

Project: Ecommerce Sales & Customer Insights

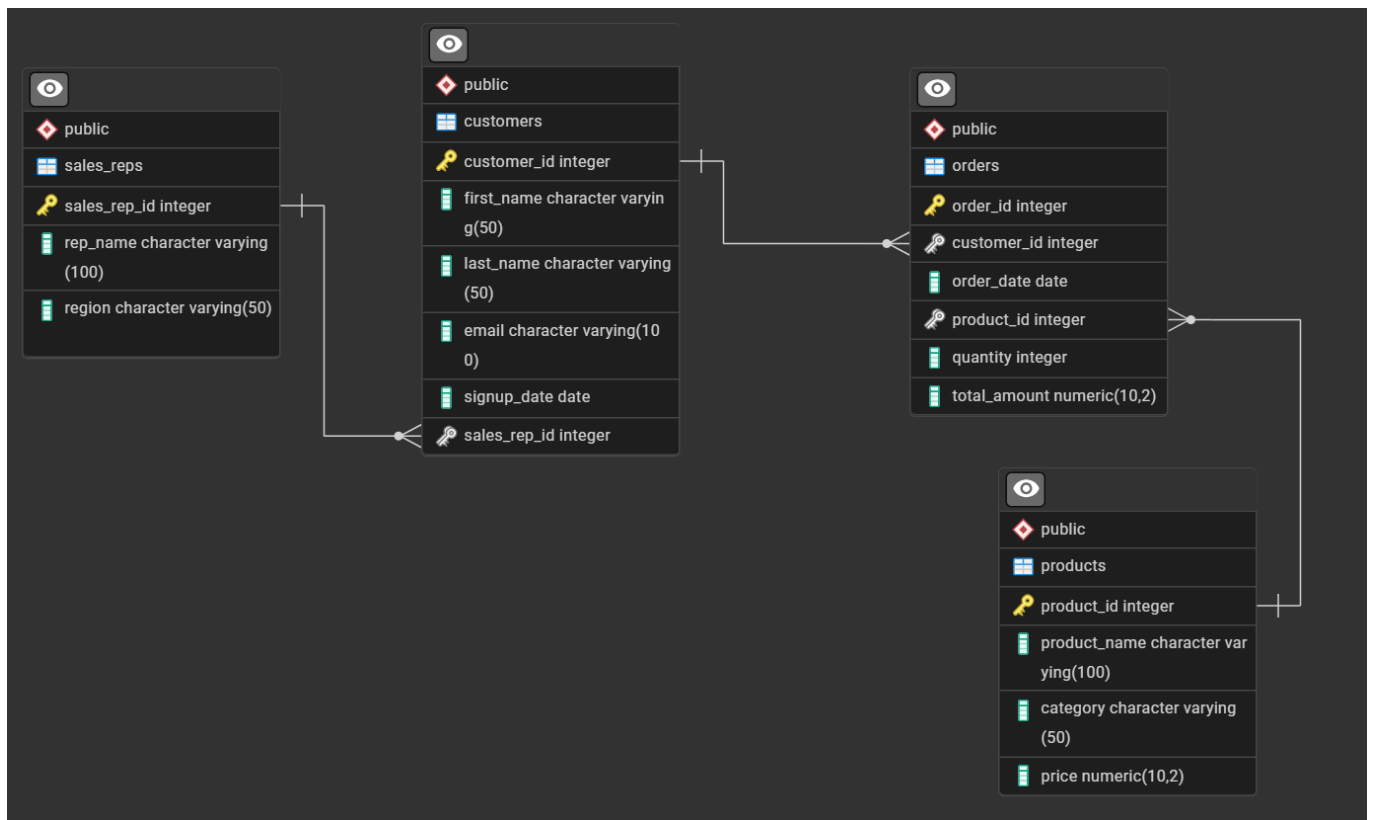
Project Description:

SQL-based project analyzing e-commerce sales and customer data to uncover actionable insights. The analysis identifies top-selling product categories, high-spending customers and regions, and segments customers by purchasing power (High, Medium, and Low Value).

What was the Scenario of the project?

I have been hired as a Data Analyst for MegaMart Online, a global e-commerce platform. My task was to analyze customer behavior, sales trends, and product performance to provide actionable insights for the marketing and operations teams.

The ERD Diagram for the Project



About the dataset:

1. customers — Information about registered customers.
2. orders — Records of orders placed.
3. products — Information about the items sold.
4. sales_reps — List of sales representatives managing certain customers

Project Questions:

- Which product categories generate the most revenue?
- Who are the highest-spending customers and which regions manage them?
- How can customers be segmented based on purchasing power?

