# Customers & Orders SQL Analysis

Real-world business questions answered with PostgreSQL

Lilla Katona

2025.05.09.

# Customer Orders Analysis Introduction

This SQL project demonstrates my ability to extract meaningful business insights from customer and order data. I used realistic datasets to write and execute queries that answer practical business questions related to customer segmentation, order behaviour, product performance, and more. Each section includes the question, SQL logic, and sample results to reflect real-world data analysis tasks.

## Business Questions

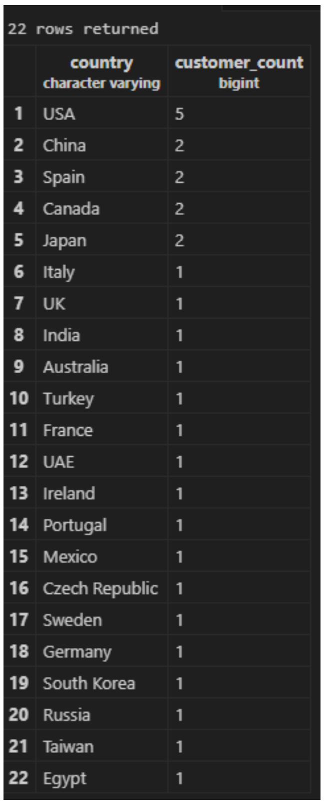
### 1. Which countries have the most registered customers?

**Purpose:** Understand geographic distribution of customers.

**Query:**

```
SELECT
  country,
  COUNT(*) AS customer_count
FROM customers
GROUP BY country
ORDER BY customer_count DESC;
```

**Result:**



22 rows returned

| | country<br>character varying | customer_count<br>bigint |
|---|---|---|
| 1 | USA | 5 |
| 2 | China | 2 |
| 3 | Spain | 2 |
| 4 | Canada | 2 |
| 5 | Japan | 2 |
| 6 | Italy | 1 |
| 7 | UK | 1 |
| 8 | India | 1 |
| 9 | Australia | 1 |
| 10 | Turkey | 1 |
| 11 | France | 1 |
| 12 | UAE | 1 |
| 13 | Ireland | 1 |
| 14 | Portugal | 1 |
| 15 | Mexico | 1 |
| 16 | Czech Republic | 1 |
| 17 | Sweden | 1 |
| 18 | Germany | 1 |
| 19 | South Korea | 1 |
| 20 | Russia | 1 |
| 21 | Taiwan | 1 |
| 22 | Egypt | 1 |

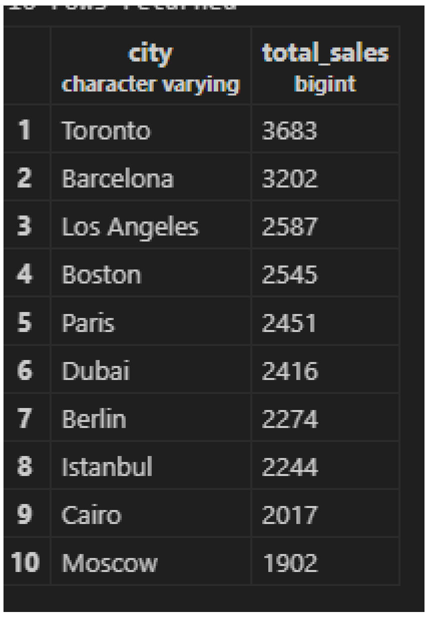
## 2. Which cities generate the highest total order amount?

