Invoice-Based Dimensional Modeling Technical Report

1. Introduction

This report outlines a dimensional modeling solution for a retail company's invoice data. The goal is to design a scalable, high-performance data warehouse that supports analytical reporting, historical tracking, and drill-down analysis. By reverse-engineering the provided invoice dataset, we identify core business dimensions (Customer, Product, Store, Time) and define strategies for handling slowly changing dimensions (SCDs), data integration, and performance optimization.

2. Data Profiling & Inference

2.1 Dataset Overview

The sample dataset includes transactional invoice line items with attributes such as:

- **Measures**: `Quantity`, `Unit_Price`, `Line_Total` (calculated as `Quantity * Unit_Price`).
- **Dimensions**: `Customer_ID`, `Product_ID`, `Store_ID`, and `Invoice_Date`.

Granularity: Each row represents a **line-item-level transaction**, enabling drill-down analysis (e.g., sales by product, customer, or store).

2.2 Identified Dimensions

1. **Customer**: Attributes include `Customer_ID`, `Customer_Name`, and inferred demographics (e.g., loyalty tier based on purchase frequency).
2. **Product**: Attributes include `Product_ID`, `Product_Name`, and inferred hierarchy (e.g., `Category → Sub-Category → Product`).
3. **Store**: Attributes include `Store_ID`, `Store_Location` (inferred from transactions), and `Store_Type`.
4. **Time**: Derived from `Invoice_Date`, with hierarchy `Year → Quarter → Month → Day`.

2.3 Fact Table Granularity

The **line-item-level granularity** of the `Fact_Sales` table (one row per product sold within an invoice) is intentionally chosen to align with the **inferred business objectives** of detailed analytical reporting and historical tracking.

Justification:

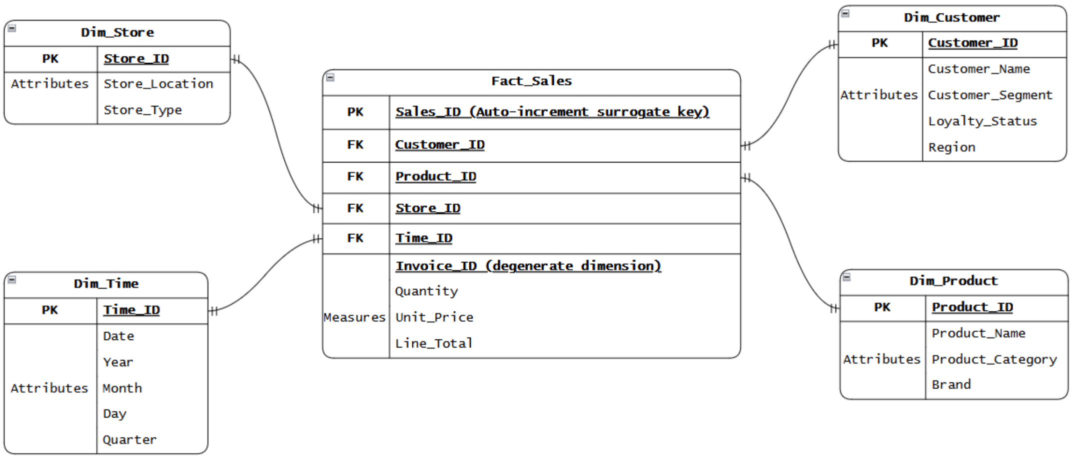
1. **Analytical Flexibility**:
    - Enables answering questions at multiple levels:
        - **Product-Centric**: "Which products drive Q3 revenue?"
        - **Customer-Centric**: "How many units of Product X did Customer Y buy last year?"
        - **Store-Centric**: "How does Store S01's electronics sales compare to S02's?"
    - Invoice-level granularity would aggregate products, limiting product/customer insights.
2. **Drill-Down Capability**:
    - Supports hierarchical analysis (e.g., sales by **Category → Sub-Category → Product** or **Region → Store → Day**).
3. **Historical Accuracy**:
    - Tracks individual product trends (e.g., USB-C Hub's Q4 spikes) and customer behavior shifts (e.g., switching from Product A to B).
4. **Scalability**:
    - Modern cloud warehouses (e.g., Snowflake) efficiently manage large datasets via partitioning and columnar storage.

Trade-Off Mitigation:

- **Storage Overhead**: Addressed via partitioning by month and compression.
- **Complexity**: Simplified querying via star schema design.

3. Dimensional Modeling Strategy

STAR SCHEMA



3.1 Star Schema Design

- **Fact_Sales**: Central table with transactional data.
  - **Surrogate Key**: `Sales_ID` (unique identifier).
  - **Foreign Keys**: `Customer_ID`, `Product_ID`, `Store_ID`, `Time_ID`.
  - **Measures**: `Quantity`, `Unit_Price`, `Line_Total`.
  - **Degenerate Dimension**: `Invoice_ID`.
- **Dimension Tables**:
  - **Dim_Customer**: `Customer_ID`, `Customer_Name`, `Loyalty_Tier`.
  - **Dim_Product**: `Product_ID`, `Product_Name`, `Category`, `Sub_Category`.
  - **Dim_Store**: `Store_ID`, `Store_Location`, `Store_Type`.
  - **Dim_Time**: `Time_ID`, `Date`, `Year`, `Quarter`, `Month`, `Day`.

