Data Analyst Task

Business Reporting Requirements

S.No	Analysis Request
1	Total amount spent and the country for the Pending delivery status for each country
2	Total number of transactions, total quantity sold, and total amount spent for each customer, along with product details
3	Maximum product purchased for each country
4	Most purchased product based on age category less than 30 and above 30
5	Country that had minimum transactions and sales amount

Concerned Department : Sales

Available Raw Data

Customer.csv

- Customer_ID: unique identifier for each customer
- First: customer's first name
- Age: customer's age in years
- Country: country of residence

Order.csv

- Order_ID: unique identifier for each order
- Item: product ordered
- Amount: sale amount in currency units.
- Customer_ID: foreign key linking back to Customer.Customer_ID

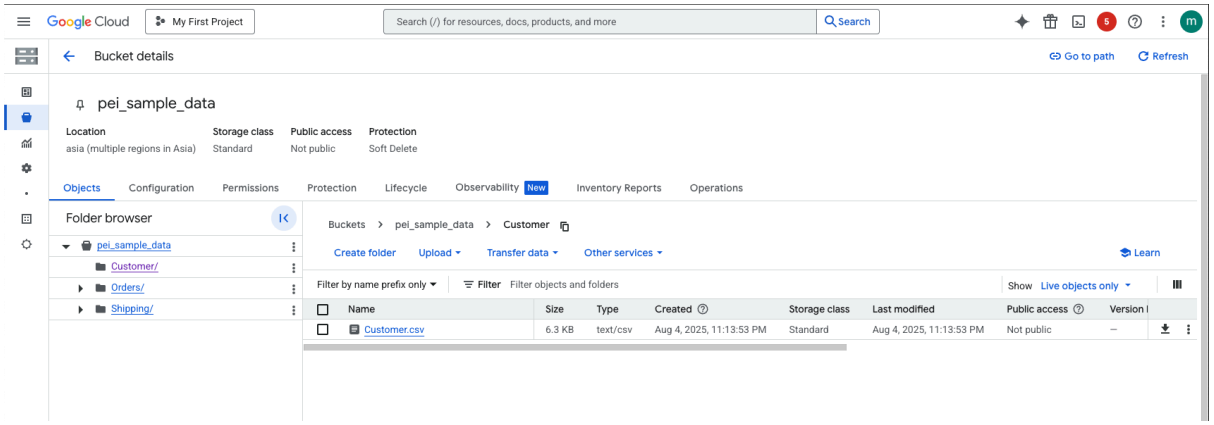
Shipping.json

- Shipping_ID: unique identifier for each shipping record
- Status: delivery status
- Customer_ID: foreign key linking back to Customer.Customer_ID

Actions Performed On Raw Data

❖ Cloud Storage Bucket

➢ Created a GCS bucket (e.g. pei-project-raw-data) to hold all raw files before ingesting into BigQuery.

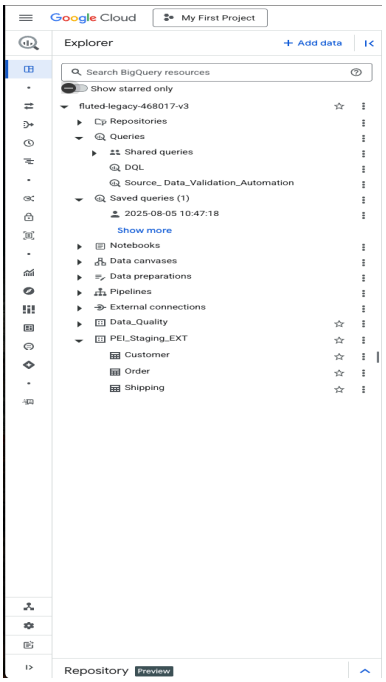


❖ Staging Environment in BigQuery

➢ Created a “PEI_Staging_EXT” dataset to store our raw source data.

➢ Defined three external tables in that dataset, each pointing to the files in Cloud Storage:

- Customer (backed by Customer.csv)
- Order (backed by Order.csv)
- Shipping (backed by Shipping.json)



❖ Data Quality Schema & Tables

- ➢ Created a “Data_Quality” dataset alongside staging.
- ➢ Built logging tables to capture DQ results:
  - ■ Data_Quality_log – completeness, domain, referential, uniqueness, range, future-date, and custom ship-order link checks.