The Codes from the Analysis

1. **INNER JOIN:** Retrieve all orders with customer names, product names, and sales rep names, and the highest grossing product.

The screenshot shows a PostgreSQL query editor with the following SQL query:

```
1 SELECT
2   o.order_id, o.order_date, c.customer_id,
3   c.first_name || ' ' || c.last_name, p.product_name,
4   s.rep_name
5 FROM products p
6 INNER JOIN orders o
7 ON p.product_id = o.product_id
8 INNER JOIN customers c
9 ON o.customer_id = c.customer_id
10 INNER JOIN sales_reps s
11 ON c.sales_rep_id = s.sales_rep_id
12 ORDER BY o.order_date;
```

The query results are displayed in a table with the following columns: order_id, order_date, customer_id, ?column?, product_name, and rep_name. The results show 5 rows of data.

order_id	order_date	customer_id	?column?	product_name	rep_name	
1	5222	2024-08-13	15	Christopher Munoz	Course Max	Steven Nichols
2	5226	2024-08-15	29	Michele Farmer	Business Max	Brian Riley
3	5039	2024-08-16	215	Mark Tucker	Gun Max	Mitchell Padilla
4	5036	2024-08-17	251	Sarah Lane	Hair Pro	Jessica Cole
5	5132	2024-08-17	281	Wesley Hernandez	Summer Plus	Taylor Castillo

Total rows: 400 Query complete 00:00:00.189

Why I used INNER JOIN:

I used an INNER JOIN to combine the **products** table with **orders**, **customers**, and **sales_reps**, so that instead of just IDs, I could see the actual customer names, product names, and sales rep names. This gives a complete picture of each order and makes the data more useful for reporting and analysis, like tracking top products, top customers, or sales performance by rep.

2. **LEFT JOIN:** List all customers with their order details, including customers who have never

The screenshot shows a PostgreSQL query editor with the following SQL query:

```
1 SELECT o.order_id, order_date, c.customer_id,
2        c.first_name, c.last_name,
3        c.signup_date, c.email
4 FROM orders o
5 LEFT JOIN customers c
6 ON o.customer_id = c.customer_id
7 ORDER BY o.order_date, o.order_date NULLS LAST;
```

The query results are displayed in a table with the following columns: order_id, order_date, customer_id, first_name, last_name, signup_date, and email. The results show 10 rows of data.

order_id	order_date	customer_id	first_name	last_name	signup_date	email	
393	5238	2025-08-05	13	Heather	Gonzalez	2024-03-02	scott38@yahoo.com
394	5112	2025-08-05	156	Patrick	Hayes	2024-02-04	bhenry@yahoo.com
395	5240	2025-08-07	383	Charles	Moreno	2023-08-26	tamarascott@hotmail.com
396	5376	2025-08-08	113	David	Stone	2024-02-29	justin42@wilson-brown.com
397	5379	2025-08-09	176	Rhonda	Price	2024-10-21	ysharp@stewart.com
398	5076	2025-08-11	215	Mark	Tucker	2024-11-09	youngsandra@hotmail.com
399	5338	2025-08-12	4	Gary	Brown	2024-09-10	reyesjeffery@rodriguez.com
400	5229	2025-08-12	270	Tyler	Hughes	2024-09-13	dennis41@trevino.net

Total rows: 400 Query complete 00:00:00.176

Why I used LEFT JOIN:

I used a **LEFT JOIN** to make sure I could list **all customers** along with their order details, while also including those customers who have **never placed an order**. This way, I don't lose any customers in my analysis. I can see both active and inactive ones, which is useful for understanding customer engagement and identifying those who may need follow-ups or promotions.

3. **RIGHT JOIN:** Show all products and the orders placed for them (including products never sold).

Servers (1)

- PostgreSQL 17
 - Databases (3)
 - E-commerce
 - Cast
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas
 - Subscriptions
 - parch_and_posey
 - postgres
 - Login/Group Roles
 - Tablespaces

E-commerce / postgres@PostgreSQL 17

Query

Query History

```
1 SELECT p.product_id, p.product_name, p.category, p.price,
2       o.order_id, o.order_date, o.quantity, o.total_amount
3 FROM products p
4 RIGHT JOIN orders o
5 ON p.product_id = o.product_id
6 ORDER BY o.order_date, o.order_date NULLS LAST;
```

