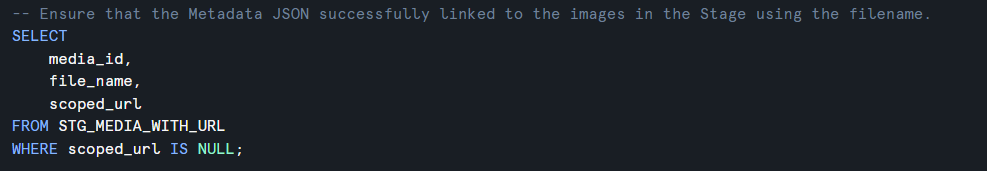
**E-HP001 in our Excel Sheet**



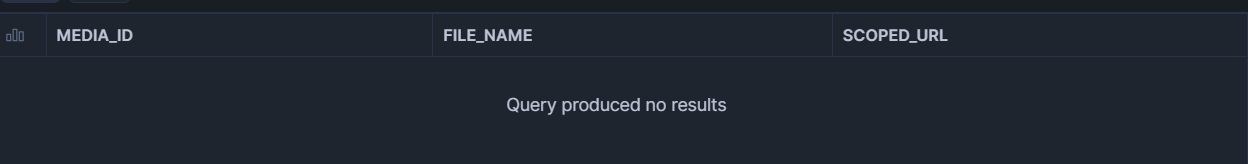
**7) Verifying that all Metadata has been linked to all the images**

Ensures that the metadata form JSON files have successfully linked to the actual images in the cloud storage. By filtering for any NULL scoped\_url values, this check identifies whether any of our media files failed to generate a valid access path during the ingestion process.

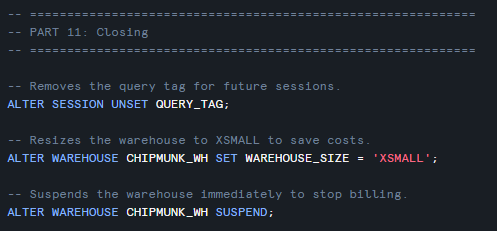
**SQL Code**



**Output**



## Part 10 - Warehouse Cleanup



* Unsets query tag which is a good housekeeping measure
* Reduces warehouse size for cost control
* Suspends warehouse to stops compute usage

# Final curated dataset



The final curated dataset, CURATED\_PRODUCT\_FEEDBACK, is designed to support the scenario “Product Sales and Customer Feedback” by integrating product master data, customer review sentiment, review media evidence, sales performance, and return signals into a single analysis-ready table.

Grain (row definition):

* 1 row = 1 review event (unique REVIEW\_EVENT\_ID).  
  This grain is chosen because customer sentiment and media are captured at the review level, and it is the most logical level to analyse feedback against product performance.

What the curated dataset contains (integrated sources):

1. Review facts (Customer Feedback):
   * REVIEW\_EVENT\_ID, FEEDBACK\_ID, PRODUCT\_ID, REVIEW\_DATE, RATING, REVIEW\_TITLE, REVIEW\_TEXT, REVIEWER\_ID  
     These fields represent customer sentiment and qualitative feedback including rating plus their review.
2. Product enrichment (Master Data):
   * BRAND, MODEL\_NAME, CATEGORY, SUBCATEGORY, BASE\_PRICE, CURRENCY  
     These attributes come from the product master JSON and provide consistent product context for analysis.
3. Feedback media evidence (Images/Videos linked to reviews):
   * MEDIA\_COUNT, MEDIA\_URLS  
     Media items are linked using FEEDBACK\_ID and resolved to scoped file URLs from the image/video stages.
   * If a review has no media attached, MEDIA\_URLS is set to “NO MEDIA” to avoid NULLs and to clearly indicate absence of media.
4. Sales performance metrics (Product-level aggregation):
   * UNITS\_SOLD, TOTAL\_REVENUE  
     These are calculated from sales transactions and aggregated by PRODUCT\_ID, allowing analysis such as whether higher-rated products generate higher revenue.
5. Return signal (Product-level aggregation):
   * RETURN\_COUNT, RETURN\_REASON\_DESCRIPTIONS  
     Returns/issues are aggregated by PRODUCT\_ID to provide a post-purchase outcome indicator.
   * If a product has no recorded returns, RETURN\_REASON\_DESCRIPTIONS is set to “NO RETURN”.

By combining review sentiment, media evidence, sales, and returns in one table, stakeholders can directly analyse questions such as:

* Do products with higher ratings generate higher revenue and unit sales?
* Are negative reviews associated with return reasons?
* Do reviews with supporting media correlate with stronger/clearer sentiment trends?

