

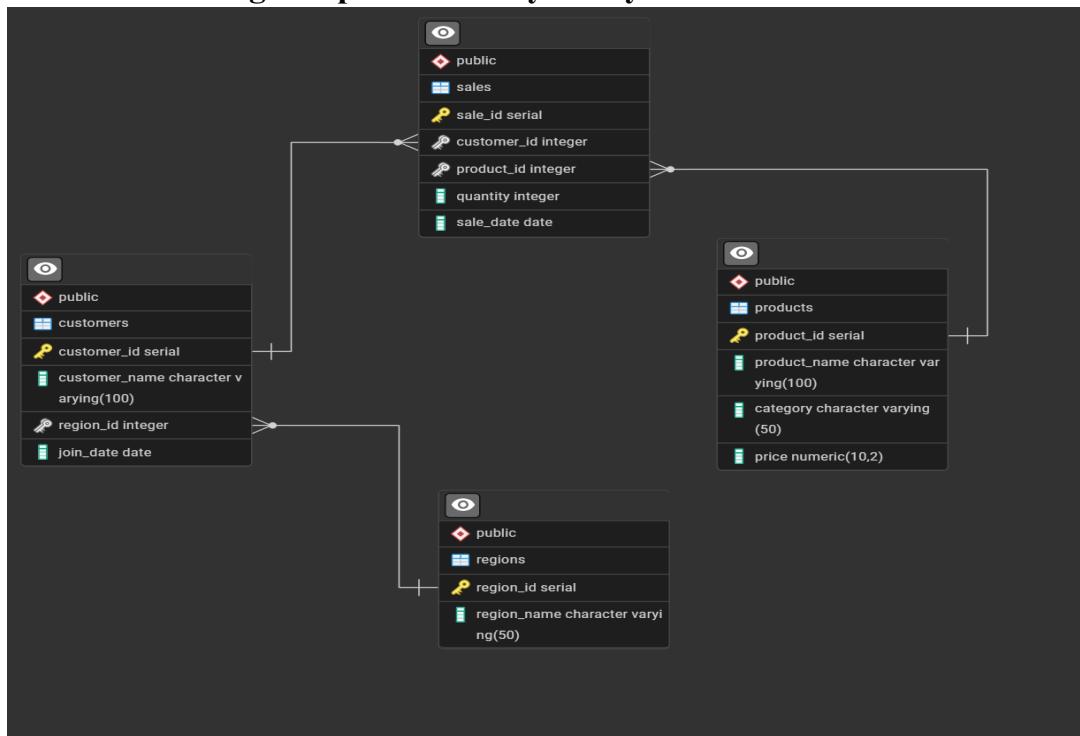
LagosExpress-Grocery FMCG Case Study – PostgreSQL Capstone Project (An Hypothetical Company)

Insights to the Analysis

Background:

LagosExpress-Grocery is an FMCG company that sells consumer goods (beverages, snacks, and toiletries) across various regions in Nigeria. The company wants to analyze its sales performance, customer distribution, and product profitability using PostgreSQL. They maintain 4 main tables: **customers, products, sales, and regions**.

ERD For our LagosExpress Grocery Analysis



Problem Questions and Solutions

Q1. Use a JOIN to find the total amount each customer spent.

SELECT

```
c.customer_name,  
SUM(p.price * s.quantity) AS total_spent  
FROM sales s  
JOIN customers c  
ON s.customer_id = c.customer_id  
JOIN products p  
ON s.product_id = p.product_id  
GROUP BY c.customer_name  
ORDER BY total_spent DESC;
```

Code:

The screenshot shows the pgAdmin 4 interface with a connection to 'LagosExpress_Grocery/postgres@PostgreSQL 17'. The query editor contains the following SQL code:

```
1 SELECT
2     c.customer_name,
3     SUM(p.price * s.quantity) AS total_spent
4 FROM
5     sales s
6 JOIN
7     customers c
8 ON s.customer_id = c.customer_id
9 JOIN
10    products p
11 ON s.product_id = p.product_id
12 GROUP BY
13     c.customer_name
14 ORDER BY
15     total_spent DESC;
```

The results pane shows the output of the query:

	customer_name	total_spent
1	Mary Johnson	3500.00

Total rows: 5 | Query complete 00:00:00.434 | CRL

Code Result:

The screenshot shows the pgAdmin 4 interface with a connection to 'LagosExpress_Grocery/postgres@PostgreSQL 17'. The query editor contains the same SQL code as the previous screenshot.

The results pane shows the output of the query with five rows:

	customer_name	total_spent
1	Mary Johnson	3500.00
2	Musa Ibrahim	3500.00
3	Chinedu Okafor	2050.00
4	Grace Udo	1800.00
5	Fatima Bello	1200.00

Total rows: 5 | Query complete 00:00:00.434 | CRL

Insight:

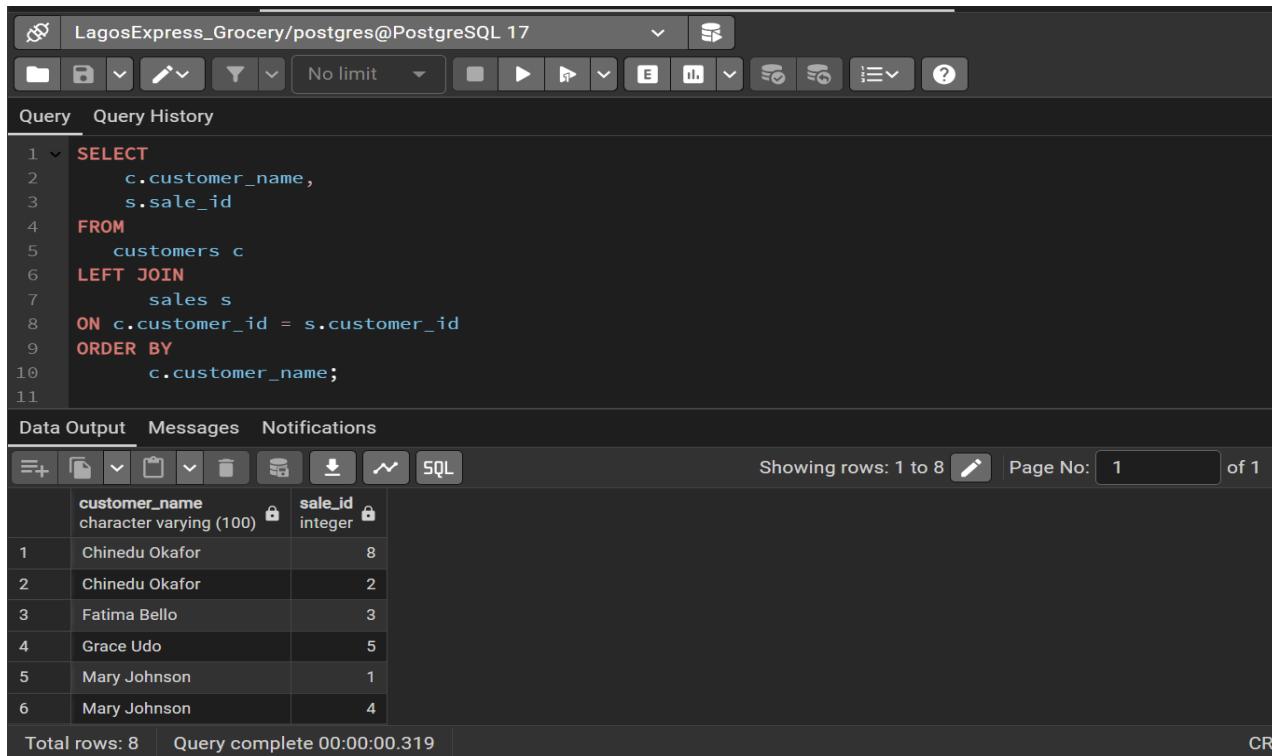
- Mary Johnson spent ₦3,500 (Spent the highest)
- Fatima Bello spent ₦1200.00 (Spent the lowest)
- Customers' total spending helps SwiftGrocer identify high-value clients.

Q2. Use a LEFT JOIN to find all customers, even those who haven't made any purchase.

SELECT

```
c.customer_name,  
s.sale_id  
FROM customers c  
LEFT JOIN sales s  
ON c.customer_id = s.customer_id  
ORDER BY c.customer_name;
```

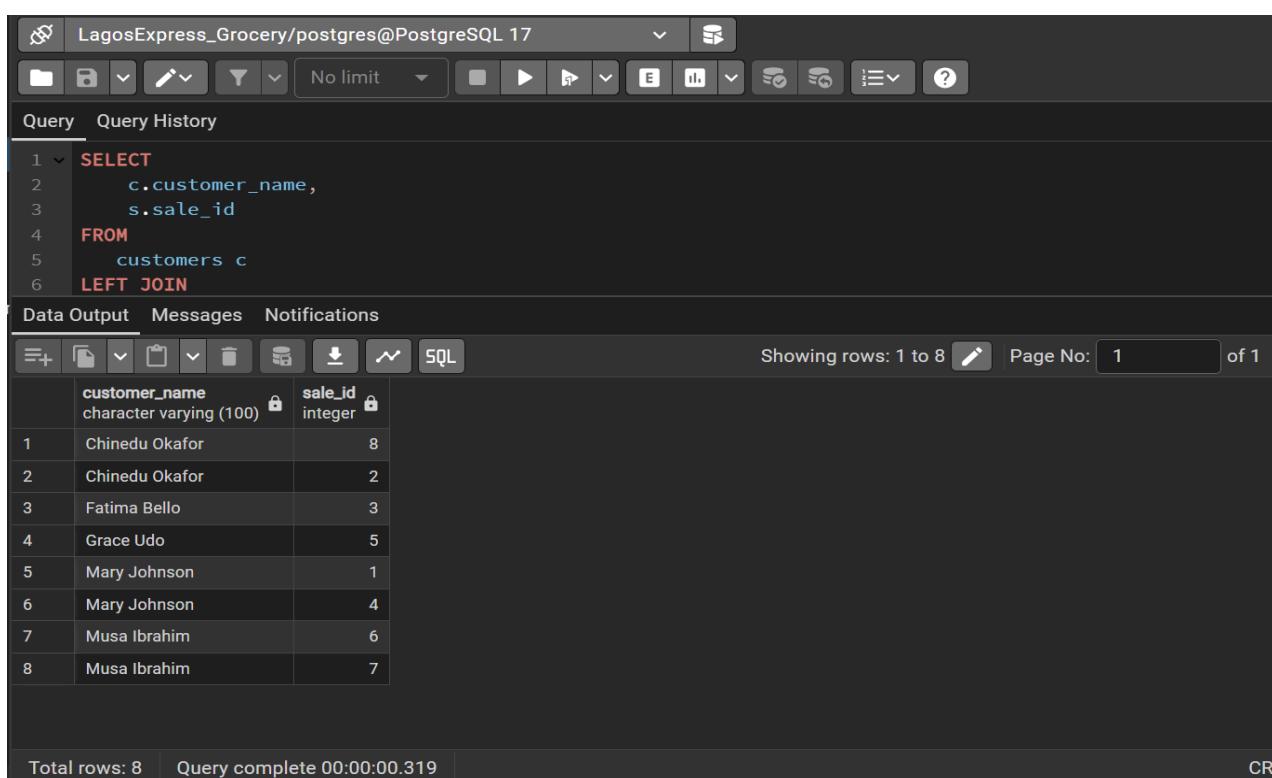
Code:



The screenshot shows a PostgreSQL database client interface. The query editor contains the SQL code for a LEFT JOIN. The results pane displays a table with two columns: customer_name and sale_id. The data shows six rows, with the last two rows being for 'Mary Johnson' which have null values in the sale_id column. The status bar at the bottom indicates 'Query complete 00:00:00.319'.

	customer_name	sale_id
1	Chinedu Okafor	8
2	Chinedu Okafor	2
3	Fatima Bello	3
4	Grace Udo	5
5	Mary Johnson	1
6	Mary Johnson	4

Code Result:



The screenshot shows a PostgreSQL database client interface. The query editor contains the same SQL code as the previous screenshot. The results pane displays a table with two columns: customer_name and sale_id. The data now includes an additional row for 'Musa Ibrahim' with a sale_id of 6. The status bar at the bottom indicates 'Query complete 00:00:00.319'.

	customer_name	sale_id
1	Chinedu Okafor	8
2	Chinedu Okafor	2
3	Fatima Bello	3
4	Grace Udo	5
5	Mary Johnson	1
6	Mary Johnson	4
7	Musa Ibrahim	6
8	Musa Ibrahim	7

Insight:

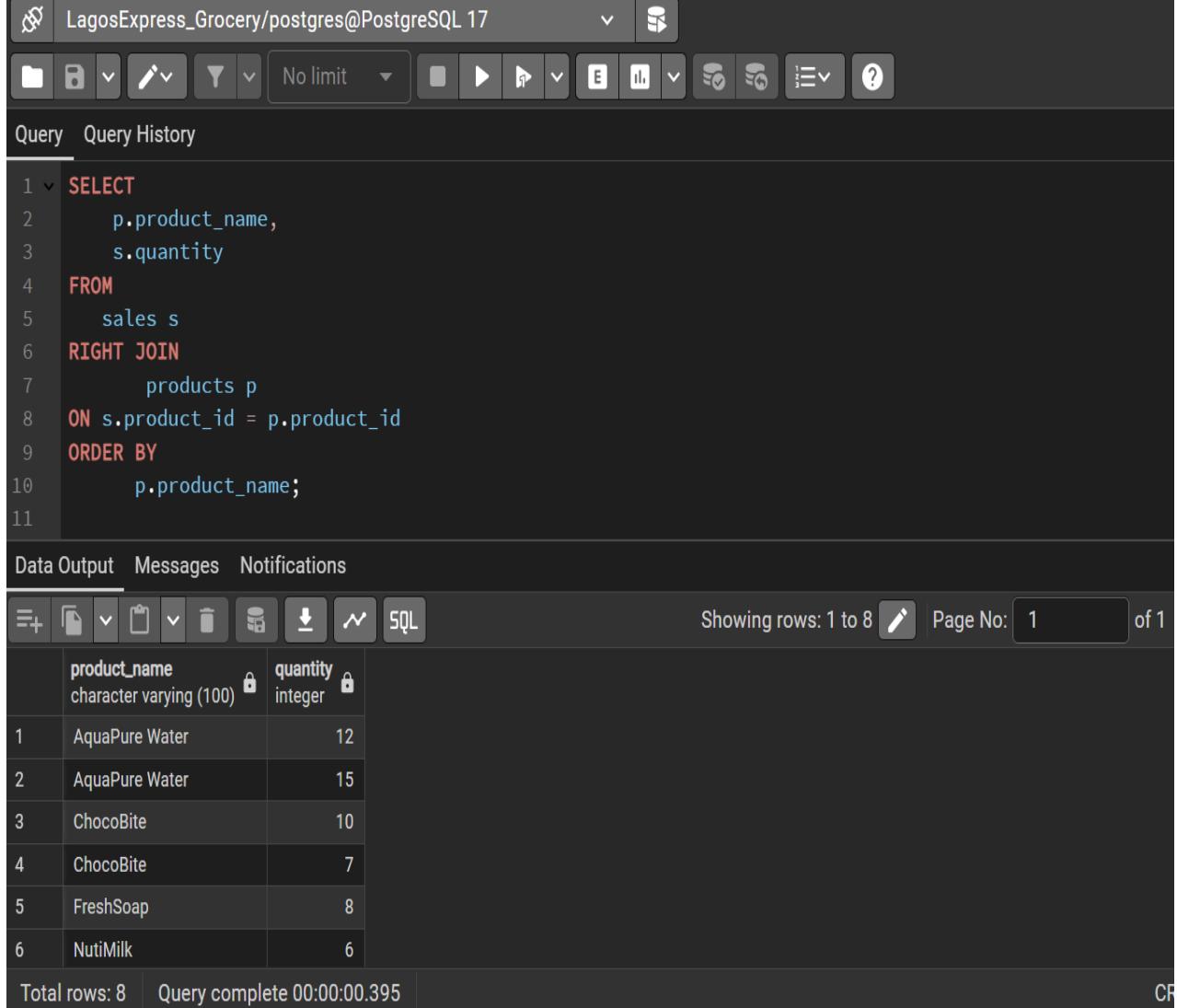
- If any **sale_id** is **NULL**, that customer hasn't made a purchase. Helps identify inactive customers.
- According to the result all customers made a purchase from the LagosExpress Grocery

Q3. Use a RIGHT JOIN to display all products and their sales (even if not sold).

