2) Anticipated Datasets & Proposed Domain Model

2.1 Design Principles

- **Keep grains clean**: **f_order** is at an order-line grain, while **shipping** is at a customer grain. I avoid mixing these grains in the same table.
- **Conform dimensions**: I reuse customer, country, and product dimensions across fact tables for consistent joins and filters.
- **Debias joins**: Because **shipping** is at the customer level, I prevent data fan-out by collapsing it to a single row per customer whenever a report needs shipping status.
- **Data quality built-in**: Data quality checks, such as name hygiene (digit/symbol mapping, stripping, and trimming), age guardrails, and referential integrity checks, are applied as part of the data load process.

2.2 Core Warehouse Entities

Fact: f_order (order-line grain)

- **Grain**: 1 row = 1 order line from the source data.
- **Columns**:
  - order_id (PK, from source)
  - customer_id (FK → d_customer)
  - product_id (FK → d_product)
  - country_id (FK → d_country) ← Stamped from the customer data at load time
  - quantity (INT, default 1)
  - amount (DECIMAL(18,2), amount > 0)
- **Notes**: This table does not include shipping status, as it's a customer-grain attribute. This design keeps the fact table clean and prevents double-counting.

Dim: d_customer (customer grain)

- **Columns**: customer_id (PK), first_name, last_name, age, country_id (FK → d_country)
- **Data Quality Rules Applied at Load**:
  - Map digits/symbols to letters (e.g., 0→o, 1→i, 3→e, 4→a, 5→s, 7→t, @→a, !→i).
  - Strip disallowed punctuation, then trim and collapse spaces.
  - Enforce age to be BETWEEN 10 AND 100.

Dim: d_country (country grain)

Note: A better approach is to have a dimension at location which can be snowflaked into other dimensions like region, country, state etc.

- **Columns**: country_id (PK), country_name (UNIQUE)
- **Population**: Populated with distinct countries from the customers source data.

Dim: d_product (product grain)

- **Columns**: product_id (PK), product_name (from Item in source data)
- **Future-proof Fields (nullable)**: category, unit_price

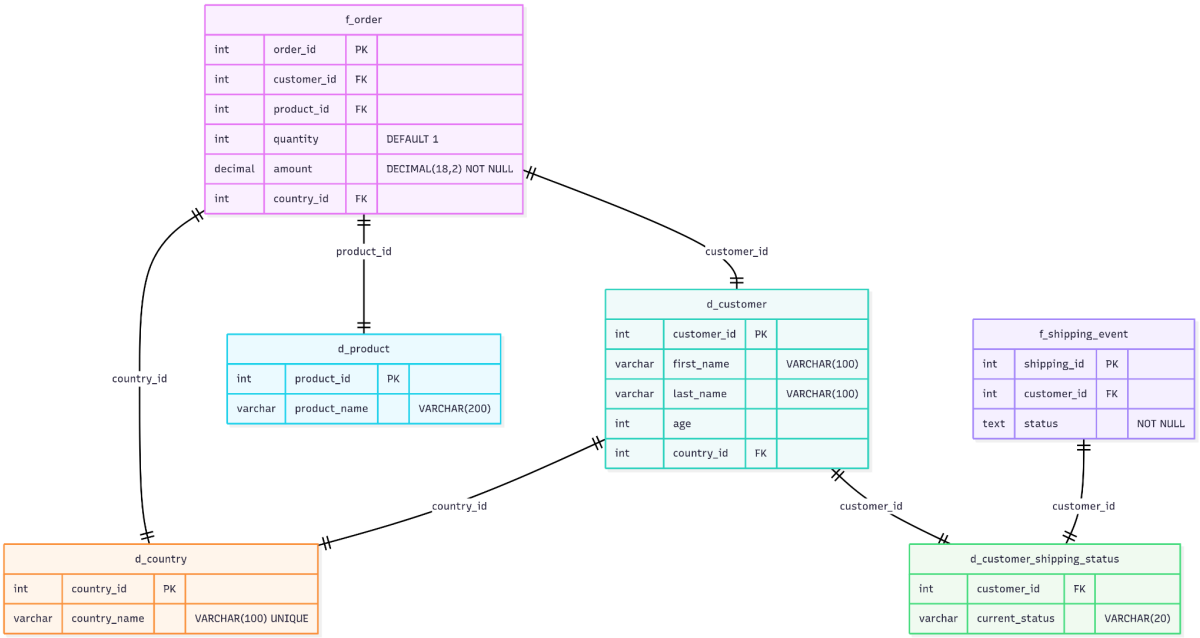
Fact (event): f_shipping_event (event grain, optional now)

- **Purpose**: To preserve raw shipping rows for lineage and history (multiple rows per customer).
- **Columns**: shipping_id (degenerate key), customer_id (FK), status_code (FK → d_customer_shipping_status).
- **Use Cases**: QA, tracking trends of status changes, and calculating time-to-delivered once timestamps become available.