Conformed Dimensions:

- `Dim_Time` is reusable across processes (e.g., inventory, promotions).

3.2 Degenerate Dimensions

- `Invoice_ID` is stored directly in `Fact_Sales` as it has no additional attributes.

4. Slowly Changing Dimensions (SCDs)

4.1 SCD Strategy

Dimension	SCD Type	Rationale
Customer	Type 2	Track historical changes (e.g., loyalty tier, address) for trend analysis.
Product	Type 2	Preserve history of product attributes (e.g., category changes).
Store	Type 1	Overwrite location/type changes if historical tracking is not critical.

Type 2 Implementation Example (SQL):

```
-- Dim_Customer with Type 2 SCD
CREATE TABLE Dim_Customer (
  Customer_SK INT PRIMARY KEY AUTOINCREMENT,
  Customer_ID VARCHAR(10),
  Customer_Name VARCHAR(50),
  Loyalty_Tier VARCHAR(20),
  Start_Date DATE,
  End_Date DATE,
  Is_Current BOOLEAN
);
```

- **New records** are inserted with `Start_Date = CURRENT_DATE` and `Is_Current = TRUE`.
- **Old records** are expired with `End_Date = CURRENT_DATE` and `Is_Current = FALSE`.

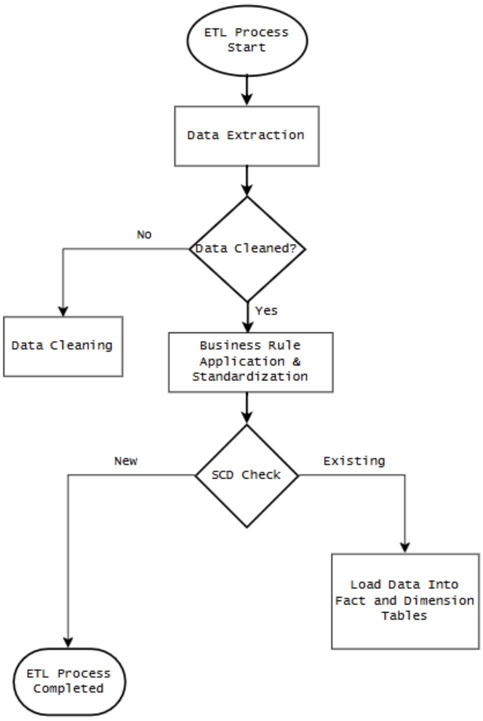
Trade-Offs:

- **Type 2**: Increases storage but enables historical accuracy.
- **Type 1**: Simplifies ETL but loses history.

5. ETL/ELT Strategy

5.1 Data Integration Workflow

ETL FLOWCHART



1. **Extract**:
    - **Initial Load**: Full extract of historical invoices.
    - **Incremental Load**: Daily extracts using `Invoice_Date` or CDC.
2. **Transform**:
    - **Data Cleansing**: Handle missing `Unit_Price` (default to average price).
    - **SCD Handling**: Compare source data with existing records for changes.
    - **Referential Integrity**: Validate `Product_ID` and `Store_ID` against dimension tables.
3. **Load**:
    - **Fact Table**: Insert new line items with surrogate keys.
    - **Late-Arriving Data**: Update `Fact_Sales` with missing dimension keys.

5.2 Data Quality & Consistency

- **Constraints**: Enforce `Quantity > 0` and `Unit_Price > 0`.
- **Line_Total Validation**: Triggers ensure `Line_Total = Quantity * Unit_Price`.

6. Advanced Analysis & Scalability

6.1 Hierarchical Drill-Down

- **Product Hierarchy**: Aggregate sales from `Category` → `Sub-Category` → `Product`.
- **Time Hierarchy**: Analyze trends by `Year → Quarter → Month`.

Sample Query (Monthly Sales by Store):

```
SELECT
  S.Store_Location,
  T.Year,
  T.Month,
  SUM(F.Line_Total) AS Total_Sales
FROM Fact_Sales F
JOIN Dim_Store S ON F.Store_ID = S.Store_ID
JOIN Dim_Time T ON F.Time_ID = T.Time_ID
GROUP BY ROLLUP(S.Store_Location, T.Year, T.Month);
```

6.2 Performance Optimization

- **Partitioning**: Split `Fact_Sales` by `Year-Month`.
- **Indexing**: Indexes on `Time_ID`, `Store_ID`, and `Product_ID`.
- **Cloud Scalability**: Use columnar storage (e.g., BigQuery).

7. Assumptions & Trade-Offs

1. **Assumptions**:
    - Source data excludes returns/discounts.
    - `Store_Location` is inferred contextually.
2. **Trade-Offs**:
    - **Star Schema**: Denormalization improves performance but increases redundancy.
    - **SCD Type 2**: Historical accuracy vs. storage cost.

8. Use Cases

1. **Customer Behavior**: Track loyalty tier changes (Type 2 SCD) vs. purchase frequency.
2. **Product Performance**: Compare sales before/after category changes.
3. **Store Trends**: Identify top performers using partitioned tables.

9. Conclusion

This model balances historical accuracy (via Type 2 SCDs) and performance (via partitioning/indexing). Line-item granularity ensures drill-down capabilities, while scalable design supports future growth. Enhancements could include handling returns or integrating promotions.