Data warehousing Project

Prepared by:

Team Name: Digital Pioneers

Beshoy Karam (Team Leader)

Fady Emad

Adel Emad

Ahd Montasser

Shahd Hussein

Dr: Wesam Ahmed

Content:

1. Introduction: The "Business Case"

Problem Statement: Traditionally, sales data in Excel is "flat" and "static." It lacks historical tracking and makes it difficult to see relationships between different entities (like how a specific category performs across different months).

The Goal: To transition from **Operational Data** (Excel) to **Analytical Data** (Data Warehouse).

Define exactly what you are measuring:

Gross Profit: Total Sales - Cost of Goods Sold (COGS).

Sales Volume: Total units (pieces) sold.

Product Penetration: Which products are moving the fastest.

2. The Data Architecture (The "How")

In this part we will describe the ETL process (Extract, Transform, Load)
Data Source Layer: our8 source was Excel, which represents "unstructured" or "semi-structured" raw business data.

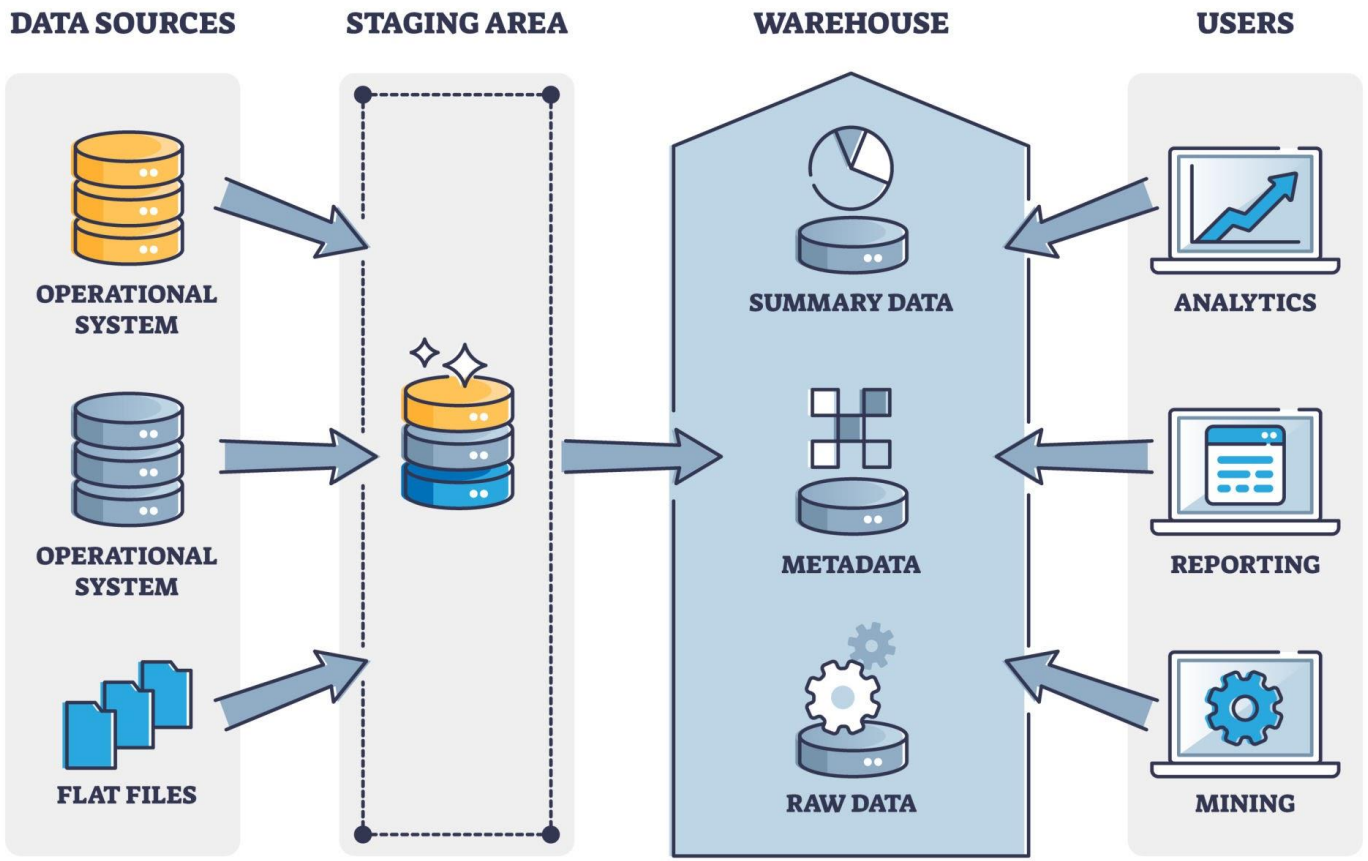
•**Staging Area (The "Buffer"):** This is crucial. Tell them you used a staging area to:

•**Cleanse:** Remove null values or fixing "messy" product names.

•**Standardize:** Ensure all currency and date formats are uniform before they hit the SQL database.