Data_Quality_log Query Open in Share Copy Snapshot Delete Export

Schema | Details | Preview | Table Explorer | Preview | Insights | Lineage | Data Profile | Data Quality

Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Data Policies	Description
check_timestamp	TIMESTAMP	NULLABLE	-	-	-	-	-	UTC timestamp when this data-quality run began
table_name	STRING	NULLABLE	-	-	-	-	-	Name of the source table evaluated in this row
completeness_errors	INTEGER	NULLABLE	-	-	-	-	-	Rows that fail completeness (required-column) rules
domain_errors	INTEGER	NULLABLE	-	-	-	-	-	Rows with invalid domain values (e.g., bad Status)
referential_errors	INTEGER	NULLABLE	-	-	-	-	-	Rows violating referential integrity (orphan FKs)
uniqueness_errors	INTEGER	NULLABLE	-	-	-	-	-	Duplicate primary-key occurrences in the source
range_errors	INTEGER	NULLABLE	-	-	-	-	-	Rows with out-of-range numeric values
future_date_errors	INTEGER	NULLABLE	-	-	-	-	-	Rows with date fields later than the run timestamp
ship_order_link_errors	INTEGER	NULLABLE	-	-	-	-	-	Shipping records lacking a linked Order record
overall_result	STRING	NULLABLE	-	-	-	-	-	Overall outcome of the checks: PASS or FAIL

❖ Data-Quality Rule Development

- ➢ Created a unified BigQuery SQL script that, for each of the three source tables, runs:
  - ■ Completeness checks (null or missing values)
  - ■ Domain validity checks (e.g. allowed status values in Shipping)
  - ■ Referential-integrity checks (orphaned Customer_IDs)
  - ■ Uniqueness checks (duplicate primary keys)
  - ■ Range checks (e.g. Age between realistic bounds)
  - ■ Future-date guards (to prevent timestamps beyond today)
  - ■ Custom ship-order linkage checks (ensuring every shipped order matches an Order record)

❖ DQ Automation & Logging

- ➢ Carried out each suite of checks in a BEGIN TRANSACTION; … COMMIT; block to ensure atomic INSERTs.
- ➢ Scheduled regularly so our Data Quality logs stay up-to-date.

REFER - EXTtable_Schema_Information.sql

REFER - EXTtable_Data_Quality.sql

Interpretation of the script-1 Schema Info Check

Attribute	Interpretation
table_name	The staging table you're inspecting (Customer, Order, or Shipping)
column_name	The name of the column in that table
data_type	The column's declared data type in BigQuery
is_nullable	YES means the DDL allows NULLs NO means BigQuery schema forbids NULLs
null_count	How many rows in that column are actually NULL
row_count	The total number of rows in the table.

Interpretation of the script-2 Data Quality Check

Attribute	Interpretation
check_timestamp	Declared value of the Time of the scrips run
table_name	source table name
completeness_errors	Number of rows missing one or more required fields
domain_errors	Number of rows whose values fall outside allowed "domains" or enumerations.
referential_errors	Count of orphan-FK violations
uniqueness_errors	Number of duplicate business keys
range_errors	Number of rows with numeric or date values outside expected ranges (e.g. Age <0 or >120, Amount ≤0).
ship_order_link_errors	Specific to Shipping: shipments whose Customer_ID has no matching Order row. Should be 0
overall_result	'PASS' if **all** the relevant error-counts are zero; otherwise 'FAIL'.

Data Quality Check

Customer

- All 250 rows are complete, unique, and within the age range 0-120.

Order

- 250 unique orders; no negative amounts; 100 % of Customer_IDs can be linked to a customer.

Shipping

- Structural quality is fit (no NULLs, valid statuses).
- **98 / 250 rows (39 %) cannot be tied to any existing order** when linked only via Customer_ID.
  - Inference : There are number of shipments to which there is no order element in the ORDERs data
  - Likely root causes
    1. **Missing Order_ID** in the Shipping feed.
    2. Orders failed to land in Order.csv but arrived in Shipping.json.

Business Reporting Requirements and Available Data

1. Total amount spent and country for “Pending” delivery status, by country

It is currently not possible to determine the total cost of "Pending" deliveries on a per-country basis. The order details in Order.csv, which would supply the required financial information, are not directly linked to in the Shipping.json file. Furthermore, time-based analysis is limited because shipping status updates lack a timestamp.

2. Per-customer: total number of transactions, total quantity sold, total amount spent, plus product details

It is challenging to create a thorough per-customer profile that includes transaction volume, total quantity purchased, and aggregate spending. Quantity, unit price, and product identification information are not included in the Order.csv file. Time-sensitive evaluations are further limited by the lack of order dates.

3. Maximum product purchased for each country

Because the business question may be seeking the quantity of product in absolute numbers rather than an aggregated monetary term, it is difficult to determine which product is most frequently purchased in each country. Quantity data at the product level and a verified correlation between orders and customer country are absent from the dataset.

4. Most purchased product by age category (< 30 vs. ≥ 30)

The lack of quantity data for individual products makes it challenging to determine which products are most popular among various age groups (<30 vs. >=30).

