Project 07
Building a Data Engineering Pipeline for Business Decision Making using Databricks

Deepika Mandapalli

22nd December, 2025

Project Overview

- This project builds an end-to-end data engineering pipeline using Databricks, Delta Lake, ADLS Gen2, and Power BI.

- The dataset is processed through Bronze, Silver, and Gold layers following the Medallion Architecture.

- The final output supports business decision making through curated KPIs and Power BI dashboards.

Environment Setup

Component	Name / Details
Databricks Workspace	databricksprojects
Cluster	All-purpose
Catalog	retail
Schemas	landing_zone/bronze / silver / gold
Compute	Databricks Cluster
File Formats	CSV /Delta
Reporting Tool	Power BI Desktop

Dataset

- Dataset name: retail_data.csv
- Source:https://www.kaggle.com/datasets/bhavikjikadara/retail-transactional-dataset
- Format: CSV
- Rows: 302,011
- Columns: 30
- Number of records: 302,011
- Size: 80 MB

Column Name	Description
Transaction_ID	Unique identifier for each transaction
Customer_ID	Unique identifier for each customer
Name	Full name of the customer
Email	Customer's email address
Phone	Customer's contact phone number
Address	Street address of the customer
City	City where the customer or transaction is located
State	State or region of the customer
Zipcode	Postal or ZIP code
Country	Country of the transaction
Age	Age of the customer
Gender	Gender of the customer
Income	Income category of the customer (Low, Medium, High)
Customer_Segment	Type of customer segment (e.g., Regular, Premium)
Date	Date when the transaction occurred
Year	Year extracted from the transaction date
Month	Month extracted from the transaction date
Time	Time at which the transaction occurred
Total_Purchases	Number of items purchased in the transaction
Amount	Price per unit of the product
Total_Amount	Total transaction value (Amount × Total_Purchases)
Product_Category	High-level category of the product
Product_Brand	Brand of the product
Product_Type	Type or sub-category of the product
Feedback	Customer feedback on the transaction
Shipping_Method	Shipping method used (e.g., Same-Day, Standard)
Payment_Method	Payment method used (e.g., Credit Card, Debit Card)
Order_Status	Current status of the order (e.g., Shipped, Processing)
Ratings	Customer rating for the transaction or product
products	Name or description of the purchased product

1 - 15 Columns

retail_d

Transaction_ID	Customer_ID	Name	Email	Phone	Address	City	State	Zipcode	Country	Age	Gender	Income	Customer_Segment	Date
8691788	37249	Michelle Harrington	Ebony39@gmail.com	1414786801	3959 Amanda Burgs	Dortmund	Berlin	77985	Germany	21	Male	Low	Regular	9/18/2023
2174773	69749	Kelsey Hill	Mark36@gmail.com	6852899987	82072 Dawn Centers	Nottingham	England	99071	UK	19	Female	Low	Premium	12/31/2023
6679610	30192	Scott Jensen	Shane85@gmail.com	8362160449	4133 Young Canyon	Geelong	New South Wales	75929	Australia	48	Male	Low	Regular	4/26/2023
7232460	62101	Joseph Miller	Mary34@gmail.com	2776751724	8148 Thomas Creek Suite 100	Edmonton	Ontario	88420	Canada	56	Male	High	Premium	05-08-23
4983775	27901	Debra Coleman	Charles30@gmail.com	9098267635	5813 Lori Ports Suite 269	Bristol	England	48704	UK	22	Male	Low	Premium	01-10-24
6095326	41289	Ryan Johnson	Haley12@gmail.com	3292677006	532 Ashley Crest Suite 014	Brisbane	New South Wales	74430	Australia	58	Female	Medium	Premium	9/21/2023
5434096	97285	Erin Lewis	Arthur76@gmail.com	1578355423	600 Brian Prairie Suite 497	Kitchener	Ontario	47545	Canada	29	Female	Low	New	6/26/2023
2344675	26603	Angela Fields	Tanya94@gmail.com	3668096144	237 Young Curve	Munich	Berlin	86862	Germany	29	Male	Medium	Premium	3/24/2023
4155845	80175	Diane Clark	Martin39@gmail.com	6219779557	8823 Mariah Heights Apt. 263	Wollongong	New South Wales	39820	Australia	46	Male	Medium	New	01-06-24
4926148	31878	Lori Bell	Jessica33@gmail.com	6004895059	6225 William Lodge	Cologne	Berlin	64317	Germany	25	Male	Medium	New	10-04-23
8493213	19136	Jonathan Eaton	Mark38@gmail.com	2996714102	9772 Sosa Coves	Portsmouth	England	59280	UK	64	Female	Low	Regular	7/20/2023
1609659	66883	Brianna Oconnor	David47@gmail.com	9398168800	95471 Jerry Hollow Suite 034	Portsmouth	England	91253	UK	31	Female	Medium	Regular	6/21/2023
2846832	31930	Kristine Williams	Charles29@gmail.com	1822767586	60629 Jones Villages	Portsmouth	England	38173	UK	56	Female	Medium	Regular	01-02-24
3217461	74671	Felicia Brown	Laura72@gmail.com	8869644441	381 Richard Plaza	Portsmouth	England	32061	UK	53	Female	Medium	Regular	05-07-23
2401331	98300	Andrew Guzman	Eric76@gmail.com	2923044936	470 Rodriguez Estate Suite 564	Portsmouth	England	5259	UK	32	Female	High	Regular	11/18/2023
4969987	64995	Justin Lara	Jacob26@gmail.com	2683281702	3577 Cole Stream Apt. 351	Portsmouth	England	9845	UK	43	Male	Medium	Regular	6/15/2023
6681000	69939	Jessica Harrison	Anne45@gmail.com	2540232555	93806 Murphy Avenue Apt. 919	Portsmouth	England	84007	UK	69	Male	Low	Regular	07-01-23
2430572	71803	Alicia Knight	Jennifer1@gmail.com	6262218202	9856 Meyers Road Apt. 698	Portsmouth	England	91967	UK	19	Female	High	Regular	4/14/2023
7728099	78376	Ronald Chen	Nicole72@gmail.com	1796133281	6048 Charles Lake	Portsmouth	England	82975	UK	49	Male	Low	Regular	02-07-24
3489336	40654	Sara Rogers	Jeff99@gmail.com	2687947826	6048 Charles Knoll Suite 755	Portsmouth	England	3035	UK	69	Male	Low	Regular	10/24/2023
6925572	42165	Daniel Hutchinson	Oscar54@gmail.com	9604443902	37402 Williams Stream Apt. 779	Portsmouth	England	29119	UK	46	Male	Low	Regular	3/21/2023
8871554	63014	Joshua Noble	Diana25@gmail.com	5283401869	7467 Henry Port	Portsmouth	England	16285	UK	61	Male	High	Regular	08-06-23

16- 30 Columns

Year	Month	Time	Total_Purchases	Amount	Total_Amount	Product_Category	Product_Brand	Product_Type	Feedback	Shipping_Method	Payment_Method	Order_Status	Ratings	products
2023	September	22:03:55	3	108.0287567	324.08627	Clothing	Nike	Shorts	Excellent	Same-Day	Debit Card	Shipped	5	Cycling shorts
2023	December	8:42:04	2	403.3539073	806.7078147	Electronics	Samsung	Tablet	Excellent	Standard	Credit Card	Processing	4	Lenovo Tab
2023	April	4:06:29	3	354.4775997	1063.432799	Books	Penguin Books	Children's	Average	Same-Day	Credit Card	Processing	2	Sports equipment
2023	May	14:55:17	7	352.4077173	2466.854021	Home Decor	Home Depot	Tools	Excellent	Standard	PayPal	Processing	4	Utility knife
2024	January	16:54:07	2	124.2765245	248.5530491	Grocery	Nestle	Chocolate	Bad	Standard	Cash	Shipped	1	Chocolate cookies
2023	September	23:24:27	4	296.2918059	1185.167224	Electronics	Apple	Tablet	Good	Express	PayPal	Pending	4	Lenovo Tab
2023	June	13:35:51	2	315.0576476	630.1152951	Electronics	Samsung	Television	Bad	Standard	Cash	Processing	1	QLED TV
2023	March	10:12:56	1	46.58807038	46.58807038	Clothing	Zara	Shirt	Bad	Same-Day	Cash	Processing	1	Dress shirt
2024	January	14:38:26	8	328.8393017	2630.714413	Grocery	Nestle	Chocolate	Bad	Same-Day	Cash	Delivered	1	Dark chocolate
2023	October	22:27:40	10	397.6112295	3976.112295	Home Decor	Home Depot	Decorations	Excellent	Standard	Cash	Delivered	4	Candles
2023	July	23:06:51	4	90.98186975	363.927479	Home Decor	Home Depot	Tools	Average	Standard	Credit Card	Shipped	2	Screwdriver set
2023	June	13:09:58	1	364.8305669	364.8305669	Books	Random House	Non-Fiction	Average	Standard	Credit Card	Pending	2	Science
2024	January	0:00:47	10	161.879361	1618.79361	Grocery	Coca-Cola	Water	Bad	Standard	PayPal	Delivered	1	Bottled water
2023	May	18:49:35	9	350.8170158	3157.353142	Grocery	Nestle	Snacks	Excellent	Express	PayPal	Delivered	4	Fruit snacks
2023	November	23:41:05	6	297.7260391	1786.356235	Clothing	Adidas	T-shirt	Bad	Same-Day	Cash	Shipped	1	V-neck tee
2023	June	23:24:22	3	23.72390122	71.17170367	Books	Random House	Literature	Bad	Express	Credit Card	Pending	1	Drama
2023	July	11:12:02	10	14.72716425	147.2716425	Grocery	Pepsi	Water	Average	Same-Day	Debit Card	Shipped	2	Flavored water
2023	April	5:50:55	3	139.786855	419.3605651	Electronics	Apple	Tablet	Good	Express	Cash	Shipped	3	Samsung Galaxy Tab
2024	February	18:29:59	1	210.1379188	210.1379188	Grocery	Coca-Cola	Juice	Bad	Express	Credit Card	Shipped	1	Orange juice
2023	October	13:01:04	2	469.6596392	939.3192783	Home Decor	IKEA	Furniture	Average	Standard	Cash	Shipped	2	Bookshelf

Top 5 Business Objectives

- **Objective 1 – Product and Brand Revenue Analysis**
  - Analyze which products and brands generate the highest total revenue to support product assortment optimization and supplier performance evaluation.

