**Detailed Data Quality Analysis**

Queries:

1. Data Completeness

* 100% completeness across all three datasets
* No null values or empty strings in any field
* All 750 records (250 per file) are complete and present

2. Structural Integrity

* No duplicate records found in any dataset
* All primary keys (Customer\_ID, Order\_ID, Shipping\_ID) are unique
* Sequential ID ranges are complete with no gaps (1-250 for each table)

3. Data Distribution

* Age range: 18-80 years with reasonable distribution across age groups
* Geographic coverage: USA (101 customers), UK (100 customers), UAE (49 customers)
* Product variety: 8 different items with consistent pricing

4. Data Inaccuracy

* 12 customers have special characters in their names
* Examples: "N!cole" (! character), "R0bert" (0 instead of O), "L@rry" (@ character), "Al1cia" (1 instead of i)
* Recommendation: Standardize names by replacing special characters with proper letters

5. Data Inconsistency

* 1 product Mousepad has inconsistency in it’s price
* 2 different pricing found for mousepad 200 & 250.

6. Referential Integrity

| **Customer Relationship** | **Count** |
| --- | --- |
| Customers without orders in order table | 90 |
| Customers with both orders and shipping | 99 |
| Customers with orders but no shipping | 61 |
| Customers with shipping but no orders | 55 |
| Customers with neither | 35 |

7. Business Logic Violations

* 94 orders without corresponding shipping records
* 98 shipping records without corresponding orders
* $207,300 in revenue at risk from orders lacking shipping information
* Shipping records without orders violate fundamental business logic
* There are few orders which are shipped at different times based on shipping records but there is no product level information in shipping to track the status at product level

**Business Impact Assessment**

Financial Impact

* $207,300 in revenue at risk from orders without shipping information
* Operational inefficiency from maintaining separate, unlinked records
* Customer service issues from inability to track order fulfilment

Data Reliability Impact

* 60.4% of customers have incomplete transaction records
* 38.19% delivery rate of valid orders
* Reporting accuracy compromised due to referential integrity issues
* Analytics and forecasting reliability significantly reduced as there is no order date or shipping date for time series analysis

**Recommendations for Resolution**

1. Reconcile Order and Shipping Relationships
   * Create shipping records for 94 orders without tracking
   * Investigate and resolve 98 orphaned shipping records
   * Establish clear business rules for order-shipping linkage
   * Establish clear business rules for shipping-product linkage
2. Data Governance Implementation
   * Implement referential integrity constraints in database
   * Establish data validation rules before record creation
   * Create automated checks for order-shipping matching
3. Add Key Fields to the Data
   * Order Placement Date/Time: When the order was created in the system (e.g., Order\_Placed\_Timestamp in Order.csv)
   * Order Shipped Date/Time: When the order was handed over to a carrier (e.g., Shipped\_Timestamp in Shipping.csv)
   * Order Delivered Date/Time: When the order was confirmed as delivered to the customer (e.g., Delivered\_Timestamp in Shipping.csv)
   * Validate that every order has a placement date
   * For every shipped order, there must be a corresponding shipped date and delivered date if completed
   * For every shipped order, there must be a corresponding order data linked to it rather than every shipment being linked to a customer.
   * Shipping ID do not hold information on which product of a single order is being shipped. Hence, add product information in shipping
4. Process Improvements
   * Establish mandatory shipping record creation upon order confirmation
   * Implement end-to-end transaction tracking
   * Whenever an order is placed, automatically record the current timestamp.
   * When an item is shipped, log the shipment time.
   * Upon delivery confirmation, update a delivery timestamp.
   * These can be automated in your order management/shipping system.
   * Establish business process to capture order-shipping linkage

**2. Dataset Requirements and Domain Model**

**Source Tables:**

Customer:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Required** | **Constraints** | **Example** |
| Customer\_ID | INT | Unique customer identifier | Yes | Primary Key, Not Null | 101 |
| First | varchar 50 | First name of the customer | Yes | Not Null | John |
| Last | varchar 50 | Last name of the customer | Yes | Not Null | Smith |
| Age | INT | Age of the customer | Yes | Between 10 and 100 | 35 |
| Country | varchar 10 | Country of residence | Yes | Not Null | USA |

