Nicole Gallo
D597 – Database Management
May 23, 2025
MKN1 Task 1: Relational Database Design and Implementation

## Table of Contents

*Part 1: Design Document* ..... 2

**A1: Business Problem** ..... 2

**A2: Proposed Data Structure** ..... 2

**A3: Database Justification** ..... 3

**A4: Data Utilization** ..... 4

**B: Logical Data Model** ..... 5

**C: Database Objects** ..... 5

**D: Scalability Strategies** ..... 5

**E: Privacy & Security Measures** ..... 6

*Part 2: Implementation* ..... 7

**F1 – Write script to create a database instance named "D597 Task 1" using SQL** ..... 7

**F2 – Write script to import the data records from the Scenario 2 CSV file into the D597 database** ..... 8

**F3 – Write script for three queries to retrieve specific information from EcoMart database.** ..... 11

**F4 - Apply optimization techniques to improve the run time of your queries from part F3, providing output results via a screenshot.** ..... 14

*Part 3: Presentation* ..... 21

**Part 1: Design Document**

# A1: Business Problem

EcoMart faces challenges managing diverse sustainability attributes across their product offering from eco-friendly vendors. This limits their commitment to providing, promoting, and connecting sustainability and environmental consciousness to consumers.

EcoMart is looking to create a database to support their business goals and values by addressing these problems:

- Lacking platform scalability and flexibility for their consumers
- Inconsistent tracking of sustainability certifications
- Difficulty linking eco-friendly products to verification of sustainability certifications
- Sacrificing platform speed and reliability for consumers

# A2: Proposed Data Structure

To solve these problems, EcoMart will adopt a **relational database** that highlights the following tables, entities, and attributes:

Product Table - *Products*

- ProductID (Integer) - PK
- ProductName (Varchar)
- ProdDescription (Varchar)
- Price (Decimal 10,2)
- CategoryID (Integer) – FK
- BrandID (Integer) - FK
- CategoryName (Varchar)

Customer Table - *Customers*

- CustomerID (Integer) - PK
- CustomerName (Varchar)
- CustomerPhone (Varchar)
- CustomerEmail (Varchar)

Categories Table – *Categories*

- CategoryID (Integer) – PK
- CategoryName (Varchar)

Brands Table – *Brands*

- BrandID (Integer) - PK
- BrandName (Varchar)

Orders Table – *Orders*

- OrderID (Integer) - PK
- CustomerID (Integer) - FK

- OrderDate (Date)
- OrderStatus (Varchar)
- TotalAmount (Decimal 10,2)

Order Details Table – *OrderDetails*
- OrderDetailID (Integer) - PK
- OrderID (Integer) - FK
- ProductID (Integer) - FK
- Quantity (Integer)
- UnitPrice (Decimal 10,2)

Reviews Table – *Reviews*
- ReviewID (Integer) - PK
- CustomerID (Integer) - FK
- ProductID (Integer) - FK
- Rating (TinyInteger)
- ReviewText (Varchar)
- ReviewDate (Date)

Certifications Table – *Certifications*
- CertificationID (Integer) - PK
- CertificationName (Varchar)

Inventory Table – *Inventory*
- ProductID (Integer) – PK, FK
- StockLevel (Integer)
- ReorderLevel (Integer)

Product Certifications Table - *ProductCertifications*
- ProductID (Integer) - FK
- CertificationID (Integer) - FK

Supplier Table - *Suppliers*
- SupplierID (Integer) – PK
- ProductID (Integer) - FK
- SupplierName (Varchar)
- ProdAvailability (Boolean)
- SustainabilityCert (Boolean)

# A3: Database Justification

A relational database for EcoMart solves these issues by:

1. Supporting **complex relationships** between entities such as, products, certifications, customers, suppliers, etc.

2. Enforcing **ACID compliance (atomicity, consistency, isolation, durability)** and **data integrity** for accurate inventory/sustainability updates
3. Enabling **OLAP (online analytical processing) queries** for flexibility and high-quality data reporting
4. Providing **scalability and performance optimization** for past, current, and new orders, products and customers.