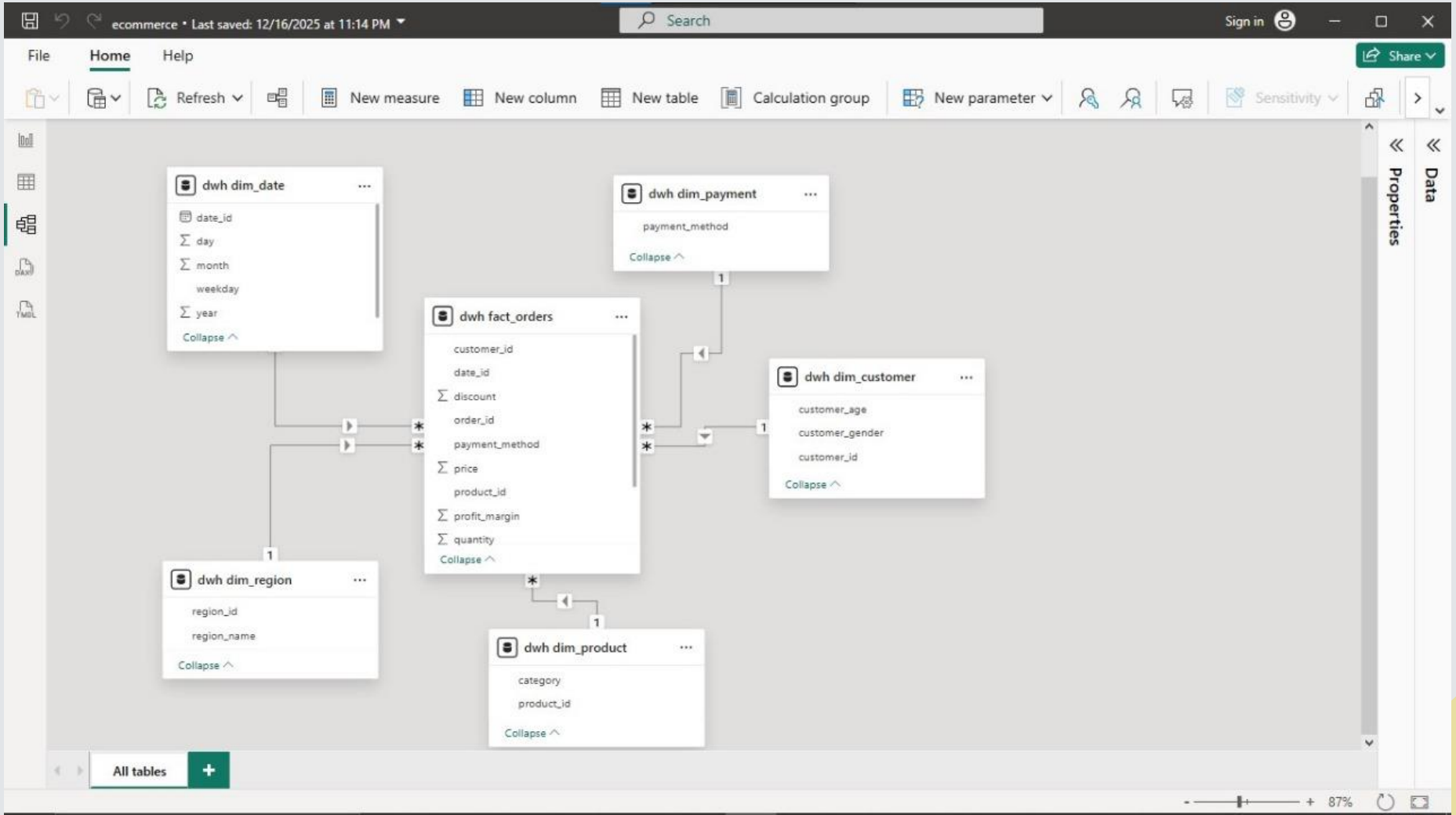
•**Data Warehouse Layer (SQL):** This is the permanent, structured storage where data is optimized for reading and reporting, not just storage.

DATA WAREHOUSE



3. The Data Model (The "Logic")

Our data model follows a **Star Schema** architecture, designed to optimize query performance and simplify the analytical process within Power BI.



A. The Central Fact Table: dwh_fact_orders

This is the core of our warehouse. It captures every business event (an order) and stores the numerical data we need for calculation.

Keys for Integration: It contains Foreign Keys such as customer_id, date_id, product_id, and region_id to connect with our descriptive dimensions.

Core Metrics: We store the fundamental values here: price, quantity, and discount.

Derived Logic: We included a profit_margin field directly in the fact table to allow for immediate profitability analysis without complex runtime calculations.

Granularity: The "grain" of this table is a **single order line item**, ensuring we can aggregate data up to any level (e.g., total sales by region or average profit by customer gender).

B. The Dimension Tables (The Descriptive Entities)

We created five distinct dimension tables to provide context to our order data:

dwh_dim_date: Enables time-series analysis through attributes like day, month, weekday, and year. This allows us to track "Pieces Sold" over specific calendar periods.

dwh_dim_product: Stores product metadata, specifically category. This is essential for identifying which product lines are the most profitable.

dwh_dim_customer: Contains demographic data like customer_age and customer_gender, allowing us to see which audience segments are buying the most pieces.

dwh_dim_region: Organized by region_id and region_name to visualize geographic sales performance.

dwh_dim_payment: Tracks the payment_method used for each order, helping us understand customer purchasing behavior.

C. Relationship Logic & Schema Integrity

Star Schema Efficiency: As seen in our model diagram, the dwh_fact_orders table is at the center, surrounded by dimensions. This reduces the number of "joins" needed, making the Power BI dashboard faster.

1:N (One-to-Many) Relationships: We established 1:N relationships between the dimensions and the fact table. For example, one entry in dwh_dim_product can relate to many rows in dwh_fact_orders.

Directional Filtering: The filters flow from the Dimension tables to the Fact table (represented by the arrows in the diagram). This ensures that when we select a specific "Category" in our dashboard, it correctly filters the "Total Profit" in our Fact table.

4. Analysis & Power BI (The "Insight“)

In this phase, **we transformed the structured data from our SQL Warehouse into a visual "Semantic Layer."** This allows stakeholders to move from viewing rows of data to identifying business trends instantly.