5. Country with minimum transactions and minimum sales amount

It is impossible to identify the nation with the fewest transactions and the lowest total sales revenue with any degree of accuracy. Quantity and unit price data, which are necessary for figuring out sales totals, are not included in the dataset.

Request/Suggestions to the Data Engineering Team

The field listed below should be added to the source data in order to facilitate the Granular, Temporal, and Dimensional analysis with the requested BPRs and more.

Customer.csv

- **Date_of_Birth** Replace the drift-prone Age field
- Add **Country_ISO** → standard ISO code

Order.csv

- **Order_Date** Enable all time-series and period-over-period sales analyses
- **Product_ID** to know the constituents of the order
- **Quantity** → required for “total quantity sold”

Shipping.json

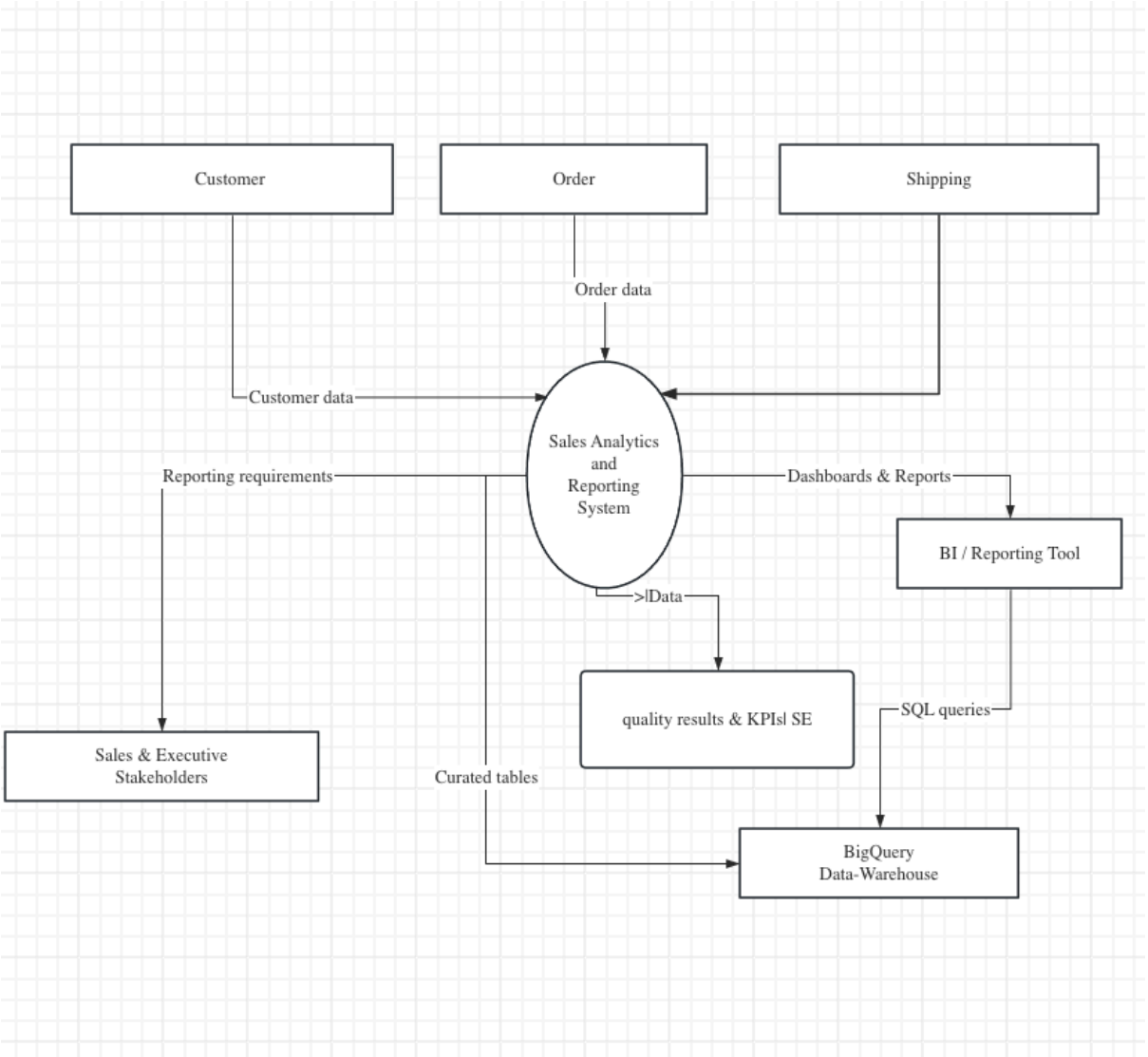
- **Order_ID** Foreign key → Order.Order_ID to link shipments back to orders
- **Shipping_Date** Timestamp when the shipment left the warehouse
- **Status_Update_Date** Record each status change “as-of” for accurate snapshot reporting

