IE 6750 – Data Warehousing & Integration

Group – 3

E-commerce Buyer Behaviour Analytics
Project Report

Members:

Param Madan, Vinyas Naidu Karri

Table of Contents

1. Executive Summary

In modern e-commerce, understanding buyer behavior is no longer optional it is a competitive necessity. Businesses require more than isolated reports on orders or revenue; they need a unified, analytics-ready view of **who their customers are**, how they interact with products, how reliably they pay, and how their experiences influence reviews and future purchases. However, operational databases store this information in separate, transactional systems, making it difficult to derive meaningful insights.

This project builds a fully integrated **E-Commerce Buyer Behavior Analytics Data Warehouse** designed to overcome these limitations. Using **Talend Open Studio** for ETL, **PostgreSQL** as the warehouse platform, and **Tableau** for visualization, the system consolidates customer demographics, product details, orders, payments, and review activity into a coherent analytical model.

A complete **dimensional star schema** was implemented, featuring conformed dimensions and three analytic fact tables: **Fact_Sales**, **Fact_Review**, and **Fact_Payment**. A Type-2 Slowly Changing Dimension (SCD) was built for the Customer dimension to track historical attribute changes, while incremental loading strategies were applied to Product and other dimensions. The ETL pipelines perform key transformations including surrogate key management, lookup resolution, calculated measures (such as payment processing fees), and historical version tracking.

The resulting data warehouse forms a unified analytical foundation supporting high-value business questions, such as:

- **Customer Insights:** Understanding purchase frequency, geographic patterns, loyalty behavior, and review quality.
- **Product Insights:** Identifying top-performing products and categories through sales and rating analysis.
- **Payment Behavior:** Analyzing payment success rates, preferred payment methods, and fee impacts.
- **Operational Trends:** Monitoring daily sales, order volumes, and review activity to support real-time decision-making.

Finally, interactive **Tableau dashboards** visualize trends across sales, customer behavior, product performance, and payment outcomes, enabling decision-makers to explore insights through intuitive charts, KPIs, and drill-downs.

Overall, the project delivers a scalable, analytics-driven data warehouse that transforms raw transactional data into a strategic asset for understanding and improving buyer behavior across the e-commerce lifecycle.

2. Business Problem & Project Definition

2.1 Business Problem

E-commerce platforms generate vast amounts of transactional data orders, payments, customer interactions, product engagement, and reviews. However, these datasets typically live in **disconnected operational systems**, each optimized for fast transactions rather than analytics. As a result, businesses struggle to answer fundamental strategic questions:

- *Who are our most valuable customers and how does their behavior evolve over time?*
- *Which products drive revenue, positive reviews, and repeat purchases?*
- *What payment methods perform reliably, and where do failures or fees impact profit margins?*
- *How do customer experience signals (ratings, verified purchases, review timing) correlate with sales and retention?*

Because operational databases are not built for OLAP analysis, companies end up with:

- fragmented reporting across teams,
- inconsistent KPIs,
- no historical tracking of customer attribute changes,
- difficulty identifying trends, anomalies, or customer behavioral segments.

In short, raw transactional systems provide *data*, but not *intelligence*.
What the business truly needs is a **unified analytical foundation** that organizes all buyer-related activity in a consistent, scalable, decision-ready model.

2.2 Project Definition

This project addresses the analytics gap by designing and implementing a complete **E-Commerce Buyer Behavior Analytics Data Warehouse**. The objective is to integrate disparate datasets — customers, products, orders, payments, and reviews — into a dimensional warehouse capable of supporting high-impact business analysis through dashboards and KPIs.

The project involves:

1. Building a Dimensional Star Schema

A robust warehouse model was created, consisting of:

- **Dimension Tables:**
  Customer (SCD Type-2), Product, Payment Method, Payment Status, Order Status, Shipping Provider, Date
- **Fact Tables:**
  Fact_Sales, Fact_Payment, and Fact_Review

These tables establish a unified analytical view of buyer behavior, linking customer identities, product attributes, order flows, and payment outcomes.

2. Implementing ETL Pipelines using Talend Open Studio

Talend jobs were developed to:

- extract data from the operational e-commerce source database,
- cleanse and standardize attributes,
- generate surrogate keys,
- apply Slowly Changing Dimension logic for Customer attributes,
- load fact tables with foreign key lookups, calculated measures, and incremental changes.

This ensures the warehouse stays synchronized with evolving business data.

3. Enabling Historical & Behavioral Analysis

The system supports both current and historical analysis:

- Customer SCD allows tracking changes in demographics, location, or segments.
- Fact tables capture the granular behavior of purchasing, reviewing, and paying.
- The Date dimension enables time-series insights across daily, monthly, and seasonal trends.

4. Preparing Analytical Outputs for Dashboards

The final warehouse supports Tableau dashboards that highlight:

- customer loyalty and churn indicators,
- top-performing products and categories,
- payment success rates and fee impacts,
- review sentiment and rating distributions,
- sales trends across regions, demographics, and time periods.

2.3 Project Value

By consolidating all buyer-related data into a single analytical environment, the warehouse empowers business teams to:

- make evidence-based product, marketing, and pricing decisions,
- design targeted customer retention and engagement strategies,
- reduce payment failures and optimize method-level costs,
- strengthen marketplace trust through consistent review insights,
- monitor revenue drivers with clarity and precision.

This transformation turns raw transaction logs into strategic intelligence — enabling organizations to understand *not just what customers do, but why they behave the way they do*.

3. Data Sources

The e-commerce data warehouse was built using multiple structured data sources originating from the operational PostgreSQL system (ecommerce_warehouse_source). These datasets capture different aspects of customer activity — purchases, payments, reviews, and product interactions. Each dataset plays a distinct role in forming a unified analytical environment.

The project integrates **two core source systems**:

3.1 Transactional Source Database: ecommerce_warehouse_source

This is the primary OLTP system containing production-level e-commerce data. The following tables were used as inputs to the ETL pipeline:

1. customers

Contains demographic and behavioral attributes about each registered customer.

This source powers the **Customer Dimension (SCD Type-2)**, enabling historical tracking of customer attribute changes.

2. products

Stores product catalog information and category metadata.

Used to populate the **Product Dimension**, with no SCD applied.

3. orders

Represents high-level order transactions placed by customers.

Used in the **Fact_Sales** table after joining with order items.

4. order_items

Contains item-level details for each order, enabling granular sales analytics.

Mapped into **Fact_Sales** to calculate revenue, discount impact, and product-level performance.

5. payments

Holds payment transaction-level data.

This table feeds into **Fact_Payment**, where additional transformations compute **processing fees** based on the payment method dimension.

6. reviews

Captures customer-generated feedback for products.

Loaded into **Fact_Review**, enabling sentiment and experience-based buyer behavior analytics.

7. Lookup Reference Tables

These tables enrich the fact tables with standardized business codes:

• **order_status -** Used to populate **dim_order_status**.

• **payment_method -** Includes processing_fee_percentage, used to calculate **processing_fee** in Fact_Payment.

• **payment_status -** Used in **dim_payment_status**.

• **shipping_provider -** Used in **dim_shipping_provider**.

3.2 Derived Source: Time Dimension

In addition to operational tables, a **custom Date Dimension (dim_date) was generated entirely within the ETL pipeline**.

