PROJECT: DESIGNING AN END-TO-END DIMENSIONAL MODEL FOR INVOICE DATA

1. Introduction

Data-driven decision-making is critical in modern business environments, particularly for organizations dealing with high volumes of transactional data. Dimensional modeling provides a structured approach for designing data warehouses optimized for analytical processing. This project outlines the development of a star schema-based dimensional model for analyzing the invoice data provided. I have ensured that our model supports scalability, data quality, historical tracking, and drill-down capabilities. This report details the data profiling process, fact and dimension table design, strategies for handling data challenges, and performance optimizations to support business intelligence (BI) reporting.

2. Data Profiling and Inference Process

Before designing the dimensional model, I carried out the below data profiling to understand the dataset and determine the required fact and dimension tables. This involved analyzing the data structure, identifying relationships, and inferring meaningful business attributes.

2.1 Understanding the Raw Data – Sample Invoice

The provided sample invoice data includes the following fields:

Column Name	**Description**
Invoice_ID	Unique identifier for each transaction
Invoice_Date	Timestamp of the purchase
Customer_ID	Unique identifier for the customer
Customer_Name	Name of the customer
Product_ID	Unique identifier for the product
Product_Name	Name of the product purchased
Quantity	Number of units bought
Unit_Price	Price per unit
Line_Total	Total price (Quantity × Unit Price)
Store_ID	Identifier for the store where the sale occurred

From the dataset, I can infer several business dimensions:

a. **Customer Dimension**: Each purchase is linked to a customer, indicating the need to store and track customer details.
b. **Product Dimension**: Products have unique IDs and names, suggesting a structured catalogue of available items.
c. **Time/Date Dimension**: Transactions occur at specific timestamps, enabling analysis of sales trends over different time periods.
d. **Store Dimension:** Each sale is associated with a store, allowing for store performance tracking.

Understanding the Fact Table: The invoice itself is as a major record of the business event, capturing key sales transactions such as Customer, Product, Time, and Store from which the Product was supplied.

3. Dimensional Model Design

I adopted a star schema approach. This ensures simplicity, fast query performance, and flexibility for analytical reporting. The model design is attached in a separate link.

3.1 Fact Table: Fact_Invoice

The **Fact_Invoice** table serves as the central table; it captures and stores every transactional event. It includes foreign keys linking to dimensional tables and key metrics that support business/sales analysis.

Column Name	Data Type	Description
Invoice_ID	String (PK)	Unique invoice identifier
Date_ID	Integer (FK)	Foreign key to `Dim_Date`
Customer_ID	String (FK)	Foreign key to `Dim_Customer`
Product_ID	String (FK)	Foreign key to `Dim_Product`
Store_ID	String (FK)	Foreign key to `Dim_Store`
Quantity	Integer	Number of items purchased
Unit_Price	Decimal	Price per unit
Line_Total	Decimal	Total revenue (Quantity × Unit_Price)

This table supports performance-driven analytical queries by reducing redundancy while maintaining a direct link to dimensional attributes.

3.2 Dimension Tables

The Dimension tables store descriptive data, providing context for business transactions.

3.2.1 `Dim_Customer` (Customer Dimension)

Captures customer-related details for segmentation and retention analysis.

Column Name	Data Type	Description
Customer_ID	String (PK)	Unique identifier for the customer
Customer_Name	String	Full name of the customer
Customer_Segment	String	Category (Regular, VIP, etc.)
Location	String	Customer's region (if applicable)

3.2.2 `Dim_Product` (Product Dimension)

Stores product details for category-wise sales analysis.

Column Name	Data Type	Description
Product_ID	String (PK)	Unique product identifier
Product_Name	String	Product name
Category	String	Product category (Electronics, Accessories, etc.)
Price	Decimal	Standard unit price

3.2.3 `Dim_Date` (Date Dimension)

Enables **historical trend analysis** by storing time-related attributes.

Column Name	Data Type	Description
Date_ID	Integer (PK)	Surrogate key for the date
Date	Date	Actual date
Day	Integer	Day of the month
Month	Integer	Month number
Month_Name	String	Month name (March, etc.)
Quarter	Integer	Quarter (Q1-Q4)
Year	Integer	Year (e.g., 2025)
Day_of_Week	String	Name of the day

3.2.4 `Dim_Store` (Store Dimension)

Facilitates **store performance analysis** across locations.

Column Name	Data Type	Description
Store_ID	String (PK)	Unique store identifier
Store_Location	String	City or region of the store
Store_Type	String	Type of store (Online, Physical)

4. Addressing the Challenges in Dimensional Modeling

4.1 Handling Slowly Changing Dimensions (SCDs)

Changes in customer segments, product categories, or store types must be managed carefully.

I used:

- SCD Type 1 (Overwrite): Used for correcting non-critical changes (e.g., typos in names or minor updates).

- SCD Type 2 (Track History): Applied to customer segments and product categories to maintain historical accuracy. It is used for critical updates (e.g., customers moving from "Regular" to "VIP").