New Product Dimension

- **Product_ID** Surrogate key for joins between orders/line-items and products
- **Product_Name** Human-readable product name
- **Cost**

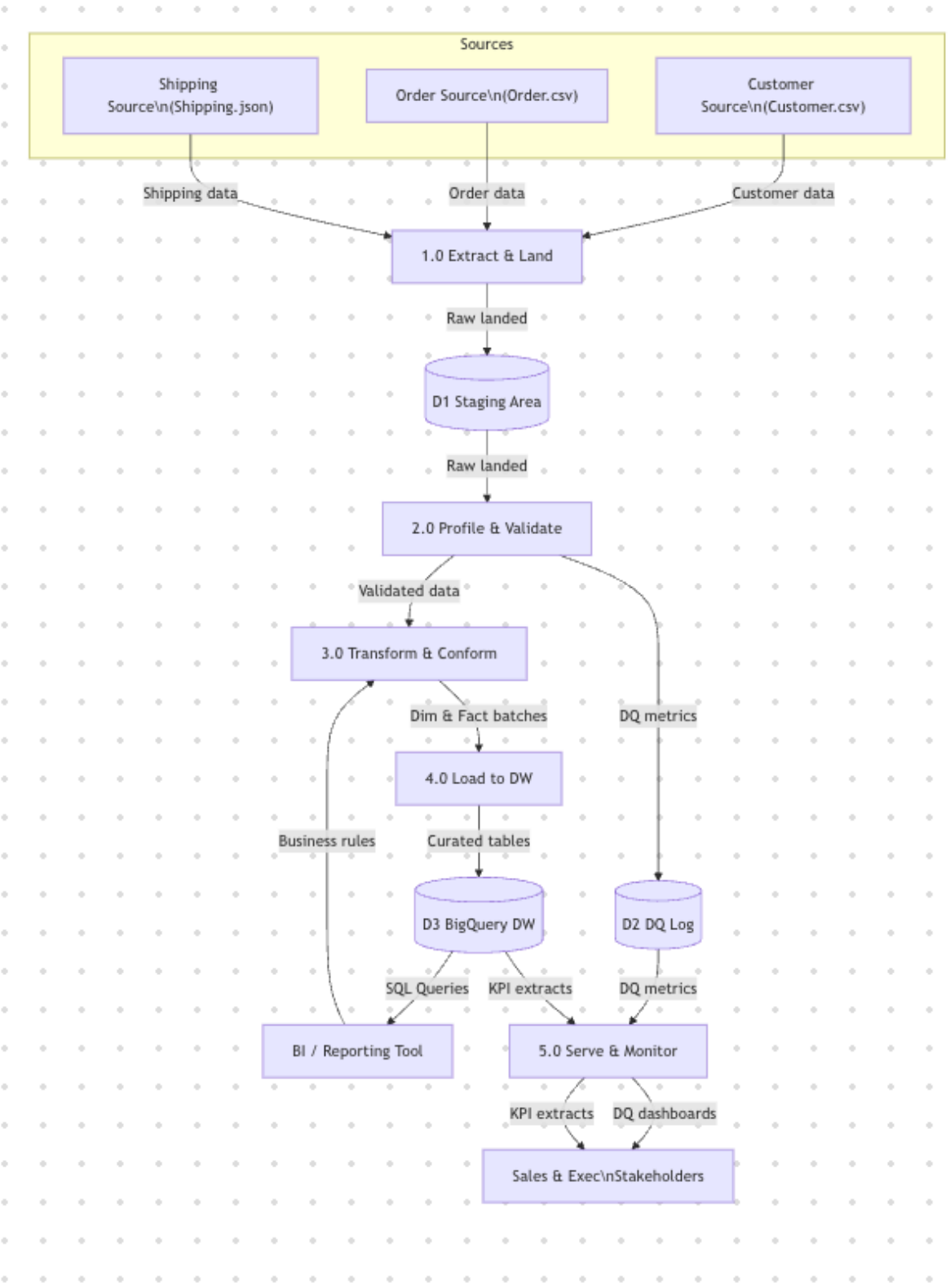
Provided next are the Data Flow Diagrams and the Entity Relation Diagram for the Data modelling for the Business task at hand.