A. Core Metrics.

We utilized high-level "Card" visualizations to provide an immediate snapshot of the business's health:

Total Revenue: indicating a high-volume operation.

Profit Performance: which is our primary indicator of business success beyond just sales.

Inventory Throughput: helping us understand the scale of physical product movement.

B. Visual Analysis & Behavioral Trends

•**Demographic Profitability (Line Chart):** We analyzed the relationship between customer_age and sales. This line chart reveals how different age groups contribute to revenue, allowing us to target marketing toward the peaks seen in the data.

•**Regional Performance (Bar Chart):** By plotting region_name, we discovered that the **South** and **North** regions are leading in sales, while the **Central** region is currently the lowest performer.

•**Gender Distribution (Donut Chart):** Our customer base is almost evenly split, with **Females** slightly outperforming **Males**. This insight is crucial for product procurement and inventory planning.

C. Operational & Payment Analysis

Integrating the dwh_dim_payment and shipping data allowed us to see how customers interact with the business:

Payment Preferences: Credit Cards are the dominant payment method.

Shipping Costs by Region: We tracked the Max of shipping_cost per region.

Discount Impact: Our analysis shows a sharp decline in sales volume as discount levels increase.

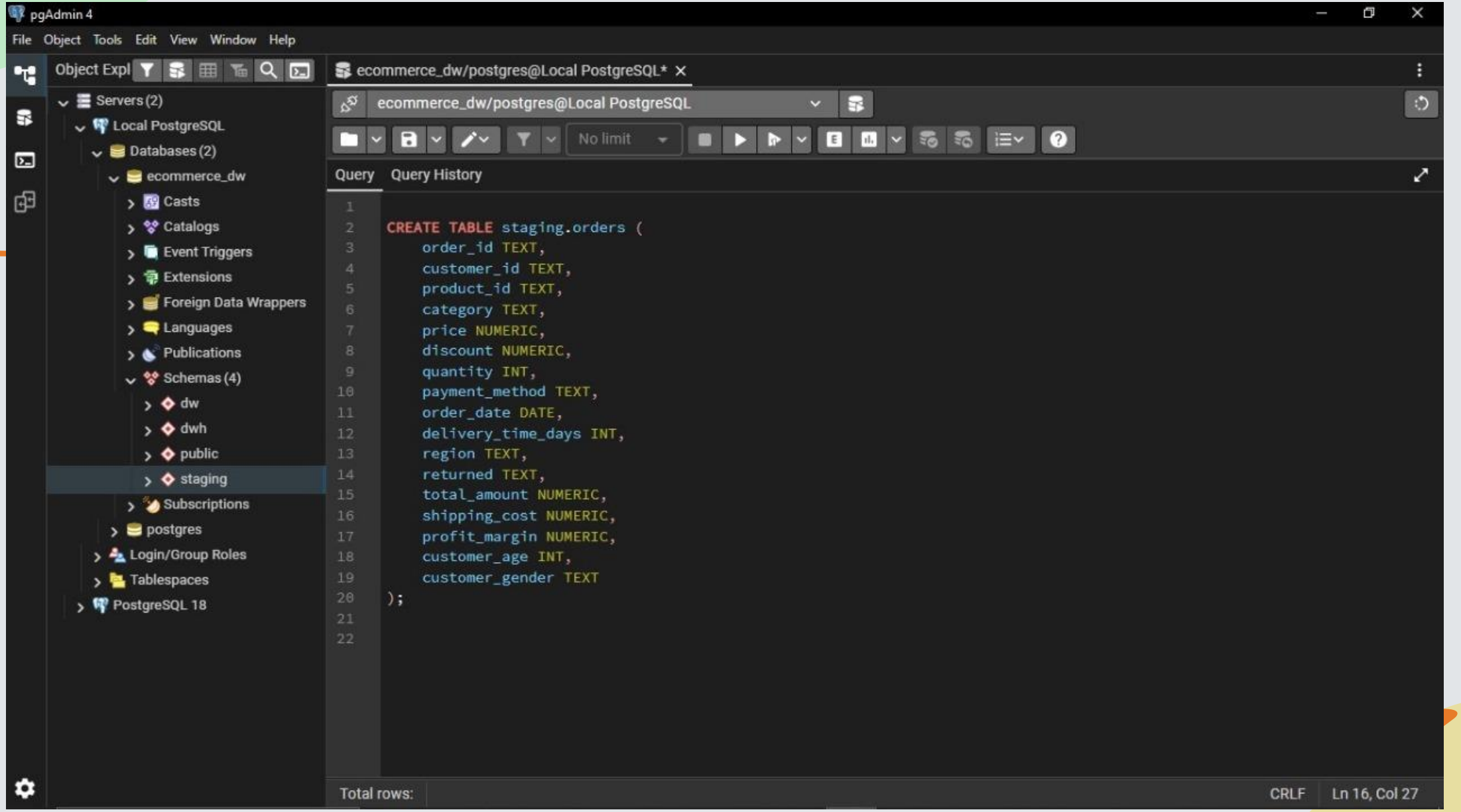
D. Operational Efficiency & Behavior

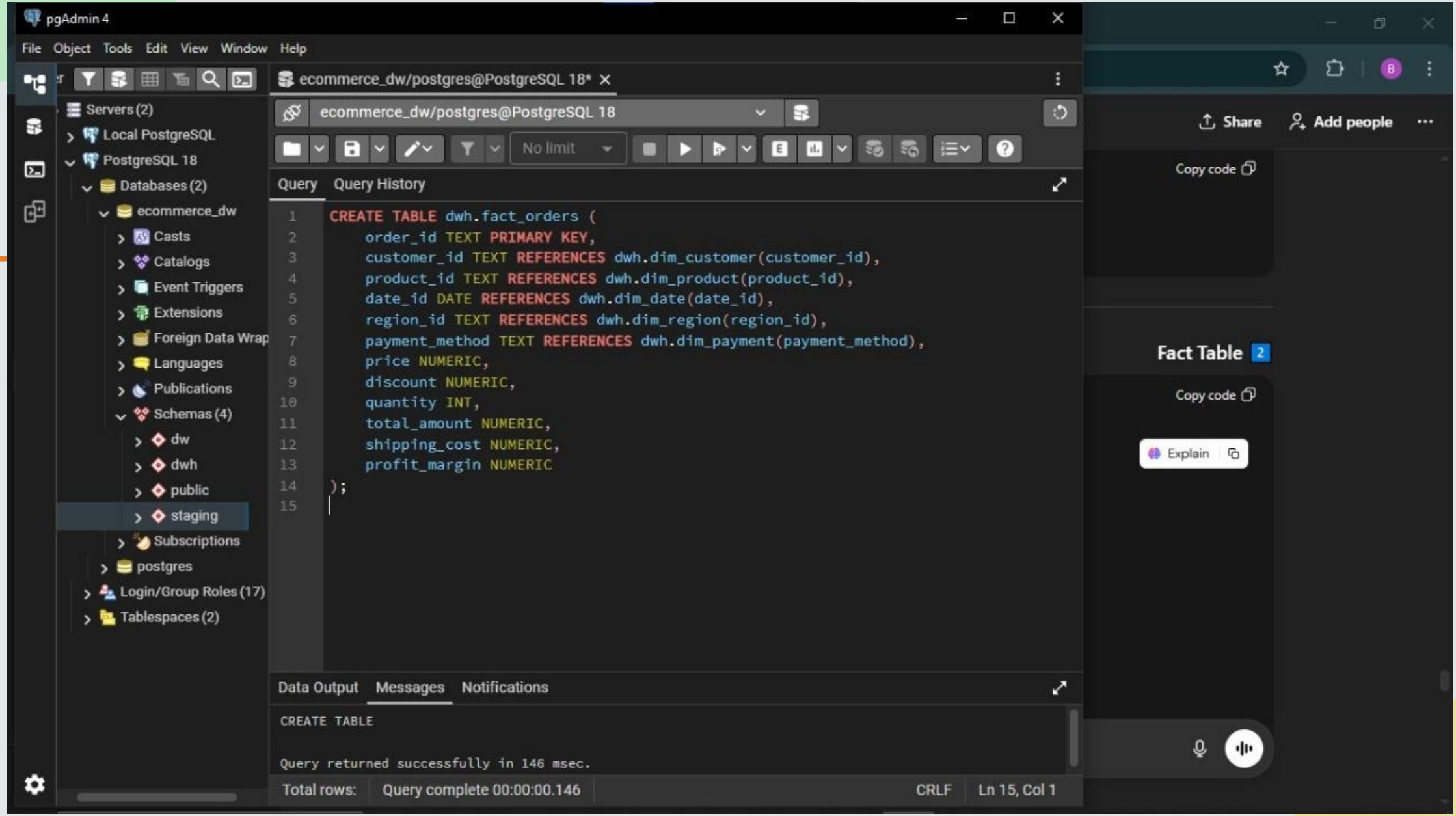
Payment Method Impact: Credit Cards.

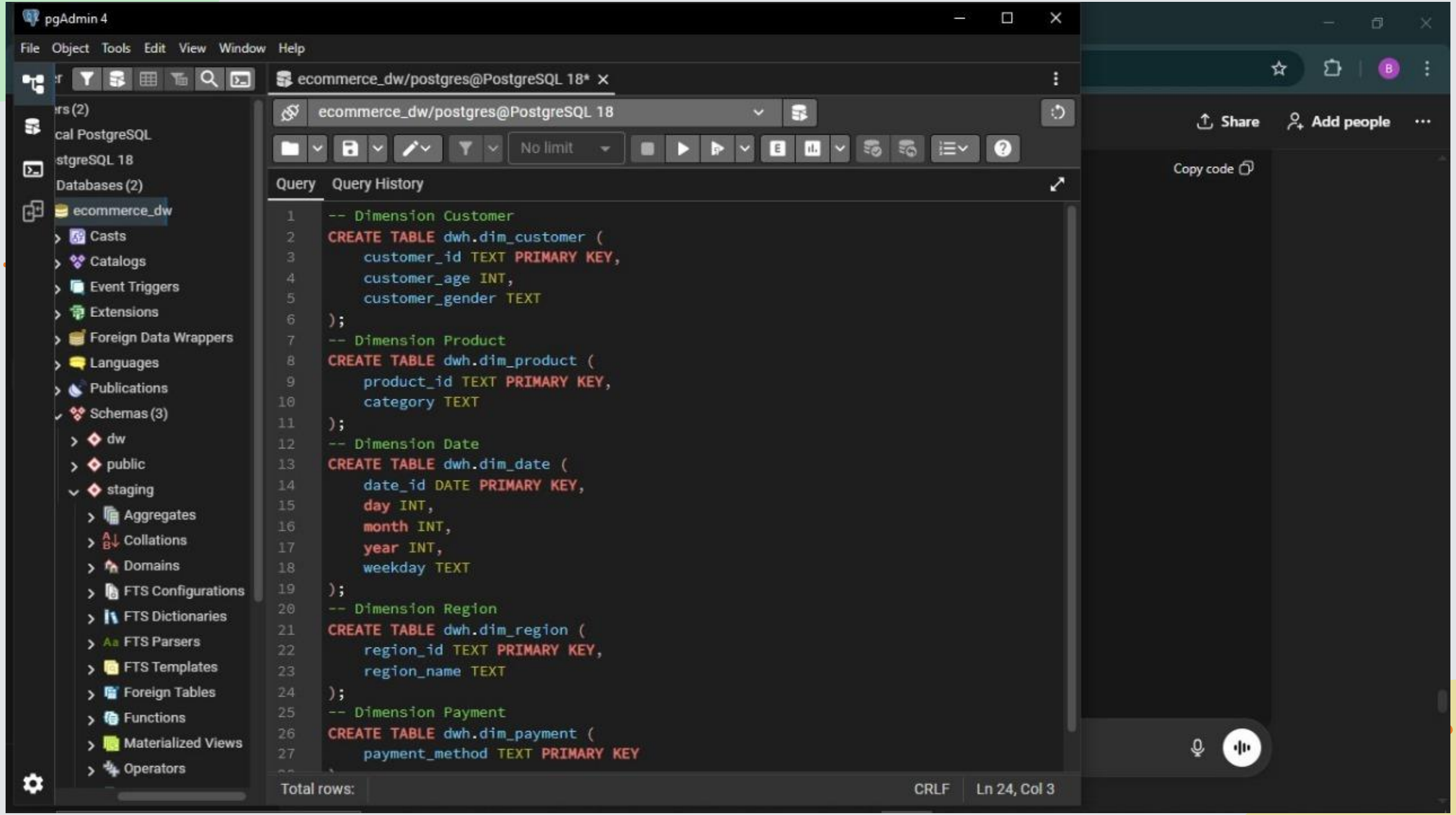
Logistics Overhead: identifying a clear area for cost reduction.