**Purpose**: Spot top-performing cities.

**Query:**

```
SELECT
c.city,
SUM(CAST(o.order_amount as INT)) AS total_sales
FROM customers c
JOIN orders o
ON c.customer_id = o.customer_id
GROUP BY c.city
ORDER BY total_sales DESC
LIMIT 10;
```

**Result:**



| | city<br>character varying | total_sales<br>bigint |
|---|---|---|
| 1 | Toronto | 3683 |
| 2 | Barcelona | 3202 |
| 3 | Los Angeles | 2587 |
| 4 | Boston | 2545 |
| 5 | Paris | 2451 |
| 6 | Dubai | 2416 |
| 7 | Berlin | 2274 |
| 8 | Istanbul | 2244 |
| 9 | Cairo | 2017 |
| 10 | Moscow | 1902 |

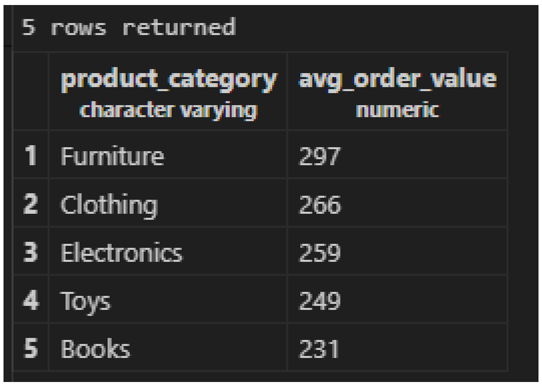
## 3. What is the average order value by product category?

**Purpose**: Compare performance across product categories.

**Query:**

SELECT
product_category,
ROUND(AVG(CAST(order_amount as int)),0) AS avg_order_value
FROM orders
GROUP BY product_category
ORDER BY avg_order_value DESC;

**Result:**



```
5 rows returned
```

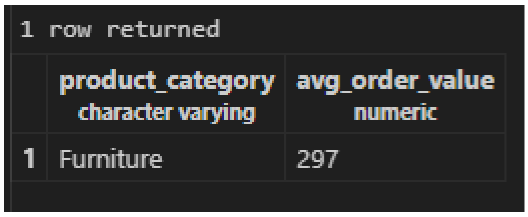
| | product_category<br>character varying | avg_order_value<br>numeric |
|---|---|---|
| 1 | Furniture | 297 |
| 2 | Clothing | 266 |
| 3 | Electronics | 259 |
| 4 | Toys | 249 |
| 5 | Books | 231 |

**Alternative Query**: Filtering for a specific product:

SELECT
product_category,
ROUND(AVG(CAST(order_amount as int)),0) AS avg_order_value
FROM orders
WHERE product_category = 'Furniture'
GROUP BY product_category
ORDER BY avg_order_value DESC;

**Result**

:



```
1 row returned
```

| | product_category<br>character varying | avg_order_value<br>numeric |
|---|---|---|
| 1 | Furniture | 297 |

## 4. How many customers registered each year?

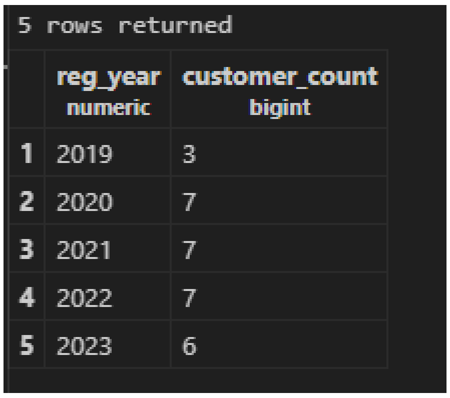
**Purpose:** Analyse customer acquisition trends.

**Query:**

SELECT
EXTRACT(YEAR FROM registration_date) AS reg_year,

COUNT(*) AS customer_count
FROM customers
GROUP BY reg_year
ORDER BY reg_year;

**Result:**



```
5 rows returned
```

| | reg_year<br>numeric | customer_count<br>bigint |
|---|---|---|
| 1 | 2019 | 3 |
| 2 | 2020 | 7 |
| 3 | 2021 | 7 |
| 4 | 2022 | 7 |
| 5 | 2023 | 6 |