Proposed Iter-1 of Level 0 Contextual Data Flow Diagram



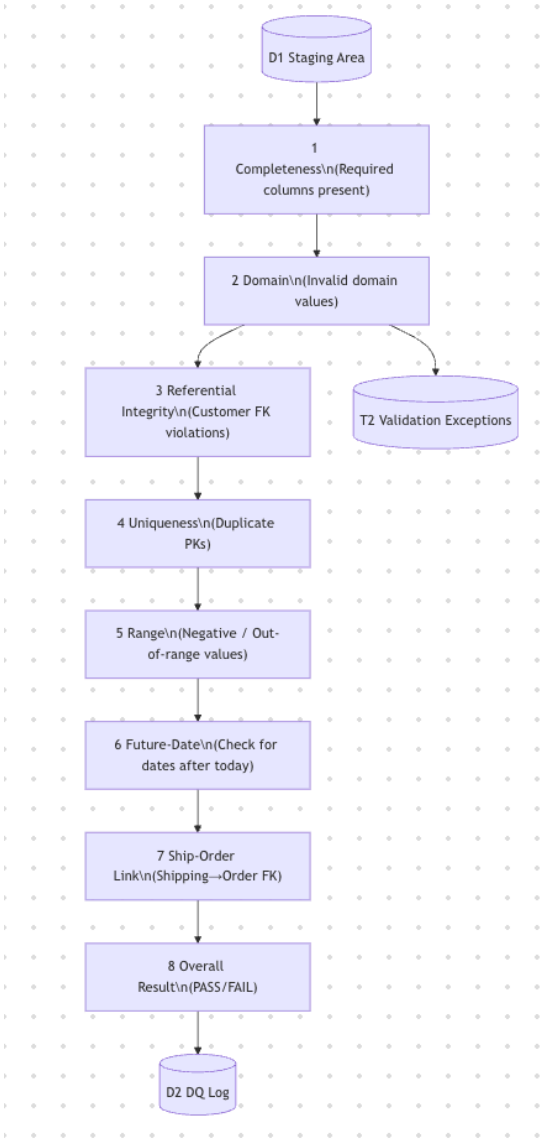
- Three operational feeds arrive daily:
    - **Customer.csv** – customer master data
    - **Order.csv** – item-level sales transactions
    - **Shipping.json** – shipment status updates
- The feeds enter a single **Sales Analytics & Reporting System** (our end-to-end pipeline).
- Stakeholders push reporting requirements into the system; these drive every downstream rule.
- The system produces three outward flows:
    - **Fact and dimension tables were curated and uploaded to the BigQuery database.**
    - **KPI snapshots and data-quality results are sent straight to executive and sales stakeholders and the QA team.**
    - **The BI tool, which makes real-time queries to BigQuery, produced dashboards and ad hoc reports.**

Proposed Iter-1 of Level 1 Data Flow Diagram



1. **Extract & Land**
    - Mount the three files as external tables and record row counts / checksums.
2. **Profile & Validate**
    - Run one SQL script that checks completeness, domain, referential integrity, duplicates, ranges, future-dates and Ship→Order links.
    - Populate the **DQ Log**
3. **Transform & Conform**
    - Clean raw data, standardise types and formats.
    - Build conformed dimension and fact batches in the staging area and attach surrogate keys.
4. **Load to DW**
    - Merge SCD-2 dimensions; upsert fact partitions into the BigQuery warehouse.
    - Capture post-load audit counts for reconciliation.
5. **Serve & Monitor**
    - Refresh dashboards, publish KPI extracts, and surface DQ statistics.
    - Accept new business rules from stakeholders and feed them back into the validation layer.