Data Output

Messages

Notifications

Showing rows: 1 to 400

Page No: 1 of 1

	product_id	product_name	category	price	order_id	order_date	quantity	total_amount
	integer	character varying (100)	character varying (50)	numeric (10,2)	integer	date	integer	numeric (10,2)
1	539	Course Max	Electronics	36.71	5222	2024-08-13	4	146.84
2	612	Business Max	Clothing	54.86	5226	2024-08-15	4	219.44
3	574	Gun Max	Accessories	498.65	5039	2024-08-16	4	1994.60
4	367	Hair Pro	Clothing	223.00	5036	2024-08-17	2	446.00
5	308	Summer Plus	Electronics	323.24	5132	2024-08-17	2	646.48
6	333	To X	Sports	30.90	5140	2024-08-18	1	30.90
7	694	Close Pro	Home Appliances	434.44	5033	2024-08-18	5	2172.20
8	474	Police Lite	Clothing	254.21	5241	2024-08-19	1	254.21
9	576	Teach Plus	Sports	91.37	5117	2024-08-19	4	365.48

Total rows: 400 Query complete 00:00:00.152 CRLF Ln 1, Col 32

Why I used RIGHT JOIN:

I used a **RIGHT JOIN** to display **all products** along with any orders linked to them, while also making sure products that have **never been sold** still appear. This is important for analyzing product performance, because it highlights not only the top-selling items but also those with little or no sales, which can guide inventory and marketing decisions.

4. Find customers who **signed up in 2023 AND** have placed at least one order worth more than \$200

Servers (1)

- PostgreSQL 17
 - Databases (3)
 - E-commerce
 - Cast
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas
 - Subscriptions
 - parch_and_posey
 - postgres
 - Login/Group Roles
 - Tablespaces

E-commerce / postgres@PostgreSQL 17

Query

Query History

```
1 SELECT c.customer_id, c.first_name, c.last_name,
2       c.email, c.signup_date, o.total_amount
3 FROM orders o
4 JOIN customers c
5 ON o.customer_id = c.customer_id
6 WHERE c.signup_date BETWEEN '2023-01-01' AND '2023-12-31' AND o.total_amount > 200
7 ORDER BY c.customer_id;
```

Data Output

Messages

Notifications

Showing rows: 1 to 63

Page No: 1 of 1

	customer_id	first_name	last_name	email	signup_date	total_amount
	integer	character varying (50)	character varying (50)	character varying (100)	date	numeric (10,2)
1	8	Megan	Banks	danawalker@woods.com	2023-08-28	299.84
2	8	Megan	Banks	danawalker@woods.com	2023-08-28	403.60
3	12	Monica	Cabrera	taylorkeith@gmail.com	2023-08-31	534.36
4	20	Jonathan	Martin	emilymcbride@ward.net	2023-11-12	365.48
5	21	Joseph	Elliott	robertskimberly@morrow.com	2023-09-16	1754.90
6	39	Kevin	Marsh	robinsoncorey@hotmail.com	2023-10-29	410.16
7	39	Kevin	Marsh	robinsoncorey@hotmail.com	2023-10-29	850.62
8	69	Christina	Turner	carpenteredward@cisneros.c...	2023-08-31	1250.43

Total rows: 63 Query complete 00:00:00.152 CRLF Ln 1, Col 49

Why I wrote this query:

I used this query to find customers who **signed up in 2023** and also made at least one **high-value order above \$200**. Filtering by signup date ensures I'm only looking at new customers, and the order amount condition helps me identify those who quickly became valuable. This is useful for analyzing the quality of recent signups and targeting early high-spending customers for loyalty programs.

5. List orders where the product category is "Electronics" **OR** "Accessories".

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT o.order_id, p.product_id, p.category, o.quantity
2 FROM products p
3 JOIN orders o
4 ON p.product_id = o.product_id
5 WHERE p.category = 'Electronics' OR p.category = 'Accessories'
6 ORDER BY o.quantity DESC;
```

The query results are displayed in a table with the following columns: order_id, product_id, category, and quantity. The results show 8 rows of data, sorted by quantity in descending order.

order_id	product_id	category	quantity
1	5318	Electronics	5
2	5091	Accessories	5
3	5194	Electronics	5
4	5049	Accessories	5
5	5343	Accessories	5
6	5107	Electronics	5
7	5337	Accessories	5
8	5061	Accessories	5

Total rows: 152 Query complete 00:00:00.159

Why I wrote this query:

I used this query to list orders where the products fall under **Electronics or Accessories**, because these categories are key revenue drivers. Filtering this way makes it easier to track performance, compare demand between the two, and identify which items sell the most within these popular categories

6. Show customers whose sales reps are **NOT** assigned to the "North Region".

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT s.sales_rep_id, s.rep_name, s.region, c.customer_id, c.first_name, c.last_name
2 FROM sales_reps s
3 JOIN customers c
4 ON s.sales_rep_id = c.sales_rep_id
5 WHERE s.region NOT IN ('North Region')
6 ORDER BY c.customer_id;
```