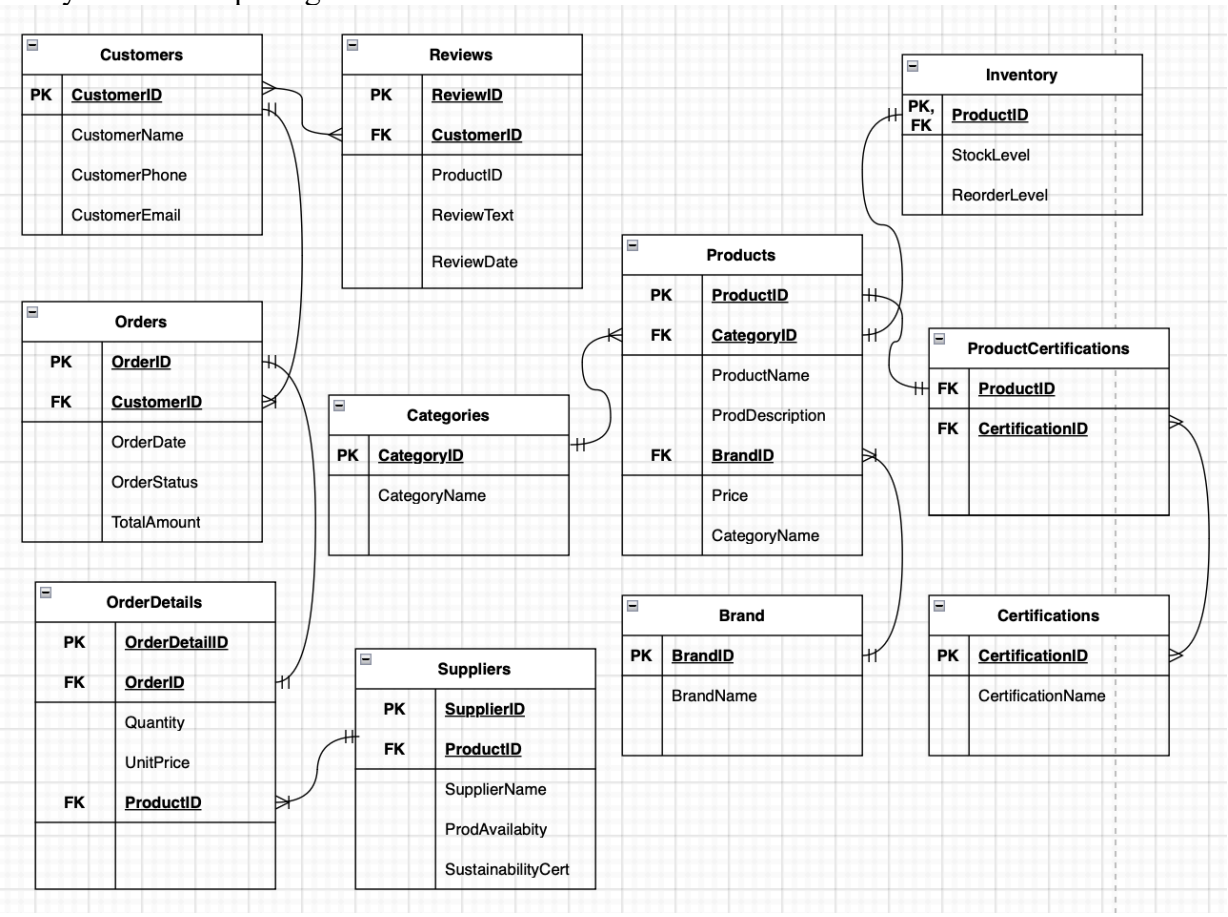
# A4: Data Utilization

The data utilization for EcoMart is as follows:

- **Product discovery** – the ability for customers to search and filter through products using entities such as, categorization, certification, availability, and price.
- **Order processing** – the system can track customer orders, inventory updates, and fulfillment details.
- **Customer engagement** – the process of submitting reviews and ratings for product support
- **Sustainability analytics** – the ability to analyze eco-friendly trends and certification statuses

# B: Logical Data Model

Entity-Relationship Diagram for EcoMart




# C: Database Objects

The database for EcoMart will contain the following:

- **Tables** – this database object is how rows and columns will be stored
- **Indexes** – this database object is used to optimize queries so the performance is enhanced
- **Views** – this database object is for simplifying complex queries to support data presentation

The files within the database will store attributes such as data tyles, default values, constraints, and if an attribute is NULL or NOT NULL.

# D: Scalability Strategies

To support scalability and future growth, we will use the following strategies:

- **Denormalization** – Combining and adding necessary information amongst tables such as creating CategoryName and SupplierName directly into the Products table.
- **Horizontal Scaling (Sharding)** – Dividing the order and supplier data into geographic regions or time zones to support high volumes of orders and shipping.

- **Vertical Scaling** – Increasing resources to servers such as expanding storage to be able to handle more orders in the future while keeping a log of past and current orders in the database.
- **Caching** – Implementing caching for specific "every day" queries such as finding most sold products or products with certifications. This will save time spent on rewriting popular queries.