Using Talend's row-generation mechanism, the project created a calendar table with attributes such as:

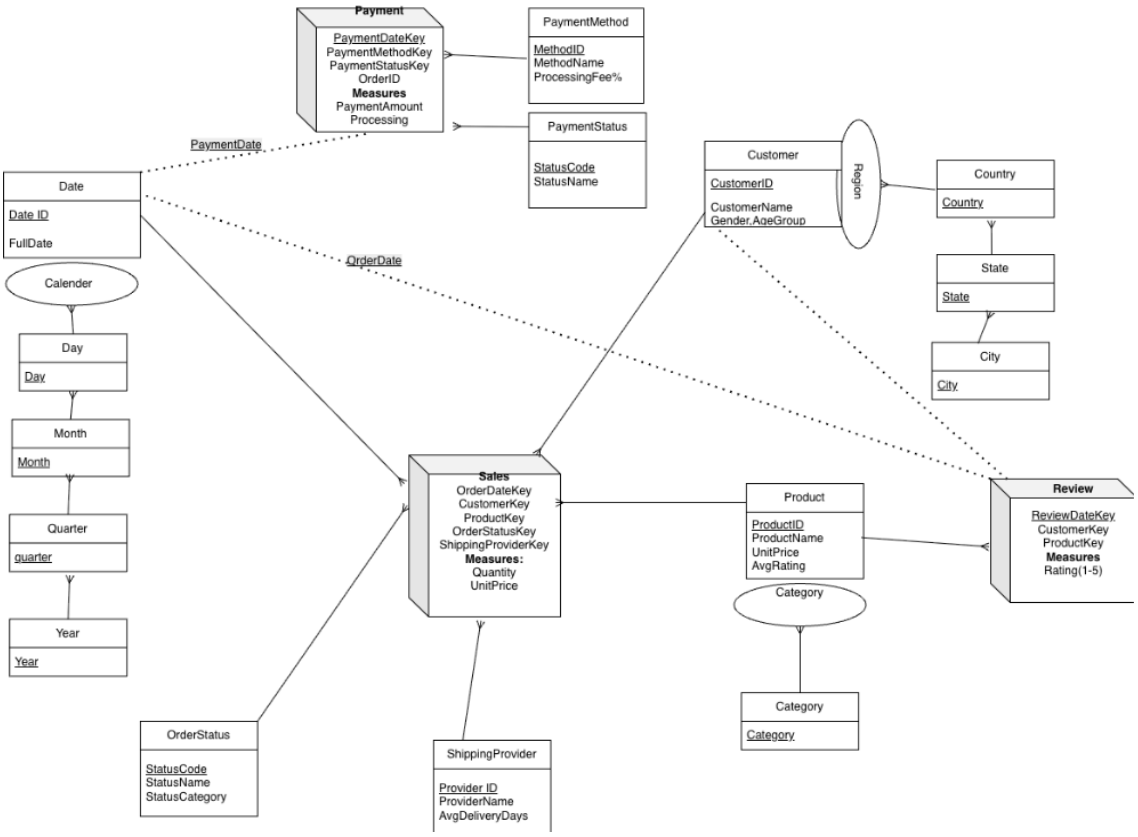
- date_key
- full_date

- day, month, year
- quarter
- day_of_week, is_weekend
- is_holiday (if applicable)

This dimension supports time-series analysis for all fact tables.

4. Conceptual Data Warehouse Model

The conceptual model follows a **star schema** architecture with three fact tables at the center surrounded by seven dimension tables. This design optimizes the data warehouse for OLAP queries and multidimensional analysis.



The diagram illustrates:

Fact Tables (3):

- **Sales** - Central fact capturing order line items with 5 dimension connections
- **Payment** - Financial transaction fact with 3 dimension connections
- **Review** - Customer feedback fact with 3 dimension connections

Dimension Tables (7):

- **Date** - Temporal dimension with 4-level hierarchy (Year → Quarter → Month → Day)
- **Customer** - Demographic dimension with 4-level geographic hierarchy (Country → State → City → Customer)
- **Product** - Catalog dimension with 2-level hierarchy (Category → Product)
- **OrderStatus, ShippingProvider, PaymentMethod, PaymentStatus** - Flat dimensions for filtering and grouping

Key Design Features:

- **Hierarchies** shown as ovals/cylinders stacked vertically beside dimension tables
- **Role-playing dimension:** Date dimension serves multiple roles (OrderDate, PaymentDate, ReviewDate) indicated by dotted lines
- **Measures** clearly separated within fact tables (Quantity, UnitPrice, LineTotal, Rating, PaymentAmount)
- *1: Cardinalities** on all relationships from dimensions to facts

This conceptual schema enables comprehensive business analysis including temporal trends, geographic patterns, product performance, customer segmentation, and payment analytics.

5. Logical Data Warehouse Model

5.1 Schema Overview

The logical model implements a **star schema** with **7 dimension tables** and **3 fact tables**, using surrogate keys (SK) for all primary keys while preserving natural keys (NK) for traceability.

5.2 Dimension Tables

Dimension	Purpose	Key Attributes	Hierarchy
Dim_Date	Temporal analysis	full_date, day, month, quarter, year, day_of_week, is_weekend, is_holiday	Year → Quarter → Month → Day
Dim_Customer	Customer segmentation	customer_id (NK), customer_name, gender, age_group, city, state, country, customer_segment, registration_year	Country → State → City → Customer
Dim_Product	Product catalog	product_id (NK), product_name, category_name, current_price, stock_status, avg_rating, product_status	Category → Product

Dimension	Purpose	Key Attributes	Hierarchy
Dim_OrderStatus	Order lifecycle	status_code (NK), status_name, status_category	Flat
Dim_ShippingProvider	Logistics tracking	provider_id (NK), provider_name, avg_delivery_days, reliability_score	Flat
Dim_PaymentMethod	Payment analysis	payment_method_id (NK), method_name, payment_type, processing_fee_percentage	Flat
Dim_PaymentStatus	Payment outcomes	status_code (NK), status_name, is_successful	Flat

5.3 Fact Tables

Fact_Sales

- **Grain:** One row per order line item
- **Dimensions:** Dim_Date, Dim_Customer, Dim_Product, Dim_OrderStatus, Dim_ShippingProvider
- **Measures:** quantity, unit_price, line_total, discount_applied
- **Degenerate Dimension:** order_id

Fact_Payment

- **Grain:** One row per payment transaction
- **Dimensions:** Dim_Date, Dim_PaymentMethod, Dim_PaymentStatus
- **Measures:** payment_amount, processing_fee
- **Bridge:** order_id (links to Fact_Sales)

Fact_Review

- **Grain:** One row per product review
- **Dimensions:** Dim_Date, Dim_Customer, Dim_Product
- **Measures:** rating (1-5 stars)
- **Bridge:** order_id (links to Fact_Sales)

5.4 Key Transformations

Source	Target	Transformation
customers.first_name + last_name	Dim_Customer.customer_name	Concatenation
Order count per customer	Dim_Customer.customer_segment	VIP (10+), Loyal (5-9), Regular (2-4), One-time (1)

Source	Target	Transformation
AVG(reviews.rating)	Dim_Product.avg_rating	Aggregation
orders.order_date	Dim_Date attributes	Date component extraction
payment_amount × fee_percentage	Fact_Payment.processing_fee	Calculation

5.5 Model Summary

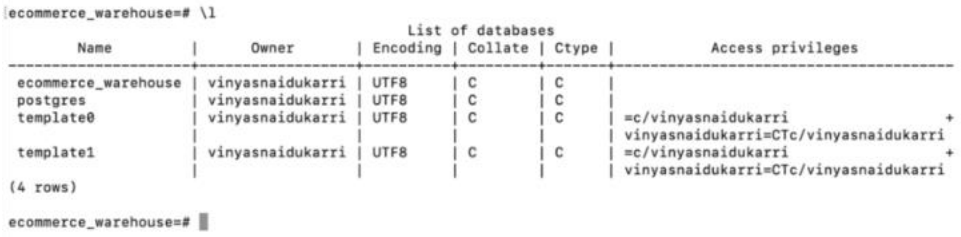
Metric	Count
Dimension Tables	7 (3 hierarchical, 4 flat)
Fact Tables	3
Total Measures	7
Total Dimension References	11

6. Database Implementation (Source DB)

6.1 Schema Creation

The e-commerce data warehouse was implemented in PostgreSQL version 14.19 on macOS. The database ecommerce_warehouse was created to house all operational tables for the e-commerce system.