SELECT

```
p.product_name,  
s.quantity  
FROM sales s  
RIGHT JOIN products p  
ON s.product_id = p.product_id  
ORDER BY p.product_name;
```

Code:



The screenshot shows the DBeaver SQL editor interface. The top bar displays the connection information: 'LagosExpress_Grocery/postgres@PostgreSQL 17'. Below the toolbar, there are tabs for 'Query' (which is selected) and 'Query History'. The main code area contains the following SQL query:

```
1 SELECT  
2     p.product_name,  
3     s.quantity  
4 FROM  
5     sales s  
6 RIGHT JOIN  
7     products p  
8 ON s.product_id = p.product_id  
9 ORDER BY  
10    p.product_name;  
11
```

Below the code, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, showing the results of the query in a table format:

	product_name	quantity
1	AquaPure Water	12
2	AquaPure Water	15
3	ChocoBite	10
4	ChocoBite	7
5	FreshSoap	8
6	NutiMilk	6

At the bottom of the results pane, it says 'Total rows: 8' and 'Query complete 00:00:00.395'. The status bar at the bottom right shows 'CR'.

Code Result:

The screenshot shows the pgAdmin 4 interface with a query editor and a results grid. The query editor contains the following SQL code:

```
1 SELECT
2     p.product_name,
3     s.quantity
4 FROM
5     sales s
6 RIGHT JOIN
7     products p
```

The results grid displays the following data:

	product_name	quantity
1	AquaPure Water	12
2	AquaPure Water	15
3	ChocoBite	10
4	ChocoBite	7
5	FreshSoap	8
6	NutMilk	6
7	Sparkle Cola	10
8	Sparkle Cola	5

Total rows: 8 | Query complete 00:00:00.395 | CRL

Insight:

- If any **quantity** is **NULL**, that product hasn't been sold. This is useful for stock control.
- At least one of every product in the LagosExpress Grocery was sold

Q4. Use AGGREGATION to find total sales revenue per category.

SELECT

```
p.category,  
SUM(p.price * s.quantity) AS total_revenue  
FROM sales s  
JOIN products p ON s.product_id = p.product_id  
GROUP BY p.category  
ORDER BY total_revenue DESC;
```

Code and Result:

The screenshot shows a DBeaver interface with a SQL editor and a results grid.

SQL Editor:

```
1 SELECT
2     p.category,
3     SUM(p.price * s.quantity) AS total_revenue
4 FROM
5     sales s
6 JOIN
7     products p
8 ON s.product_id = p.product_id
9 GROUP BY
10    p.category
11 ORDER BY
12    total_revenue DESC;
```

Results Grid:

	category	total_revenue
1	Beverages	7500.00
2	Snacks	2550.00
3	Toiletries	2000.00

Total rows: 3 | Query complete 00:00:00.395 | CRL

Insight:

- The beverages category led in sales.
- Management can focus marketing on best-selling categories.

Q5. Use DATE FUNCTIONS to find monthly sales totals for 2025.

SELECT

```
DATE_TRUNC('month', sale_date) AS month,
SUM(p.price * s.quantity) AS monthly_revenue
FROM sales s
JOIN products p
ON s.product_id = p.product_id
WHERE sale_date BETWEEN '2025-01-01' AND '2025-12-31'
GROUP BY month
ORDER BY monthly_revenue DESC;
```

Code:

LagosExpress_Grocery/postgres@PostgreSQL 17

```
1 SELECT
2     DATE_TRUNC('month', sale_date) AS month,
3     SUM(p.price * s.quantity) AS monthly_revenue
4 FROM
5     sales s
6 JOIN
7     products p
8 ON s.product_id = p.product_id
9 WHERE
10    sale_date
11        BETWEEN '2025-01-01' AND '2025-12-31'
12 GROUP BY
13    month
14 ORDER BY
15    monthly_revenue DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 4 | Page No: 1 of 1

	month	monthly_revenue
1	2025-03-01 00:00:00+01	3800.00
2	2025-04-01 00:00:00+01	2550.00
3	2025-02-01 00:00:00+01	2000.00

Total rows: 4 | Query complete 00:00:00.275 | CRL

Code Result:

LagosExpress_Grocery/postgres@PostgreSQL 17