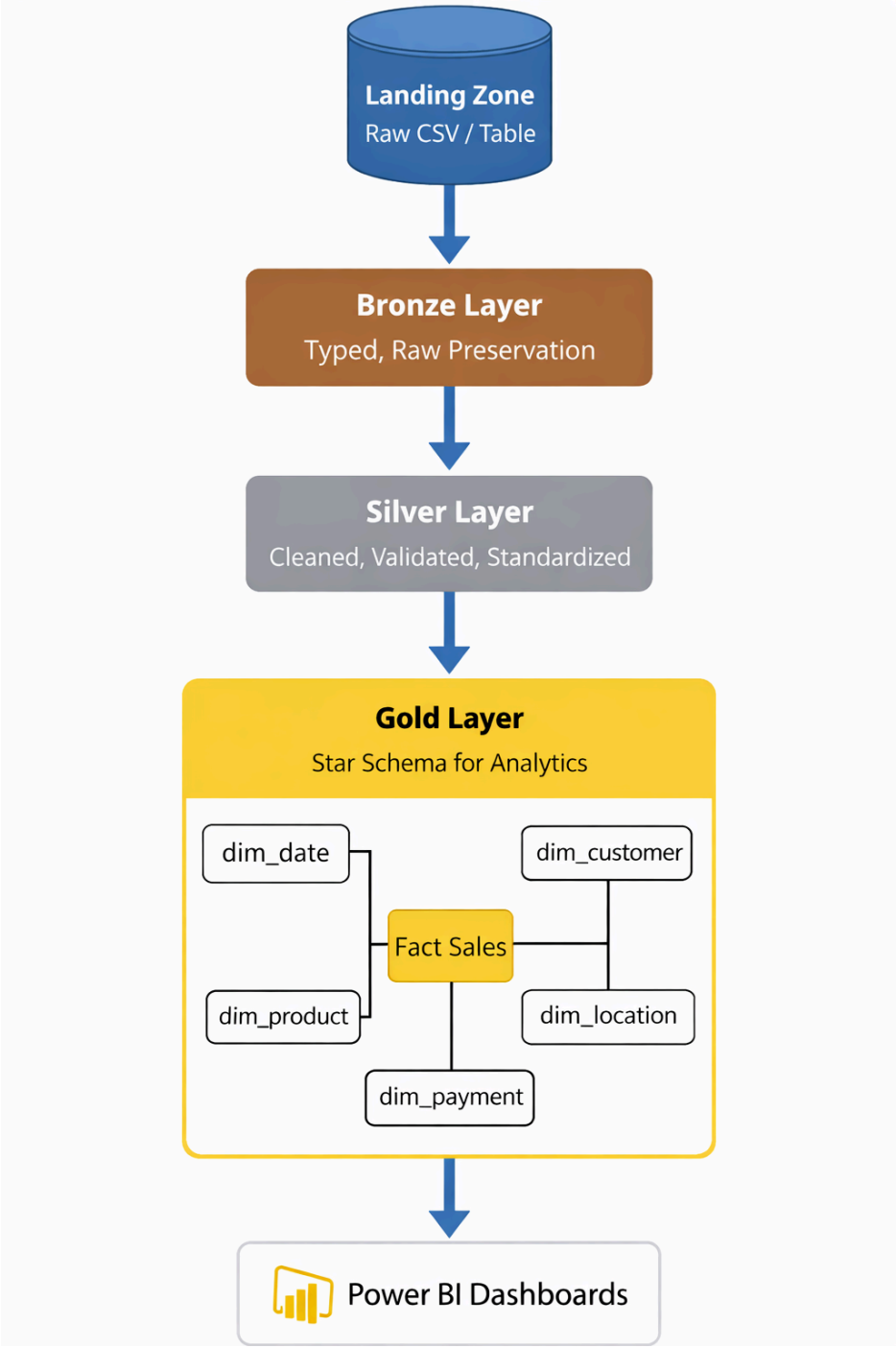
- **Objective 2 – Underperforming Product Category Identification**
  - Identify product categories with low sales volume or revenue contribution to support inventory reduction and shelf-space optimization.

- **Objective 3 – Shipping Method Impact Analysis**
  - Evaluate how different shipping methods impact order volume and customer satisfaction by analyzing order status and customer feedback.

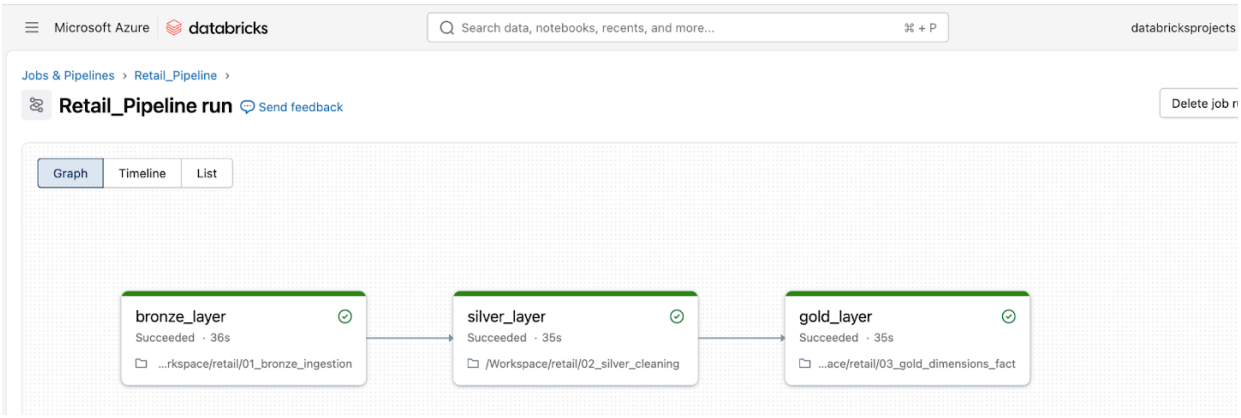
- **Objective 4 – Revenue Contribution by Customer Segment and Income**
  - Analyze which customer segments and income groups contribute the most to total revenue to improve targeting and promotional strategies.

- **Objective 5 – Gender-Based Purchasing Behavior Analysis**
  - Compare purchasing behavior across male and female customers based on total revenue and quantity purchased to support personalized marketing strategies.

