**Star Schema Data Warehouse Implementation**

# **Objective**

To design and implement a star schema in PostgreSQL that supports analytical queries on a retail sales dataset downloaded from Kaggle. This includes creating normalized dimension tables and a centralized fact table for efficient querying and reporting.

# **Dataset Overview**

**Source File:** retail\_sales\_dataset.csv

Fields:

* - Transaction\_ID: Unique transaction ID
* - Date: Transaction date
* - Customer\_ID: Unique ID of the customer (e.g., CUST001)
* - Gender: Gender of the customer
* - Age: Age of the customer
* - Product\_Category: Category of product sold
* - Quantity: Number of units sold
* - Price\_Per\_Unit: Price of a single unit

# **Schema Design**

**The star schema consists of the following tables:**

1. Fact Table: fact\_sales

* - transaction\_id (Primary Key)
* - date\_key (FK to dim\_date)
* - customer\_key (FK to dim\_customer)
* - product\_key (FK to dim\_product)
* - quantity
* - price\_per\_unit
* - total\_amount

2. Dimension Table: dim\_date

* - date\_key (Primary Key)
* - date
* - day
* - month
* - year
* - quarter

3. Dimension Table: dim\_customer

* - customer\_key (Primary Key)
* - customer\_id
* - gender
* - age

4. Dimension Table: dim\_product

* - product\_key (Primary Key)
* - product\_category

# **Implementation Steps**

**1. Database Creation**

CREATE DATABASE retail\_dw OWNER postgres;

**2. Schema & Table Setup**

CREATE SCHEMA IF NOT EXISTS star;

SET search\_path = star;

**Staging table for raw CSV**

CREATE TABLE IF NOT EXISTS staging\_sales (  
 transaction\_id INTEGER,  
 date DATE,  
 customer\_id VARCHAR(20),  
 gender VARCHAR(10),  
 age INTEGER,  
 product\_category VARCHAR(50),  
 quantity INTEGER,  
 price\_per\_unit NUMERIC(10,2),  
 total\_amount NUMERIC(10,2)  
);

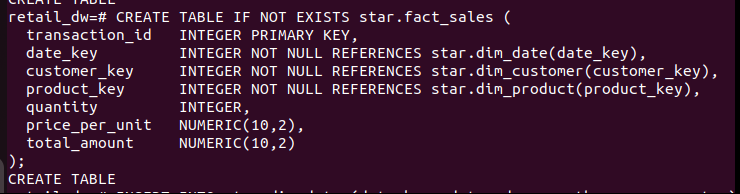
**Dimension Tables**

CREATE TABLE IF NOT EXISTS dim\_date (  
 date\_key INTEGER PRIMARY KEY,  
 date DATE NOT NULL,  
 day SMALLINT NOT NULL,  
 month SMALLINT NOT NULL,  
 year INTEGER NOT NULL,  
 quarter SMALLINT NOT NULL  
);

CREATE TABLE IF NOT EXISTS dim\_customer (  
 customer\_key SERIAL PRIMARY KEY,  
 customer\_id VARCHAR(20) UNIQUE,  
 gender VARCHAR(10),  
 age INTEGER  
);  
  
CREATE TABLE IF NOT EXISTS dim\_product (  
 product\_key SERIAL PRIMARY KEY,  
 product\_category VARCHAR(50) UNIQUE  
);

**Fact Table**

CREATE TABLE IF NOT EXISTS fact\_sales (  
 transaction\_id INTEGER PRIMARY KEY,  
 date\_key INTEGER NOT NULL REFERENCES dim\_date(date\_key),  
 customer\_key INTEGER NOT NULL REFERENCES dim\_customer(customer\_key),  
 product\_key INTEGER NOT NULL REFERENCES dim\_product(product\_key),  
 quantity INTEGER,  
 price\_per\_unit NUMERIC(10,2),  
 total\_amount NUMERIC(10,2)  
);



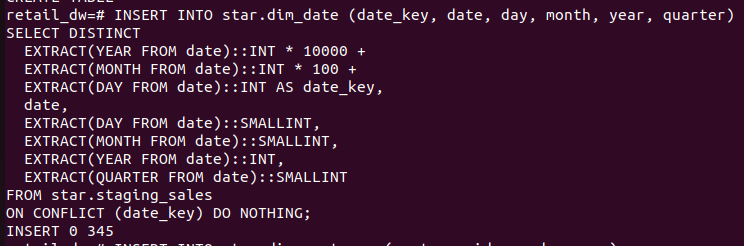
**3. Data Import Using \copy**

\copy star.staging\_sales(transaction\_id, date, customer\_id, gender, age, product\_category, quantity, price\_per\_unit, total\_amount)  
FROM '/home/rishabh/task/star\_schema/retail\_sales\_dataset.csv'  
WITH (FORMAT csv, HEADER true);

## **4. Populating Dimension Tables**

**dim\_date:**

INSERT INTO dim\_date (date\_key, date, day, month, year, quarter)  
SELECT DISTINCT  
 EXTRACT(YEAR FROM date)::INT \* 10000 +  
 EXTRACT(MONTH FROM date)::INT \* 100 +  
 EXTRACT(DAY FROM date)::INT AS date\_key,  
 date,  
 EXTRACT(DAY FROM date)::SMALLINT,  
 EXTRACT(MONTH FROM date)::SMALLINT,  
 EXTRACT(YEAR FROM date)::INT,  
 EXTRACT(QUARTER FROM date)::SMALLINT  
FROM star.staging\_sales  
ON CONFLICT (date\_key) DO NOTHING;