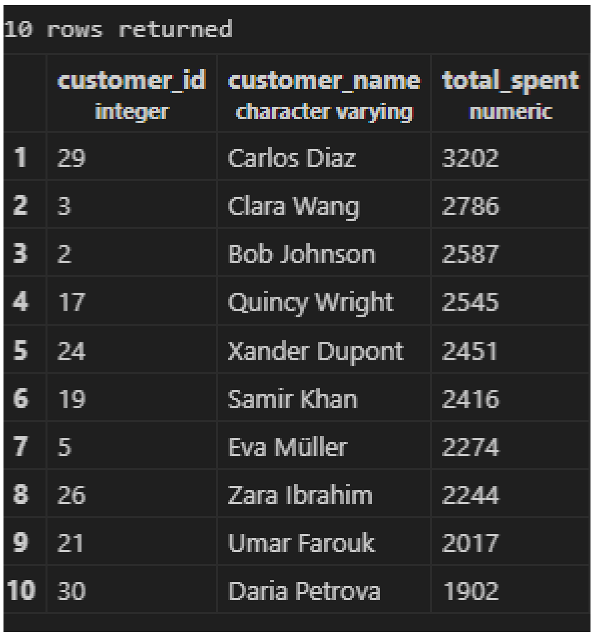
## 5. Which customers have spent the most overall?

**Purpose:** Identify top customers.

**Query:**

**SELECT**
c.customer_id,
c.customer_name,
ROUND(SUM(o.order_amount),0) AS total_spent
FROM customers c
JOIN orders o
ON c.customer_id = o.customer_id
GROUP BY 1,2
ORDER BY total_spent DESC
LIMIT 10;

**Result:**


10 rows returned

| | customer_id integer | customer_name character varying | total_spent numeric |
|---|---|---|---|
| 1 | 29 | Carlos Diaz | 3202 |
| 2 | 3 | Clara Wang | 2786 |
| 3 | 2 | Bob Johnson | 2587 |
| 4 | 17 | Quincy Wright | 2545 |
| 5 | 24 | Xander Dupont | 2451 |
| 6 | 19 | Samir Khan | 2416 |
| 7 | 5 | Eva Müller | 2274 |
| 8 | 26 | Zara Ibrahim | 2244 |
| 9 | 21 | Umar Farouk | 2017 |
| 10 | 30 | Daria Petrova | 1902 |

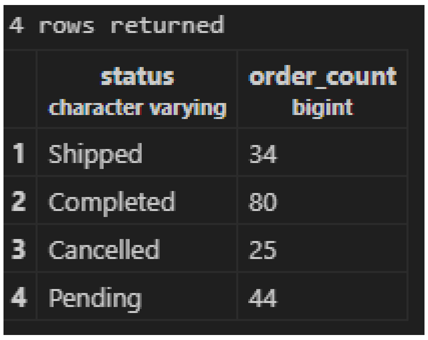
## 6. What is the order status breakdown?

**Purpose:** Understand fulfilment success rate.

**Query:**

```
SELECT
status,
COUNT(*) AS order_count
FROM orders
GROUP BY status;
```

**Result:**


4 rows returned

| | status character varying | order_count bigint |
|---|---|---|
| 1 | Shipped | 34 |
| 2 | Completed | 80 |
| 3 | Cancelled | 25 |
| 4 | Pending | 44 |

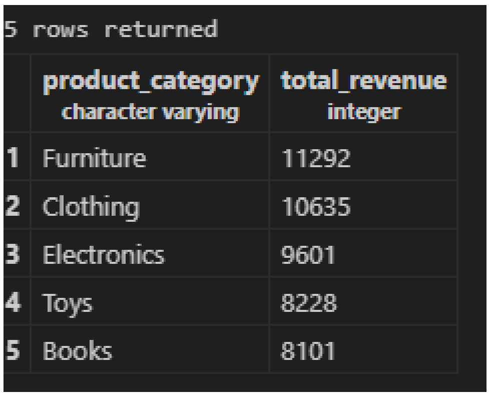
## 7. What are the top 5 most ordered product categories by revenue?