4.2 Strategies I applied in Ensuring Data Quality:

- **Use Standardize format:** (e.g., consistent date formats).
- **Data Validation:** Check for missing values in key attributes.
- **Remove duplicates:** Remove duplicate records to prevent incorrect reporting (e.g., prevent duplicate invoices).
- **Referential Integrity:** Ensure that all foreign keys in the fact table exist in dimension tables.

4.3 Performance Optimization process:

- Indexing foreign keys: for faster queries.
- Partitioning fact table by date for performance improvement.
- Pre-aggregations: Store precomputed summaries for faster reporting.

5. Meeting Business Objectives with the Model

5.1 Business Insights Enabled

- **Sales Analysis:** Track revenue trends, analyze sales by products, store, customer segment, and time.
- **Customer Behavior:** Track customer purchases, identify high-value customers, Identify frequent buyers and segment customer behavior.
- **Product & Store Performance Evaluation:** Identify best-selling products and top-performing stores.
- **Historical Analysis:** The **Date Dimension** enables trend analysis over months, quarters, and years.

5.2 Scalability

- Handles large transaction volumes efficiently
- Supports new products, customers, and stores without redesigning the schema

6. ETL Process

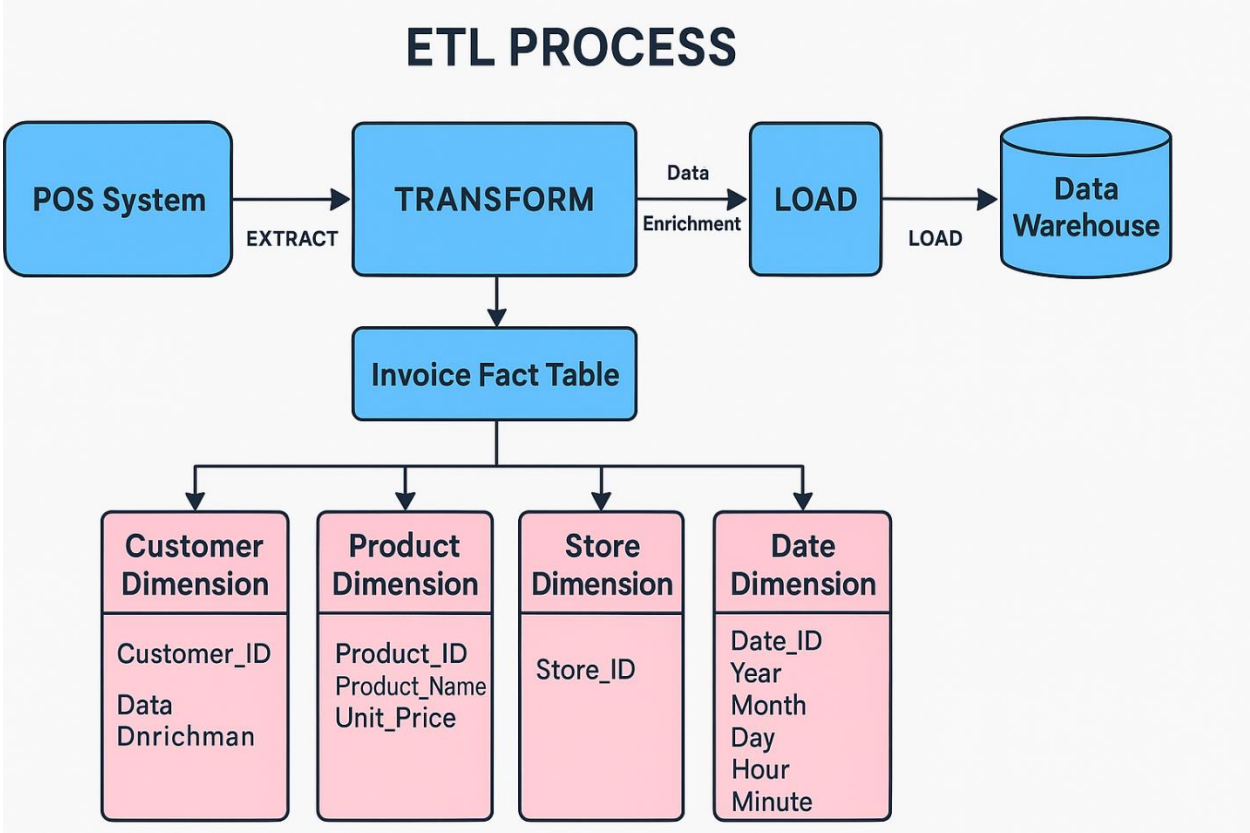
We utilized ETL process flow, involving Data Extraction (Extract raw invoice data from operational systems), **Data Transformation (**Mapping raw data to the dimensional model; Sorting missing values, clean, standardize, data), and **Data Loading (**Load dimension tables first, then fact tables, ensure referential integrity using foreign key constraints).