```
1 SELECT
2     DATE_TRUNC('month', sale_date) AS month,
3     SUM(p.price * s.quantity) AS monthly_revenue
4 FROM
5     sales s
6 JOIN
7     products p
8 ON s.product_id = p.product_id
9 WHERE
10    sale_date
11        BETWEEN '2025-01-01' AND '2025-12-31'
12 GROUP BY
13    month
```

Data Output Messages Notifications

Showing rows: 1 to 4 | Page No: 1 of 1

	month	monthly_revenue
1	2025-03-01 00:00:00+01	3800.00
2	2025-04-01 00:00:00+01	2550.00
3	2025-02-01 00:00:00+01	2000.00
4	2025-01-01 00:00:00+01	1200.00

Total rows: 4 | Query complete 00:00:00.275 | CRL

Insight:

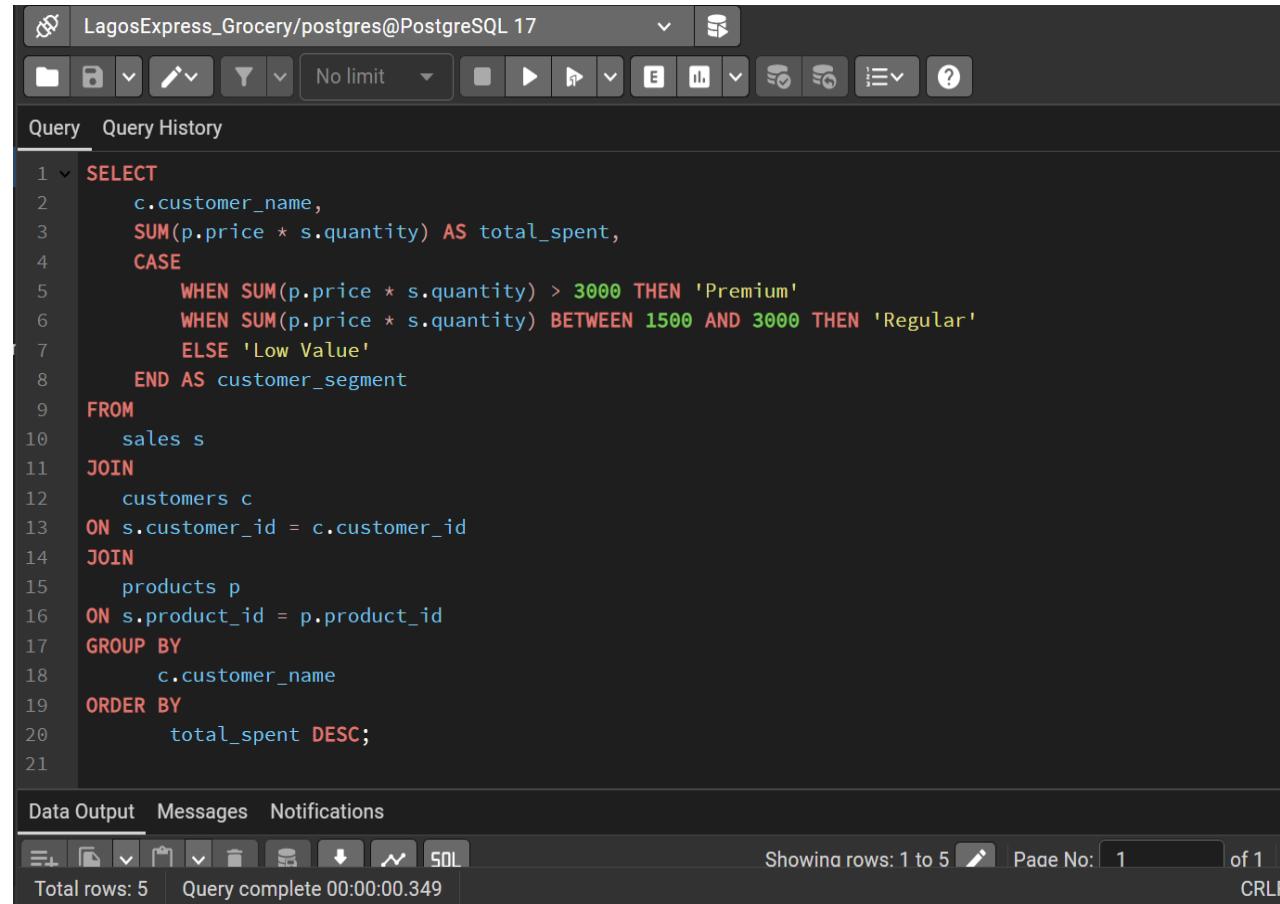
Shows sales growth or increase and decline month-by-month. This helps identify seasonal patterns.
LagosExpress Grocery experienced a decline in sales between March 2025 to April 2025

Q6. Use CASE STATEMENT to classify customers based on spending.

SELECT

```
c.customer_name,  
SUM(p.price * s.quantity) AS total_spent,  
CASE  
    WHEN SUM(p.price * s.quantity) > 3000 THEN 'Premium'  
    WHEN SUM(p.price * s.quantity) BETWEEN 1500 AND 3000 THEN  
        'Regular'  
    ELSE 'Low Value'  
END AS customer_segment  
FROM sales s  
JOIN customers c  
ON s.customer_id = c.customer_id  
JOIN products p  
ON s.product_id = p.product_id  
GROUP BY c.customer_name  
ORDER BY total_spent DESC;
```

Code:



The screenshot shows a PostgreSQL client interface with the following details:

- Connection:** LagosExpress_Grocery/postgres@PostgreSQL 17
- Toolbar:** Includes icons for file, edit, search, and navigation.
- Query Bar:** Shows "No limit".
- Buttons:** Execute, Refresh, Stop, etc.
- Query Editor:** Displays the SQL code from the previous block, numbered 1 to 21.
- Data Output:** Shows the results of the query:

customer_name	total_spent	customer_segment
John Doe	1200	Low Value
Jane Smith	1800	Regular
Mike Johnson	2500	Regular
Sarah Williams	3200	Premium

- Messages:** No messages displayed.
- Notifications:** No notifications displayed.
- Bottom Status:** Total rows: 5 | Query complete 00:00:00.349 | Showing rows: 1 to 5 | Page No: 1 of 1 | CRLF

Code Result:

The screenshot shows a pgAdmin interface with a query editor and a results grid. The query is a SELECT statement that joins the 'customers' and 'sales' tables to calculate total spent and classify customers into segments based on their spending. The results grid displays 5 rows of data with columns for customer name, total spent, and segment.