Screenshot 1: Database Creation



```
ecommerce_warehouse=# \l
                                  List of databases
        Name         |      Owner       | Encoding | Collate | Ctype |           Access privileges
---------------------+------------------+----------+---------+-------+---------------------------------------
 ecommerce_warehouse | vinyasnaidukarri | UTF8     | C       | C     |
 postgres            | vinyasnaidukarri | UTF8     | C       | C     |
 template0           | vinyasnaidukarri | UTF8     | C       | C     | =c/vinyasnaidukarri                  +
                     |                  |          |         |       | vinyasnaidukarri=CTc/vinyasnaidukarri
 template1           | vinyasnaidukarri | UTF8     | C       | C     | =c/vinyasnaidukarri                  +
                     |                  |          |         |       | vinyasnaidukarri=CTc/vinyasnaidukarri
(4 rows)

ecommerce_warehouse=#
```

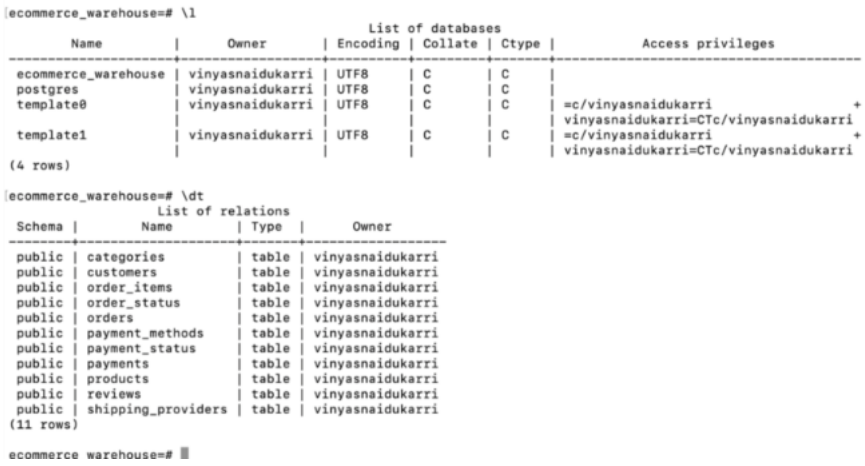
The database was successfully created with UTF8 encoding, ensuring proper handling of international characters and special symbols commonly found in e-commerce data.

Table Creation Strategy

Tables were created in a specific order to respect foreign key dependencies:

1. **Independent tables** (no foreign keys): customers, categories, order_status, shipping_providers, payment_methods, payment_status
2. **Dependent tables** (with foreign keys): products, orders, order_items, payments, reviews

Screenshot 2: All Tables Created



```
ecommerce_warehouse=# \l
                                                List of databases
        Name         |      Owner       | Encoding | Collate | Ctype |           Access privileges
---------------------+------------------+----------+---------+-------+---------------------------------------
 ecommerce_warehouse | vinyasnaidukarri | UTF8     | C       | C     |
 postgres            | vinyasnaidukarri | UTF8     | C       | C     |
 template0           | vinyasnaidukarri | UTF8     | C       | C     | =c/vinyasnaidukarri                  +
                     |                  |          |         |       | vinyasnaidukarri=CTc/vinyasnaidukarri
 template1           | vinyasnaidukarri | UTF8     | C       | C     | =c/vinyasnaidukarri                  +
                     |                  |          |         |       | vinyasnaidukarri=CTc/vinyasnaidukarri
(4 rows)

ecommerce_warehouse=# \dt
                List of relations
 Schema |        Name         | Type  |      Owner
--------+---------------------+-------+------------------
 public | categories          | table | vinyasnaidukarri
 public | customers           | table | vinyasnaidukarri
 public | order_items         | table | vinyasnaidukarri
 public | order_status        | table | vinyasnaidukarri
 public | orders              | table | vinyasnaidukarri
 public | payment_methods     | table | vinyasnaidukarri
 public | payment_status      | table | vinyasnaidukarri
 public | payments            | table | vinyasnaidukarri
 public | products            | table | vinyasnaidukarri
 public | reviews             | table | vinyasnaidukarri
 public | shipping_providers  | table | vinyasnaidukarri
(11 rows)

ecommerce_warehouse=#
```

Primary Keys and Foreign Keys

Each table was created with appropriate primary key constraints to ensure entity integrity. Foreign key constraints were implemented to maintain referential integrity across related tables.

6.2 Data Population

The database was populated with realistic synthetic data using Python with the Faker library to generate USA-specific information.

Data Volume

The following data volumes were achieved:

Screenshot 6: Row Counts

```
ecommerce_warehouse=# SELECT
    'customers' AS table_name, COUNT(*) as count FROM customers
UNION ALL
SELECT 'products', COUNT(*) FROM products
UNION ALL
SELECT 'categories', COUNT(*) FROM categories
UNION ALL
SELECT 'orders', COUNT(*) FROM orders
UNION ALL
SELECT 'order_items', COUNT(*) FROM order_items
UNION ALL
SELECT 'payments', COUNT(*) FROM payments
UNION ALL
SELECT 'reviews', COUNT(*) FROM reviews
UNION ALL
SELECT 'shipping_providers', COUNT(*) FROM shipping_providers
ORDER BY count DESC;
     table_name     | count
--------------------+-------
 order_items        | 26918
 orders             | 15000
 payments           | 14556
 customers          | 10000
 reviews            |  5508
 products           |   500
 categories         |    10
 shipping_providers |     5
(8 rows)

ecommerce_warehouse=#
```

Summary of populated data:

- **10,000 customers** across 15 major US cities
- **500 products** across 10 categories
- **15,000 orders** spanning 12 months
- **26,918 order items** (average 1.8 items per order)
- **14,656 payments** (some cancelled orders have no payment)
- **5,508 reviews** (approximately 55% of delivered orders)

USA-Specific Data

Customer data was generated with realistic USA demographics including major cities such as New York, Los Angeles, Chicago, Houston, and others.

Screenshot 7: Sample Customer Data

```
ecommerce_warehouse=# SELECT customer_id, first_name, last_name, email, city, state
FROM customers
LIMIT 8;
 customer_id | first_name | last_name |            email             |   city   | state
-------------+------------+-----------+------------------------------+----------+-------
           1 | Katie      | Osborne   | toddchristensen@example.org  | New York | NY
           2 | Rebecca    | Lawrence  | apark@example.com            | Dallas   | TX
           3 | Gary       | Bryant    | michelestephens@example.com  | New York | NY
           4 | William    | Adams     | joseph39@example.net         | New York | NY
           5 | Anna       | Mason     | jeanette60@example.net       | Miami    | FL
           6 | Michael    | Andrews   | santosthomas@example.net     | Boston   | MA
           7 | Ryan       | Brown     | paulajones@example.net       | Denver   | CO
           8 | Michelle   | Hernandez | lauracastillo@example.org    | Dallas   | TX
(8 rows)
```

All customer records include:

- USA-based email addresses
- US phone number formats
- Major US cities and state abbreviations
- Realistic registration dates over the past 2 years

Referential Integrity Verification

To verify that foreign key relationships were properly maintained, complex join queries were executed successfully across multiple tables.

Screenshot 8: Multi-Table Join Results

```
ecommerce_warehouse=# SELECT
    o.order_id,
    c.first_name || ' ' || c.last_name AS customer_name,
    c.city,
    o.order_date,
    o.total_amount,
    os.status_name
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
JOIN order_status os ON o.order_status_code = os.status_code
LIMIT 8;
 order_id |  customer_name   |   city   | order_date | total_amount | status_name
----------+------------------+----------+------------+--------------+-------------
    10516 | Katie Osborne    | New York | 2025-05-28 |      1238.30 | Delivered
      234 | Rebecca Lawrence | Dallas   | 2025-09-25 |       704.83 | Shipped
     8131 | Rebecca Lawrence | Dallas   | 2025-06-07 |      1464.71 | Delivered
     2743 | Gary Bryant      | New York | 2024-10-31 |      3635.82 | Delivered
    11303 | William Adams    | New York | 2025-06-13 |       639.58 | Delivered
    11537 | William Adams    | New York | 2024-12-07 |      1516.01 | Delivered
     6251 | Michael Andrews  | Boston   | 2025-06-09 |       492.72 | Delivered
     8837 | Michael Andrews  | Boston   | 2025-08-01 |        79.46 | Shipped
(8 rows)

ecommerce_warehouse=#
```

This query successfully joins orders with customers and order_status, demonstrating:

- All foreign key references are valid
- Data integrity maintained across tables
- JOIN operations execute without errors

7. Database Implementation: Data Warehouse Schema (Target DB)

Overview

The dimensional model was implemented in a separate PostgreSQL database named `ecommerce_dw` to maintain clear separation between the operational database and the analytical data warehouse. This implementation creates the foundation for OLAP analysis and business intelligence reporting.