The query results are displayed in a table with the following columns: sales_rep_id, rep_name, region, customer_id, first_name, and last_name. The results show 10 rows of data, sorted by customer_id in ascending order.

sales_rep_id	rep_name	region	customer_id	first_name	last_name
267	Christopher Mayo	South Region	1	Benjamin	Fuentes
424	Anthony Santos	South Region	2	Brian	Waters
321	Alexander Wilson	South Region	3	George	Gonzalez
381	Adrian Price	West Region	4	Gary	Brown
188	Randy Henderson	West Region	5	James	Harrell
160	Bryan Pearson	West Region	6	Ann	Oliver
477	Hannah Price	South Region	7	Yolanda	Villarreal
199	Mary Rowland MD	West Region	9	Gregory	Kelley
220	Sara Jackson	West Region	10	Susan	Matthews

Total rows: 293 Query complete 00:00:00.139

Why I wrote this query:

I used this query to find customers whose sales reps are **not assigned to the North Region**. This helps compare customer distribution and sales activity across other regions, ensuring that analysis isn’t limited to just one area and making it easier to spot differences in performance between regions.

7. **DATE_PART**: Count how many orders were placed in each **month** of 2024.

The screenshot shows a PostgreSQL query editor with the following query:

```
1 SELECT DATE_PART('month', order_date) AS months, COUNT(*) AS order_count
2 FROM orders
3 GROUP BY 1
4 ORDER BY order_count DESC;
```

The query results are displayed in a table with two columns: **months** (double precision) and **order_count** (bigint). The results show the number of orders for each month of 2024, ordered by count in descending order.

months	order_count
1	43
2	40
3	40
4	37
5	35
6	35
7	35
8	33
9	29
10	26

Total rows: 12 Query complete 00:00:00.125 CRLF Ln 4, Col 27

Why I wrote this query:

I used **DATE_PART** to count how many orders were placed in each **month of 2024**. This breaks down sales activity month by month, making it easier to spot seasonal trends, peak periods, and months with low performance that may need attention.

8. **DATE_TRUNC**: Calculate total revenue for each **quarter** in 2024.

The screenshot shows a PostgreSQL query editor with the following query:

```
1 SELECT DATE_TRUNC('quarter', order_date) AS rev_quarter, SUM(total_amount) AS total_revenue
2 FROM orders
3 WHERE DATE_TRUNC('quarter', order_date) BETWEEN '2024-01-01' AND '2024-12-31'
4 GROUP BY 1
5 ORDER BY total_revenue DESC;
```

The query results are displayed in a table with two columns: **rev_quarter** (timestamp with time zone) and **total_revenue** (numeric). The results show the total revenue for each quarter of 2024, ordered by revenue in descending order.

rev_quarter	total_revenue
2024-10-01 00:00:00+01	80563.94
2024-07-01 00:00:00+01	40796.69

Total rows: 2 Query complete 00:00:00.133 CRLF Ln 4, Col 11

Why I wrote this query:

I used **DATE_TRUNC** to calculate the total revenue for each **quarter in 2024**. Looking at revenue by quarter gives a clearer view of broader sales performance, highlights growth patterns over time, and helps compare business results across different parts of the year.

9. Create a column **spending_category** to classify customers based on their total spending:

The screenshot shows a PostgreSQL query editor with a query that classifies customers into three spending categories: High Value, Medium Value, and Low Value. The query uses a CASE statement to categorize customers based on their total spending. The results are displayed in a table with columns: customer_id, first_name, last_name, total_spent, and spending_category.

```
1 SELECT c.customer_id, c.first_name, c.last_name, SUM(o.total_amount) total_spent,
2 CASE WHEN SUM(o.total_amount) > 1000 THEN 'High Value'
3 WHEN SUM(o.total_amount) BETWEEN 500 AND 1000 THEN 'Medium Value'
4 ELSE 'Low Value' END AS spending_category
5 FROM orders o
6 JOIN customers c
7 ON o.customer_id = c.customer_id
8 GROUP BY c.customer_id, c.first_name, c.last_name
9 ORDER BY total_spent DESC;
```

customer_id	first_name	last_name	total_spent	spending_category
1	129	Maria	6974.96	High Value
2	102	Andrea	4805.73	High Value
3	221	Todd	4314.44	High Value
4	214	Jared	4266.56	High Value
5	281	Wesley	3540.16	High Value
6	312	Dylan	3200.32	High Value
7	263	Richard	3062.52	High Value

Why I wrote this query:

I created a **spending_category** column using a CASE statement to classify customers based on their **total spending**. This makes it easy to separate customers into **High, Medium, and Low Value** groups, which is useful for targeted marketing, loyalty programs, and understanding which customer segments drive the most revenue.