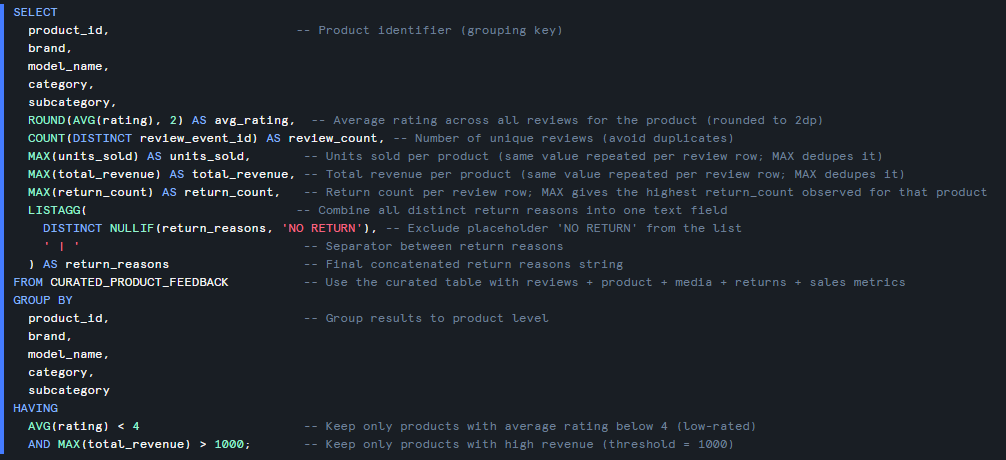
## Sample Analytical Queries on the Final Curated Dataset

Query 1: Products with High Revenue but Low Average Ratings

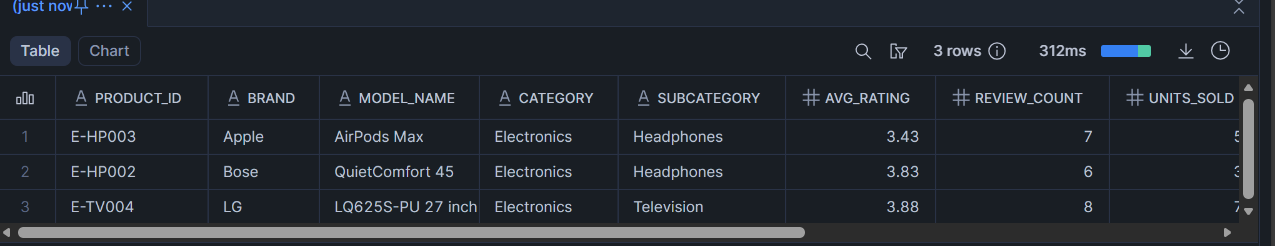
Business Question: Which products are generating strong sales revenue despite receiving lower customer satisfaction scores?

These products may indicate quality issues that needs attention or opportunities for product improvement:

**SQL Code**



**Output**

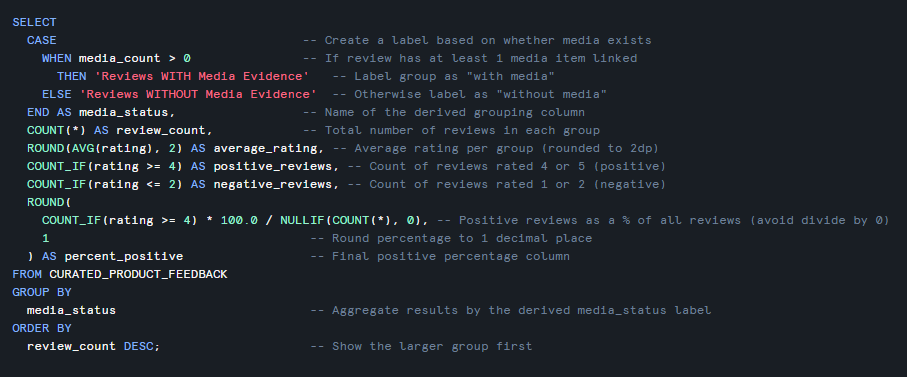


QUERY 2: Impact of Media Evidence on Review Ratings

Business Questions: Do customer reviews that include photo or video evidence tend to have different ratings compared to text-only reviews?

Media evidence may indicate stronger sentiment (very positive or very negative experiences).