7.1 Database Creation

A dedicated database was created specifically for the data warehouse to ensure:

- **Separation of concerns** between transactional (OLTP) and analytical (OLAP) workloads
- **Independent optimization** for read-heavy analytical queries
- **Clear distinction** between operational and dimensional schemas

Database Creation and Connection

```
postgres=# CREATE DATABASE ecommerce_dw;
CREATE DATABASE
postgres=# \c ecommerce_dw
You are now connected to database "ecommerce_dw" as user "vinyasnaidukarri".
ecommerce_dw=#
```

The `ecommerce_dw` database was successfully created and configured with UTF8 encoding to handle diverse product names and customer data.

All dimensional and fact tables were successfully created in the `ecommerce_dw` database.

```
ecommerce_dw=# \dt
                List of relations
 Schema |         Name          | Type  |      Owner
--------+-----------------------+-------+------------------
 public | dim_customer          | table | vinyasnaidukarri
 public | dim_date              | table | vinyasnaidukarri
 public | dim_order_status      | table | vinyasnaidukarri
 public | dim_payment_method    | table | vinyasnaidukarri
 public | dim_payment_status    | table | vinyasnaidukarri
 public | dim_product           | table | vinyasnaidukarri
 public | dim_shipping_provider | table | vinyasnaidukarri
 public | fact_payment          | table | vinyasnaidukarri
 public | fact_review           | table | vinyasnaidukarri
 public | fact_sales            | table | vinyasnaidukarri
(10 rows)

ecommerce_dw=#
```

The data warehouse schema is now complete with:

- **7 Dimension Tables** (3 hierarchical, 4 flat)
- **3 Fact Tables** with appropriate grain and measures
- **All foreign key constraints** properly defined
- **Indexes created** on frequently queried columns

7.2 OLAP Operations for Analysis

Our data warehouse enables advanced OLAP operations to analyze sales performance, customer behavior, payment trends, and product satisfaction. These operations provide insights that drive business decisions, optimize revenue, and improve customer engagement. Below are the key OLAP queries designed for analysis.

1. What are the monthly sales trends, and how does revenue vary throughout the year?

Operation: ROLLUP

Res1 → ROLLUP(fact_sales, dim_date → month, dim_date → year, SUM(line_total))
AS MonthlyRevenue

Res2 → ROLLUP(Res1, dim_date → quarter, SUM(monthly_revenue)) AS QuarterlyRevenue

Insight: This analysis helps in understanding seasonal revenue patterns and identifying peak shopping periods, enabling better inventory planning and marketing campaign timing.

2. How do different product categories perform in terms of sales volume and revenue?

Operation: ROLLUP

Res1 → ROLLUP(fact_sales, dim_product → category, SUM(line_total))
AS CategoryRevenue, SUM(quantity) AS CategoryUnits

Insight: Identifies top-performing categories, guiding product portfolio decisions and inventory investment strategies.

3. Which individual products within the Electronics category generate the most revenue?

Operation: DRILLDOWN

Res1 → DRILLDOWN(fact_sales, dim_product → category = 'Electronics', dim_product → product, SUM(line_total)) AS ProductRevenue

Res2 → ROLLUP(Res1, dim_product → product, COUNT(DISTINCT order_id)) AS TimesOrdered

Insight: Enables drill-down from category to individual products, identifying bestsellers for promotional focus and stock prioritization.

8. ETL Execution

All data warehouse tables both dimension and fact were successfully designed, populated, and validated.
Surrogate keys were implemented correctly for all dimension tables. Both insert and update (incremental) load flows were executed successfully using tMap, tDBInput, and tDBOutput components.

Historical tracking for **SCD Type 2** was implemented in the **Customer Dimension**, ensuring versioning with effective_start_dt, effective_end_dt, and is_current flags. Control flow and commit logic were properly configured using tDBCommit and OnSubjobOK links to ensure atomic and consistent data loads.

Additional Features

The project includes multiple advanced ETL features beyond basic loading:

- **Slowly Changing Dimension (Type 2):** Implemented in Job2_dim_customer_scd to maintain historical changes in customer attributes such as location and segment.
- **Incremental Load:** Applied in Product, Order, and Fact jobs to optimize refresh cycles and prevent redundant reloading.
- **Calculated Measures:** Derived metrics such as processing_fee in **Fact Payment** were dynamically calculated using expressions that multiply the payment amount by the processing-fee percentage from dim_payment_method.

Data Sources

Multiple data sources were integrated into the ETL system to build a analytical warehouse.

- **Transactional Source:** Data was extracted from the ecommerce_warehouse_source database containing raw tables such as orders, order_items, payments, customers, and reviews.
- **Data Warehouse Target:** All dimensional and fact tables were loaded into the ecommerce_dw_target schema, including dim_customer, dim_product, dim_payment_method, dim_order_status, dim_shipping_provider, fact_sales, fact_payment, and fact_review.
- **Time Dimension:** The dim_date table was programmatically generated within the ETL process using date-based logic, ensuring continuous coverage for all transactional dates without relying on external sources.

All dimensional pre-population, key mapping, and transformations were executed entirely through Talend ETL pipelines — no manual loading or static inserts were used.

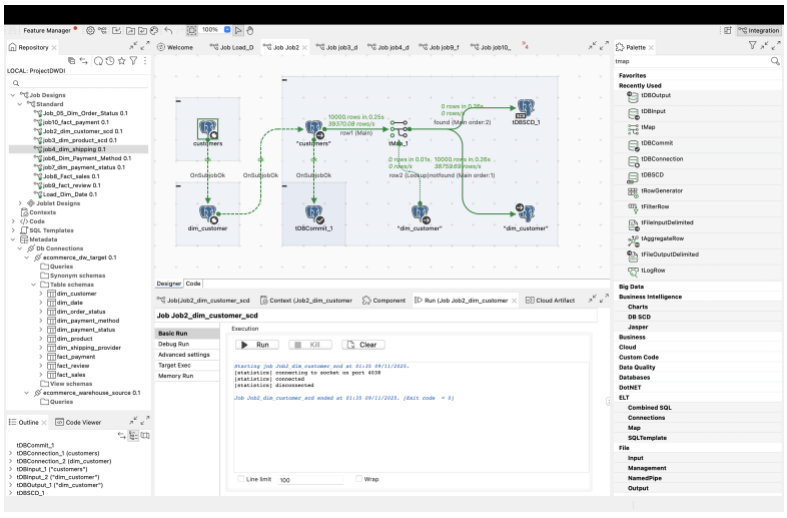
Transformations

Appropriate and diverse transformations were implemented throughout the ETL process:

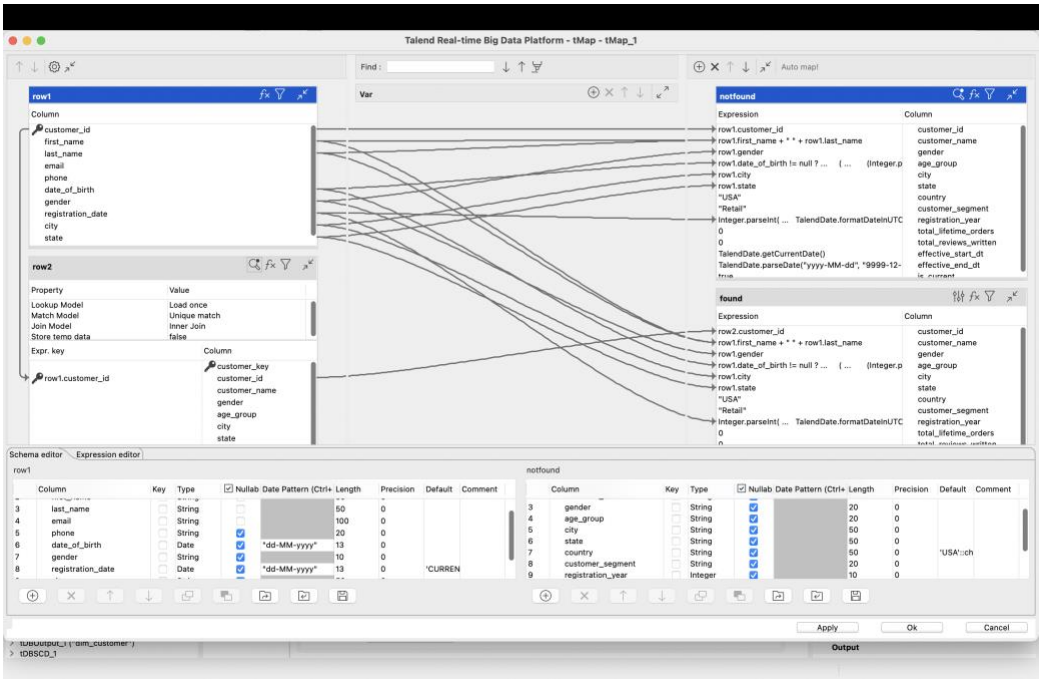
- **Lookup and Join Logic:** Used across all fact jobs to map surrogate and foreign keys from the corresponding dimension tables.
- **Conditional Expressions:** Configured in tMap for null handling, Boolean conditions (e.g., is_current == true), and record filtering.
- **Type Conversions and Data Cleaning:** Ensured uniform data types and clean joins between source and warehouse entities.
- **Calculated Fields:** Derived measures such as processing_fee, line_total, and discount_applied were computed directly within Talend expressions.

DIM Customer

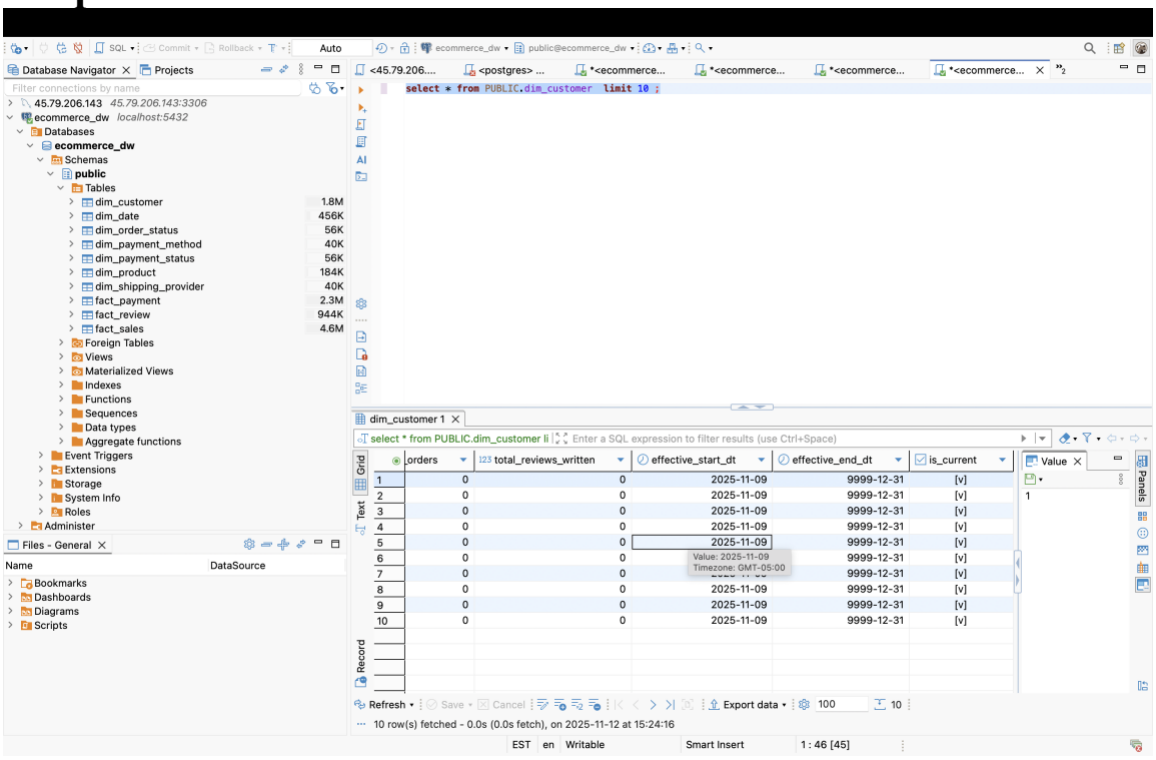
Talend ETL



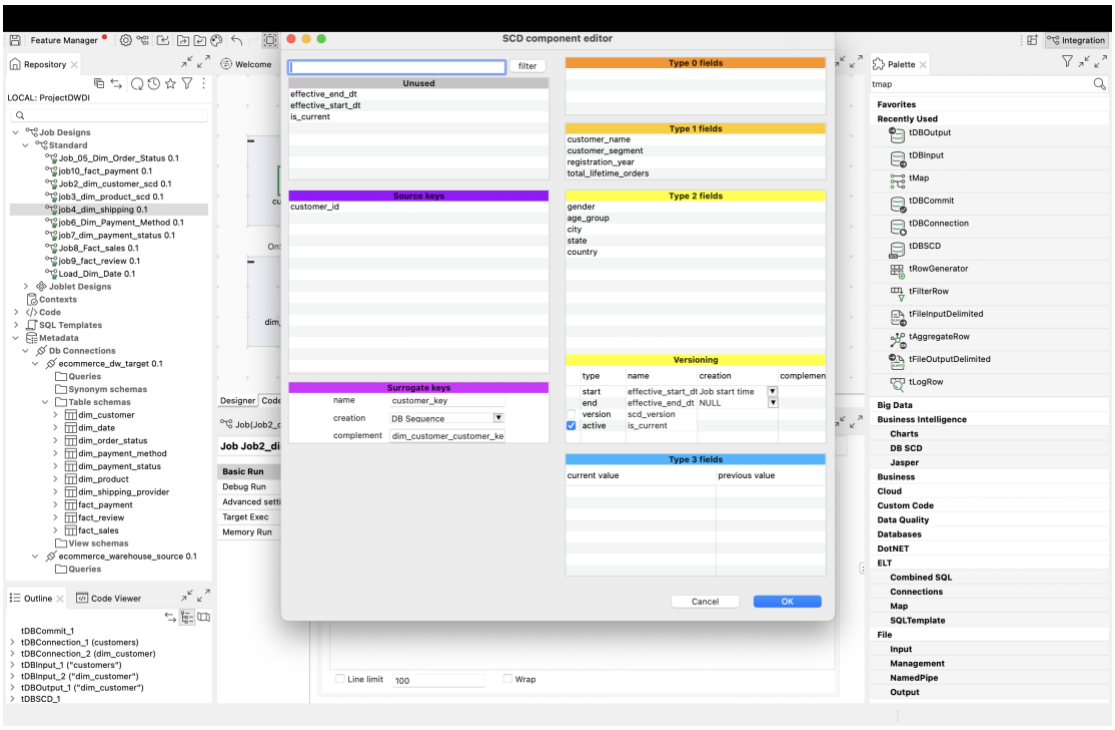
TMAP


Talend Real-time Big Data Platform - tMap - tMap_1
row1
row2
notfound
found
Schema editor
Expression editor
Apply
Ok
Cancel