Order:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Required** | **Constraints** | **Example** |
| Order\_ID | INT | Unique order identifier | Yes | Primary Key, Not Null | 5001 |
| Line Number | INT | Unique Line number for each Item in an Order | Yes | Not Null | 1 |
| Item | STRING | Product or item name | Yes | Not Null | Running Shoes |
| Invoice Amount | FLOAT | Total order amount | Yes | Must be greater than 0 | 120.5 |
| Quantity | INT | No of items for each product | Yes | Not Null | 10 |
| Unit Price | FLOAT | Price of single Item | Yes | Must be greater than 0 | 20.5 |
| Customer\_ID | INT | Link to Customer table | Yes | Foreign Key, Not Null | 101 |
| Order\_Date | DATETIME | Date of order placed | Yes | Not Null | 2025-01-01 00:00:00 |

Shipping:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Required** | **Constraints** | **Example** |
| Shipping\_ID | INT | Unique shipping identifier | Yes | Primary Key, Not Null | 9001 |
| Status | STRING | Delivery status | Yes | should be either 'Pending', 'Delivered' | Delivered |
| Customer\_ID | INT | Link to Customer table | Yes | Foreign Key, Not Null | 101 |
| Order\_ID | INT | Link to Order table | Yes | Foreign Key, Not Null | 101 |
| Line Number | INT | Link to Order table | Yes | Not Null | 1 |
| Shipping\_Date | DATETIME | Date of shipment handed over to a carrier | Yes | Not Null | 2025-01-01 00:00:00 |
| Delivery\_Date | DATETIME | Date of shipment delivered to customer | Yes | NULLABLE IF PENDING | 2025-01-01 00:00:00 |

**Target Tables:**

Create a star schema with Facts and Dimensions with 3NF normalisation to avoid data redundancy and to ensure the data model is future proof ready and allows scaling without any modifications.

|  |  |  |
| --- | --- | --- |
| Table | Purpose | Grain |
| dim\_customer | Customer Data | One row per customer |
| dim\_product | Product data | One row per unique product (by name) |
| dim\_geography | Country data | One row per unique Country |
| dim\_date | Date table | One row per calendar day |
| fact\_sales | Sales transactions | One row per order (unique Order\_ID+Product\_ID) |
| fact\_shipping | Shipping status | One row per shipping record (unique (Order\_ID + Line\_Number)) |

Dim\_Customer:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data Type** | **Description** | **Transform/Constraints** |
| customer\_key | INT, PK | Surrogate key (auto) | Identity(1,1) |
| customer\_id | INT, UNIQUE | Business key | from source customer |
| first\_name | VARCHAR(50) | Cleaned first name | trim spaces, remove special chars |
| last\_name | VARCHAR(50) | Cleaned last name | trim spaces, remove special chars |
| age | INT | Customer age | Check: age between 18 & 100 |
| Country | VARCHAR(10) | Country Code |  |
| is\_active | BOOL | All set to TRUE unless deactivated |  |
| created\_timestamp | DATETIME | Timestamp of load date |  |
| Begindate\_timestamp | DATETIME | Timestamp of date when details was updated |  |
| Enddate\_timestamp | DATETIME | Timestamp until when the customer will stay active |  |

Dim\_Product:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data Type** | **Description** | **Transform/Constraints** |
| product\_key | INT, PK | Surrogate key | Identity (1,1) |
| product\_name | VARCHAR(100) | Business key | from Orders, deduplicate |
| is\_active | BOOL | TRUE unless deactivated |  |
| created\_timestamp | DATETIME | Timestamp of load date |  |
| updated\_timestamp | DATETIME | Timestamp of date when details was updated |  |
| Enddate\_timestamp | DATETIME | Timestamp until when the customer will stay active |  |

Dim\_Geography: Optional, it is useful for geography specific analysis with more granularity

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Description** | **Transform/Constraints** |
| geography\_key | INT, PK | Surrogate key | Identity(1,1) |
| country\_code | VARCHAR(10) | Standardized (e.g. USA) | From Customers, dedup |
| country\_name | VARCHAR(50) | e.g. "United States" | Lookup table if needed |
| is\_active | BOOL | TRUE unless deactivated |  |
| created\_timestamp | DATETIME | Timestamp of load date |  |
| updated\_timestamp | DATETIME | Timestamp of date when details was updated |  |
| Enddate\_timestamp | DATETIME | Timestamp until when the customer will stay active |  |

**NOTE:** Dimension tables above are designed to be implemented as SCD type 2 by BEGIN\_DATE,END\_DATE along with the Is\_Active flag that is present in the table to track all the changes in the dimensions.