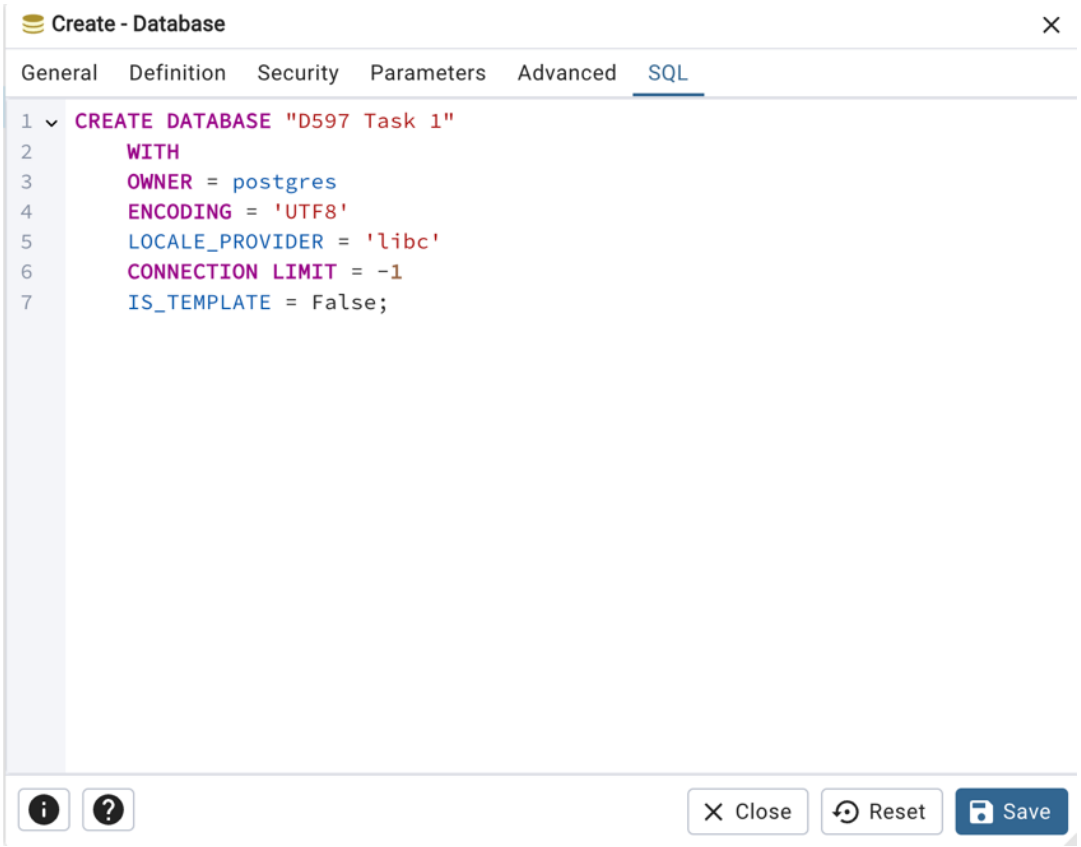
# E: Privacy & Security Measures

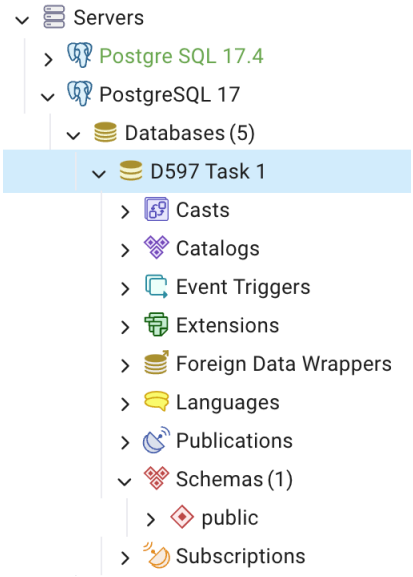
The privacy and security measures that should be implemented are as follows:

1. **Data Protection (Encryption):**
    a. Sensitive data like customer information (contact info, billing, etc.) needs to be encrypted and protected from possible data breaches.
2. **Access Control**:
    a. Assignment of roles such as a "sustainability auditor" should be implemented to restrict data visibility on specific and relevant tables
3. **Compliance:**
    a. There are laws that deal with storing and handling personal information and data. There needs to be compliance checks in place to ensure these laws (GDPR, PCI-DSS, etc.) are being upheld.
4. **Monitoring:**
    a. The monitoring of data is important because as EcoMart grows, it's more susceptible to cyber-attacks and threats
    b. There should be an implementation for real-time alerting for suspicious queries, such as bulk supplier data exports.

## Part 2: Implementation

### F1 – Write script to create a database instance named “D597 Task 1” using SQL



**F2 – Write script to import the data records from the Scenario 2 CSV file into the D597 database**

Query | Query History

```
--CREATE TABLE in D597 Task1 Database
CREATE TABLE EcoMart (
    region VARCHAR(100),
    country VARCHAR(100),
    item_Type VARCHAR(100),
    sales_Channel VARCHAR(50),
    order_Priority VARCHAR(1),
    order_Date DATE,
    order_ID INT,
    ship_Date DATE,
    units_Sold INT,
    unit_Price DECIMAL(10,2),
    unit_Cost DECIMAL(10,2),
    total_Revenue DECIMAL(10,2),
    total_Cost DECIMAL(10,2),
    total_Profit DECIMAL(10,2)
)
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 60 msec.
```

COPY data from CSV file

Query | Query History

```
--COPY command to copy the data in EcoMart table
COPY ecomart
FROM '/Applications/PostgreSQL 17/100000_Sales_Records.csv'
DELIMITER ','
CSV HEADER;
```

Data Output | Messages | Notifications

```
COPY 100000

Query returned successfully in 404 msec.
```