Data Processing Architecture

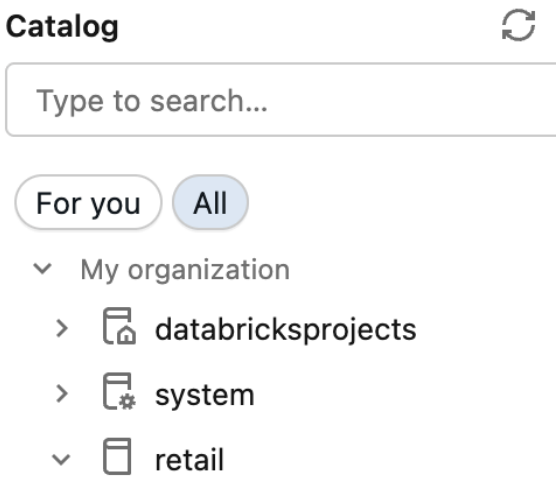


Pipeline for Business Intelligence Solution



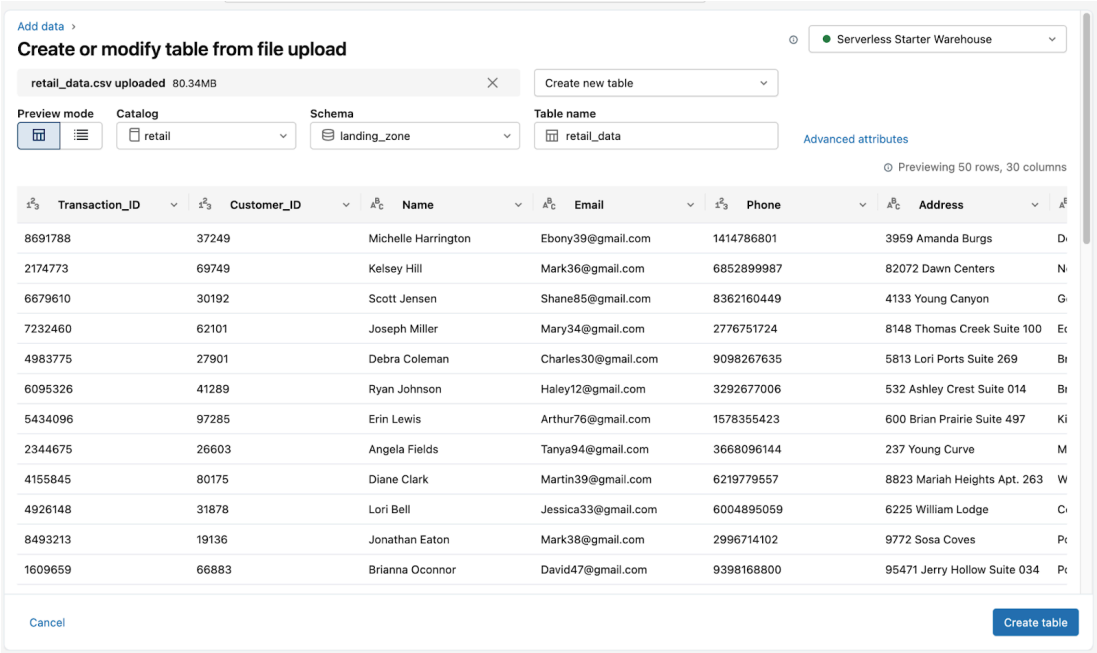
Catalog

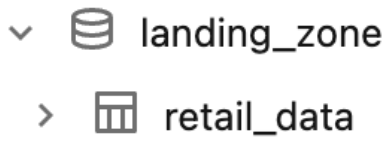
- A catalog named retail was created to organize all project-related data objects.
- The catalog provides a logical container for schemas and tables across different layers.
- All Landing_zone, Bronze, Silver, and Gold schemas are created under the retail catalog.
- Helps maintain clear separation and governance of data assets within the project.



Landing Layer

- Stores the **raw retail dataset** exactly as received from the source.
- Acts as the **initial ingestion layer** with no transformations applied.
- Data is loaded in its **original format** (CSV / table).
- Used as a **reference point** for auditing and reprocessing if needed.
- Ensures raw data is preserved before any cleaning or validation steps.





Bronze Layer – Data Ingestion

Purpose

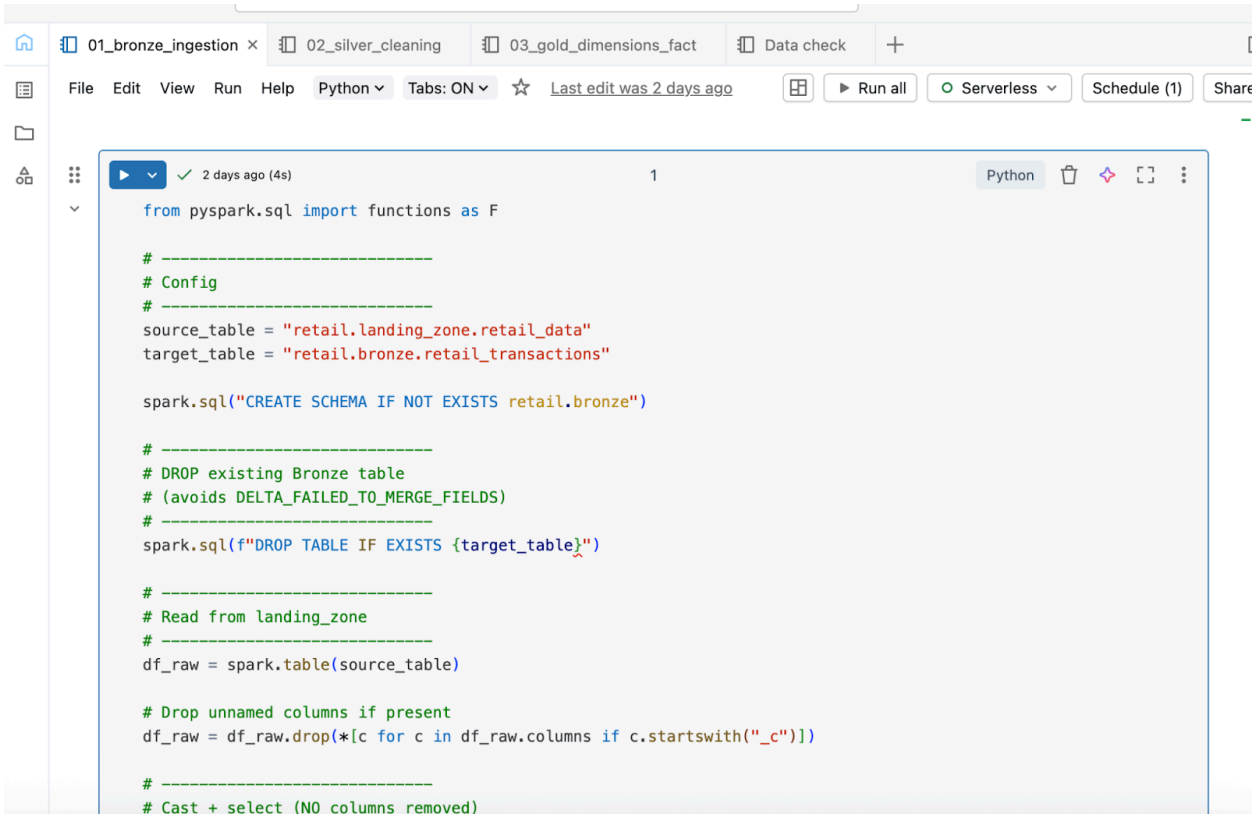
- Load raw retail CSV data into the Bronze layer in Delta format without transformations.
- Data is ingested from the Landing Layer without business transformations.

Bronze Code

```
from pyspark.sql import functions as F

# Config

source_table = "retail.landing_zone.retail_data"

target_table = "retail.bronze.retail_transactions"

spark.sql("CREATE SCHEMA IF NOT EXISTS retail.bronze")

# ----------------------------

# DROP existing Bronze table

# (avoids DELTA_FAILED_TO_MERGE_FIELDS)

spark.sql(f"DROP TABLE IF EXISTS {target_table}")

# ----------------------------

# Read from landing_zone

# ----------------------------

df_raw = spark.table(source_table)

# Drop unnamed columns if present

df_raw = df_raw.drop(*[c for c in df_raw.columns if c.startswith("_c")])

# ----------------------------
```

```
# Cast + select (NO columns removed)
# ---------------------------
df_bronze = df_raw.select(
    F.col("Transaction_ID").cast("int").alias("Transaction_ID"),
    F.col("Customer_ID").cast("int").alias("Customer_ID"),
    F.col("Name").cast("string").alias("Name"),
    F.col("Email").cast("string").alias("Email"),
    F.col("Phone").cast("string").alias("Phone"),
    F.col("Address").cast("string").alias("Address"),
    F.col("City").cast("string").alias("City"),
    F.col("State").cast("string").alias("State"),
    F.col("Zipcode").cast("string").alias("Zipcode"),
    F.col("Country").cast("string").alias("Country"),
    F.col("Age").cast("int").alias("Age"),
    F.col("Gender").cast("string").alias("Gender"),
    F.col("Income").cast("string").alias("Income"),
    F.col("Customer_Segment").cast("string").alias("Customer_Segment"),

    # Keep raw in Bronze; parse in Silver
    F.col("Date").cast("string").alias("Date"),
    F.col("Year").cast("int").alias("Year"),
    F.col("Month").cast("string").alias("Month"),
```

```
    F.col("Time").cast("string").alias("Time"),


    F.col("Total_Purchases").cast("int").alias("Total_Purchases"),
    F.col("Amount").cast("double").alias("Amount"),
    F.col("Total_Amount").cast("double").alias("Total_Amount"),


    F.col("Product_Category").cast("string").alias("Product_Category"),
    F.col("Product_Brand").cast("string").alias("Product_Brand"),
    F.col("Product_Type").cast("string").alias("Product_Type"),
    F.col("Feedback").cast("string").alias("Feedback"),
    F.col("Shipping_Method").cast("string").alias("Shipping_Method"),
    F.col("Payment_Method").cast("string").alias("Payment_Method"),
    F.col("Order_Status").cast("string").alias("Order_Status"),
    F.col("Ratings").cast("int").alias("Ratings"),
    F.col("products").cast("string").alias("products")
)
# Write Bronze table
# ---------------------------
df_bronze.write \
    .format("delta") \
    .mode("overwrite") \
    .saveAsTable(target_table)
```



Silver Layer – Data Cleaning & Transformation

Purpose

- Convert raw data into clean, standardized, analysis-ready data

Data Quality Checks (Silver)

Removed fully duplicated rows

Dropped rows with invalid or missing critical values

Filtered out invalid dates

Removed records with negative numeric values

Transformations Applied

Column Handling

- Dropped PII and unused columns:
    - Email, Phone, Address, State, Time, Year, Month, Name
- Renamed products to Product_Name