Dim\_Date:

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| date\_key | INT, PK | e.g., YYYYMMDD |
| full\_date | DATE | Calendar date |
| year | INT | Year |
| quarter | INT | Calendar quarter |
| month | INT | Month |
| day | INT | Day of month |

Fact\_Sales:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Description** | **Transform/Constraints** |
| sale\_id | INT, PK | Surrogate key (auto) | Identity(1,1) |
| order\_id | INT | Natural key | from Source Order table |
| customer\_key | INT, FK | Customer dimension | Join by customer\_key dim customer |
| product\_key | INT, FK | Product dimension | Join by product\_key dim product |
| geography\_key | INT, FK | Geography dimension | Join by geography\_key dim geography |
| order\_date\_key | INT/FK | Date dimension | Join by Date\_Key dim\_date |
| Line Number | INT | From Source order | Not Null and >0 |
| Invoice amount | DECIMAL(10,2) | Order value | from Source Order table ,Always > 0 |
| Quantity | INT | Number of items | from Source Order table, always >0 |
| Unit Price | FLOAT (10,2) | Price of single Item | from Source Order table, always >0 |
| Line\_Total | FLOAT (10,2) | Total amount for each line (quantity \* unit price) | always >0 |
| created\_timestamp | DATETIME | Timestamp of load date |  |
| updated\_timestamp | DATETIME | Timestamp of updated date |  |

Fact Shipping:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Description** | **Transform/Constraints** |
| shipping\_key | INT, PK | Surrogate key | Identity (1,1) |
| shipping\_id | INT | Shipping natural key | From Source Shipping |
| order\_id | INT, FK | Order natural key | Join source Shipping ,dim\_customer and Order to extract, must not be NULL |
| Line Number | INT | From Source order | Not Null and >0 |
| status | VARCHAR(20) | 'Pending', 'Delivered', etc | Normalize case |
| shipping\_date\_key | INT/FK | Ship date for delivery analysis | From Source Shipping |
| delivery\_date\_key | INT/FK | delivery date for delivery analysis | From Source Shipping |
| created\_timestamp | DATETIME | Timestamp of load date |  |
| updated\_timestamp | DATETIME | Timestamp of updated date |  |

NOTE: The above two fact tables are implemented with SCD 1 types to simply overwrite any changes.

**User Story: Fact\_sales Table Implementation**

As a: Data Engineer  
I want to: Build a fact\_sales table that supports order-line level grain  
So that: We can track multi-product orders, partial shipments, and product-specific analytics

**Acceptance Criteria**

* Support order-line level tracking (one row per order line)
* Handle multi-product orders with separate line tracking
* Maintain referential integrity with all dimension tables
* Support quantity-based pricing (unit\_price × quantity = line\_total)
* Enable partial order fulfillment analysis
* Provide complete audit trail with ETL timestamps
* Ensure data quality through validation rules
* Support efficient querying for business analytics

**Technical Specifications:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Constarints** | **NULLABLE** | **Transform/Constraints** |
| sale\_id | PK, IDENTITY(1,1) | NO | Clustered PK |
| order\_id | NOT NULL | NO | Composite with Line\_number |
| line\_number | NOT NULL | NO | Composite with order\_id |
| customer\_key | FK, NOT NULL | NO | None |
| product\_key | FK, NOT NULL | NO | None |
| geography\_key | FK, NOT NULL | NO | None |
| order\_date\_key | FK, NOT NULL | NO | None |
| unit\_price | NOT NULL, > 0 | NO | None |
| quantity | NOT NULL, > 0 | NO | None |
| line\_total | NOT NULL, = unit\_price \* quantity | NO | None |
| created\_timestamp | NOT NULL, DEFAULT GETDATE() | NO | None |
| updated\_timestamp | NOT NULL, DEFAULT GETDATE() | NO | None |

Column-by-Column ETL Transformations

* 1. sale\_id
* Source: System enerated
* Logic: Auto-increment surrogate key starting from 1
* Validation: Ensure uniqueness and sequential numbering
* Error Handling: System-generated, no validation needed
  1. **order\_id**
* Source: Order.Order\_ID
* Logic: Direct mapping from source Order\_ID field
* Validation: Must exist in source, must be positive integer
* Error Handling: Reject row if NULL or ≤ 0

**3. line\_number**

* Source: Order.Line\_Number
* Logic: Direct mapping, represents line sequence within order
* Validation: Must be positive integer, unique within order\_id
* Error Handling: Reject if NULL, ≤ 0, or duplicate within same order