Query Query History

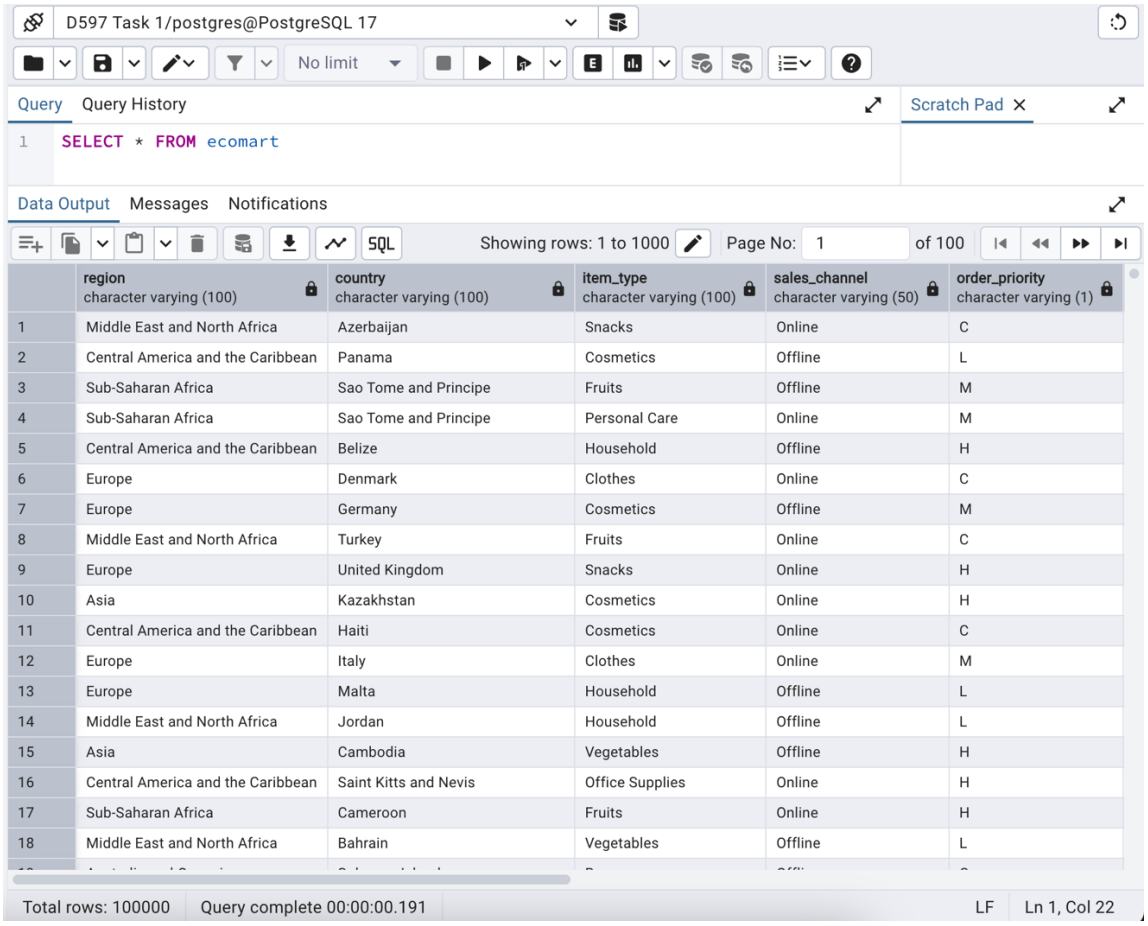
```
--COPY command to copy the data in EcoMart table
COPY ecomart
FROM '/Applications/PostgreSQL 17/100000_Sales_Records.csv'
DELIMITER ','
CSV HEADER;

--COUNT to ensure amount of records in dataset (100,000)
SELECT COUNT(*)
FROM ecomart;
```

Data Output Messages Notifications

Showing row

| | count bigint |
|---|---|
| 1 | 100000 |

SELECT the entire dataset for EcoMart



INSERT INTO command to insert new data entry (new order)

Query | Query History

```
90
91  --INSERT INTO command to insert new order into EcoMart table
92  INSERT INTO ecomart(region, country, item_type, sales_channel,
93  order_priority, order_date, order_id, ship_date, units_sold,
94  unit_price, unit_cost, total_revenue, total_cost, total_profit)
95  VALUES
96  ('North America', 'Canada',
97  'Beverages', 'Online', 'H', '5/27/25',
98  '999999998', '6/3/25', '8759', '47.45',
99  '31.79', '415614.55', '278448.61', '137165.94');
100
101 SELECT * FROM ecomart WHERE order_id = '999999998';
102
```

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 75 msec.
```

Query | Query History

```
91  --INSERT INTO command to insert new order into EcoMart table
92  INSERT INTO ecomart(region, country, item_type, sales_channel,
93  order_priority, order_date, order_id, ship_date, units_sold,
94  unit_price, unit_cost, total_revenue, total_cost, total_profit)
95  VALUES
96  ('North America', 'Canada',
97  'Beverages', 'Online', 'H', '5/27/25',
98  '999999998', '6/3/25', '8759', '47.45',
99  '31.79', '415614.55', '278448.61', '137165.94');
100
101 SELECT region, country, item_type, order_id
102 FROM ecomart WHERE order_id = '999999998';
```

Data Output | Messages | Notifications

Showing rows: 1 to

| | region character varying (100) | country character varying (100) | item_type character varying (100) | order_id integer |
|---|---|---|---|---|
| 1 | North America | Canada | Beverages | 999999998 |

**F3 – Write script for three queries to retrieve specific information from EcoMart database.**

Q1 – What are the top 5 products by total profit?

Query | Query History

```sql
--Q1 - What are the top 5 products by total profit?
SELECT
    item_type,
    SUM (total_profit) as total_profit
FROM ecomart
GROUP BY item_type
ORDER BY total_profit DESC
LIMIT 5;
```

Data Output | Messages | Notifications

Showing row

| | item_type character varying (100) | total_profit numeric |
|---|---|---|
| 1 | Cosmetics | 7289406555.68 |
| 2 | Household | 6870966095.35 |
| 3 | Office Supplies | 5339532912.50 |
| 4 | Baby Food | 4017647893.20 |
| 5 | Cereal | 3743318890.62 |

Query | Query History

```sql
--Q1 - What are the top 5 products by total profit?
SELECT
    item_type,
    SUM (total_profit) as total_profit
FROM ecomart
GROUP BY item_type
ORDER BY total_profit DESC
LIMIT 5;
```

Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 131 msec.
5 rows affected.
```

Q2 – What is the average delivery time by sales channel?