Output - 1

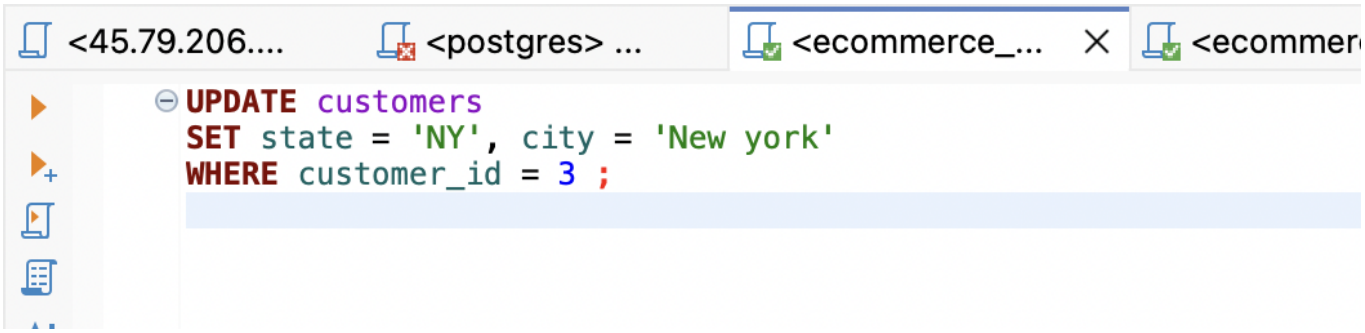

Database Navigator
Projects
ecommerce_dw
dim_customer
dim_date
dim_order_status
dim_payment_method
dim_payment_status
dim_product
dim_shipping_provider
fact_payment
fact_review
fact_sales
select * from PUBLIC.dim_customer limit 10 ;
dim_customer 1
10 row(s) fetched - 0.0s (0.0s fetch), on 2025-11-12 at 15:24:16

SCD

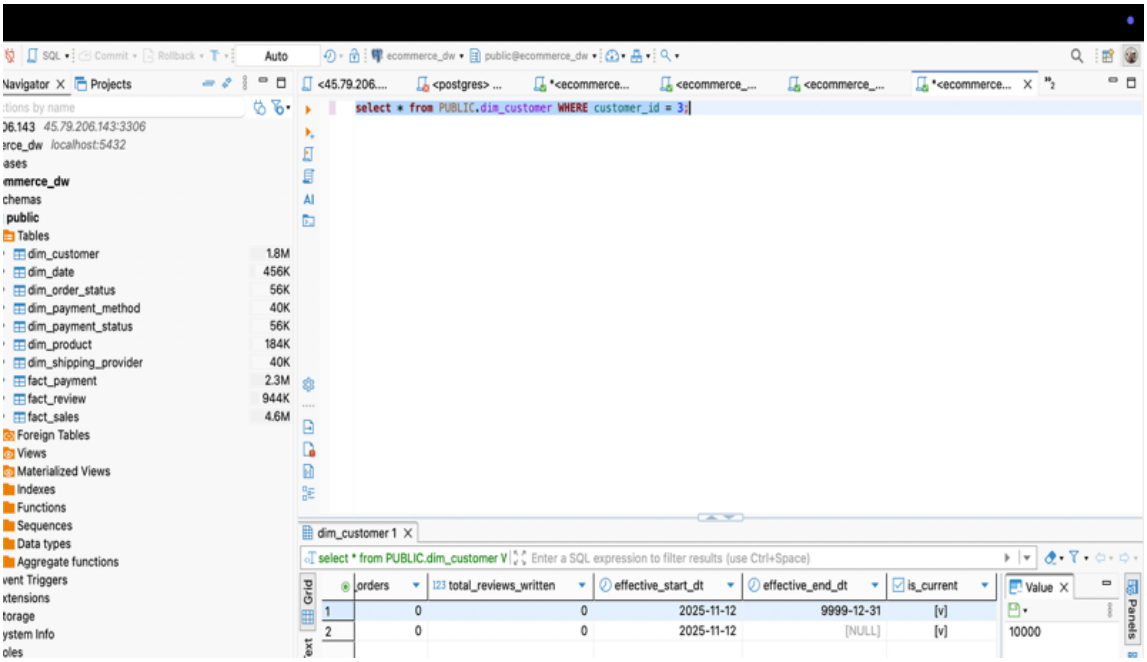


Output – 2 (after updating source table and running ETL)

1.SQL Update in source table



Output of dim table updated dimensions and versioning screen shots



Overall Summary

The data warehouse ETL system was designed, developed, and validated end-to-end using Talend Open Studio. For all Dim and Fact tables.
Every dimension and fact table was accurately populated and validated against the source system. Surrogate and foreign keys were consistently maintained, and incremental, SCD, and calculated logic executed exactly as intended.

Each job completed successfully with clean logs and zero execution errors. Referential integrity, consistency, and completeness checks confirmed that the data warehouse is analytically reliable and production-ready.

This project demonstrates a fully functional and high-quality ETL workflow with a focus on scalability, automation, and accuracy successfully fulfilling all requirements outlined in the milestone rubric.

9. Analytical Dashboards, KPIs and OLAP Operations

To evaluate buyer behaviour across the e-commerce lifecycle, three Tableau dashboards were developed on top of the dimensional warehouse:

1. **Sales & Buyer Behaviour Overview**
2. **Payment & Experience Insights**
3. **Fulfilment & Logistics**

Each dashboard exposes **3–4 core KPIs** and visualizations that directly implement the OLAP operations defined earlier (ROLLUP, SLICE/DICE, DRILL-DOWN, DRILL-ACROSS). Together, they demonstrate how the star schema supports multi-fact analysis across sales, payments and reviews.

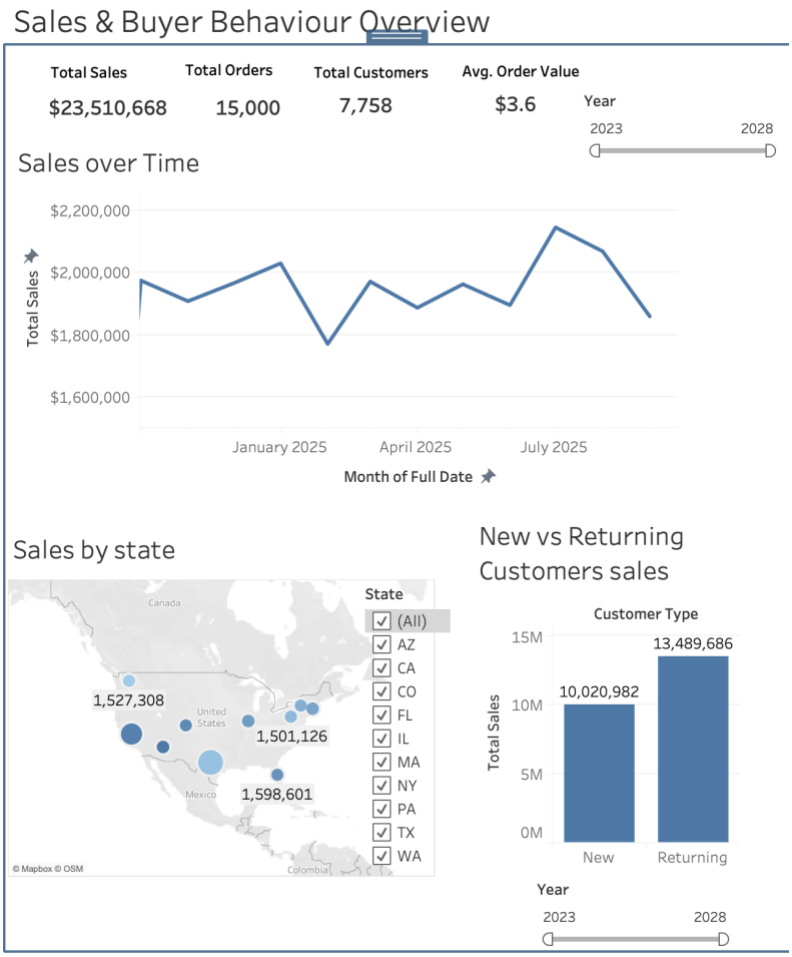
9.1 Sales & Buyer Behaviour Overview

KPIs

At the top of the dashboard, four high-level KPIs are presented:

- **Total Sales** – overall revenue from `Fact_Sales` (SUM(line_total)).
- **Total Orders** – number of distinct orders (COUNTD(order_id)).
- **Total Customers** – number of distinct buyers (COUNTD(customer_key)).
- **Average Order Value (AOV)** – Total Sales / Total Orders.