**4. customer\_key**

* Source: Lookup via Customer\_ID
* Logic: JOIN Order.Customer\_ID with dim\_customer.customer\_id to get customer\_key
* Validation: Must resolve to valid customer\_key in dim\_customer
* Error Handling: Reject row if customer not found in dimension

**5. product\_key**

* Source: Lookup via Item name
* Logic: JOIN Order.Item with dim\_product.product\_name to get product\_key
* Validation: Must resolve to valid product\_key in dim\_product
* Error Handling: Create new product record if not found, then use key

**6. geography\_key**

* Source: Lookup via Customer\_ID to fetch Country
* Logic: Chain lookup: Customer\_ID to dim\_customer.country to dim\_geography.geography\_key
* Validation: Must resolve to valid geography\_key
* Error Handling: Create new geography record if not found

**7. order\_date\_key**

* Source: Order.Order\_Date
* Logic: Convert date to YYYYMMDD integer format for dim\_date lookup
* Validation: Must be valid date, must exist in dim\_date
* Error Handling: Reject if invalid date or not in dim\_date

**8. unit\_price**

* Source: Order.Unit\_Price
* Logic: Direct mapping with decimal precision validation
* Validation: Must be > 0, max 2 decimal places, ≤ 999999.99
* Error Handling: Reject if ≤ 0, NULL, or invalid format

**9. quantity**

* Source: Order.Quantity
* Logic: Direct mapping as positive integer
* Validation: Must be positive integer, ≤ 9999
* Error Handling: Reject if ≤ 0, NULL, or not integer

**10. line\_total**

* Source: Calculated
* Logic: unit\_price \* quantity with precision handling
* Validation: Must equal unit\_price \* quantity, ≤ 9999999.99
* Error Handling: Recalculate if mismatch, reject if overflow

**11. Audit Timestamps**

* Source: ETL Runtime
* Logic: Current timestamp when record is inserted/updated
* Validation: updated\_timestamp ≥ created\_timestamp
* Error Handling: System-generated, always valid

**User Acceptance Test Scenarios**

**QA Test Cases:**

1. Functional Test Cases

TC1.1: Insert Valid Sales Row

Insert a row with all valid keys and positive unit\_price/quantity.

Expected: Row is inserted, line\_total = unit\_price × quantity, all FKs resolve.

TC1.2: MultiProduct Order

Insert three rows for an order each for a different product.

Expected: All rows inserted; unique (order\_id, line\_number) constraint holds.

TC1.3: MultiQuantity Order Line

Insert row with order\_id, line\_number, product\_key, quantity = 5, unit\_price = 100.

Expected: line\_total must be 500; row inserted.

TC1.4: Audit Columns Generation

Insert any valid row.

Expected: created\_timestamp and updated\_timestamp are autopopulated with current time.

2. Negative/Validation Test Cases

TC2.1: Null order\_id

Attempt insert with order\_id = NULL.

Expected: Reject (not null constraint).

TC2.2: Duplicate (order\_id, line\_number)

Insert row, then insert another row with same order\_id, line\_number.

Expected: Reject (unique constraint).

TC2.3: Invalid Foreign Key

Insert row with non existent customer\_key/product\_key/geography\_key/order\_date\_key.

Expected: Reject (FK violation).

TC2.4: unit\_price ≤ 0

Insert row with unit\_price = 0 or <0.

Expected: Reject (check constraint).

TC2.5: quantity ≤ 0

Insert row with quantity = 0 or <0.

Expected: Reject (check constraint).

TC2.6: order\_id Not Integer

Attempt insert with text or float for order\_id.

Expected: Reject (type constraint).

3. Data Integrity Test Cases

TC3.1: Double Insertion Check

Attempt loading same sales file twice.

Expected: Duplicate (order\_id, line\_number) rows rejected.

TC3.2: Missing Required Columns

Attempt insert missing product\_key or geography\_key.

Expected: Reject (not null).

4. Boundary Test Cases

TC4.1: Min and Max Values

Insert row with quantity = 1 (min), quantity = 9999 (max), unit\_price very small (0.01) or very large (999999.99).

Expected: Row accepted if within bound, rejected if exceeds specified max/min.

TC4.2: Precision Verification

Insert decimal unit\_price like 123.45 × 67 (quantity).

Expected: line\_total precision and scale correct.

5. Referential Integrity Test Cases

TC5.1: Deleting Referenced Dimension Row

Try deleting a product from dim\_product which is referenced in fact\_sales.

Expected: Reject (FK constraint), unless cascade delete is set (should not be).