**SQL Code**



**Output**

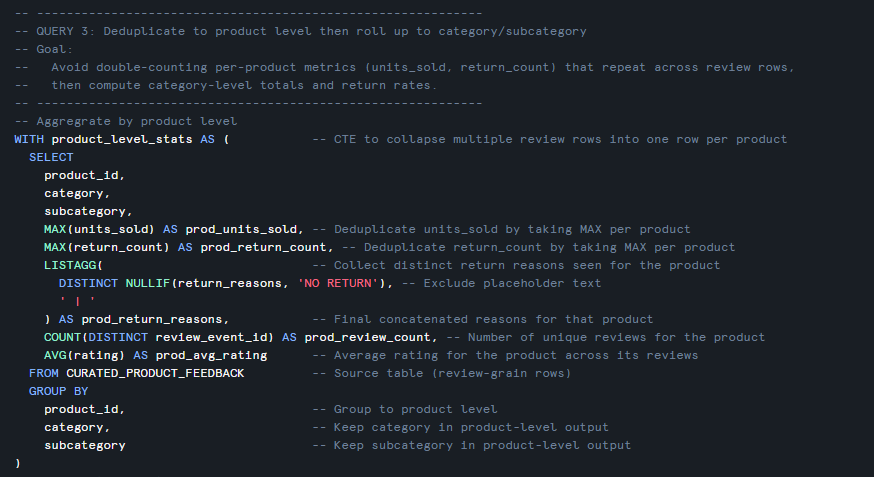


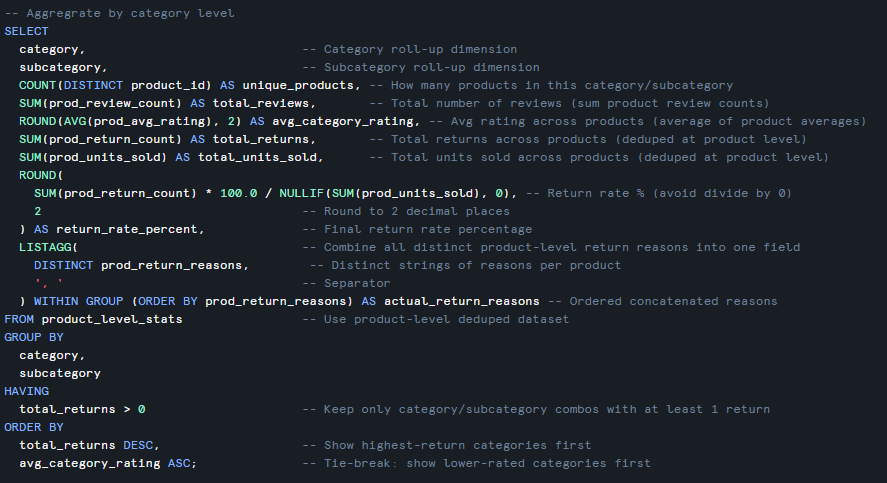
QUERY 3: Product Categories with Highest Return Rates and Average Ratings

*Business Question*: Which product categories have the most returns, and how do their customer ratings compare?

This helps identify systematic quality issues by category and prioritize improvement efforts.