Proposed Iter-1 of Level 1.1 Data Flow Diagram

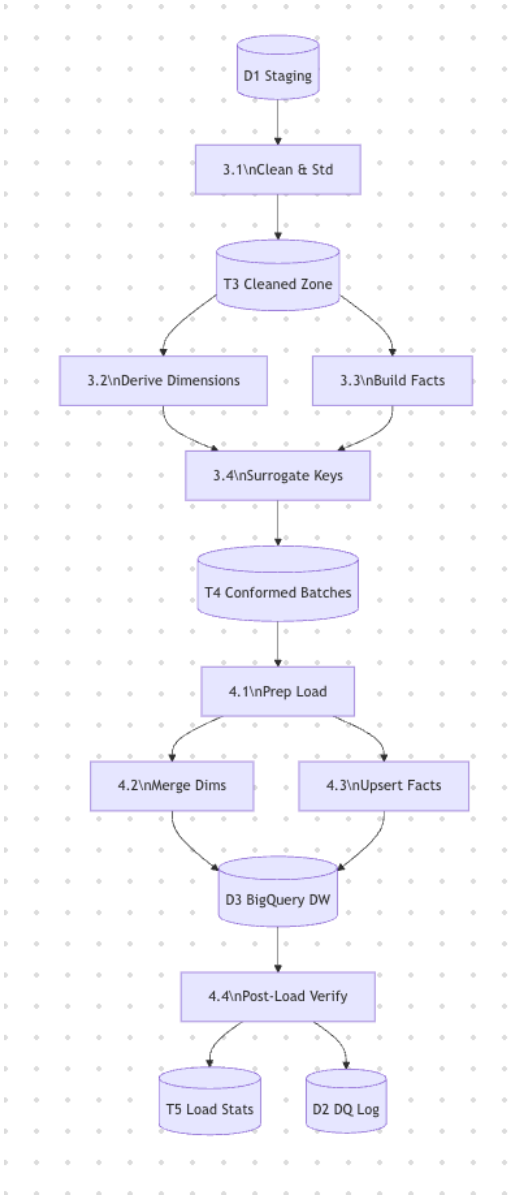


The staging tables pass through eight sequential checks:

1. **Completeness** – every required column is non-NULL.
2. **Domain** – values fall within allowed vocabularies.
3. **Referential integrity** – foreign keys resolve to parent tables.
4. **Uniqueness** – primary keys are not duplicated.
5. **Range** – numeric values stay within business limits (e.g., Age 0-120, Amount > 0).
6. **Future-date** – date fields are not after today.
7. **Ship→Order link** – every shipping record matches an order.
8. **Overall result** – a PASS is recorded only if all preceding checks return zero errors.

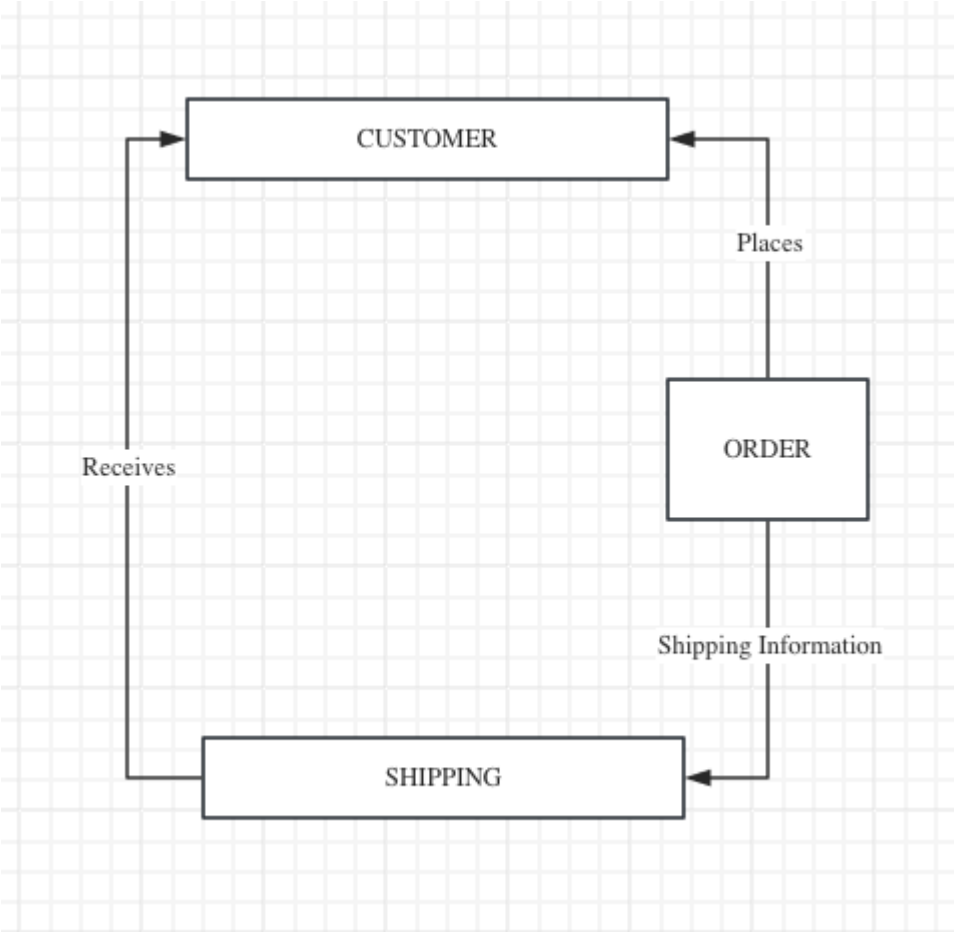
Rows that break domain rules branch off to an **exceptions store** for later triage.
The final PASS / FAIL flag is written to the DQ Log and drives the go / no-go decision for the entire run.