Date Handling

- Parsed Date into transaction_date using try_to_date
- Removed invalid dates
- Dropped original Date column

Null Handling

- Critical columns: rows dropped if null
- Non-critical categorical columns: nulls replaced with "UNKNOWN"

Data Quality Rules

- Age restricted to **0–100**
- Ratings restricted to **1–5** (invalid values set to null)
- Numeric fields must be **non-negative**

Text Standardization

- Trimmed whitespace
- Standardized categorical values for consistent reporting

Silver Code

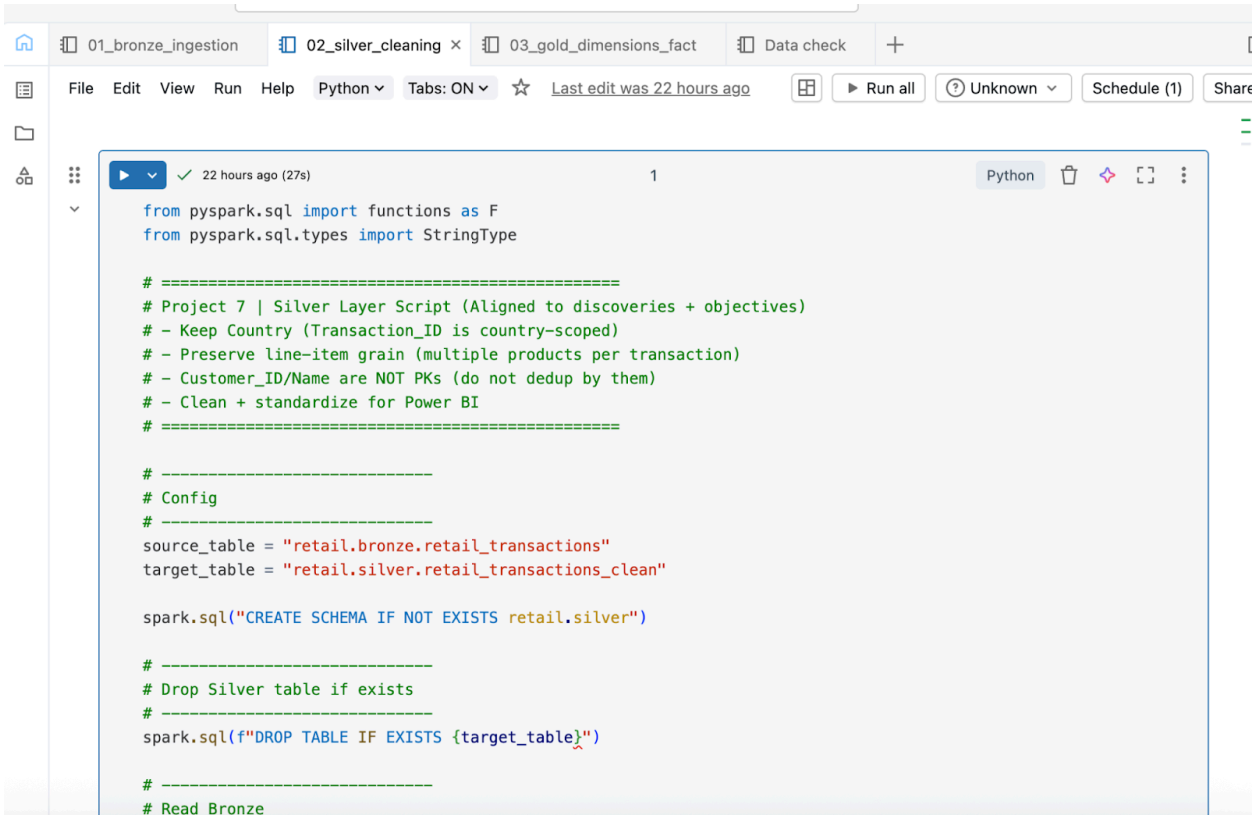
```
from pyspark.sql import functions as F
from pyspark.sql.types import StringType

# Project 7 | Silver Layer
# ================================================
# Config
# ----------------------------
source_table = "retail.bronze.retail_transactions"
target_table = "retail.silver.retail_transactions_clean"

spark.sql("CREATE SCHEMA IF NOT EXISTS retail.silver")

# ----------------------------
# Drop Silver table if exists
# ----------------------------
spark.sql(f"DROP TABLE IF EXISTS {target_table}")

# ----------------------------
# Read Bronze
# ----------------------------
df = spark.table(source_table)

# ================================================
# 1) Drop unwanted columns (PII + not needed)
#    Keep Country (crucial); keep Name/Customer_ID as attributes if you want,
#    but DO NOT treat them as keys.
# ================================================
drop_cols = ["Email", "Phone", "Address", "State", "City", "Zipcode", "Time", "Year",
"Month", "Name"]
df = df.drop(*[c for c in drop_cols if c in df.columns])

# ================================================
# 2) Date cleanup -> transaction_date (DATE), filter invalid, drop raw Date
# ================================================
df = df.withColumn(
```

```
    "transaction_date",
    F.coalesce(
        F.try_to_date("Date", "M/d/yyyy"),
        F.try_to_date("Date", "MM/dd/yyyy"),
        F.try_to_date("Date", "MM-dd-yy"),
        F.try_to_date("Date", "yyyy-MM-dd")
    )
).filter(F.col("transaction_date").isNotNull()) \
 .drop("Date")

# ================================================
# 3) string cleanup
# ================================================
string_cols = [f.name for f in df.schema.fields if isinstance(f.dataType, StringType)]
for c in string_cols:
    df = df.withColumn(c, F.trim(F.col(c)))

# Country standardization
if "Country" in df.columns:
    df = df.withColumn(
        "Country",
        F.when(F.upper(F.col("Country")).isin("USA", "US", "UNITED STATES",
"UNITED STATES OF AMERICA"), F.lit("USA"))
         .when(F.upper(F.col("Country")).isin("UK", "U.K.", "UNITED KINGDOM",
"GREAT BRITAIN"), F.lit("UK"))
         .otherwise(F.initcap(F.col("Country")))
    )

# Normalize categoricals used in objectives
cat_upper = [
    "Gender", "Income", "Customer_Segment",
    "Product_Category", "Product_Brand", "Product_Type",
    "Shipping_Method", "Order_Status","Feedback"
]
cat_upper = [c for c in cat_upper if c in df.columns]
for c in cat_upper:
    df = df.withColumn(c, F.upper(F.col(c)))
```

```
# Product name readable & consistent + rename for business clarity
if "products" in df.columns:
    df = df.withColumn("products", F.initcap(F.col("products")))
    df = df.withColumnRenamed("products", "Product_Name")


# ================================================
# 4) Preserve true grain

df = df.dropDuplicates()
# ================================================
# 5) Cast numerics + validations

# Cast numeric columns safely
for c in ["Total_Purchases", "Amount", "Total_Amount"]:
    if c in df.columns:
        df = df.withColumn(c, F.col(c).cast("double"))

# Non-negative numeric checks
for c in ["Total_Purchases", "Amount", "Total_Amount"]:
    if c in df.columns:
        df = df.filter(F.col(c) >= 0)

# Age sanity (optional but safe)
if "Age" in df.columns:
    df = df.withColumn("Age", F.col("Age").cast("int"))
    df = df.filter((F.col("Age") >= 0) & (F.col("Age") <= 100))

# Ratings: keep numeric INT; invalid -> NULL (Power BI friendly)
if "Ratings" in df.columns:
    df = df.withColumn("Ratings", F.col("Ratings").cast("int"))
    df = df.withColumn(
        "Ratings",
        F.when((F.col("Ratings") >= 1) & (F.col("Ratings") <= 5), F.col("Ratings"))
         .otherwise(F.lit(None).cast("int"))
    )
```

```
# ============================================================
# 6) Null handling aligned to discoveries + 5 objectives
# ============================================================
required_cols = [
    "Transaction_ID", "Country", "transaction_date",
    "Gender", "Income", "Customer_Segment",
    "Product_Category", "Product_Brand", "Product_Name",
    "Total_Amount",
    "Shipping_Method", "Order_Status", "Feedback"
]

required_cols = [c for c in required_cols if c in df.columns]
df = df.dropna(subset=required_cols)

# Optional fill UNKNOWN for remaining categoricals (after dropping null required)
fill_unknown_cols = [
    "Product_Type", "Gender", "Income", "Customer_Segment",
    "Product_Category", "Product_Brand", "Product_Name",
    "Shipping_Method", "Order_Status"
]

fill_unknown_cols = [c for c in fill_unknown_cols if c in df.columns]
df = df.fillna("UNKNOWN", subset=fill_unknown_cols)

# ============================================================
# 7) Add Transaction_BK (deterministic, not surrogate)

if "Transaction_ID" in df.columns and "Country" in df.columns:
    df = df.withColumn(
        "Transaction_BK",
        F.concat_ws("|", F.col("Transaction_ID").cast("string"), F.col("Country"))
    )

# ============================================================
# 8) Write to Silver

df.write.format("delta").mode("overwrite").saveAsTable(target_table)
```

```
print("Silver table created:", target_table)
print("Rows:", df.count())
df.printSchema()
```



Gold Layer – Business Modeling

Purpose

- Build an analytics-ready star schema for Power BI.
- Ensure each dimension has unique keys so relationships are 1 (dim) → many (fact)
- Enable fast analytics and Power BI reporting
- Used deterministic Business Keys (not random IDs) for stable joins.