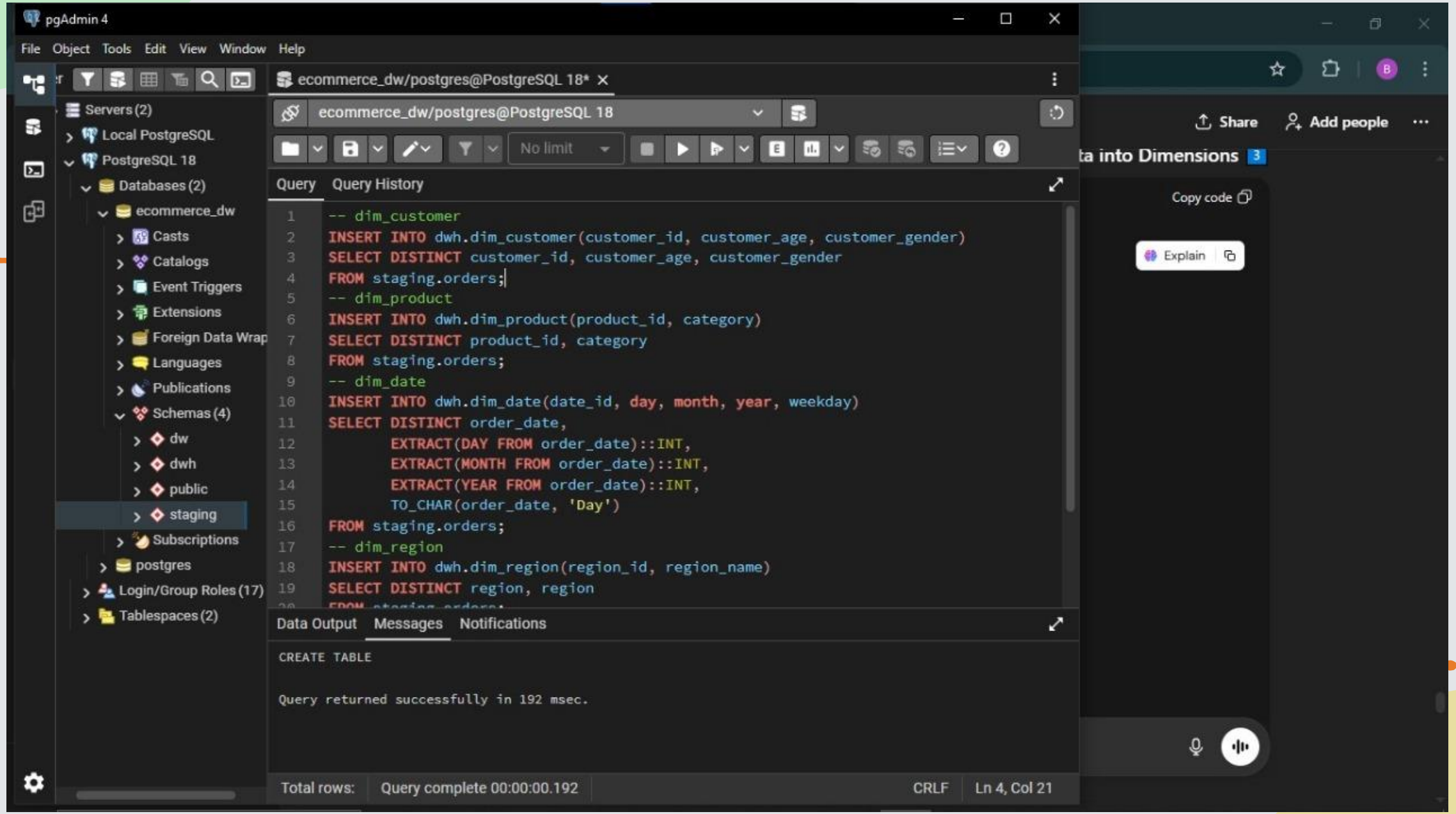
Discount Sensitivity: Our analysis shows that volume peaks at **18.9K** when discounts are at **0%**, suggesting that our customers are currently more driven by product value than price-slashing.