Proposed Iter-1 of Level 1.2 Data Flow Diagram



3.x: Conformed Batch Prep & Cleaning
3.1 Clean & Standardize: cast datatypes, trim strings, and capitalize codes.
3.2 Derive Dimensions: Using different item names, generate mini-dims (Country, Age-Band) and the entire Product dimension.
3.3 Build Facts: Put together shipment-status and order-line facts (add Quantity default = 1).
3.4 Assign Surrogate Keys: Create SKs for each dimension and associate them with natural keys.
4.x: Enter Data-Warehouse
Landing conformed batches into temporary DW staging tables is known as prep load (4.1).
4.2 Merge Dims: maintain Is_Current flags, close old records, open new ones, and apply SCD-2 logic.
4.3 Upsert Facts: enforce the new Ship→Order key, update late-arriving rows, and add new partitions.
4.4 Post-Load Verify: restore load statistics to the DQ Log, run FK checks again, and reconcile row counts.

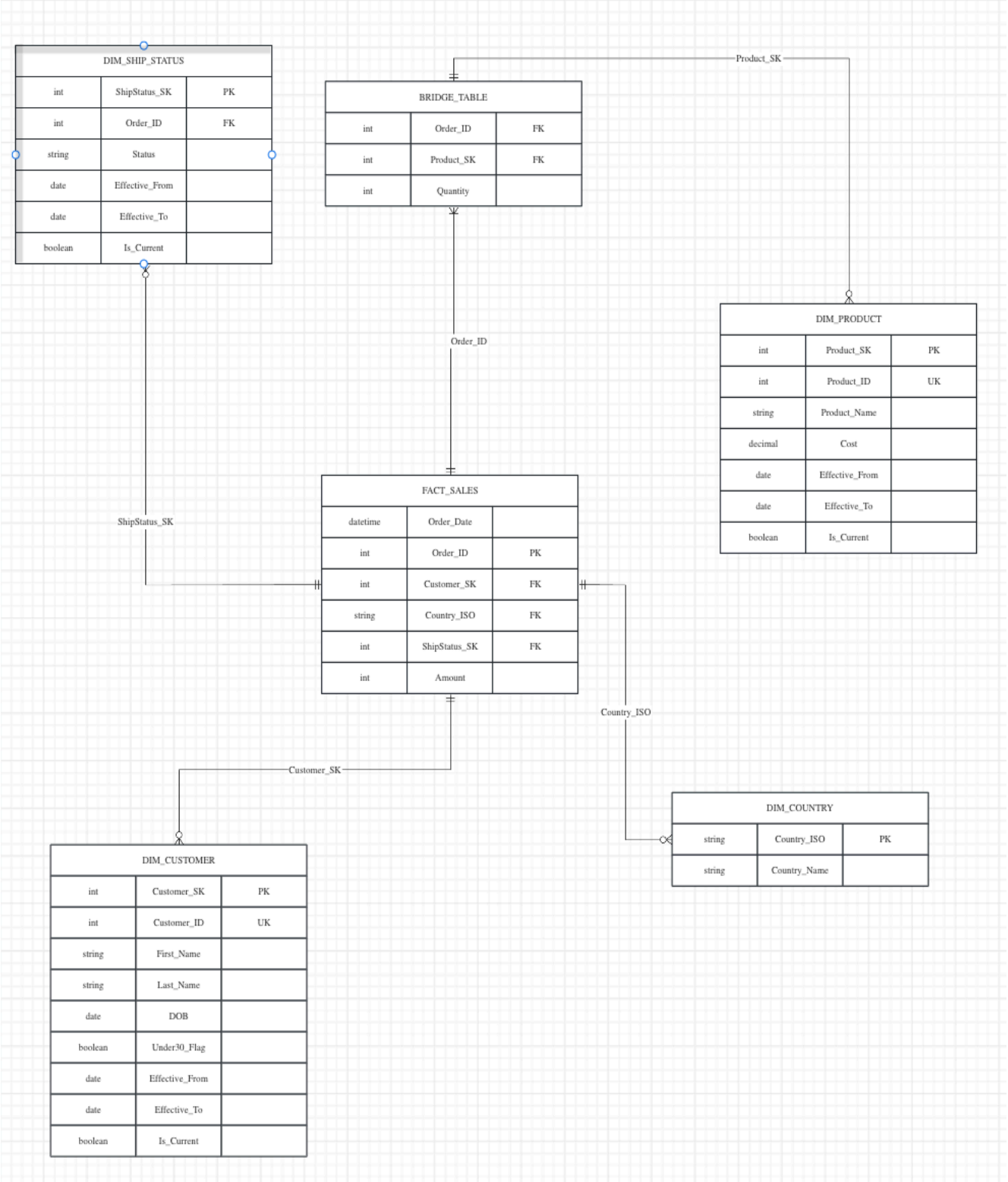
Proposed Iter-1 Of Conceptual ER Diagram



Conceptual model Overview

- **Three core entities only:**
  - **CUSTOMER** – the entity buying from us.
  - **ORDER** – a commercial transaction with an entity placed by a customer.
  - **SHIPPING** – the physical fulfilment of the order by an entity.
- **Simple, business-friendly relationships:**
  - A customer *places* an order (1-to-many).
  - An order results in shipping information (1-to-1 / 1-to-many, depending on split shipments).
  - A customer *receives* shipments (many-to-one).