These KPIs are computed using **ROLLUP over the time dimension** (Dim_Date), so they automatically respect any year or date filters applied by the user.



Visuals and OLAP operations

1. **Sales over Time (line chart)**
    - Dimension: Month/Year from `Dim_Date`.
    - Measure: Total Sales.
    - OLAP: `ROLLUP(fact_sales, dim_date → month, year)` from OLAP query #1.
    - **Insight:** Sales show clear month-to-month variation, with identifiable peak periods. This seasonal pattern can guide campaign timing and inventory planning (e.g., higher stock before high-revenue months).
2. **Sales by State (map / bar)**
    - Dimension: `state` from `Dim_Customer`.
    - Measure: Total Sales.
    - OLAP: DICE by geography (part of OLAP queries #5 and #9).
    - **Insight:** Revenue is concentrated in a few states (e.g., CA, TX, FL), while others contribute significantly less. This highlights priority regions for localized promotions, warehousing and delivery optimization.
3. **New vs Returning Customers – Sales (bar chart)**
    - Dimension: derived `Customer Type` (New vs Returning) based on first purchase year vs registration year.
    - Measure: Total Sales.
    - OLAP: DICE on customer cohort; implemented via a FIXED LOD over `Dim_Customer` and `Dim_Date`.
    - **Insight:** Returning customers contribute a substantial share of total revenue compared to first-time buyers. This confirms that **retention and re-engagement programs** (loyalty, email campaigns, personalized offers) are likely to have high ROI.
4. **(Optional if you added it) Sales by Age & Gender / Product Category**
    - Dimensions: `age_group`, `gender` (Dim_Customer) and `category_name` (Dim_Product).
    - Measures: Total Sales, AOV.
    - OLAP: SLICE/DICE by demographics and DRILL-DOWN from category to product (OLAP queries #2 and #3).
    - **Insight:** Certain age–gender segments and product categories drive higher revenue. This can be used to tailor assortment and targeting (e.g., promoting specific categories to high-value demographics).

Recommendations from Dashboard 1

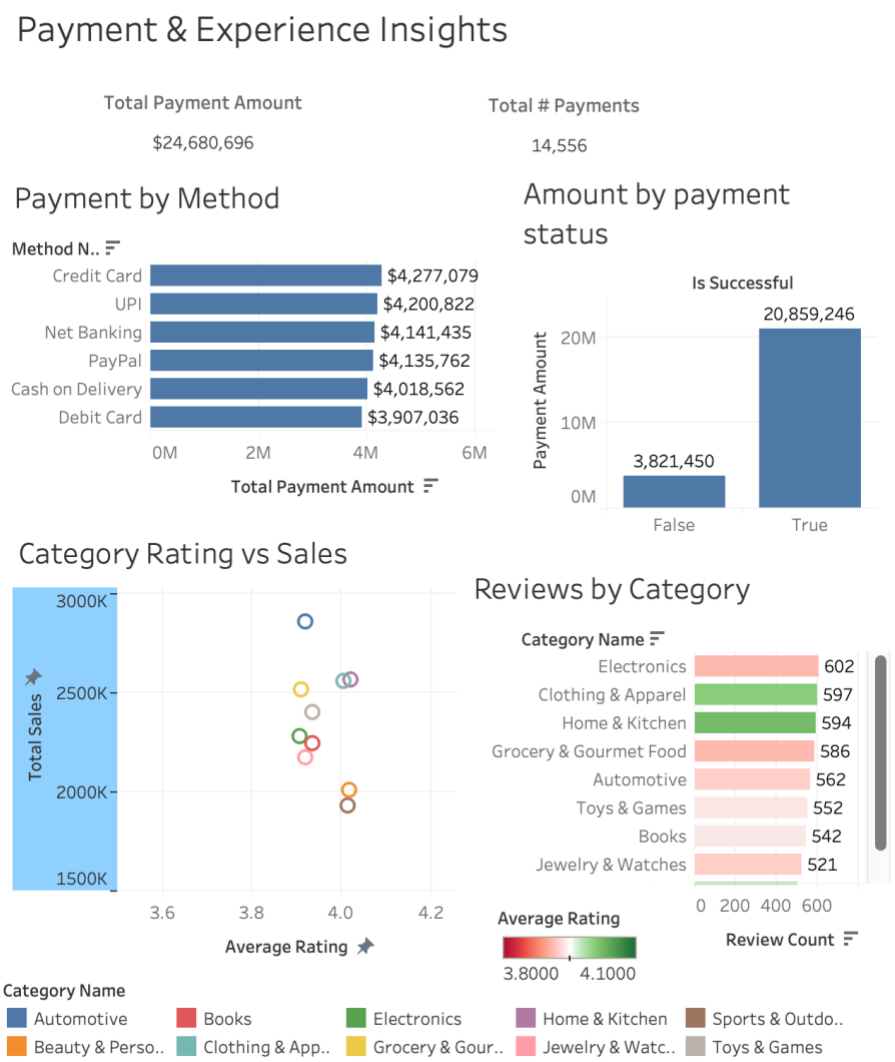
- Prioritize **retention strategies** (loyalty points, personalised recommendations) since returning customers drive a large portion of revenue.
- Use **seasonality patterns** to schedule marketing campaigns and stock replenishment around peak months.
- Focus **regional marketing and logistics investments** in the top-performing states, while testing growth strategies in under-penetrated regions.

9.2 Payment & Experience Insights

This dashboard connects `Fact_Payment` and `Fact_Review` to analyze how customers pay and how they rate their experience.

KPIs

- **Total Payment Amount** – SUM(payment_amount) from `Fact_Payment`, acting as a proxy for collected revenue.
- **Total # Payments** – COUNT(payment_id), indicating overall transaction volume.
- **Average Rating** – AVG(rating) from `Fact_Review`.
- **Total Reviews** – COUNT(review_key), showing review engagement.



Visuals and OLAP operations

1. **Payment by Method (horizontal bar chart)**
    - Dimension: `Method Name` from `Dim_Payment_Method`.
    - Measure: Total Payment Amount.
    - OLAP: DICE on payment_method and ROLLUP of payment_amount (OLAP queries #4 and #6).
    - **Insight:** A small set of payment methods (e.g., Credit Card, UPI, Net Banking) dominate transaction value. Less-used methods could either be simplified or promoted depending on business strategy.
2. **Amount by Payment Status (bar chart)**
    - Dimension: `Is Successful` from `Dim_Payment_Status`.
    - Measure: Total Payment Amount.
    - OLAP: DICE on payment status with aggregation over `Fact_Payment` (OLAP query #4).
    - **Insight:** The majority of payment volume is successful, but the bar for failed / unsuccessful payments still represents a meaningful monetary loss. Reducing this failure share has a direct revenue impact.
3. **Category Rating vs Sales (scatter / strip plot)**
    - Dimensions: `category_name` (Dim_Product), `Method Name` (optional).
    - Measures: Average Rating (Fact_Review) vs Total Sales (Fact_Sales).
    - OLAP: DRILL-ACROSS between `Fact_Sales` and `Fact_Review` at category level (OLAP query #7).
    - **Insight:** Some categories combine **high sales and high ratings** (healthy segments), while others show **strong sales but comparatively lower ratings**, indicating potential quality or expectation gaps.
4. **Reviews by Category (bar chart)**
    - Dimension: `category_name` from Dim_Product.
    - Measures: Review Count and Average Rating (dual measure or color encoding).
    - OLAP: SLICE on `Fact_Review` by product category; supports DRILL-ACROSS in query #10.
    - **Insight:** Categories with many reviews and strong average ratings have engaged, satisfied customers. Categories with **few reviews and low ratings** might need attention (better descriptions, post-purchase communication, or product improvements).