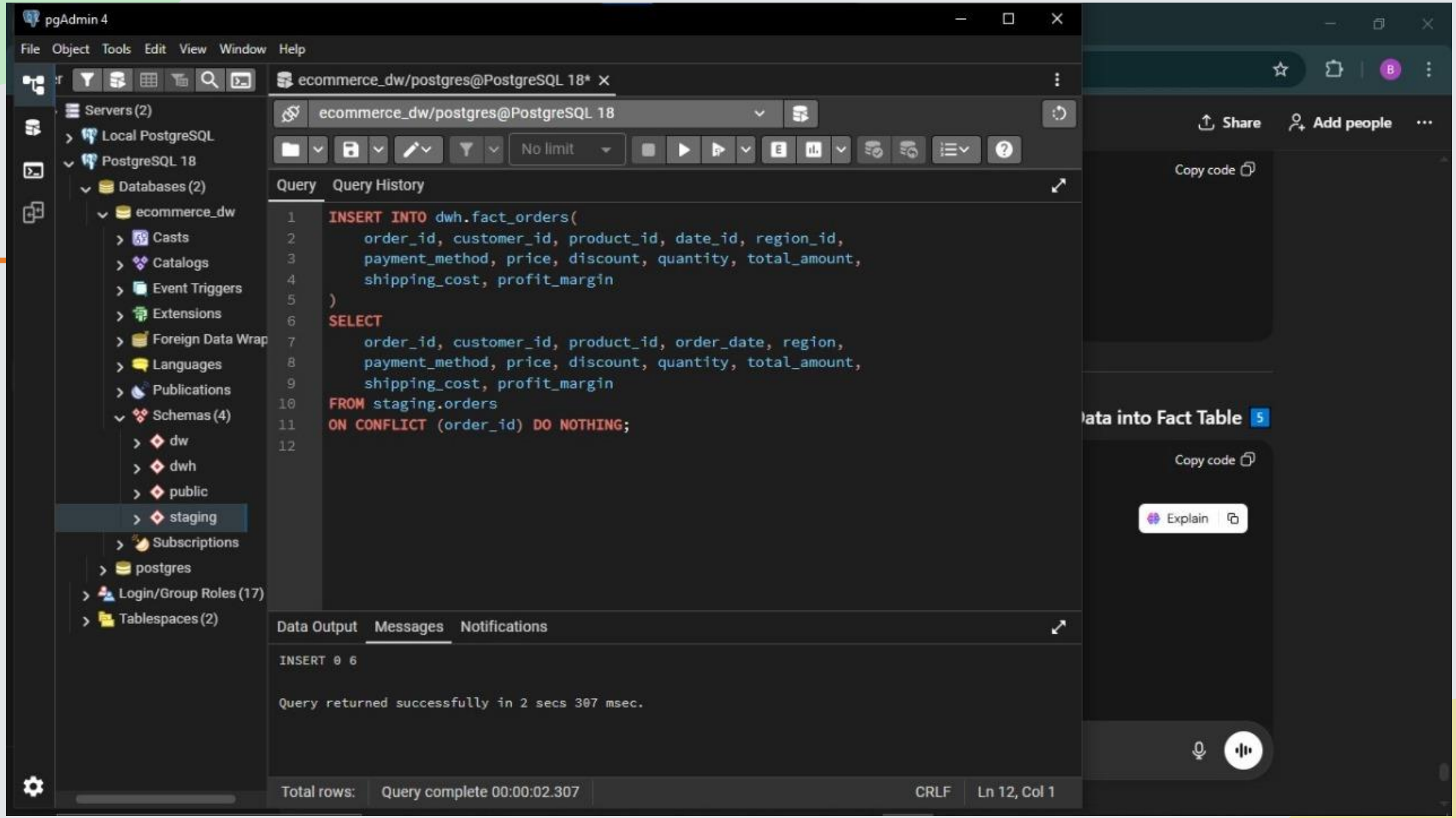
Screens from Our Project.


pgAdmin 4
File Object Tools Edit View Window Help
Object Expl
Servers (2)
Local PostgreSQL
Databases (2)
ecommerce_dw
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (4)
dw
dwh
public
staging
Subscriptions
postgres
Login/Group Roles
Tablespaces
PostgreSQL 18
ecommerce_dw/postgres@Local PostgreSQL*
ecommerce_dw/postgres@Local PostgreSQL
No limit
Query
Query History
CREATE TABLE staging.orders (
order_id TEXT,
customer_id TEXT,
product_id TEXT,
category TEXT,
price NUMERIC,
discount NUMERIC,
quantity INT,
payment_method TEXT,
order_date DATE,
delivery_time_days INT,
region TEXT,
returned TEXT,
total_amount NUMERIC,
shipping_cost NUMERIC,
profit_margin NUMERIC,
customer_age INT,
customer_gender TEXT
);
Total rows:
CRLF
Ln 16, Col 27