**SQL CODE**

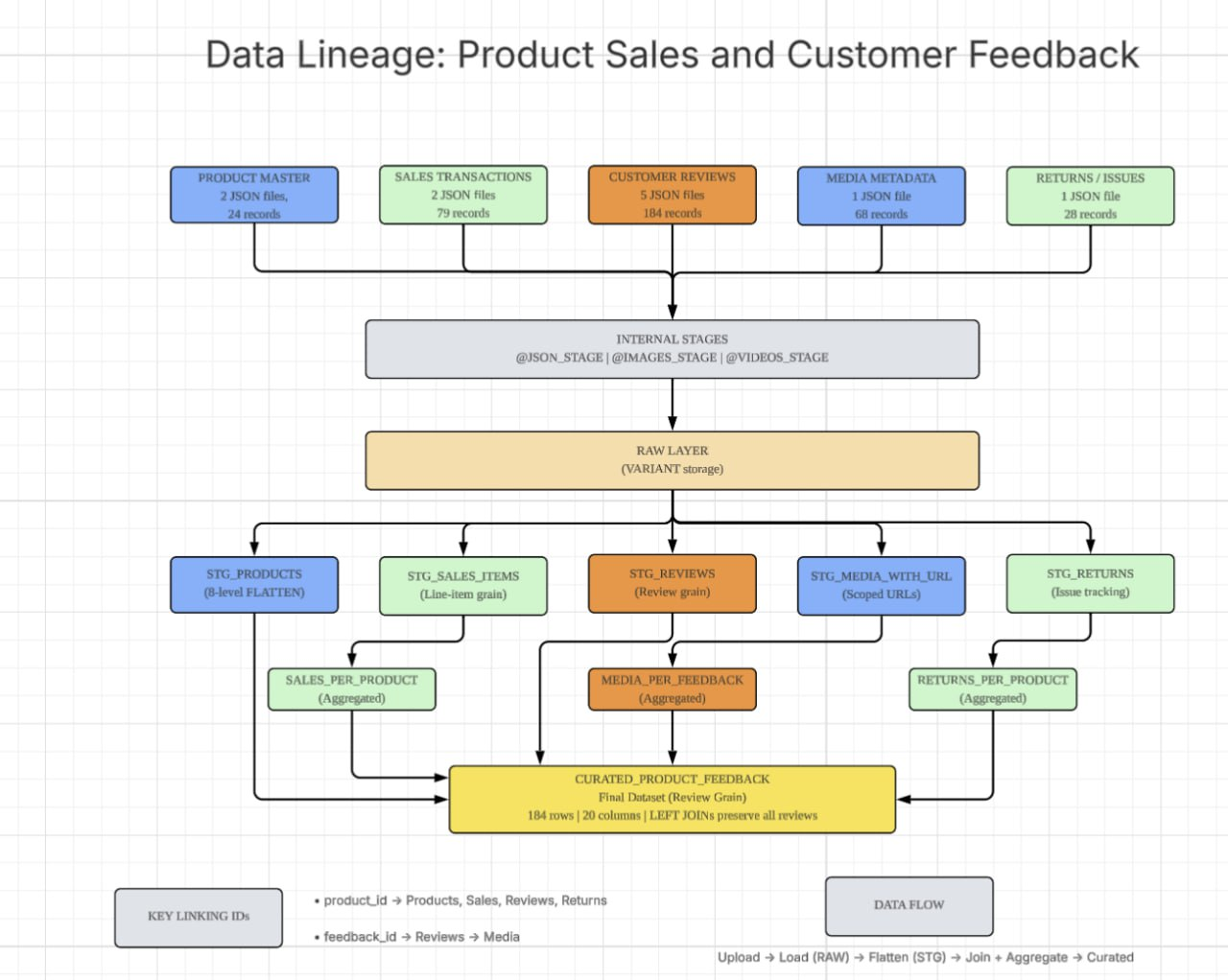




**Output**



# Data Lineage Diagram



# Summary

This assignment implements a comprehensive data engineering solution for analysing product sales performance against customer feedback and sentiment. Our team designed and generated 11 JSON files across 5 data domains, creating a production-ready data pipeline in Snowflake that integrates structured, semi-structured, and unstructured data.

Dataset Overview:

* 184 customer reviews with ratings, feedback analysis, and reviewer geo-profiles
* 24 unique products across Electronics and Sporting Goods categories
* 79 Sales transactions capturing purchase activity and order fulfilment
* 68 combined total of videos and images linked to customer feedback
* 28 return/issue records tracking post-purchase product problems

Technical Implementation: Our collective group effort demonstrates advanced snowflake capabilities including:

* 8-level JSON hierarchy flattening in Product Master files (60% above the 5-level requirement)
* Multi-step LATERAL FLATTEN operations for complex nested arrays at different hierarchy depths.
* BUILD\_SCOPED\_FILE\_URL generation for secure access to unstructured media files
* Review-grain curated dataset design with pre-aggregated CTEs to prevent row duplication
* Used LEFT JOIN to preserve all 184 reviews with product, sales, media, and return context.

Data Quality:

* No inconsistencies with all the 184 reviews in the JSON files as it exactly matches the 184 rows in Excel source tracking which can be found here:



* Included geospatial coverage as we provided coordinates for all 4 different levels such as supplier HQ, store locations, customer locations, and review locations.
* Built-in validation: duplicate checks, NULL handling with COALESCE, media URL coverage verification
* Professional ID linking strategy: product\_id and feedback\_id maintain referential integrity across all domains

Business Value: The final CURATED\_PRODUCT\_FEEDBACK table enables stakeholder to analyse:

* Correlation between sales performance (units sold & revenue) and customer ratings
* The Impact of having media (Images & Videos) on reviews
* Product categories with high return rates and their associated quality issues
* Geographic patterns in customer satisfaction and purchasing behaviour.

Overall, the solution follows industry-standard data engineering practices with a layered architecture (RAW → STG → CURATED). This makes it suitable for production deployment and easy analytical workloads.