```
1  SELECT
2      c.customer_name,
3          SUM(p.price * s.quantity) AS total_spent,
4          CASE
5              WHEN SUM(p.price * s.quantity) > 3000 THEN 'Premium'
6              WHEN SUM(p.price * s.quantity) BETWEEN 1500 AND 3000 THEN 'Regular'
7              ELSE 'Low Value'
8          END AS customer_segment
9  FROM
10         sales s
11     JOIN
```

	customer_name	total_spent	customer_segment
1	Mary Johnson	3500.00	Premium
2	Musa Ibrahim	3500.00	Premium
3	Chinedu Okafor	2050.00	Regular
4	Grace Udo	1800.00	Regular
5	Fatima Bello	1200.00	Low Value

Total rows: 5 | Query complete 00:00:00.349 | CRLF

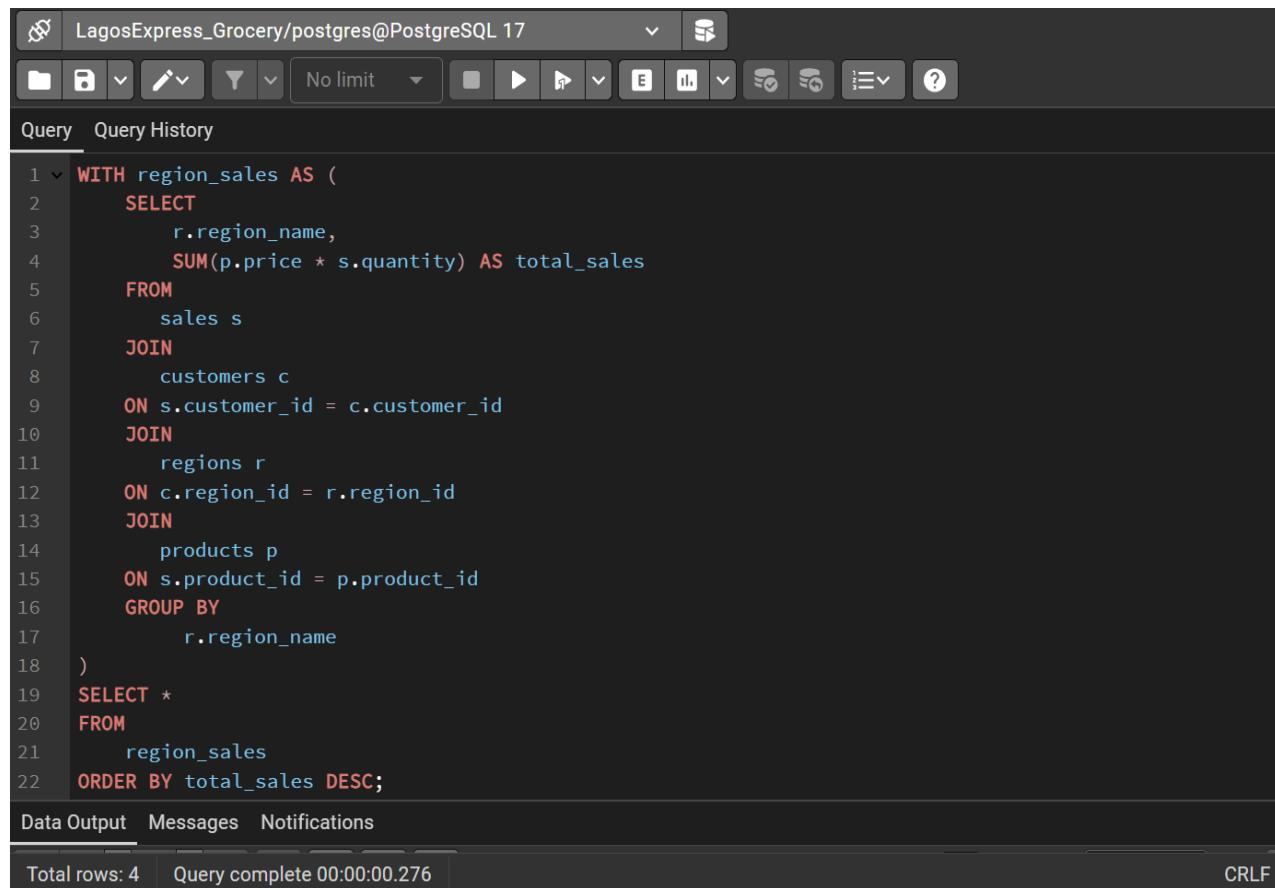
Insight:

- Classifies customers for loyalty programs
- *Mary Johnson was a Premium Customer*

Q7. Use WITH STATEMENT to find the best-performing region.

