Northwind SQL Analytics Project

Team	
Status	Done

About project

PROBLEM STATEMENT

In any retail or distribution business, data holds the key to strategic decisions ; from identifying top customers and understanding product performance to tracking employee efficiency and operational bottlenecks.

The **Northwind database** provides a realistic business dataset that simulates customer orders, product sales, shipping timelines, and employee activities.

However, raw data alone doesn't provide value without analysis.

PROJECT OBJECTIVES

- Analyze sales performance and revenue distribution
- Identify high- and low-performing products and customers
- Evaluate employee productivity and supplier contribution
- Detect inefficiencies in operations (e.g., shipping times, unsold products)
- **Analyze cumulative sales trends year-over-year using time-based window functions**
- Use SQL analytics (e.g., CTEs, window functions, views) to build dynamic reports
- Visualize insights in Excel dashboards to support strategic business decisions.

BUSINESS PROBLEMS/QUESTIONS ANSWERED

1. Who are the top revenue-generating customers?
2. Which suppliers contribute the most to product volume?
3. Which employees process the most orders?
4. How efficient are our shippers?
5. What are the monthly sales trends for 1997?
6. Which countries generate the highest average order value?
7. Top-selling employees by year.
8. Which products were never ordered?
9. Which customers had no orders in 1997?
10. Revenue distribution by product category.
11. Returning customers and time between orders.
12. Products sold only once per order.
13. Product revenue percentile rankings.
14. Employees’ first order handled.
15. Customer revenue quartiles.

16. Yearly cumulative sales growth.

The first 7 questions were visualized in the dashboard; the remaining demonstrate SQL proficiency.

TOOLS & TECHNOLOGIES USED

- **PostgreSQL** – for querying and analyzing data using SQL
- **SQL Window Functions –** for calculating cumulative metrics, rankings, and trends
- **Common Table Expressions (CTEs) –** for readable, modular queries
- **Views –** for reusable data summaries
- **Northwind Traders Dataset –** as the simulated business database

Data Analysis & Methodology

- **Data Sources:** Orders, Order Details, Customers, Suppliers, Employees, Shippers, Products tables in the Northwind Database.
- **SQL Techniques Used:**
    - Joins to consolidate data across multiple tables.
    - Aggregations (SUM, AVG, COUNT) for key metrics.
    - Window functions for ranking, cumulative totals, and percentiles.
    - CTEs for modular and readable queries.
- **Excel Dashboarding:**
    - Pivot tables and charts to visualize results.
    - Slicers for filtering by employees.

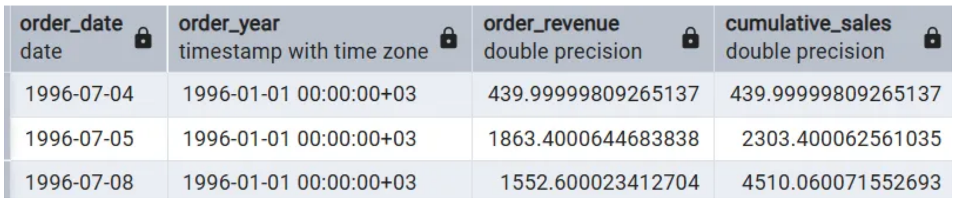
QUERY/CODE

-Problem:
-The Northwind company lacks visibility into how revenue accumulates throug hout the year, making it difficult to identify sales trends, seasonal performanc e, or growth patterns within each year

-Goal
-Analyze the cumulative sales performance per year, based on individual orde r revenue, to understand sales growth trends throughout each year.

QUERY:
```
WITH order_revenue AS (
  SELECT
    o.order_id,
    o.order_date,
    DATE_TRUNC('year', o.order_date) AS order_year,
    SUM(od.unit_price * od.quantity * (1 - od.discount)) AS order_revenue
  FROM orders o
  JOIN order_details od ON o.order_id = od.order_id
  WHERE o.order_date IS NOT NULL
  GROUP BY o.order_id, o.order_date
)
SELECT
  order_date,
  order_year,
  order_revenue,
  SUM(order_revenue) OVER (
    PARTITION BY order_year
    ORDER BY order_date
  ) AS cumulative_sales
FROM order_revenue
ORDER BY order_year, order_date;
```

RESULT:



order_date date	order_year timestamp with time zone	order_revenue double precision	cumulative_sales double precision
1996-07-04	1996-01-01 00:00:00+03	439.99999809265137	439.99999809265137
1996-07-05	1996-01-01 00:00:00+03	1863.4000644683838	2303.400062561035
1996-07-08	1996-01-01 00:00:00+03	1552.600023412704	4510.060071552693

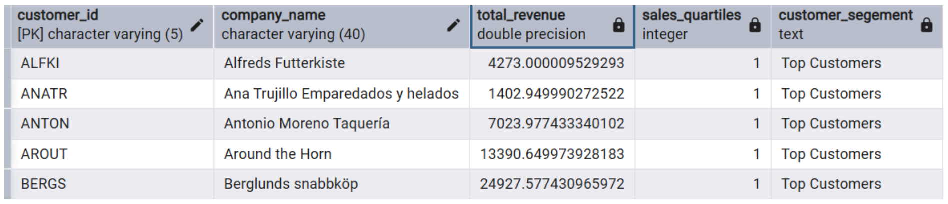
```
/* Classify Customers into Revenue-Based Quartiles

Problem:
You want to segment customers into quartiles based on their total revenue for
tailored marketing
*/

QUERY:
WITH total_customer_revenue AS(
SELECT c.customer_id, c.company_name, SUM(unit_price * quantity *(1-disco
unt)) AS total_revenue
FROM customers AS c
JOIN orders AS o
ON c.customer_id = o.customer_id
JOIN order_details AS od
ON o.order_id = od.order_id
GROUP BY c.customer_id, c.company_name
),
grouped_customers AS (
SELECT*, NTILE(4) OVER (PARTITION BY customer_id ORDER BY total_revenu
e DESC) AS sales_quartiles
FROM total_customer_revenue
)
SELECT *,
CASE
   WHEN sales_quartiles = 1 THEN 'Top Customers'
   WHEN sales_quartiles = 2 THEN 'High Value Customers'
   WHEN sales_quartiles = 3 THEN 'Middle Value Customers'
   WHEN sales_quartiles = 4 THEN 'low Value Customers'
```

```
END AS customer_segement
FROM grouped_customers;

RESULT:
```