Business Keys Created

- Transaction_BK = Transaction_ID | Country
- Product_BK = Category | Brand | Type | Product_Name
- CustomerProfile_BK = Gender | Income | Customer_Segment
- Feedback_BK = normalized Feedback

Dimension Tables

Dimension Table	Primary Key	Description	Key Attributes
dim_date	transaction_date	Date dimension for time-based analysis	year, month_num, month_name, quarter, year_month
dim_country	Country	Geographic dimension	Country
dim_product	Product_BK	Product-level details	Product_Category, Product_Brand, Product_Type, Product_Name
dim_customer_profile	CustomerProfile_BK	Customer grouping based on profile	Gender, Income, Customer_Segment
dim_shipping	Shipping_Method	Shipping method details	Shipping_Method
dim_order_status	Order_Status	Order lifecycle status	Order_Status
dim_feedback	Feedback	Customer feedback dimension	Feedback

Fact Table

Fact Table	Grain	Description	Measures
fact_sales	Line-item (per product per transaction)	Stores transactional sales data	Total_Purchases, Amount, Total_Amount

Gold Code

```
from pyspark.sql import functions as F

# ============================================================

# Project 7 | GOLD Layer

# ============================================================

# Config

# ----------------------------

silver_table = "retail.silver.retail_transactions_clean"

gold_schema  = "retail.gold"

spark.sql(f"CREATE SCHEMA IF NOT EXISTS {gold_schema}")

# ----------------------------

# Read Silver

df = spark.table(silver_table)

# ============================================================

# 0) Create deterministic Business Keys (NOT surrogate keys)

# Transaction_BK = Transaction_ID|Country

if "Transaction_BK" not in df.columns:

    df = df.withColumn(

        "Transaction_BK",

        F.concat_ws("|", F.col("Transaction_ID").cast("string"), F.col("Country"))

    )


# Product_BK = Category|Brand|Type|Product_Name
```

```
df = df.withColumn(
    "Product_BK",
    F.concat_ws("|",
        F.col("Product_Category"),
        F.col("Product_Brand"),
        F.coalesce(F.col("Product_Type"), F.lit("UNKNOWN")),
        F.col("Product_Name")
    )
)
# CustomerProfile_BK = Gender|Income|Segment (since Customer_ID/Name are not unique)
df = df.withColumn(
    "CustomerProfile_BK",
    F.concat_ws("|", F.col("Gender"), F.col("Income"), F.col("Customer_Segment"))
)


# Feedback_BK = normalized Feedback
df = df.withColumn("Feedback_BK", F.upper(F.trim(F.col("Feedback"))))



# 1) DROP Gold tables first (idempotent runs)
drop_tables = [
    f"{gold_schema}.dim_date",
    f"{gold_schema}.dim_country",
```

```
    f"{gold_schema}.dim_product",
    f"{gold_schema}.dim_customer_profile",
    f"{gold_schema}.dim_shipping",
    f"{gold_schema}.dim_order_status",
    f"{gold_schema}.dim_feedback",
    f"{gold_schema}.fact_sales"
]
for t in drop_tables:
    spark.sql(f"DROP TABLE IF EXISTS {t}")
# ============================================================
# 2) DIMENSIONS (each key UNIQUE => "1" side)
# dim_date (key = transaction_date)
dim_date = (
    df.select("transaction_date")
      .dropDuplicates()
      .withColumn("year", F.year("transaction_date"))
      .withColumn("month_num", F.month("transaction_date"))
      .withColumn("month_name", F.date_format("transaction_date", "MMM"))
      .withColumn("year_month", F.date_format("transaction_date", "yyyy-MM"))
      .withColumn("quarter", F.concat(F.lit("Q"), F.quarter("transaction_date")))
)
```

transaction_date	year	month_num	month_name	year_month
Wednesday, March 1, 2023	2023	3	Mar	2023-03
Thursday, March 2, 2023	2023	3	Mar	2023-03
Friday, March 3, 2023	2023	3	Mar	2023-03
Saturday, March 4, 2023	2023	3	Mar	2023-03
Sunday, March 5, 2023	2023	3	Mar	2023-03
Monday, March 6, 2023	2023	3	Mar	2023-03
Tuesday, March 7, 2023	2023	3	Mar	2023-03
Wednesday, March 8, 2023	2023	3	Mar	2023-03
Thursday, March 9, 2023	2023	3	Mar	2023-03
Friday, March 10, 2023	2023	3	Mar	2023-03
Saturday, March 11, 2023	2023	3	Mar	2023-03

dim_country (key = Country)

dim_country = df.select("Country").dropDuplicates()

Country
Australia
Canada
Germany
UK
USA

dim_shipping (key = Shipping_Method)

dim_shipping = df.select("Shipping_Method").dropDuplicates()

Shipping_Method
EXPRESS
SAME-DAY
STANDARD

```
# dim_order_status (key = Order_Status)
dim_order_status = df.select("Order_Status").dropDuplicates()
```

Order_Status ⇅
DELIVERED
PENDING
PROCESSING
SHIPPED

```
# dim_feedback (key = Feedback_BK)
dim_feedback = (
    df.select("Feedback_BK")
      .dropDuplicates()
      .withColumnRenamed("Feedback_BK", "Feedback")
)
```

Feedback ⇅
AVERAGE
BAD
EXCELLENT
GOOD

```
# dim_product (key = Product_BK)
dim_product = (
    df.select("Product_BK", "Product_Category", "Product_Brand", "Product_Type", "Product_Name")
      .dropDuplicates(["Product_BK"])
)
```

Product_BK	Product_Category	Product_Brand	Product_Type	Product_Name
BOOKS|HARPERCOLLINS|FICTION|Adventure	BOOKS	HARPERCOLLINS	FICTION	Adventure
BOOKS|HARPERCOLLINS|FICTION|Dystopian	BOOKS	HARPERCOLLINS	FICTION	Dystopian
BOOKS|HARPERCOLLINS|FICTION|Fantasy	BOOKS	HARPERCOLLINS	FICTION	Fantasy
BOOKS|HARPERCOLLINS|FICTION|Historical Fiction	BOOKS	HARPERCOLLINS	FICTION	Historical Fiction
BOOKS|HARPERCOLLINS|FICTION|Horror	BOOKS	HARPERCOLLINS	FICTION	Horror
BOOKS|HARPERCOLLINS|FICTION|Literary Fiction	BOOKS	HARPERCOLLINS	FICTION	Literary Fiction
BOOKS|HARPERCOLLINS|FICTION|Mystery	BOOKS	HARPERCOLLINS	FICTION	Mystery
BOOKS|HARPERCOLLINS|FICTION|Romance	BOOKS	HARPERCOLLINS	FICTION	Romance
BOOKS|HARPERCOLLINS|FICTION|Science Fiction	BOOKS	HARPERCOLLINS	FICTION	Science Fiction
BOOKS|HARPERCOLLINS|FICTION|Thriller	BOOKS	HARPERCOLLINS	FICTION	Thriller

```
# dim_customer_profile (key = CustomerProfile_BK)
dim_customer_profile = (
    df.select("CustomerProfile_BK", "Gender", "Income", "Customer_Segment")
      .dropDuplicates(["CustomerProfile_BK"])
)
```

CustomerProfile_BK	Gender	Income	Customer_Segment
FEMALE|HIGH|NEW	FEMALE	HIGH	NEW
FEMALE|HIGH|PREMIUM	FEMALE	HIGH	PREMIUM
FEMALE|HIGH|REGULAR	FEMALE	HIGH	REGULAR
FEMALE|LOW|NEW	FEMALE	LOW	NEW
FEMALE|LOW|PREMIUM	FEMALE	LOW	PREMIUM
FEMALE|LOW|REGULAR	FEMALE	LOW	REGULAR
FEMALE|MEDIUM|NEW	FEMALE	MEDIUM	NEW
FEMALE|MEDIUM|PREMIUM	FEMALE	MEDIUM	PREMIUM
FEMALE|MEDIUM|REGULAR	FEMALE	MEDIUM	REGULAR
MALE|HIGH|NEW	MALE	HIGH	NEW
MALE|HIGH|PREMIUM	MALE	HIGH	PREMIUM
MALE|HIGH|REGULAR	MALE	HIGH	REGULAR
MALE|LOW|NEW	MALE	LOW	NEW
MALE|LOW|PREMIUM	MALE	LOW	PREMIUM
MALE|LOW|REGULAR	MALE	LOW	REGULAR
MALE|MEDIUM|NEW	MALE	MEDIUM	NEW
MALE|MEDIUM|PREMIUM	MALE	MEDIUM	PREMIUM

```
# ============================================================

# 3) FACT TABLE (many side) — line-item grain (Silver grain)

fact_sales = df.select(

    # Relationship keys

    "Transaction_BK",

    "transaction_date",

    "Country",

    "Product_BK",

    "CustomerProfile_BK",

    "Shipping_Method",

    "Order_Status",

    "Feedback_BK",

    # Measures
```