```
WITH region_sales AS (
    SELECT
        r.region_name,
        SUM(p.price * s.quantity) AS total_sales
    FROM sales s
    JOIN customers c
    ON s.customer_id = c.customer_id
    JOIN regions r
    ON c.region_id = r.region_id
    JOIN products p
    ON s.product_id = p.product_id
    GROUP BY r.region_name
)
SELECT *
FROM region_sales
ORDER BY total_sales DESC;
```

Code:



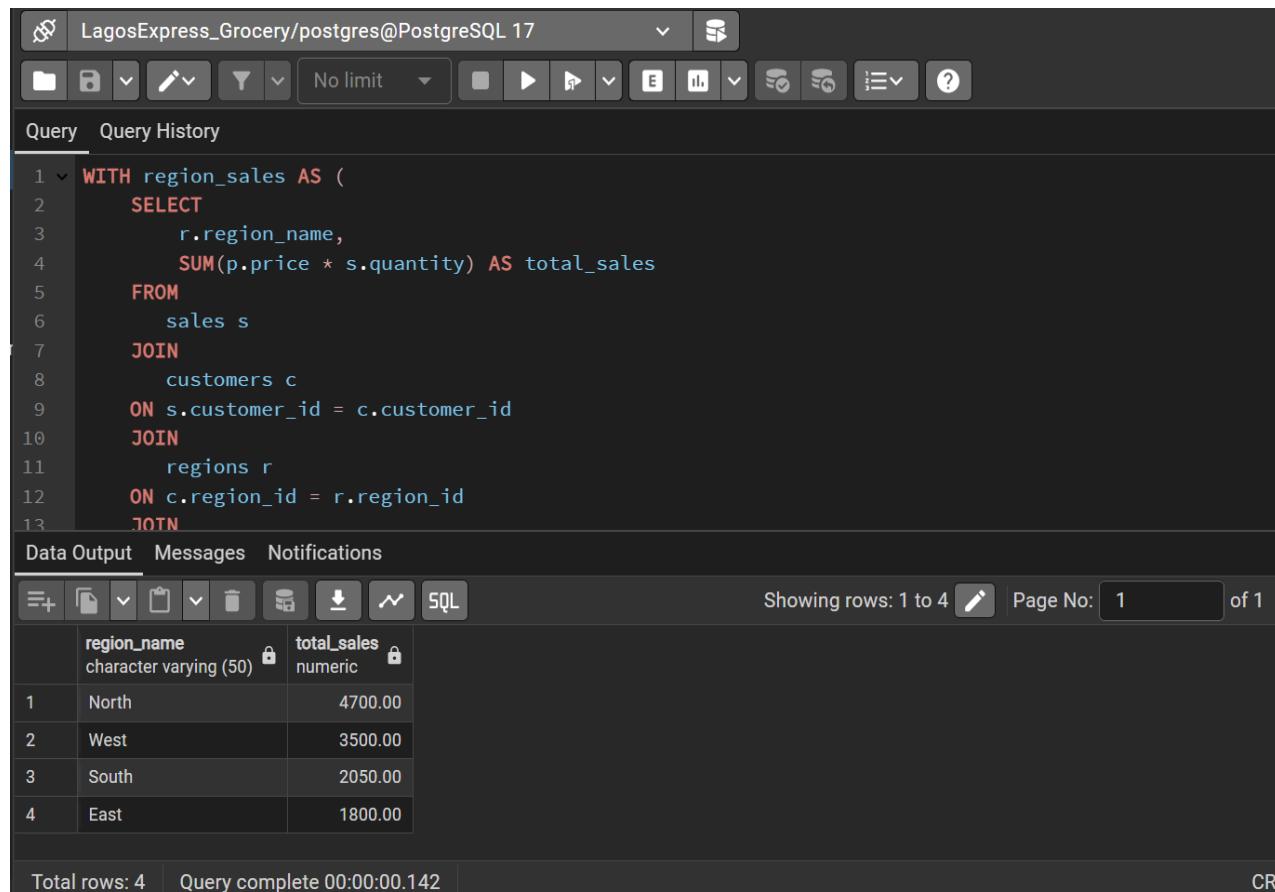
```
1 WITH region_sales AS (
2     SELECT
3         r.region_name,
4         SUM(p.price * s.quantity) AS total_sales
5     FROM
6         sales s
7     JOIN
8         customers c
9     ON s.customer_id = c.customer_id
10    JOIN
11        regions r
12    ON c.region_id = r.region_id
13    JOIN
14        products p
15    ON s.product_id = p.product_id
16    GROUP BY
17        r.region_name
18 )
19 SELECT *
20 FROM
21     region_sales
22 ORDER BY total_sales DESC;
```

Data Output Messages Notifications

Total rows: 4 Query complete 00:00:00.276

CRLF

Code Result:



```
1 WITH region_sales AS (
2     SELECT
3         r.region_name,
4         SUM(p.price * s.quantity) AS total_sales
5     FROM
6         sales s
7     JOIN
8         customers c
9     ON s.customer_id = c.customer_id
10    JOIN
11        regions r
12    ON c.region_id = r.region_id
13    JOIN
```

Data Output Messages Notifications

Showing rows: 1 to 4 Page No: 1 of 1

	region_name character varying (50)	total_sales numeric
1	North	4700.00
2	West	3500.00
3	South	2050.00
4	East	1800.00

Total rows: 4 Query complete 00:00:00.142

CRL

Insight:

- Identifies which region contributes most to total revenue. This supports resource allocation.
- The best performing region to our sales was the North. Which is the Northern part of Lagos

Q8. Use UNION to list both product categories and customer regions (as a combined business domain list).