10. Find all customers whose **total spending** is above the **average spending** of all customers.

The screenshot shows a PostgreSQL query editor with a query that finds customers whose total spending is above the average spending of all customers. The query uses a subquery to calculate the average spending and then compares each customer's total spending against it. The results are displayed in a table with columns: customer_id, first_name, last_name, and total_spent.

```
1 SELECT c.customer_id, c.first_name, c.last_name, SUM(o.total_amount) AS total_spent
2 FROM orders o
3 JOIN customers c
4 ON o.customer_id = c.customer_id
5 GROUP BY c.customer_id, c.first_name, c.last_name
6 HAVING SUM(o.total_amount) > (SELECT AVG(total_spent)
7 FROM (
8 SELECT customer_id, SUM(total_amount) AS total_spent
9 FROM orders
10 GROUP BY customer_id
11 ) sub)
12 ORDER BY total_spent DESC;
```

customer_id	first_name	last_name	total_spent
1	129	Maria	6974.96
2	102	Andrea	4805.73
3	221	Todd	4314.44
4	214	Jared	4266.56

Why I wrote this query:

I used a subquery to compare each customer's total spending against the **average spending of all customers**. This helps identify the **above-average spenders**, who are valuable for loyalty programs, upselling, or personalized offers since they contribute more than the typical customer.

INSIGHTS FROM MY PROJECT ANALYSIS

1. Top-Selling Categories

Analysis shows that **Accessories** and **Clothing** stand out as the key categories due to the large amount of revenue generated.

- **Accessories** dominate in terms of sales volume, making them the strongest revenue driver.
- **Clothing** and **Sports** products also contributed significantly, reinforcing their importance in overall sales performance.

2. Using INNER JOIN to Find Top-Grossing Products

I applied an INNER JOIN to connect the **products** table with **orders**, **customers**, and **sales_reps**.

- This allowed me to not only identify the **highest-grossing products** but also see **which customers purchased them** and **which sales reps managed those customers**, giving a complete view of product performance and customer reach.

3. High-Spending Customers & Regions

Using the CASE STATEMENT, I was able to identify the **top-spending customers** and the **regions** that manage them.

- Insight: The **East Region** consistently handles higher-value customers according to the top 5 high value customers in our company, suggesting either stronger customer relationships or greater purchasing power in that region.

Customer Segmentation (Based on Purchasing Power)

I categorized customers into three groups, which are the **High Value**, **Medium Value**, and **Low Value**, based on their total spending.

- This segmentation allows the company to design targeted strategies such as **discounts**, **loyalty rewards**, or **subscription plans** for different customer groups.
- Notably, the analysis revealed that the company has a relatively **large proportion of high-value customers**, which is a strong indicator of customer loyalty and spending potential.

IDEAS AND RECOMMENDATIONS FOR MORE SALES IMPROVEMENT

1. Double Down on Accessories & Clothing (Category Focus)

Since **Accessories** dominate in sales volume and **Clothing** contributes significantly to revenue, the company should:

- Expand product variety in these categories.
- Offer bundled deals (e.g., *"Buy Clothing + Accessory and get 10% off"*).
- Run targeted marketing campaigns highlighting best-sellers in these categories.

Benefit: Boosts sales in the strongest-performing categories while encouraging cross-sells.

2. Target High-Value Customers with Loyalty Programs

The analysis showed a **large proportion of high-value customers**. To retain and grow this segment:

- Introduce **exclusive loyalty rewards** (points, VIP discounts, early access to new products).
- Personalized offers via email or app notifications.

Benefit: Keeps big spenders engaged, increases repeat purchases, and reduces churn.

3. Strengthen Sales Strategy in the East Region

The **East Region** consistently manages higher-value customers, signaling strong purchasing power. The company should:

- Increase inventory allocation in this region.
- Invest in **regional promotions** (ads, influencer marketing, or local partnerships).
- Equip sales reps with additional support or incentives to grow accounts further.

Benefit: Maximizes revenue in a proven high-potential region.

4. Re-Engage Low & Medium Value Customers

Customer segmentation revealed that not all customers spend at the same level. To improve sales:

- Offer **discount vouchers** or “first repeat order” incentives for low-value customers.
- Upsell to medium-value customers with subscription plans (e.g., “*Subscribe & Save*” offers).
- Use remarketing ads to bring back inactive customers.

Benefit: Expands overall customer lifetime value by moving more customers into the medium/high-value brackets.