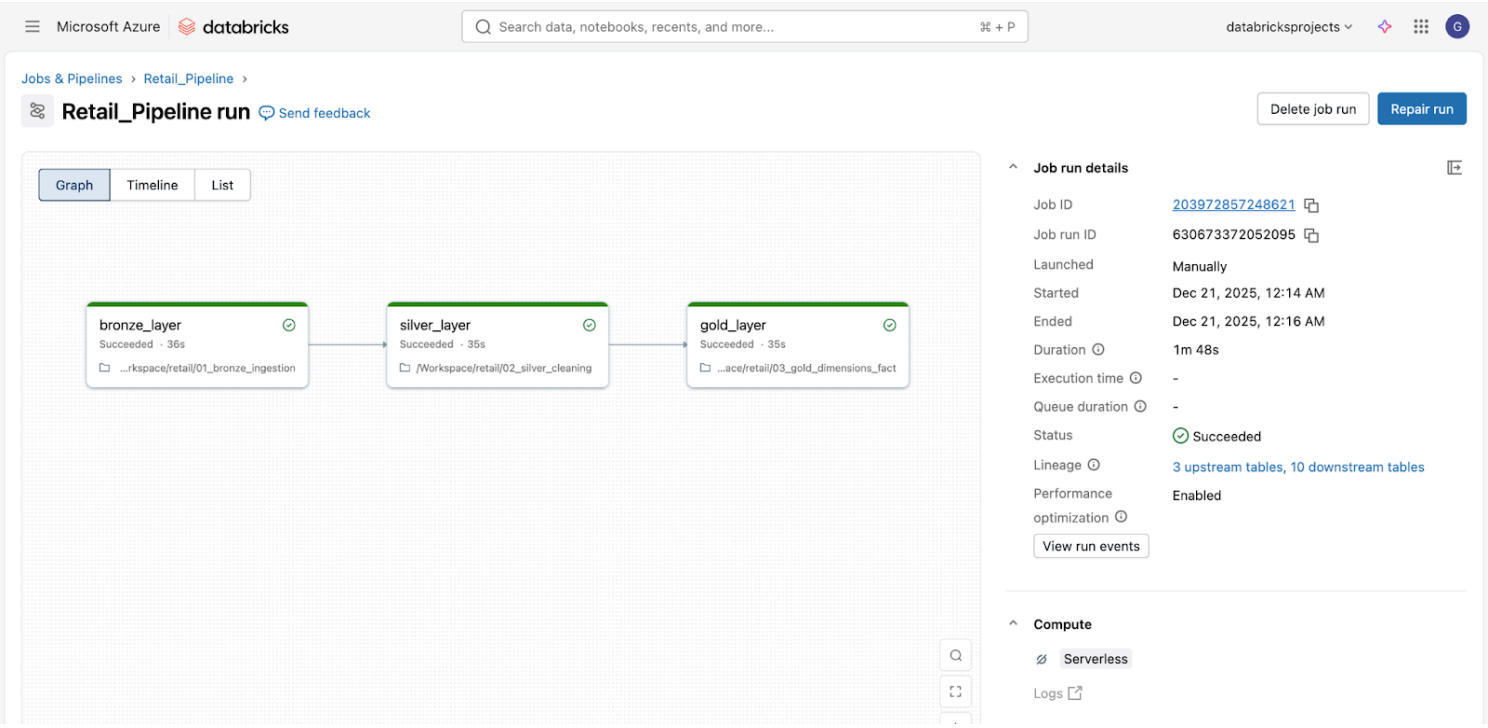
```
    "Total_Purchases",
    F.round(F.col("Amount"), 2).alias("Amount"),
    F.round(F.col("Total_Amount"), 2).alias("Total_Amount")
)
```

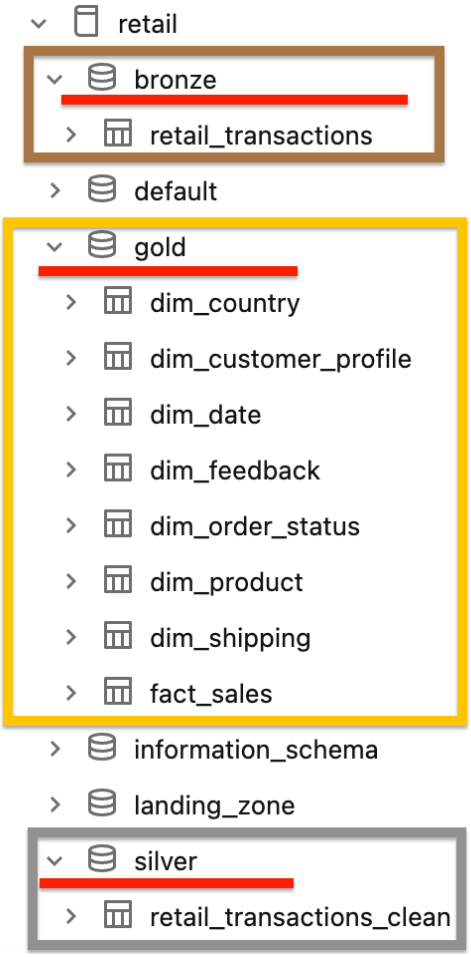
Transaction_BK	transaction_date	Country	Product_BK	CustomerProfile_BK	Shipping_Method	Order_Status
1000007|USA	Sunday, April 23, 2023	USA	GROCERY|NESTLE|COFFEE|Cappuccino	FEMALE|HIGH|NEW	SAME-DAY	SHIPPED
1000043|USA	Thursday, November 23, 2023	USA	ELECTRONICS|APPLE|SMARTPHONE|Oneplus	MALE|LOW|NEW	SAME-DAY	DELIVERED
1000073|UK	Friday, July 7, 2023	UK	BOOKS|RANDOM HOUSE|NON-FICTION|Cooking	MALE|MEDIUM|NEW	SAME-DAY	PROCESSING
1000088|Canada	Thursday, April 6, 2023	Canada	CLOTHING|ZARA|JEANS|Flare Jeans	MALE|MEDIUM|REGULAR	STANDARD	DELIVERED
1000154|Canada	Sunday, August 13, 2023	Canada	ELECTRONICS|SAMSUNG|SMARTPHONE|L g G	FEMALE|LOW|NEW	EXPRESS	PROCESSING
1000183|USA	Monday, December 4, 2023	USA	HOME DECOR|HOME DEPOT|FURNITURE|Nightstand	MALE|HIGH|REGULAR	EXPRESS	PENDING
1000195|Australia	Tuesday, October 3, 2023	Australia	CLOTHING|ADIDAS|T-SHIRT|Crop Top	FEMALE|HIGH|REGULAR	STANDARD	PENDING
1000219|USA	Sunday, January 28, 2024	USA	CLOTHING|NIKE|SHOES|High Heels	FEMALE|MEDIUM|REGULAR	EXPRESS	PENDING
1000294|UK	Tuesday, January 16, 2024	UK	GROCERY|COCA-COLA|WATER|Bottled Water	MALE|HIGH|PREMIUM	EXPRESS	DELIVERED
1000307|UK	Sunday, June 4, 2023	UK	CLOTHING|ZARA|DRESS|Cocktail Dress	MALE|HIGH|REGULAR	EXPRESS	PROCESSING
1000367|USA	Tuesday, May 30, 2023	USA	ELECTRONICS|SAMSUNG|SMARTPHONE|Sony Xperia	MALE|MEDIUM|NEW	STANDARD	DELIVERED

```
# ============================================================
# 4) WRITE Gold tables
dim_date.write.format("delta").mode("overwrite").saveAsTable(f"{gold_schema}.dim_date")
dim_country.write.format("delta").mode("overwrite").saveAsTable(f"{gold_schema}.dim_country")
dim_product.write.format("delta").mode("overwrite").saveAsTable(f"{gold_schema}.dim_product")
dim_customer_profile.write.format("delta").mode("overwrite").saveAsTable(f"{gold_schema}.dim_customer_profile")
dim_shipping.write.format("delta").mode("overwrite").saveAsTable(f"{gold_schema}.dim_shipping")
dim_order_status.write.format("delta").mode("overwrite").saveAsTable(f"{gold_schema}.dim_order_status")
```

```
dim_feedback.write.format("delta").mode("overwrite").saveAsTable(f"{gold_schema}.dim_feedback")

fact_sales.write.format("delta").mode("overwrite").saveAsTable(f"{gold_schema}.fact_sales")

# ============================================================

# 5) Quick validation prints

print("Gold Star Schema created (DROP + CREATE) using Feedback:")

for t in drop_tables:

    print("-", t)

print("\nCounts:")

print("dim_date:", spark.table(f"{gold_schema}.dim_date").count())

print("dim_country:", spark.table(f"{gold_schema}.dim_country").count())

print("dim_product:", spark.table(f"{gold_schema}.dim_product").count())

print("dim_customer_profile:", spark.table(f"{gold_schema}.dim_customer_profile").count())

print("dim_shipping:", spark.table(f"{gold_schema}.dim_shipping").count())

print("dim_order_status:", spark.table(f"{gold_schema}.dim_order_status").count())

print("dim_feedback:", spark.table(f"{gold_schema}.dim_feedback").count())

print("fact_sales:", spark.table(f"{gold_schema}.fact_sales").count())
```

Final Pipeline Execution





After the job ran, all tables were created successfully.