pgAdmin 4
File Object Tools Edit View Window Help
ecommerce_dw/postgres@PostgreSQL 18*
ecommerce_dw/postgres@PostgreSQL 18
No limit
Query Query History
CREATE TABLE dwh.fact_orders (
order_id TEXT PRIMARY KEY,
customer_id TEXT REFERENCES dwh.dim_customer(customer_id),
product_id TEXT REFERENCES dwh.dim_product(product_id),
date_id DATE REFERENCES dwh.dim_date(date_id),
region_id TEXT REFERENCES dwh.dim_region(region_id),
payment_method TEXT REFERENCES dwh.dim_payment(payment_method),
price NUMERIC,
discount NUMERIC,
quantity INT,
total_amount NUMERIC,
shipping_cost NUMERIC,
profit_margin NUMERIC
);
Servers (2)
Local PostgreSQL
PostgreSQL 18
Databases (2)
ecommerce_dw
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrap
Languages
Publications
Schemas (4)
dw
dwh
public
staging
Subscriptions
postgres
Login/Group Roles (17)
Tablespaces (2)
Data Output Messages Notifications
CREATE TABLE
Query returned successfully in 146 msec.
Total rows:
Query complete 00:00:00.146
CRLF
Ln 15, Col 1
Share
Add people
Copy code
Fact Table 2
Copy code
Explain


pgAdmin 4
File Object Tools Edit View Window Help
ecommerce_dw/postgres@PostgreSQL 18*
ecommerce_dw/postgres@PostgreSQL 18
No limit
Query Query History
-- Dimension Customer
CREATE TABLE dwh.dim_customer (
customer_id TEXT PRIMARY KEY,
customer_age INT,
customer_gender TEXT
);
-- Dimension Product
CREATE TABLE dwh.dim_product (
product_id TEXT PRIMARY KEY,
category TEXT
);
-- Dimension Date
CREATE TABLE dwh.dim_date (
date_id DATE PRIMARY KEY,
day INT,
month INT,
year INT,
weekday TEXT
);
-- Dimension Region
CREATE TABLE dwh.dim_region (
region_id TEXT PRIMARY KEY,
region_name TEXT
);
-- Dimension Payment
CREATE TABLE dwh.dim_payment (
payment_method TEXT PRIMARY KEY
Total rows:
CRLF
Ln 24, Col 3
Databases (2)
ecommerce_dw
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (3)
dw
public
staging
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Share
Add people
Copy code


pgAdmin 4
File Object Tools Edit View Window Help
ecommerce_dw/postgres@PostgreSQL 18*
Servers (2)
Local PostgreSQL
PostgreSQL 18
Databases (2)
ecommerce_dw
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrap
Languages
Publications
Schemas (4)
dw
dwh
public
staging
Subscriptions
postgres
Login/Group Roles (17)
Tablespaces (2)
ecommerce_dw/postgres@PostgreSQL 18
No limit
Query
Query History
-- dim_customer
INSERT INTO dwh.dim_customer(customer_id, customer_age, customer_gender)
SELECT DISTINCT customer_id, customer_age, customer_gender
FROM staging.orders;
-- dim_product
INSERT INTO dwh.dim_product(product_id, category)
SELECT DISTINCT product_id, category
FROM staging.orders;
-- dim_date
INSERT INTO dwh.dim_date(date_id, day, month, year, weekday)
SELECT DISTINCT order_date,
EXTRACT(DAY FROM order_date)::INT,
EXTRACT(MONTH FROM order_date)::INT,
EXTRACT(YEAR FROM order_date)::INT,
TO_CHAR(order_date, 'Day')
FROM staging.orders;
-- dim_region
INSERT INTO dwh.dim_region(region_id, region_name)
SELECT DISTINCT region, region
Data Output
Messages
Notifications
CREATE TABLE
Query returned successfully in 192 msec.
Total rows:
Query complete 00:00:00.192
CRLF
Ln 4, Col 21
Share
Add people
ta into Dimensions 3
Copy code
Explain

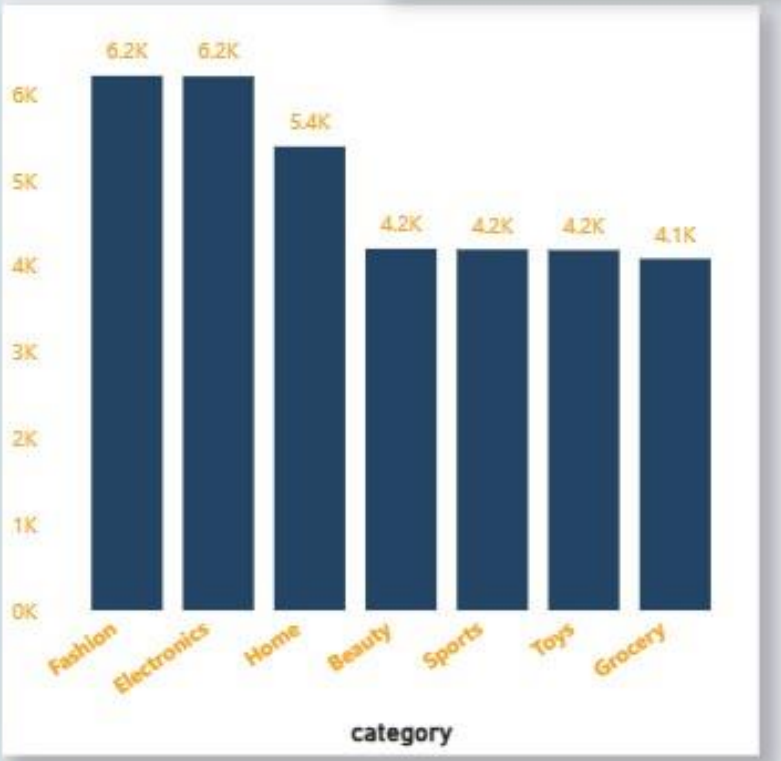

pgAdmin 4
File Object Tools Edit View Window Help
ecommerce_dw/postgres@PostgreSQL 18*
ecommerce_dw/postgres@PostgreSQL 18
No limit
Query Query History
INSERT INTO dwh.fact_orders(
order_id, customer_id, product_id, date_id, region_id,
payment_method, price, discount, quantity, total_amount,
shipping_cost, profit_margin
)
SELECT
order_id, customer_id, product_id, order_date, region,
payment_method, price, discount, quantity, total_amount,
shipping_cost, profit_margin
FROM staging.orders
ON CONFLICT (order_id) DO NOTHING;
Data Output Messages Notifications
INSERT 0 6
Query returned successfully in 2 secs 307 msec.
Total rows:
Query complete 00:00:02.307
CRLF
Ln 12, Col 1
Servers (2)
Local PostgreSQL
PostgreSQL 18
Databases (2)
ecommerce_dw
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrap
Languages
Publications
Schemas (4)
dw
dwh
public
staging
Subscriptions
postgres
Login/Group Roles (17)
Tablespaces (2)
Share
Add people
Copy code
ata into Fact Table 5
Copy code
Explain