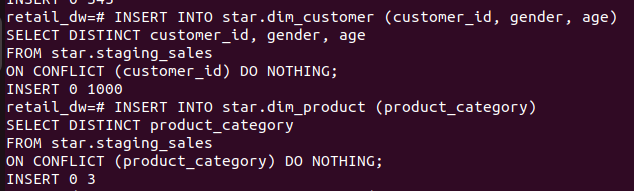


**dim\_customer:**

INSERT INTO dim\_customer (customer\_id, gender, age)  
SELECT DISTINCT customer\_id, gender, age  
FROM star.staging\_sales  
ON CONFLICT (customer\_id) DO NOTHING;

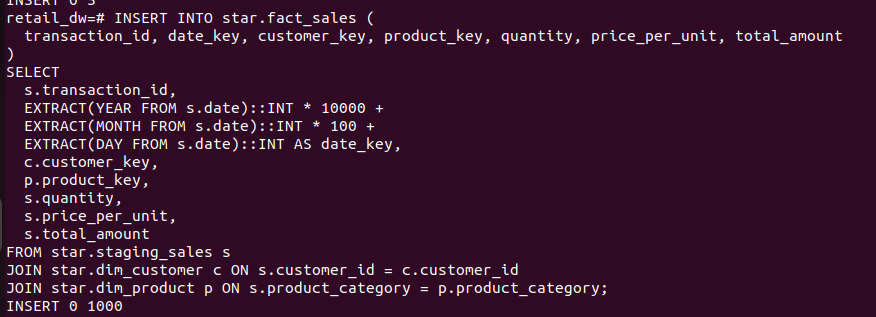
**dim\_product:**

INSERT INTO dim\_product (product\_category)  
SELECT DISTINCT product\_category  
FROM star.staging\_sales  
ON CONFLICT (product\_category) DO NOTHING;



**5. Populating Fact Table**

INSERT INTO fact\_sales (transaction\_id, date\_key, customer\_key, product\_key, quantity, price\_per\_unit, total\_amount)  
SELECT  
 s.transaction\_id,  
 EXTRACT(YEAR FROM s.date)::INT \* 10000 +  
 EXTRACT(MONTH FROM s.date)::INT \* 100 +  
 EXTRACT(DAY FROM s.date)::INT AS date\_key,  
 c.customer\_key,  
 p.product\_key,  
 s.quantity,  
 s.price\_per\_unit,  
 s.total\_amount  
FROM star.staging\_sales s  
JOIN dim\_customer c ON s.customer\_id = c.customer\_id  
JOIN dim\_product p ON s.product\_category = p.product\_category;



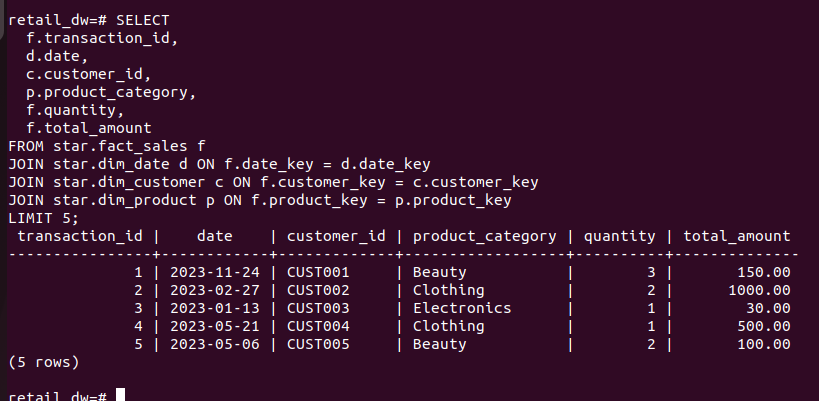
**Validation & Testing**

**Check row counts:**

SELECT   
 (SELECT COUNT(\*) FROM star.dim\_date) AS dim\_date\_rows,  
 (SELECT COUNT(\*) FROM star.dim\_customer) AS dim\_customer\_rows,  
 (SELECT COUNT(\*) FROM star.dim\_product) AS dim\_product\_rows,  
 (SELECT COUNT(\*) FROM star.fact\_sales) AS fact\_sales\_rows;

**Sample join query:**

SELECT   
 f.transaction\_id,   
 d.date,   
 c.customer\_id,   
 p.product\_category,   
 f.quantity,   
 f.total\_amount  
FROM star.fact\_sales f  
JOIN star.dim\_date d ON f.date\_key = d.date\_key  
JOIN star.dim\_customer c ON f.customer\_key = c.customer\_key  
JOIN star.dim\_product p ON f.product\_key = p.product\_key  
LIMIT 5;



**Outcome**

• Star schema successfully implemented with normalized dimensions.  
• Referential integrity established via foreign keys.  
• Dataset prepared for analytical workloads, dashboards, or BI tools.