Power BI Report

Connection

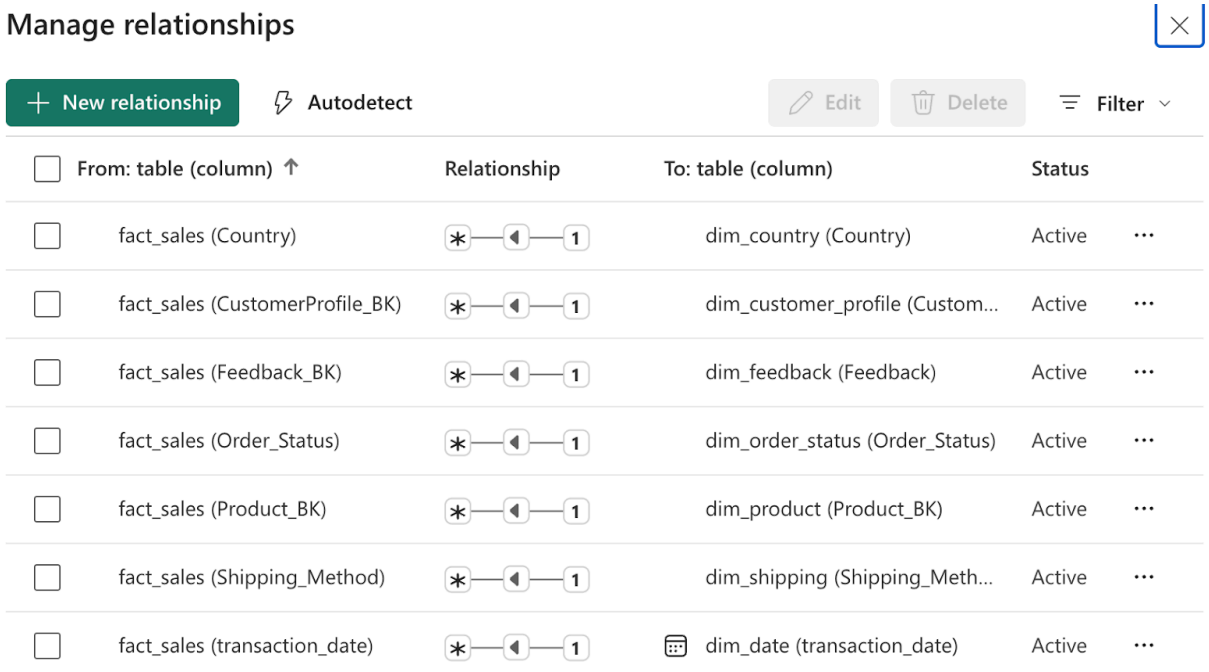
- Azure Databricks

Data Model

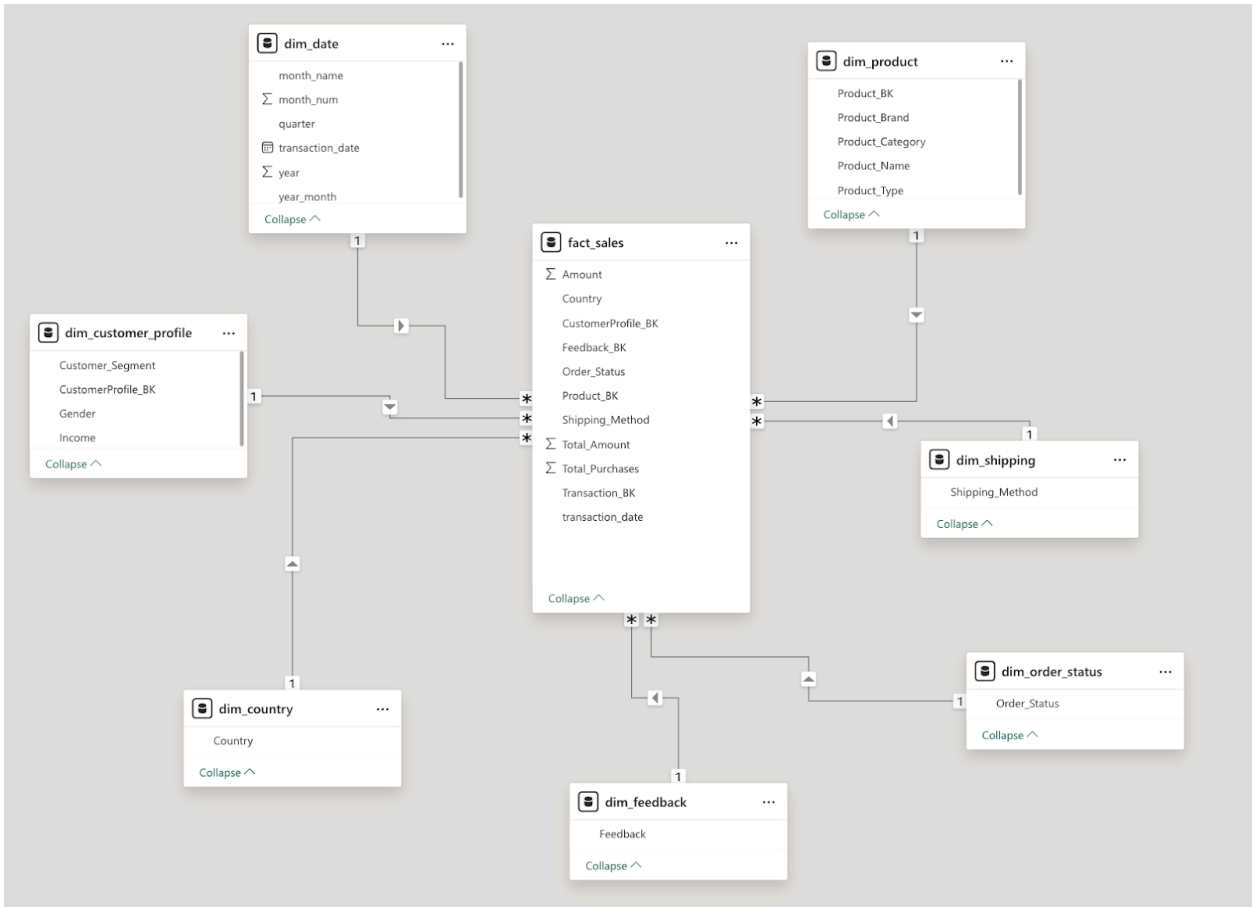
- Gold tables used
- Star schema

Relationships

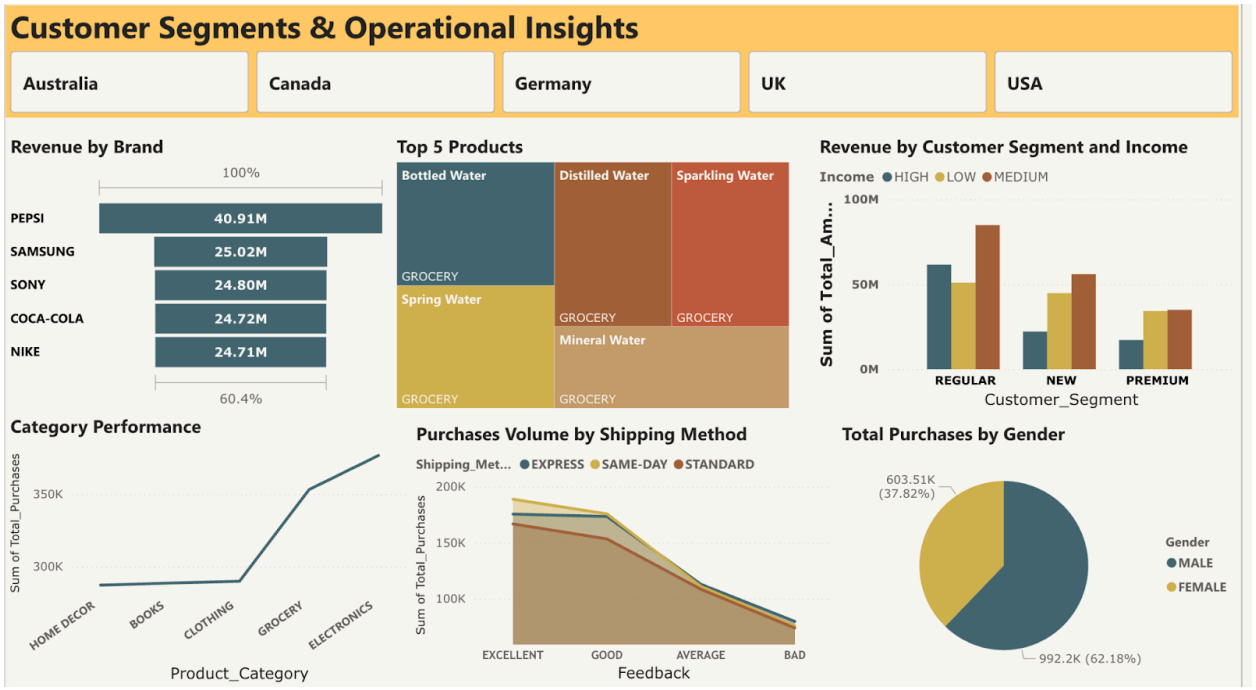
- dim_customer (1) → fact_sales (many)
- dim_date (1) → fact_sales (many)
- dim_product (1) → fact_sales (many)
- dim_location (1) → fact_sales (many)
- Other dimensions connected via natural keys



Semantic Model



Power BI Dashboard



Business Objectives

Objective 1 – Product and Brand Revenue Analysis

- Top 5 brands: Pepsi, Samsung, Sony, Coca-Cola, Nike

- Top 5 products: Bottled Water, Distilled Water, Spring Water, Sparkling Water, Mineral Water

Objective 2 – Underperforming Product Category Identification

- Home Decor, Books, and Clothing perform less compared to Grocery and Electronics

Objective 3 – Shipping Method Impact Analysis

- Same-day delivery received the most excellent feedback
- Followed by Express delivery
- Standard delivery received the lowest feedback

Objective 4 – Revenue Contribution by Customer Segment and Income

- Regular customers generate more income
- Highest contribution comes from medium-income customers

Objective 5 – Gender-Based Purchasing Behavior Analysis

- Males made a higher total number of purchases compared to females

Conclusion/Errors

Problems were reflected in the Power BI semantic model.