**Purpose:** Know which categories drive income.

**Query:**

```
SELECT
product_category,
CAST(SUM(order_amount) as INT) AS total_revenue
FROM orders
GROUP BY product_category
ORDER BY total_revenue DESC
LIMIT 5;
```

**Result:**



5 rows returned

| | product_category<br>character varying | total_revenue<br>integer |
|---|---|---|
| 1 | Furniture | 11292 |
| 2 | Clothing | 10635 |
| 3 | Electronics | 9601 |
| 4 | Toys | 8228 |
| 5 | Books | 8101 |

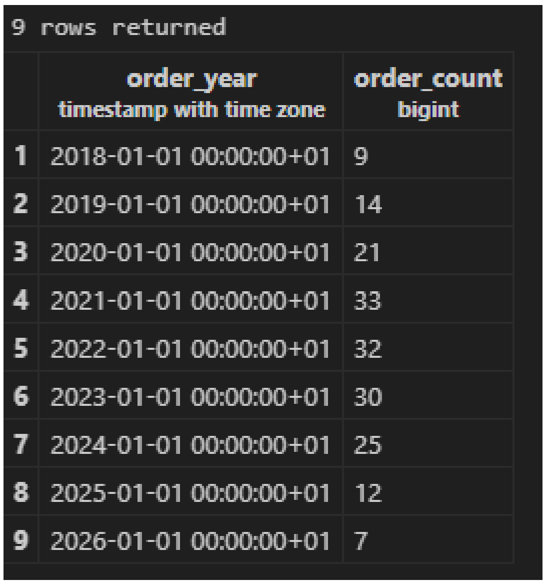
## 8. How does order volume change year by year?

**Purpose:** Spot seasonality or trends.

**Query:**

```
SELECT
DATE_TRUNC('year', order_date) AS order_year,
COUNT(*) AS order_count
FROM orders
GROUP BY order_year
ORDER BY order_year;
```

**Result:**


```
9 rows returned
```

| | order_year<br>timestamp with time zone | order_count<br>bigint |
|---|---|---|
| 1 | 2018-01-01 00:00:00+01 | 9 |
| 2 | 2019-01-01 00:00:00+01 | 14 |
| 3 | 2020-01-01 00:00:00+01 | 21 |
| 4 | 2021-01-01 00:00:00+01 | 33 |
| 5 | 2022-01-01 00:00:00+01 | 32 |
| 6 | 2023-01-01 00:00:00+01 | 30 |
| 7 | 2024-01-01 00:00:00+01 | 25 |
| 8 | 2025-01-01 00:00:00+01 | 12 |
| 9 | 2026-01-01 00:00:00+01 | 7 |

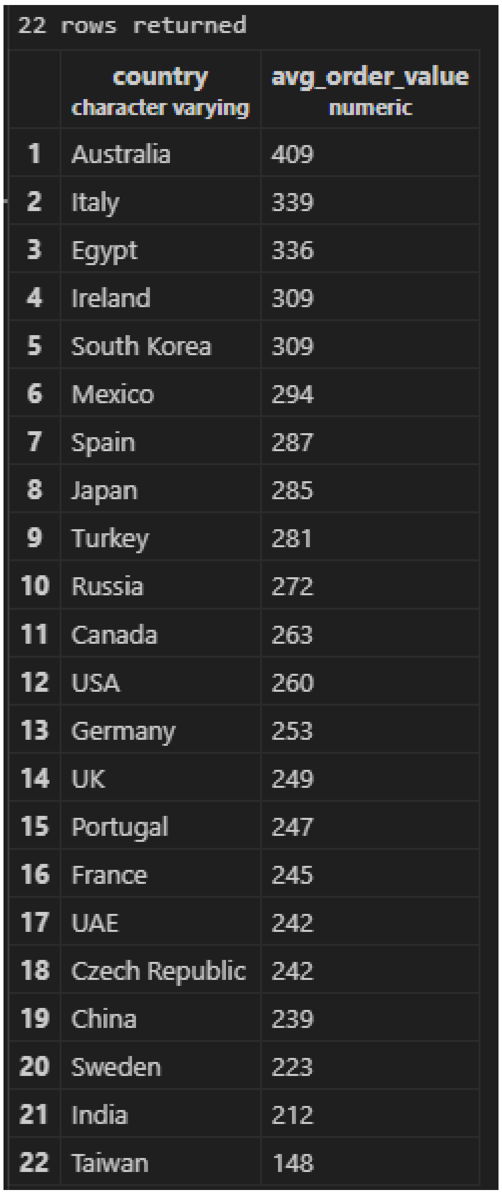
## 9. Which country has the highest average order amount?