Dim (outrigger): d_customer_shipping_status (customer grain, collapsed)

- **Grain**: 1 row per customer, representing their current/collapsed shipping status.
- **Columns**: customer_id (PK), current_status ∈ {Pending, Delivered, NoRecord}
- **Collapse Rule**: "Pending wins"; otherwise, Delivered; if no shipping rows exist, set to NoRecord.
- **Usage**: Used to join when a report needs the "current" shipping status (e.g., requirement A) without duplicating orders.

ER Diagram:



2.3 Dataset Layer (BI-facing views)

I'm exposing thin, reusable views on top of the core model so the BI tool can plug in directly.

- **DS-1 — Pending spend by country (Req A)**:
  - **Logic**: f_order → d_country, filtered by customer_id with a Pending status via d_customer_shipping_status.
  - **Fields**: country_name, total_amount_spent.
- **DS-2 — Customer × Product summary (Req B)**:
  - **Logic**: f_order → d_customer → d_product.
  - **Metrics**: total_transactions (count orders), total_quantity, total_amount.
  - **Dims Shown**: customer name, product name.
- **DS-3 — Top product by country (Req C)**:
  - **Logic**: Aggregate SUM(quantity) by (country, product) and select the max per country.
  - **Fields**: country_name, product_name, max_quantity.
- **DS-4 — Most purchased product by age category (Req D)**:
  - **Logic**: Derive age_category (<30 vs. >=30) from d_customer, then aggregate SUM(quantity) by (age_category, product) and select the max per band.
  - **Fields**: age_category, product_name, total_quantity.
- **DS-5 — Country with minimum transactions and sales (Req E)**:
  - **Logic**: f_order → d_country; group by country and order by transactions ASC, sales_amount ASC, returning the top row.
  - **Fields**: country_name, transactions, sales_amount.

2.4 Data Preparation & Transformation Requirements (for ELT jobs)

Sources (as landed):

- customers_raw (from Customer.csv)
- orders_raw (from Order.csv)
- shipping_raw (from Shipping.json)

Load/Transform Sequence:

1. **d_country**: INSERT DISTINCT country_name from customers_raw to get country_id.
2. **d_product**: INSERT DISTINCT product_name from orders_raw.Item.
3. **d_customer**:
   - Clean First and Last names (map, strip, trim/collapse).
   - Lookup country_id from d_country.
4. **f_order**:
   - Join orders_raw to the cleaned d_customer to get customer_id and country_id.
   - Lookup product_id from d_product.
   - Set quantity = 1.
   - Hard check: amount > 0.
5. **d_customer_shipping_status**:

- Collapse shipping_raw per customer using the "Pending wins" rule. Outer join to all customers so those without shipping data get a NoRecord status.
- Keep f_shipping_event (optional) to persist source events for lineage.

Data Quality:

- **Mandatory keys**: Customer_ID, Order_ID, Shipping_ID must be unique.
- **FK coverage**: All foreign keys must exist in their respective dimension tables.
- **Domains**: shipping.status is limited to {'Pending','Delivered'}; amount > 0; age is between 10-100.
- **Name Policy**: Implemented via a documented transformation.
- **Shipping Caveat**: The current feed is at a customer-grain. For future order-level shipping support, the feed would need to include Order_ID and timestamps.

2.5 Acceptance Criteria (Model-Level)

- **Grain integrity**:
  - f_order is strictly order-line; no customer attributes beyond keys.
  - d_customer_shipping_status is strictly customer-grain (1 row per customer).
- **Row parity & join safety**:
  - COUNT(f_order) must equal COUNT(orders_raw) (after filtering).
  - Joining f_order to d_customer_shipping_status must not change the row count (no fan-out).
  - d_customer row count must equal the number of valid customers.
- **Conformance**: All five reporting datasets (A–E) can be produced using only these entities/views without ad-hoc fixes.
- **Data Quality Gates**: No NULL foreign keys; amount > 0; names are cleaned; domain values are locked.

2.6 How This Model Answers the Business Questions

Requirement	Dataset	Join Path	Notes
A. Pending spend by country	DS-1	f_order → d_country + d_customer_shipping_status	Status collapsed per customer; avoids duplicates.
B. Customer totals w/ product	DS-2	f_order → d_customer → d_product	Counts, quantities, spend per customer × product.
C. Max product per country	DS-3	f_order → d_country → d_product	Max SUM(quantity) per country; deterministic tie-break if needed.
D. Top product by age band	DS-4	f_order → d_customer(age) → d_product	age_category derived at query/view layer.
E. Min country by txns & sales	DS-5	f_order → d_country	Order by (transactions, sales_amount) ascending.