```
SELECT category AS name FROM products
UNION
SELECT region_name AS name FROM regions
ORDER BY name;
```

Code:

The screenshot shows the pgAdmin interface with a query editor and a results grid. The query editor contains the code provided above. The results grid shows a single column named 'name' with four rows: 'Beverages', 'East', 'North', and 'Snacks'. The bottom status bar indicates 'Total rows: 7' and 'Query complete 00:00:00.228'.

name
character varying (50)
Beverages
East
North
Snacks

Code Result:

The screenshot shows the pgAdmin interface with a query editor and a results grid. The query editor contains the code provided above. The results grid shows a single column named 'name' with seven rows: 'Beverages', 'East', 'North', 'Snacks', 'South', 'Toiletries', and 'West'. The bottom status bar indicates 'Total rows: 7' and 'Query complete 00:00:00.228'.

name
character varying (50)
Beverages
East
North
Snacks
South
Toiletries
West

Insight:

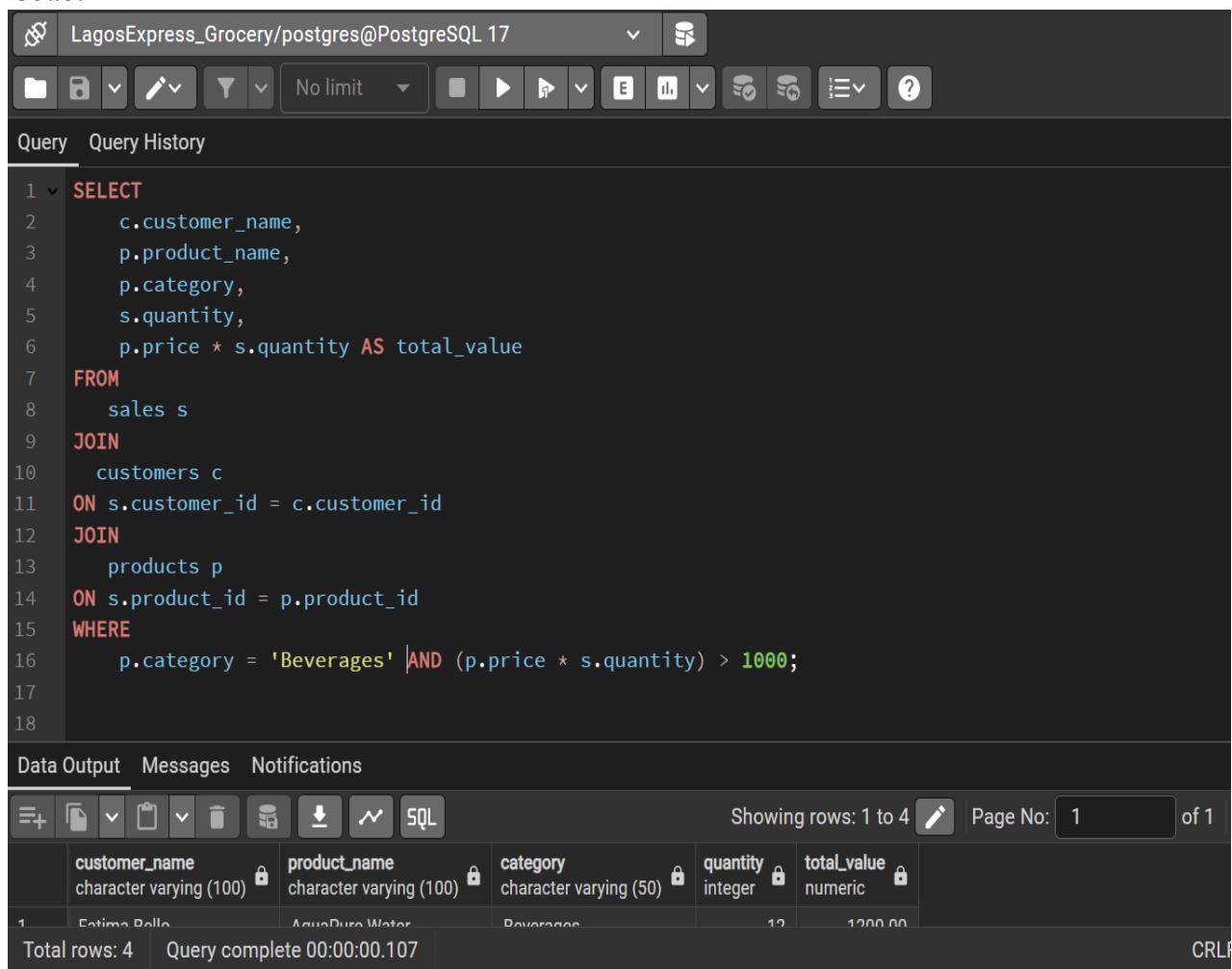
Demonstrates merging of different datasets. This is useful for creating dropdown menus or combined reports.

Q9. Use Logical Operators (AND/OR) to find sales of beverages above ₦1,000.

SELECT

```
c.customer_name,  
p.product_name,  
p.category,  
s.quantity,  
p.price * s.quantity AS total_value  
FROM sales s  
JOIN customers c  
ON s.customer_id = c.customer_id  
JOIN products p  
ON s.product_id = p.product_id  
WHERE p.category = 'Beverages' AND (p.price * s.quantity) > 1000;
```

Code:



The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar says "LagosExpress_Grocery/postgres@PostgreSQL 17". The toolbar has various icons for file operations, search, and database management. Below the toolbar is a menu bar with "Query" and "Query History". The main area contains the SQL code for the query. The code is numbered from 1 to 18. Lines 16 and 17 show the WHERE clause with a logical AND operator. The bottom part of the window shows the results of the query execution. The results table has columns: customer_name, product_name, category, quantity, and total_value. There is one row returned, with values: Estima Ballo, Aquaduro Water, Beverages, 12, and 1200.00. The status bar at the bottom shows "Total rows: 4" and "Query complete 00:00:00.107".

```
1  SELECT  
2      c.customer_name,  
3      p.product_name,  
4      p.category,  
5      s.quantity,  
6      p.price * s.quantity AS total_value  
7  FROM  
8      sales s  
9  JOIN  
10     customers c  
11  ON s.customer_id = c.customer_id  
12  JOIN  
13     products p  
14  ON s.product_id = p.product_id  
15  WHERE  
16      p.category = 'Beverages' AND (p.price * s.quantity) > 1000;  
17  
18
```

	customer_name	product_name	category	quantity	total_value
1	Estima Ballo	Aquaduro Water	Beverages	12	1200.00

Total rows: 4 Query complete 00:00:00.107

Code: Result:

The screenshot shows a PostgreSQL database client interface. The top bar displays the connection information: 'LagosExpress_Grocery/postgres@PostgreSQL 17'. Below the connection bar are various toolbar icons for managing databases, tables, and queries. The main area is divided into two tabs: 'Query' (selected) and 'Query History'. The 'Query' tab contains the following SQL code:

```
1  SELECT
2      c.customer_name,
3      p.product_name,
4      p.category,
5      s.quantity,
6      p.price * s.quantity AS total_value
7  FROM
8      sales s
9  JOIN
10     customers c
11    ON s.customer_id = c.customer_id
12   JOIN
```

Below the code, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is selected and displays a table with the results of the query. The table has the following columns:

	customer_name character varying (100)	product_name character varying (100)	category character varying (50)	quantity integer	total_value numeric
1	Fatima Bello	AquaPure Water	Beverages	12	1200.00
2	Grace Udo	NutiMilk	Beverages	6	1800.00
3	Musa Ibrahim	Sparkle Cola	Beverages	10	2000.00
4	Musa Ibrahim	AquaPure Water	Beverages	15	1500.00

At the bottom of the interface, there are status messages: 'Total rows: 4', 'Query complete 00:00:00.107', and 'CRLF'.

Insight:

Helps spot large beverage orders. This is good for upselling premium drinks.