**Purpose:** Combine customer origin and order behaviour.

**Query:**

```
SELECT
c.country,
ROUND(AVG(o.order_amount),0) AS avg_order_value
FROM customers c
JOIN orders o
ON c.customer_id = o.customer_id
GROUP BY c.country
ORDER BY avg_order_value DESC;
```

**Result:**



```
22 rows returned
```

| | country<br>character varying | avg_order_value<br>numeric |
|---|---|---|
| 1 | Australia | 409 |
| 2 | Italy | 339 |
| 3 | Egypt | 336 |
| 4 | Ireland | 309 |
| 5 | South Korea | 309 |
| 6 | Mexico | 294 |
| 7 | Spain | 287 |
| 8 | Japan | 285 |
| 9 | Turkey | 281 |
| 10 | Russia | 272 |
| 11 | Canada | 263 |
| 12 | USA | 260 |
| 13 | Germany | 253 |
| 14 | UK | 249 |
| 15 | Portugal | 247 |
| 16 | France | 245 |
| 17 | UAE | 242 |
| 18 | Czech Republic | 242 |
| 19 | China | 239 |
| 20 | Sweden | 223 |
| 21 | India | 212 |
| 22 | Taiwan | 148 |

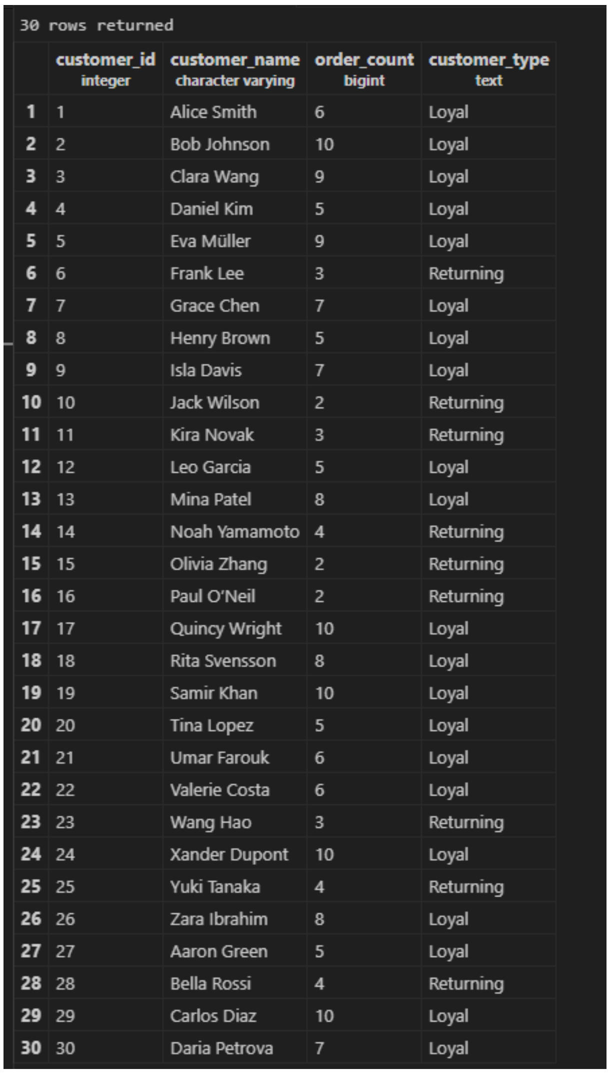
10. How engaged are our customers based on their order activity? Specifically, how many orders has each customer placed, and how can we classify them into meaningful segments like 'One-time', 'Returning', or 'Loyal' customers?