Proposed Iter-1 Of Logical ER Diagram



1. Fact Table
FACT_SALES

- One row per order
- **Columns:**
  - Order_ID (PK)
  - Order_Date
  - Customer_SK → DIM_CUSTOMER
  - Country_ISO → DIM_COUNTRY
  - ShipStatus_SK → DIM_SHIP_STATUS
  - Amount

2. Bridge Table
BRIDGE_TABLE

- Captures order–product details (many-to-many)
- **Columns:**
  - Order_ID → FACT_SALES.Order_ID
  - Product_SK → DIM_PRODUCT.Product_SK
  - Quantity

3. Dimensions

- **DIM_CUSTOMER** (Type 2 SCD)
  - Customer_SK (PK)
  - Customer_ID (natural key, UK)
  - First_Name, Last_Name, DOB
  - Under30_Flag

    (CASE

     WHEN DATE_ADD(Date_of_Birth, INTERVAL 30 YEAR) > CURRENT_DATE() THEN TRUE

     ELSE FALSE

    END AS Under30_Flag)

  - Effective_From, Effective_To, Is_Current
- **DIM_COUNTRY**
  - Country_ISO (PK & natural key)
  - Country_Name
- **DIM_PRODUCT** (Type 2 SCD)
  - Product_SK (PK)
  - Product_ID (natural key, UK)
  - Product_Name, Cost
  - Effective_From, Effective_To, Is_Current
- **DIM_SHIP_STATUS** (Type 2 SCD)
  - ShipStatus_SK (PK)
  - Order_ID (FK for status history)
  - Status
  - Effective_From, Effective_To, Is_Current

4. Relationships & Grain

1. FACT_SALES ← DIM_CUSTOMER: 1 customer → many sales
2. FACT_SALES ← DIM_COUNTRY: orders tagged by customer's country
3. FACT_SALES ← DIM_SHIP_STATUS: each fact carries current status
4. BRIDGE_TABLE connects FACT_SALES to DIM_PRODUCT for line-item quantities
5. SCD Type 2 in DIM_CUSTOMER, DIM_PRODUCT, DIM_SHIP_STATUS via surrogate keys + Effective_From/To + Is_Current

The Final Physical Model can be prepared once the proposed logical model is agreed upon.

SQL-DDL Statement for Table Creations

Attached .sql file

QA Test Checklist

Attached QA Test File, EXTtable_Schema_Information.sql and EXTtable_Data_Quality.sql file.

Refer Interpretation of the script-1 Schema Info Check and Interpretation of the script-2 Data Quality Check

Schema Validation

1. Execute the query INFORMATION_SCHEMA.COLUMNS.
2. Each of the eight fields (Date_of_Birth, Country_ISO, Order_Date, Product_ID, Quantity, Order_ID, Shipping_Date, and Status_Update_Date) should have exactly one row with the appropriate data_type and is_nullable flags.
3. Any missing row ⇒ columns were either incorrectly named or not added.

Orphan Shipments

1. Calculate the number of orphan shipments.
2. A single row should appear: orphan_shipments = >0 ⇒ there are shipments that don't map to any sale.

Orphan Orders → Customers

1. Run the orphan-orders-customers anti-join.
2. **Expect** orphan_orders_customers = 0
3. **>0** ⇒ some orders point at non-existent customers.

Orphan Bridge → Product

1. Run the bridge-product anti-join.
2. **Expect** orphan_bridge_products = 0
3. **>0** ⇒ there are (Order_ID,Product_ID) pairs not found in DIM_PRODUCT.

SCD-Type-2 Versioning

1. Run the “versions >1” query on DIM_CUSTOMER.
2. **Initial load ⇒ 0 rows** (no history yet).
3. **After applying your change ⇒ exactly the Customer_ID(s)** you updated, each with version_count = number of versions.
4. **Unexpected rows** or **wrong counts** ⇒ your history-tracking logic malfunctioned.

Under30_Flag Validation

1. Run the Under30_Flag != expected_flag query.
2. **Expect 0 rows** (every Under30_Flag must match DATE_DIFF(CURRENT_DATE(),DOB,YEAR)<30).
3. **Any rows** ⇒ those Customer_SK have incorrect flags and need correction.