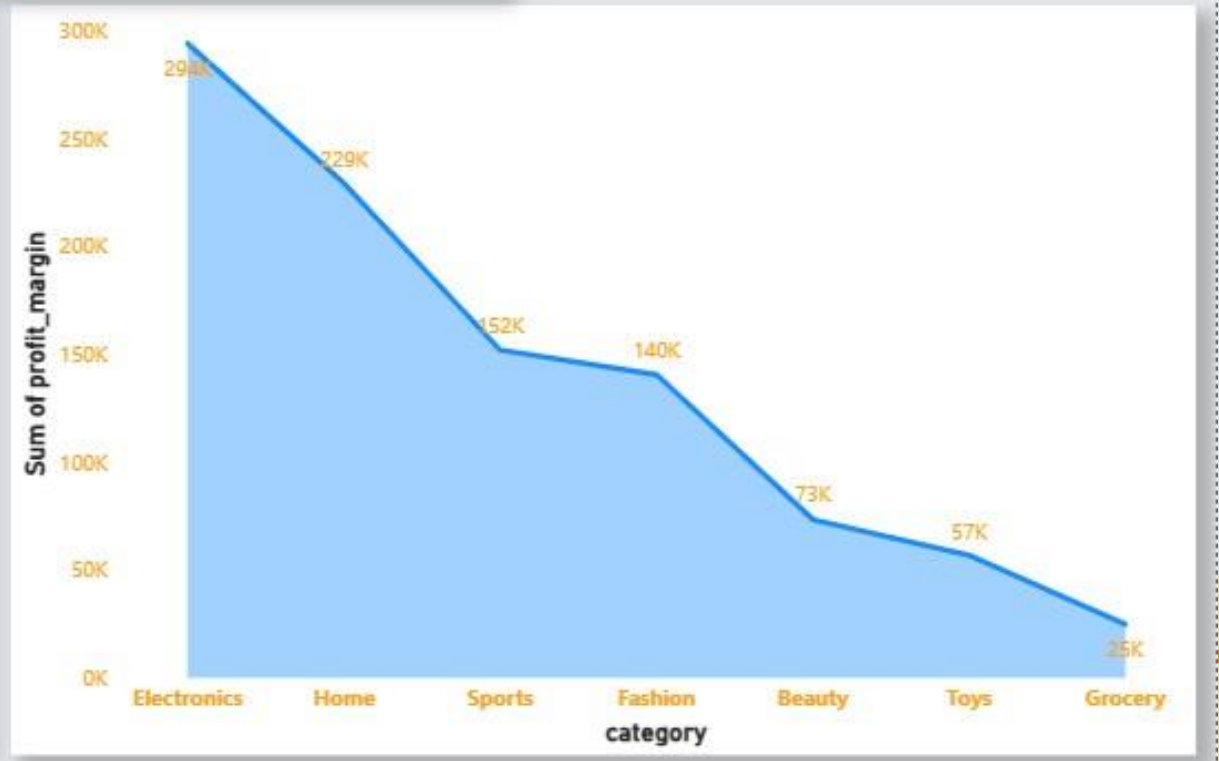
5.87M
Sum of total_amount

970K
Sum of profit_margin

51K
Sum of quantity

category
All






5.87M
Sum of total_amount
970K
Sum of profit_margin
51K
Sum of quantity
800
700
600
500
764
742
747
662
616
564
558
573
536
592
20
30
40
50
60
70
customer_age
region_name
Select all
North
Central
South
East
West
region_name
South
North
East
West
Central
7.6K
7.6K
6.9K
6.8K
5.6K
0K
2K
4K
6K
8K
Female
Male
Other
1.4K (4.07%)
17.09K
(49.54%)
16.01K (46.39%)


5.87M
Sum of total_amount
970K
Sum of profit_margin
51K
Sum of quantity
payment_method
Select all
Credit Card
PayPal
Wallet
COD
Debit Card
UPI
region_name
West
East
South
North
Central
15.7
14.7
14.5
14.4
14.0
0
5
10
15
Max of shipping_cost
12.2K
10K
8.5K
5K
4.2K
4.2K
3.4K
2.1K
0K
Credit Card
Debit Card
COD
UPI
PayPal
Wallet
payment_method
20K
18.9K
15K
10K
6.2K
5K
4.2K
2.8K
1.7K
0K
0.00
0.05
0.10
0.15
0.20
0.25
0.30
discount

6. Conclusion & Value.

This project successfully bridged the gap between raw data collection and strategic decision-making.

From Data to Strategy: We can now see that while we sell many "pieces" in Fashion, our true "profit" engine is the Electronics category.

Scalability: Because we built this in a **SQL Star Schema** rather than a flat Excel file, this warehouse can handle future growth without needing to redesign the logic.

Final Impact: We have created a transparent, automated system that gives the business a **Single Version of the Truth** for every piece sold and every dollar earned.

Bye-Bye