**Purpose:** Segment customers based on purchasing frequency to better understand engagement levels.

**Query:**

```
SELECT
c.customer_id,
c.customer_name,
order_stats.order_count,
CASE
WHEN order_stats.order_count = 1 THEN 'One-time'
WHEN order_stats.order_count BETWEEN 2 AND 4 THEN 'Returning'
ELSE 'Loyal'
END AS customer_type
FROM customers c
JOIN (
SELECT
customer_id,
COUNT(*) AS order_count
FROM orders
GROUP BY customer_id
) AS order_stats
ON c.customer_id = order_stats.customer_id;
```

**Result:**



30 rows returned

| | customer_id | customer_name | order_count | customer_type |
|---|---|---|---|---|
| | integer | character varying | bigint | text |
| 1 | 1 | Alice Smith | 6 | Loyal |
| 2 | 2 | Bob Johnson | 10 | Loyal |
| 3 | 3 | Clara Wang | 9 | Loyal |
| 4 | 4 | Daniel Kim | 5 | Loyal |
| 5 | 5 | Eva Müller | 9 | Loyal |
| 6 | 6 | Frank Lee | 3 | Returning |
| 7 | 7 | Grace Chen | 7 | Loyal |
| 8 | 8 | Henry Brown | 5 | Loyal |
| 9 | 9 | Isla Davis | 7 | Loyal |
| 10 | 10 | Jack Wilson | 2 | Returning |
| 11 | 11 | Kira Novak | 3 | Returning |
| 12 | 12 | Leo Garcia | 5 | Loyal |
| 13 | 13 | Mina Patel | 8 | Loyal |
| 14 | 14 | Noah Yamamoto | 4 | Returning |
| 15 | 15 | Olivia Zhang | 2 | Returning |
| 16 | 16 | Paul O'Neil | 2 | Returning |
| 17 | 17 | Quincy Wright | 10 | Loyal |
| 18 | 18 | Rita Svensson | 8 | Loyal |
| 19 | 19 | Samir Khan | 10 | Loyal |
| 20 | 20 | Tina Lopez | 5 | Loyal |
| 21 | 21 | Umar Farouk | 6 | Loyal |
| 22 | 22 | Valerie Costa | 6 | Loyal |
| 23 | 23 | Wang Hao | 3 | Returning |
| 24 | 24 | Xander Dupont | 10 | Loyal |
| 25 | 25 | Yuki Tanaka | 4 | Returning |
| 26 | 26 | Zara Ibrahim | 8 | Loyal |
| 27 | 27 | Aaron Green | 5 | Loyal |
| 28 | 28 | Bella Rossi | 4 | Returning |
| 29 | 29 | Carlos Diaz | 10 | Loyal |
| 30 | 30 | Daria Petrova | 7 | Loyal |

## Conclusion:

This analysis showcases my ability to turn raw data into business insights using PostgreSQL. Through techniques like joins, aggregation, subqueries, and conditional logic, I demonstrated real-world reporting skills. These are key tools I'd bring to any data-driven role to help teams make informed decisions.

**Techniques Demonstrated:**

- Filtering & Sorting with **WHERE, ORDER BY**
- Joins (**INNER JOIN, LEFT JOIN**) between multiple tables
- Aggregations using **COUNT, AVG, SUM, GROUP BY**
- Conditional logic and Data classification with **CASE WHEN**
- **Subqueries** for intermediate calculations
- Date functions