Query | Query History

```sql
--Q2 - What is the average delivery time by sales channel?
SELECT
    sales_channel,
    AVG(ship_date - order_date) AS avg_delivery_days
FROM ecomart
GROUP BY sales_channel;
```

Data Output | Messages | Notifications

Showing rows: 1 to 2

| | sales_channel character varying (50) | avg_delivery_days numeric |
|---|---|---|
| 1 | Offline | 24.9830817282665279 |
| 2 | Online | 25.0886642426179726 |

Query | Query History

```sql
--Q2 - What is the average delivery time by sales channel?
SELECT
    sales_channel,
    AVG(ship_date - order_date) AS avg_delivery_days
FROM ecomart
GROUP BY sales_channel;
```

Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 94 msec.
2 rows affected.
```

Q3 – What is the total revenue by region?

Query | Query History

```sql
--Q3 - What is the total revenue by region?
SELECT
    region,
    SUM(total_revenue) AS total_revenue
FROM ecomart
GROUP BY region
ORDER BY total_revenue DESC;
```

Data Output | Messages | Notifications

| | region<br>character varying (100) | total_revenue<br>numeric |
|---|---|---|
| 1 | Sub-Saharan Africa | 34958453406.17 |
| 2 | Europe | 34241150923.39 |
| 3 | Asia | 19293401219.82 |
| 4 | Middle East and North Africa | 16921412794.52 |
| 5 | Central America and the Caribbean | 14553730165.29 |
| 6 | Australia and Oceania | 10701522223.73 |
| 7 | North America | 2937002333.49 |

```sql
--Q3 - What is the total revenue by region?
SELECT
    region,
    SUM(total_revenue) AS total_revenue
FROM ecomart
GROUP BY region
ORDER BY total_revenue DESC;
```

Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 77 msec.
7 rows affected.
```

**F4 - Apply optimization techniques to improve the run time of your queries from part F3, providing output results via a screenshot.**

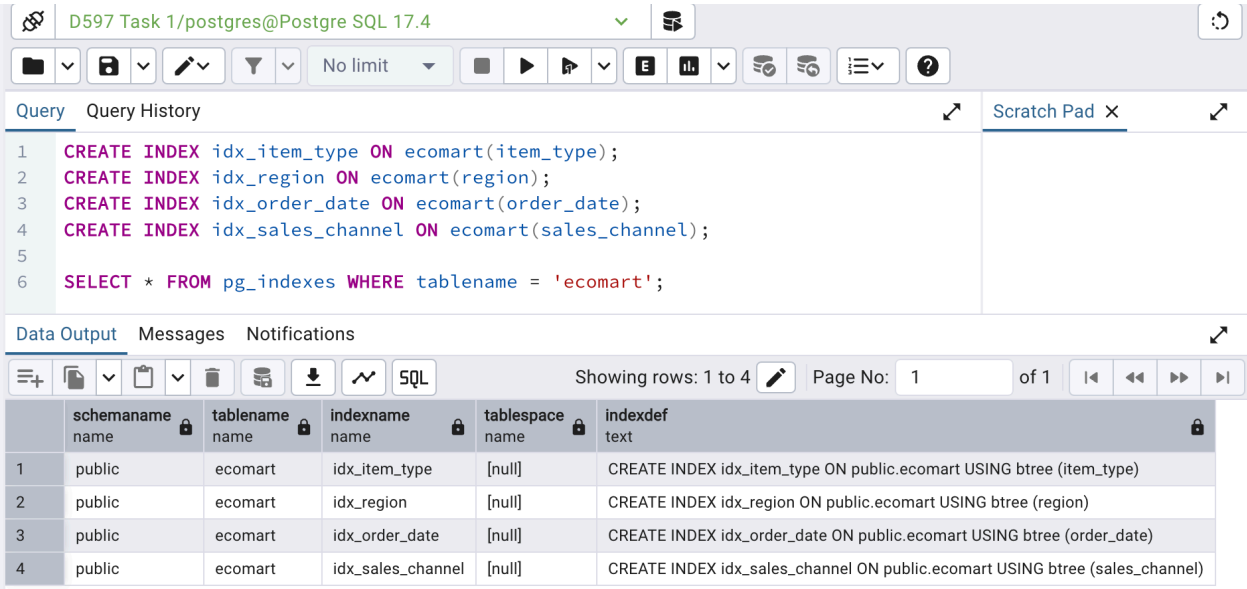
1. Creating indexes for the frequently used columns (item types, regions, order dates, sales channel)

Query | Query History

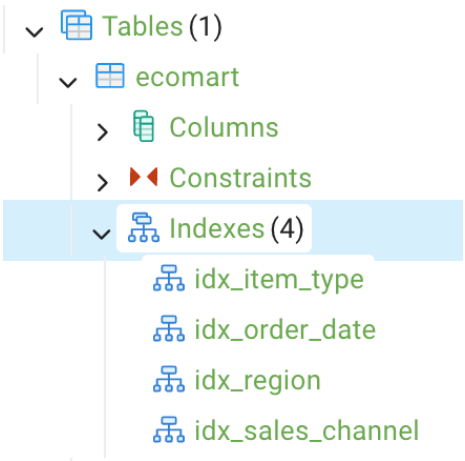
```
1  --F4
2  --Create Indexes for frequently used columns
3  CREATE INDEX idx_item_type ON ecomart(item_type);
4  CREATE INDEX idx_region ON ecomart(region);
5  CREATE INDEX idx_order_date ON ecomart(order_date);
6  CREATE INDEX idx_sales_channel ON ecomart(sales_channel);
7
8  SELECT * FROM pg_indexes WHERE tablename = 'ecomart';
9
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 243 msec.
```