1) Duplicate Customer IDs

Problem: Power BI could not create relationships because the same Customer_ID appeared multiple times.

Fix: Keep only one customer record per Customer_ID by selecting the latest transaction record.

2) Zipcode Caused Many-to-Many Relationships

Problem: Zipcode was not unique and caused many-to-many relationships in Power BI.

Fix: Stopped using Zipcode as a key and used Country + City to uniquely identify locations.

3) Transaction IDs Were Not Globally Unique

Problem: Transaction IDs repeated across countries.

Fix: Created a combined key using Transaction_ID + Country.

- The following queries were used to validate key uniqueness and identify data modeling issues.
- Grouping and count-based checks revealed that some identifiers appeared multiple times, indicating they could not be used as primary keys.
- By inspecting repeated values across related columns, it became clear that certain fields were context-dependent rather than globally unique.

```
%sql
SELECT *
FROM retail.gold.dim_customer
WHERE Customer_ID IN (
    SELECT Customer_ID
    FROM retail.gold.dim_customer
    GROUP BY Customer_ID
    HAVING COUNT(*) > 1
)
ORDER BY Customer_ID;
```

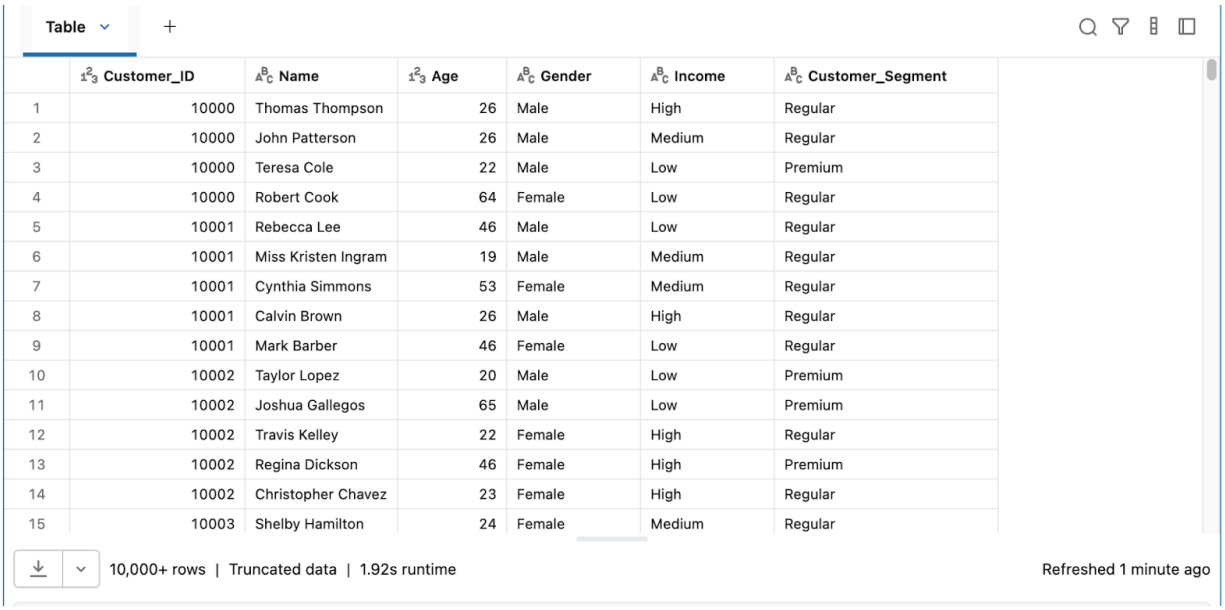


Table +

	Customer_ID	Name	Age	Gender	Income	Customer_Segment
1	10000	Thomas Thompson	26	Male	High	Regular
2	10000	John Patterson	26	Male	Medium	Regular
3	10000	Teresa Cole	22	Male	Low	Premium
4	10000	Robert Cook	64	Female	Low	Regular
5	10001	Rebecca Lee	46	Male	Low	Regular
6	10001	Miss Kristen Ingram	19	Male	Medium	Regular
7	10001	Cynthia Simmons	53	Female	Medium	Regular
8	10001	Calvin Brown	26	Male	High	Regular
9	10001	Mark Barber	46	Female	Low	Regular
10	10002	Taylor Lopez	20	Male	Low	Premium
11	10002	Joshua Gallegos	65	Male	Low	Premium
12	10002	Travis Kelley	22	Female	High	Regular
13	10002	Regina Dickson	46	Female	High	Premium
14	10002	Christopher Chavez	23	Female	High	Regular
15	10003	Shelby Hamilton	24	Female	Medium	Regular

10,000+ rows | Truncated data | 1.92s runtime

Refreshed 1 minute ago

```
%sql
SELECT *
FROM retail.bronze.retail_transactions
WHERE Transaction_ID IN (
  SELECT Transaction_ID
  FROM retail.bronze.retail_transactions
  GROUP BY Transaction_ID
  HAVING COUNT(*) > 1
)
ORDER BY Transaction_ID;
```

	Transaction_ID	Customer_ID	Name	Email	Phone	Address
1	1003092	97124	Yolanda Blankenship	Kristine9@gmail.com	2360380301	0933 Yolanda Island
2	1003092	27378	Nicholas Moore	Danielle79@gmail.com	6988330857	5564 Daniel Garden Apt.
3	1005039	84239	John Tate	Andrea27@gmail.com	9064050806	7563 Pittman Tunnel
4	1005039	84239	John Tate	Andrea27@gmail.com	9064050806	7563 Pittman Tunnel
5	1005669	39211	James Clark	William33@gmail.com	5697402092	190 Murray Trail Apt. 129
6	1005669	36372	Marcus Hess	Allison19@gmail.com	5056087581	018 Vanessa Mountain Su
7	1008244	52266	Hannah Pena	Kristen4@gmail.com	6145224687	3668 Vasquez Key Suite 8
8	1008244	51936	David Frost	Kevin56@gmail.com	1558075505	205 Spencer Rue
9	1010919	12847	Tyler Ward	Louis68@gmail.com	4553740081	945 Justin Street
10	1010919	12847	Tyler Ward	Louis68@gmail.com	4553740081	945 Justin Street
11	1015985	14362	Patricia Anderson	Nicole67@gmail.com	9044075577	4347 Lisa Ford

	ress	City	State	Zipcode	Country	Age	Gender	In
1	landa Island	St. John's	Ontario	9692	Canada	23	Female	Low
2	niel Garden Apt. 073	Chicago	Connecticut	6790	USA	34	Male	Low
3	ttman Tunnel	Bonn	Berlin	35327	Germany	23	Female	Mediu
4	ttman Tunnel	Bonn	Berlin	35327	Germany	68	Female	High
5	ray Trail Apt. 129	Dresden	Berlin	54281	Germany	22	Female	Mediu
6	essa Mountain Suite 024	Wollongong	New South Wales	86001	Australia	36	Female	Low
7	squez Key Suite 857	Sydney	New South Wales	85363	Australia	54	Female	Mediu
8	ncer Rue	Darwin	New South Wales	85949	Australia	20	Male	Mediu
9	tin Street	Vancouver	Ontario	79443	Canada	51	Male	High
10	tin Street	Vancouver	Ontario	79443	Canada	51	Male	High
11	sa Ford	Chicago	Connecticut	13172	USA	34	Female	Mediu

21 hours ago (2s) 9 SQL

```
%sql
SELECT
  Customer_ID,
  COUNT(DISTINCT Name)    AS distinct_names,
  COUNT(DISTINCT Country) AS distinct_countries,
  COUNT(*)                AS total_rows
FROM retail.silver.retail_transactions_clean
GROUP BY Customer_ID
HAVING COUNT(DISTINCT Name) > 1
    OR COUNT(DISTINCT Country) > 1
ORDER BY distinct_names DESC, distinct_countries DESC, total_rows DESC;
```

See performance (1) Optimize

Table +

	Customer_ID	distinct_names	distinct_countries	total_rows
1	60341	13	5	13
2	48453	13	5	13
3	49274	13	4	13
4	47382	13	4	13
5	99355	13	4	13