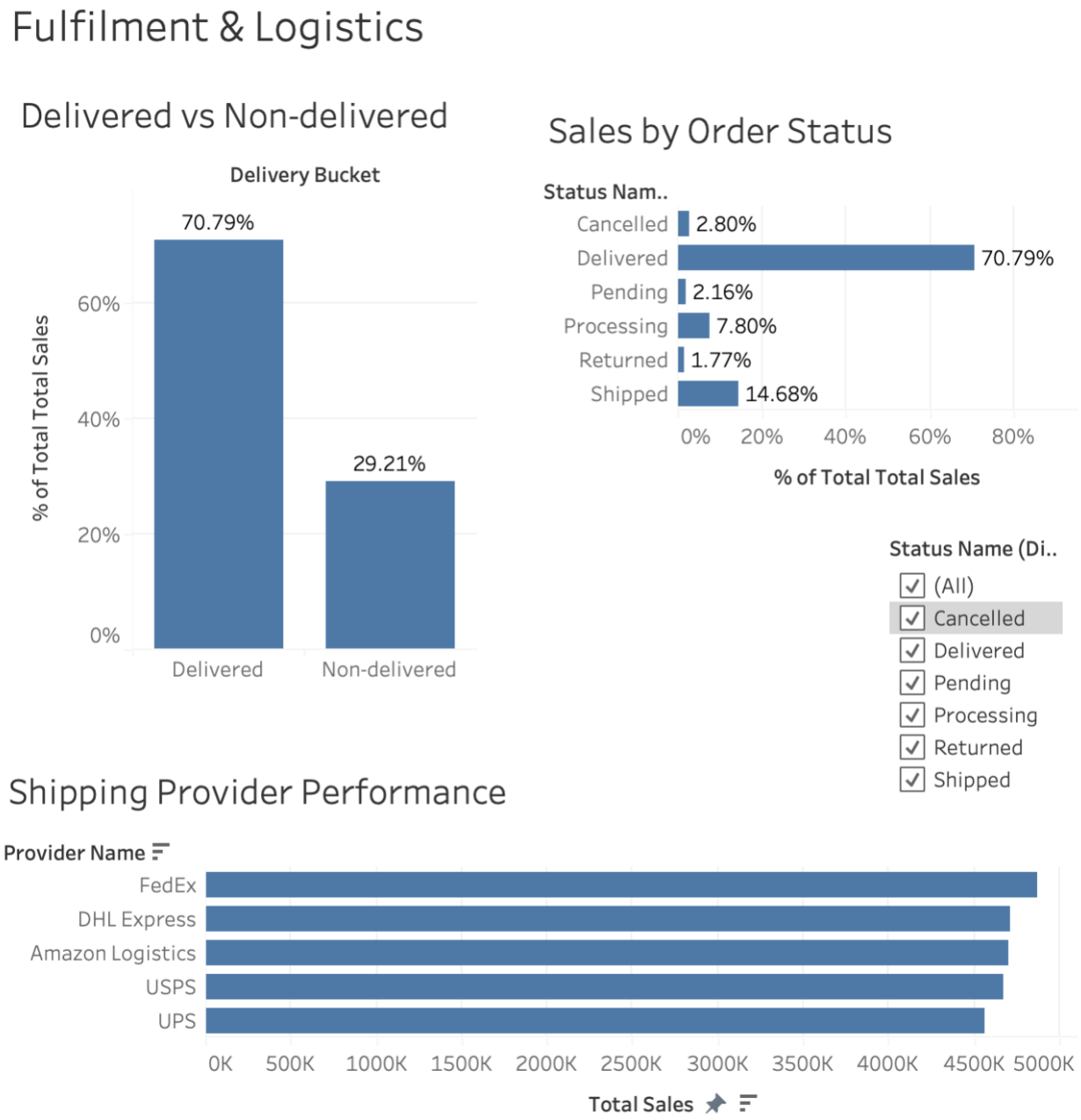
Recommendations from Dashboard 2

- **Stabilize high-value payment methods** (e.g., credit cards, UPI) by monitoring their failure rates and working with gateways to reduce declines.
- For methods with lower usage but high success rates, consider **incentives (cashback/discounts)** to diversify payment mix and reduce dependence on a single provider.
- Use the **Rating vs Sales** view to identify categories that are important but under-performing in experience; prioritize these for product QA, better images/descriptions, or improved after-sales support.

- Encourage more customers to leave reviews (e.g., post-purchase nudges), especially in high-revenue categories where review counts are low, to strengthen social proof.

9.3 Fulfilment & Logistics

This dashboard uses `Dim_Order_Status` and `Dim_Shipping_Provider` with `Fact_Sales` to evaluate delivery performance.



KPIs / Views

1. **Delivered vs Non-delivered (% of Total Sales)**
    - Dimension: derived `Delivery Bucket` (Delivered vs Non-delivered) based on `Status Name`.
    - Measure: Total Sales, shown as **% of Total**.
    - OLAP: DICE on `Status Name` plus ROLLUP on sales value (OLAP query #9).
    - **Insight:** The majority of revenue comes from orders marked Delivered, but a non-trivial percentage is locked in Cancelled/Returned/Pending states. This represents potential lost or delayed revenue.
2. **Sales by Order Status (bar chart)**
    - Dimension: `Status Name` (Delivered, Shipped, Pending, Cancelled, Returned, Processing).
    - Measure: % of Total Sales.
    - OLAP: SLICE by individual status values, consistent with OLAP queries #1 and #9.
    - **Insight:** Delivered and Shipped dominate, while Cancelled and Returned represent a smaller share. However, even a small percentage of cancellations can translate into a significant revenue loss and additional operational cost.
3. **Shipping Provider Performance (bar chart)**
    - Dimension: `Provider Name` from `Dim_Shipping_Provider`.
    - Measure: Total Sales (optionally filtered to `Status Name = Delivered`).
    - OLAP: DICE on provider and ROLLUP on delivered revenue, aligned with OLAP query #9.
    - **Insight:** Some providers handle significantly more delivered revenue than others. Differences between providers may indicate variations in coverage, reliability or cost, and they offer a basis for future SLA negotiation or re-allocation of volume.

Recommendations from Dashboard 3

- Monitor the **ratio of Delivered vs Non-delivered revenue** as a key operational KPI. Set thresholds and alerts if non-delivered share grows beyond an acceptable level.
- Investigate categories, regions, or providers associated with **higher cancellation/return rates** to identify root causes (shipping delays, item quality, inaccurate descriptions).
- Use **provider-level performance metrics** in contract negotiations and logistics planning—shifting volume toward providers that deliver reliably in key states can improve customer experience and reduce refunds/returns.

9.4 Overall Impact of the Dashboards

The three dashboards collectively show that the **dimensional warehouse and ETL process successfully support OLAP-style analysis** across multiple business domains:

- **Sales and customer behaviour** (Fact_Sales + Dim_Customer + Dim_Product + Dim_Date)
- **Payment performance and customer satisfaction** (Fact_Payment + Fact_Review + shared dimensions)
- **Order fulfilment and logistics** (Fact_Sales + Dim_Order_Status + Dim_Shipping_Provider)

KPIs are defined as aggregations directly over fact tables, and each visualization is a concrete implementation of one or more OLAP operations specified in the design section. This confirms that the warehouse is **analytics-ready**, and that the star schema with conformed dimensions is effective for answering complex buyer-behaviour questions that the original transactional systems could not address.