The detailed process flow is below:

ETL Process Flow for the sample invoice data:

1. **Extract**
    - **Source:** POS System, CSV files, or a relational database (e.g., MySQL, PostgreSQL).
    - **Extraction tools:** Python (Pandas), SQL queries, Apache NiFi, or ETL tools like Talend.
2. **Transform**
    - **Data Cleansing:** Handling missing values, standardizing formats.
    - **Data Enrichment:** Adding calculated fields (e.g., total revenue).
    - **Data Normalization:** Structuring data into **fact and dimension tables**:
        - **Fact Table: Invoice Fact Table**
        - **Dimension Tables:**
            - **Customer Dimension** (Customer_ID, Customer_Name)
            - **Product Dimension** (Product_ID, Product_Name, Unit_Price)
            - **Store Dimension** (Store_ID)
            - **Date Dimension** (Date_ID, Year, Month, Day, Hour, Minute)
3. **Load**
    - Data is loaded into a Data Warehouse (e.g., Snowflake, Redshift, BigQuery).
    - Common tools: Apache Airflow, AWS Glue, or custom ETL scripts.

See Diagram of the Process Flow below:



7. Assumptions & Trade-Offs

7.1 Assumptions

These are informed guesses or decisions made due to limited information or business rules inferred from the data. They ensure clarity in the design choices.

A. Data & Business Rules Assumptions

1. ***Invoice Data is Immutable:*** Once an invoice is generated, it cannot be modified. If changes are required, a new invoice must be issued.

2. ***Customers Can Make Multiple Purchases:*** A customer (Customer_ID) may appear in multiple invoices.

3. ***One Invoice Can Contain Multiple Products:*** Each Invoice_ID can have multiple Product_IDs, meaning a single invoice records multiple line items.

4. ***Each Product Has a Fixed Price Per Invoice:*** The Unit_Price is determined at the time of the invoice and does not change dynamically.

5. ***Stores Operate Independently:*** The same product can be sold in different stores (Store_ID), but sales transactions are separate.

6. ***No Discount or Tax Information:*** Since the dataset lacks discount or tax fields, they are not included in the dimensional model.

7. ***Date Granularity is at the Invoice Level****:* The Invoice_Date represents when the invoice was issued, and I assume that's sufficient for historical analysis.

B. Data Processing & ETL Assumptions

8. ***No Null Values in Critical Columns:*** I assume that key columns (Invoice_ID, Customer_ID, Product_ID, etc.) do not have missing values.

9. ***Data is Loaded in Batches:*** The ETL/ELT process will handle data in scheduled batches, rather than real-time streaming.

10. ***Slowly Changing Dimensions (SCD) Type 2 for Customers & Products:*** I assume that customer and product details may change over time, so I keep historical records (e.g., if a product is renamed, I track old and new names).

C. Technical Assumptions

11. ***Star Schema is Sufficient for Query Performance:*** A star schema is chosen over a snowflake schema to optimize query speed.

12. ***Storage & Compute Costs are Manageable:*** I assume that the chosen database can handle the storage and performance needs efficiently.

7.2. Trade-Offs

Trade-offs highlight the pros and cons of design choices and why I picked one option over another.

A. Star Schema vs. Snowflake Schema

Star Schema (Chosen)

- Simpler and faster queries for reporting.
- Easier to understand and maintain.

Snowflake Schema (Not Chosen)

- Saves storage by normalizing dimensions, but increases query complexity.
- Requires more joins, leading to slower performance.

Trade-Off: I prioritized query speed over storage efficiency, making Star Schema the better choice.

B. Handling Slowly Changing Dimensions (SCDs)

Type 2 (Track Historical Changes) - Chosen

- Keeps past records of customer and product changes.
- Enables better historical analysis.

Type 1 (Overwrite Data) – Not chosen

- Simplifies updates but loses historical data.

Trade-Off: I chose SCD Type 2 to retain history at the cost of extra storage.

C. Real-Time ETL vs. Batch Processing

Batch Processing (Chosen)

- More efficient for large data loads.
- Reduces system complexity.

Real-Time Processing (Not Chosen)

- Requires more infrastructure and processing power.

Trade-Off: I sacrificed real-time updates for better performance and simplicity.

D. Date Granularity (Invoice Level)

Invoice Date Granularity (Chosen)

- Simple and meets most reporting needs.

Transaction Timestamp Granularity (Not Chosen)

- More precise but increases storage and complexity.

Trade-Off: I accepted less granular data in exchange for simpler storage and querying.

Conclusion on Trade-Offs and Assumptions

This section ensures that stakeholders understand the assumptions underlying the design and the trade-offs made to balance performance, data integrity, and scalability.

8. FINAL CONCLUSION

This end-to-end dimensional model provides an optimized data structure for business intelligence. It ensures data integrity, scalability, and performance, making it an ideal foundation for analytical processing. It enables businesses to track sales performance, customer behavior, product demand, and store efficiency while supporting historical and drill-down reporting.