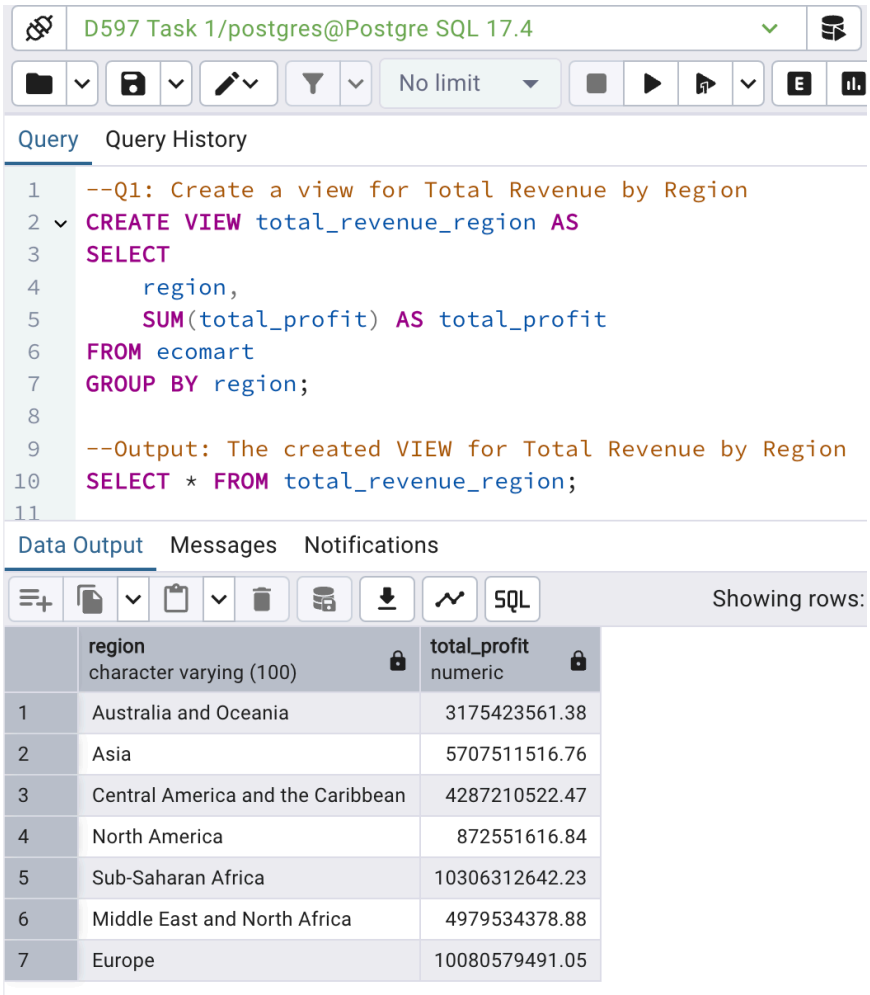



| | schemaname name | tablename name | indexname name | tablespace name | indexdef text |
|---|---|---|---|---|---|
| 1 | public | ecomart | idx_item_type | [null] | CREATE INDEX idx_item_type ON public.ecomart USING btree (item_type) |
| 2 | public | ecomart | idx_region | [null] | CREATE INDEX idx_region ON public.ecomart USING btree (region) |
| 3 | public | ecomart | idx_order_date | [null] | CREATE INDEX idx_order_date ON public.ecomart USING btree (order_date) |
| 4 | public | ecomart | idx_sales_channel | [null] | CREATE INDEX idx_sales_channel ON public.ecomart USING btree (sales_channel) |




2. Creating VIEWs to store summary tables for Total Revenue by Region, Top Profit Products, Avg Delivery Time by Sales Channel

Q1- Create a view for Total Revenue by Region




| | region character varying (100) | total_profit numeric |
|---|---|---|
| 1 | Australia and Oceania | 3175423561.38 |
| 2 | Asia | 5707511516.76 |
| 3 | Central America and the Caribbean | 4287210522.47 |
| 4 | North America | 872551616.84 |
| 5 | Sub-Saharan Africa | 10306312642.23 |
| 6 | Middle East and North Africa | 4979534378.88 |
| 7 | Europe | 10080579491.05 |

Before Optimization (Q1 - Total Revenue by Region):

```
23  --Before Optimization / No indexes or views applied yet
24  SELECT region, SUM(total_revenue) AS total_revenue
25  FROM ecomart
26  GROUP BY region
27  ORDER BY total_revenue DESC;
28
```

Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 90 msec.
7 rows affected.
```

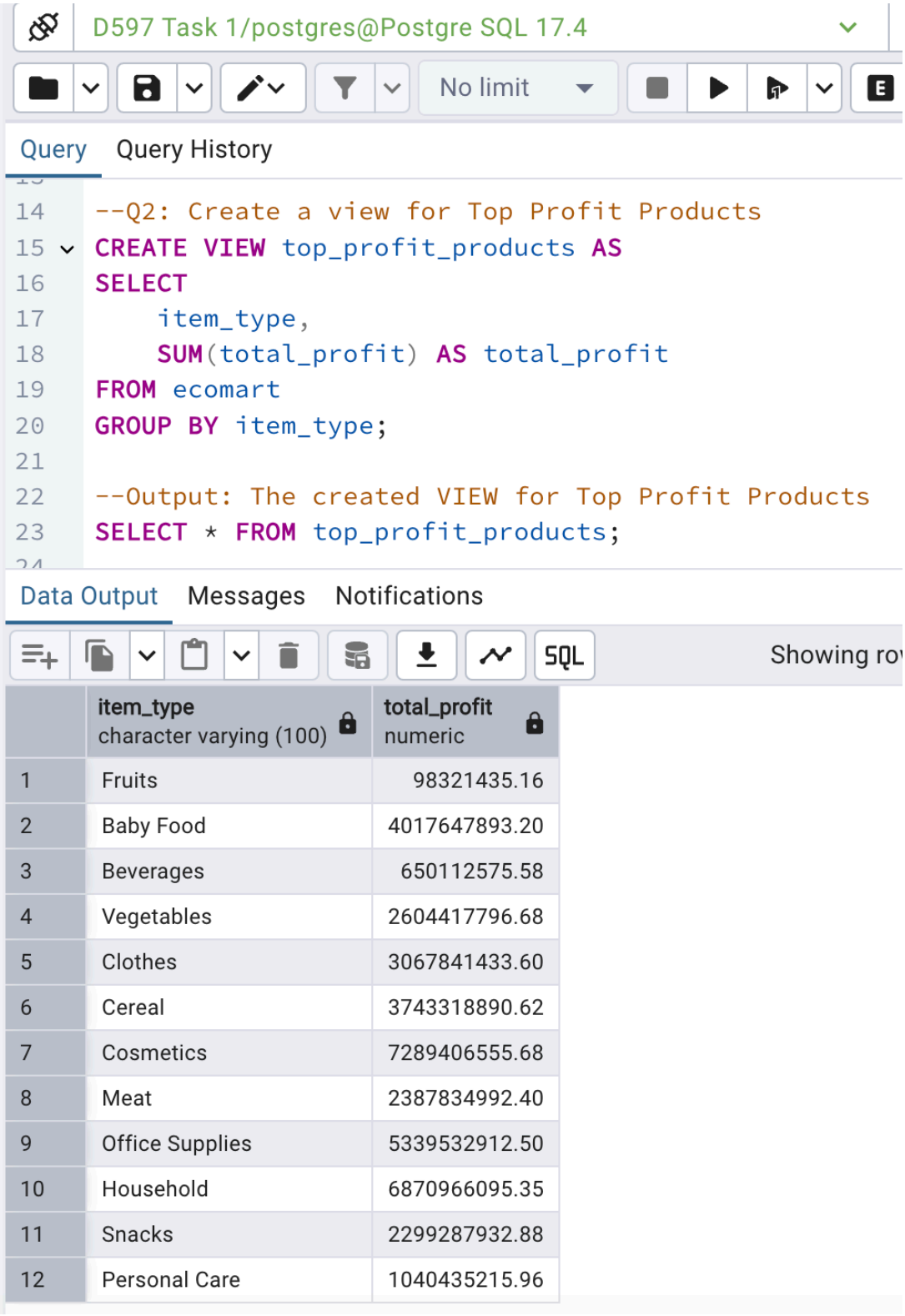
After Optimization (Q1 - Total Revenue by Region):

```
20  --Output: The created VIEW for Total Revenue by Region
21  SELECT * FROM total_revenue_region;
```

Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 85 msec.
7 rows affected.
```

Q2 – Create a view for Top Profit Products




| | item_type character varying (100) | total_profit numeric |
|---|---|---|
| 1 | Fruits | 98321435.16 |
| 2 | Baby Food | 4017647893.20 |
| 3 | Beverages | 650112575.58 |
| 4 | Vegetables | 2604417796.68 |
| 5 | Clothes | 3067841433.60 |
| 6 | Cereal | 3743318890.62 |
| 7 | Cosmetics | 7289406555.68 |
| 8 | Meat | 2387834992.40 |
| 9 | Office Supplies | 5339532912.50 |
| 10 | Household | 6870966095.35 |
| 11 | Snacks | 2299287932.88 |
| 12 | Personal Care | 1040435215.96 |

Before Optimization (Q2 - Top Profit Products):

```
43  --Before Optimization / No indexes or views applied yet
44  SELECT item_type, SUM(total_profit) AS total_profit
45  FROM ecomart
46  GROUP BY item_type
47  ORDER BY total_profit DESC
```

Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 113 msec.
12 rows affected.
```

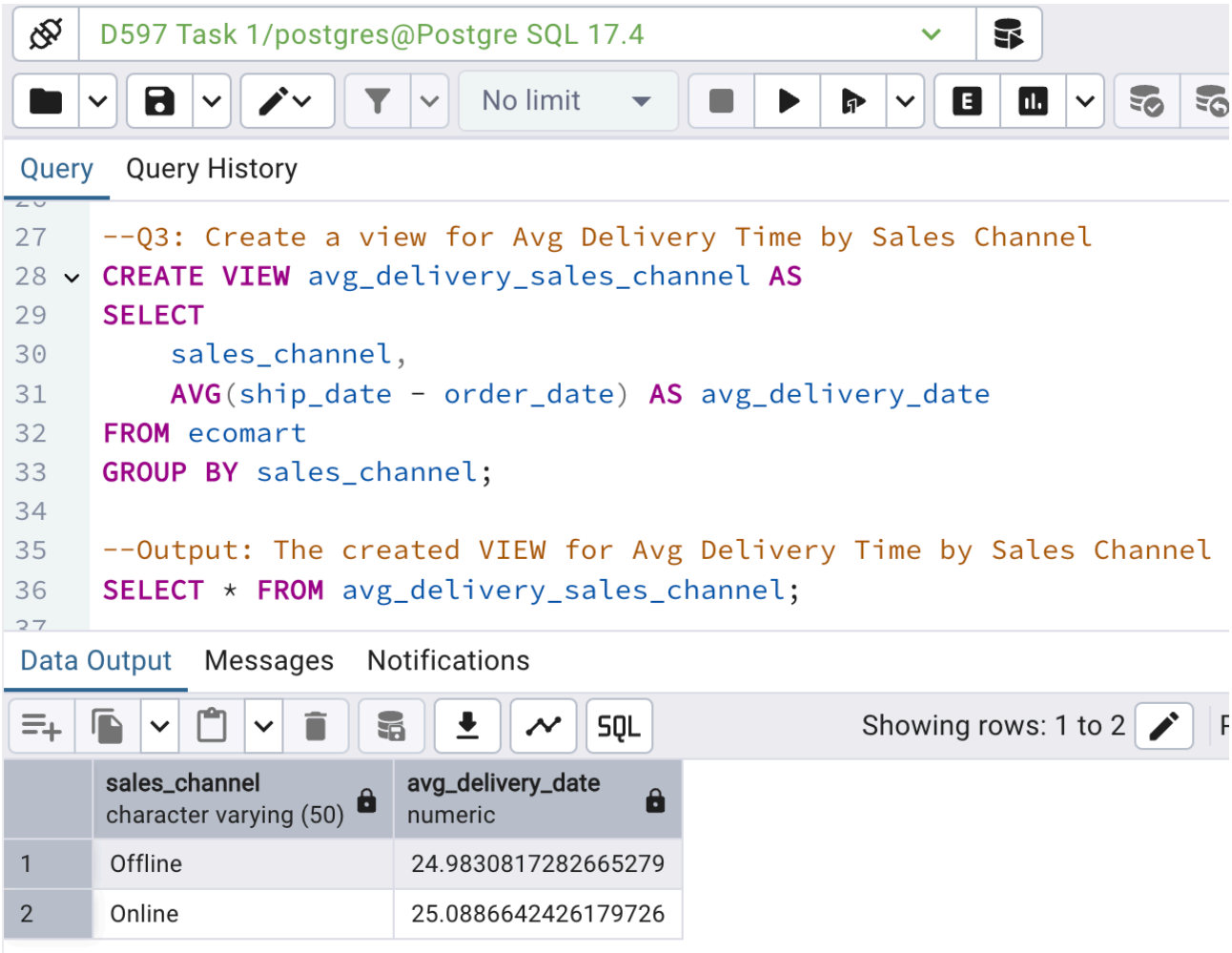
After Optimization (Q2 - Top Profit Products):

```
40  --Output: The created VIEW for Top Profit Products
41  SELECT * FROM top_profit_products;
42
```

Data Output | Messages | Notifications

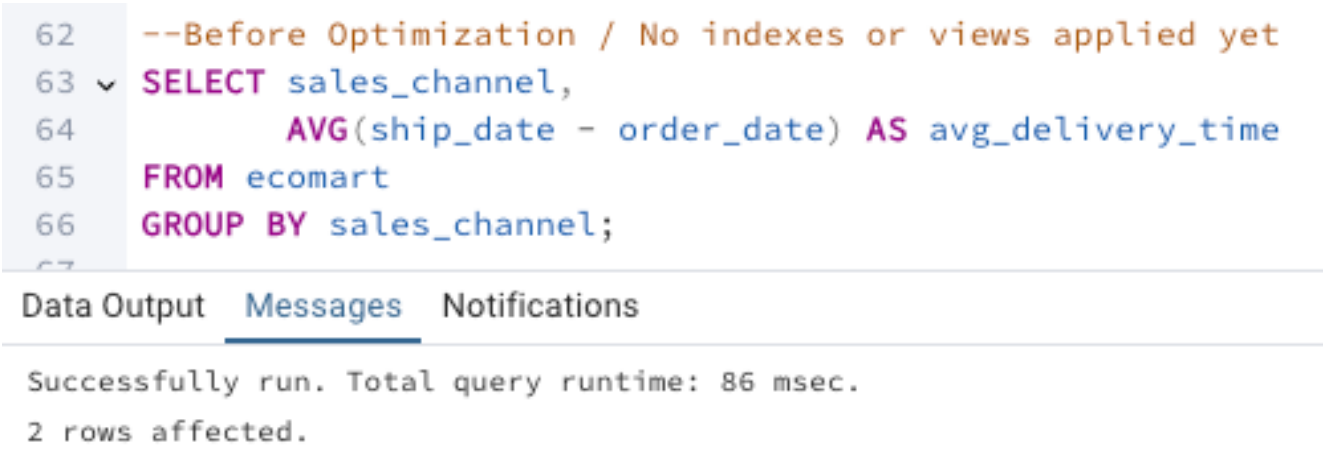
```
Successfully run. Total query runtime: 90 msec.
12 rows affected.
```

Q3 – Create a view for Avg Delivery Time by Sales Channel




| | sales_channel character varying (50) | avg_delivery_date numeric |
|---|---|---|
| 1 | Offline | 24.9830817282665279 |
| 2 | Online | 25.0886642426179726 |

Before Optimization (Q3 – Avg Delivery Time by Sales Channel):



After Optimization (Q3 – Avg Delivery Time by Sales Channel):

```
59  --Output: The created VIEW for Avg Delivery Time by Sales Channel
60  SELECT * FROM avg_delivery_sales_channel;
61
```

Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 77 msec.
2 rows affected.
```

**Part 3: Presentation**
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3bbaee3a-35a0-4df7-a9b4-b2f601285033

**H: Sources**
No external sources were referenced throughout this submission.