Data Warehouse Design

Project Title: Consumer Electronics Retail Data Warehouse
Dataset was synthetic.

1. Project Overview

The goal of this project is to design and implement a **Data Warehouse (DW)** for a consumer electronics retail company operating both online and offline across major U.S. cities. The company seeks to better understand its **sales performance** and **inventory management** by enabling detailed reporting and business intelligence through structured analytical queries.

2. Business Objectives

- Analyze total sales revenue by **year**, **month**, and **quarter**, across **cities**, **product categories**, and **stores**.
- Enable multi-dimensional reporting with high performance.
- Standardize and consolidate data from operational systems for analytical use.
- Allow for historical analysis and trend forecasting.

3. Star Schema Design

The **Star Schema** is chosen for its simplicity, performance, and ability to support ad-hoc queries by business analysts. It consists of one **Fact Table** surrounded by multiple **Dimension Tables**.

Key Features:

- Central **Fact Table**: MyFactSales
- Surrounding **Dimension Tables**: MyDimDate, MyDimProduct, MyDimCustomerSegment, and MyDimCity
- Granularity: One row per sales transaction

4. Tables and Fields

4.1. Fact Table: MyFactSales

Stores all sales transactions.

Field	Description
salesid	Unique ID for the sales record

Field	Description
productid	Foreign key to MyDimProduct
quantitysold	Number of units sold
priceperunit	Unit price of the product
segmentid	Foreign key to MyDimCustomerSegment
dateid	Foreign key to MyDimDate
cityid	Foreign key to MyDimCity

4.2. Dimension Table: MyDimDate

Supports time-based analysis.

Field	Description
dateid	Surrogate key (e.g., YYYYMMDD)
year	Year (e.g., 2024)
month	Month number (1-12)
monthname	Name of the month (e.g., April)
day	Day of the month (1-31)
weekday	Weekday number (1=Sunday, etc.)
weekdayname	Name of the weekday (e.g., Monday)

4.3. Dimension Table: MyDimProduct

Details of products sold.

Field	Description
productid	Unique identifier for the product
productname	Name or type of the product

4.4. Dimension Table: MyDimCustomerSegment

Stores customer segmentation.

Field	Description
segmentid	Unique ID for customer segment
segmentname	Description (e.g., Online, In-Store)

4.5. Dimension Table: MyDimCity

Captures city/store-related info.

Field	Description
cityid	Unique identifier for city
cityname	City name (e.g., New York)
region	Region/State (optional)

5. Creating Tables in PostgreSQL

5.1 Database Creation

A new PostgreSQL database named CER_Database was created to serve as the central data warehouse for this project. Create all the tables by running the cer_db.sql script.

5.2 Loading Data

Load the dataset by running the data_loading.sh. This downloads the data and loads the data into the corresponding tables.

Make sure the script is executable:

chmod +x load_data.sh

Run the script:

bash data_loading.sh

6. Aggregation Queries and Materialized View

This section focuses on advanced SQL analytical functions using aggregation operations such as *GROUPING SETS*, *ROLLUP*, *CUBE*, and *Materialized Views*. These queries provide multi-dimensional analysis on sales data stored in the data warehouse.

6.1 Grouping Sets Query

A **GROUPING SETS** query was used to generate multiple groupings in a single query execution. This allowed aggregation of total sales at various levels, including combinations of Product ID and Product Type, individual columns, and an overall grand total. It facilitated flexible reporting without the need for multiple queries.

6.2 Rollup Query

The **ROLLUP** operation was employed to aggregate sales data across hierarchical dimensions: Year, City, and Product ID. This created subtotal rows for each level and a final grand total, helping in hierarchical reporting such as yearly summaries and city-level comparisons.

6.3 Cube Query

A **CUBE** query was designed to generate every possible combination of Year, City, and Product ID, along with their corresponding average sales values. This provided a full multidimensional breakdown of the sales data, allowing exploration of correlations and trends across all groupings.

6.4 Materialized View

A **Materialized View** named max_sales was created to store the maximum sales per City, Product ID, and Product Type. This precomputed view enhances performance by reducing the computational load during repeated queries. The view can be refreshed using the REFRESH MATERIALIZED VIEW command to keep data up to date.