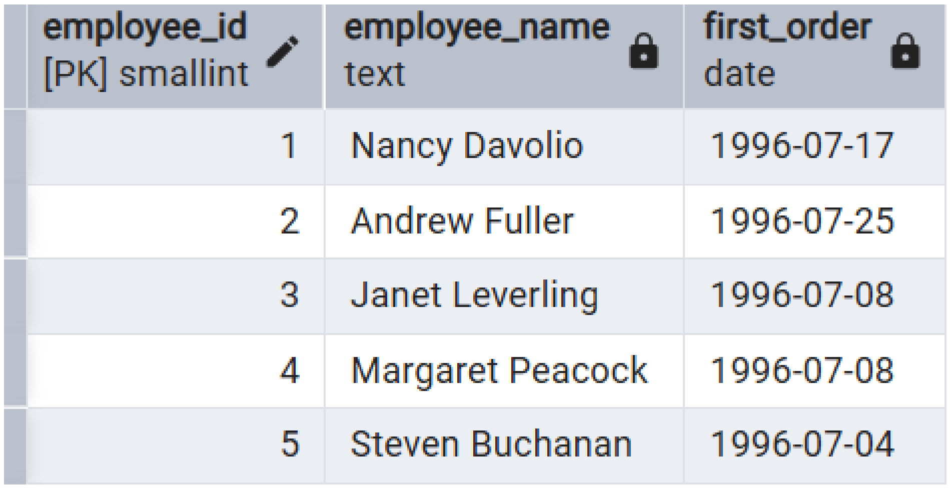


customer_id [PK] character varying (5)	company_name character varying (40)	total_revenue double precision	sales_quartiles integer	customer_segement text
ALFKI	Alfreds Futterkiste	4273.000009529293	1	Top Customers
ANATR	Ana Trujillo Emparedados y helados	1402.949990272522	1	Top Customers
ANTON	Antonio Moreno Taquería	7023.977433340102	1	Top Customers
AROUT	Around the Horn	13390.649973928183	1	Top Customers
BERGS	Berglunds snabbköp	24927.577430965972	1	Top Customers

```
/* Track the First Order Date for Each Employee
Problem:
You want to find out when each employee handled their first order, using orde
r dates from the orders table.
*/

QUERY:
WITH employee_first_order AS(
SELECT e.employee_id, first_name || ' ' || last_name AS employee_name, orde
r_date,
FIRST_VALUE(order_date) OVER (PARTITION BY e.employee_id ORDER BY ord
er_date) AS first_order_date
FROM employees AS e
JOIN orders AS o
ON e.employee_id = o.employee_id
JOIN order_details AS od
ON o.order_id = od.order_id
)
SELECT DISTINCT employee_id, employee_name, first_order_date
FROM employee_first_order
WHERE order_date = first_order_date;
```

RESULT:



employee_id [PK] smallint	employee_name text	first_order date
1	Nancy Davolio	1996-07-17
2	Andrew Fuller	1996-07-25
3	Janet Leverling	1996-07-08
4	Margaret Peacock	1996-07-08
5	Steven Buchanan	1996-07-04

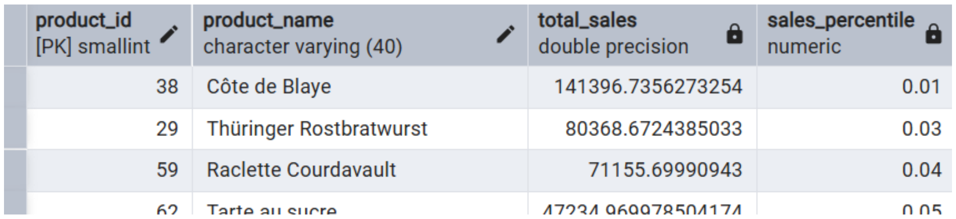
/* Ranking Products by Sales Performance Percentile
 Problem:
The sales department wants to understand how products rank in terms of tota
l revenue performance, so they can target marketing campaigns for the top pe
rcentile performers.

 Goal:
Use CUME_DIST() to assign a percentile ranking to each product based on tot
al revenue
*/

QUERY:

```
WITH total_revenue AS(
SELECT p.product_id, p.product_name, SUM(od.unit_price * od.quantity * (1-o
d.discount)) AS total_sales
FROM products AS p
JOIN order_details AS od
ON p.product_id = od.product_id
JOIN orders AS o
ON od.order_id = o.order_id
GROUP BY p.product_id, p.product_name
)
SELECT *, ROUND(CUME_DIST() OVER (ORDER BY total_sales DESC)::numeri
c, 2) AS sales_percentile
FROM total_revenue;

RESULT:
```

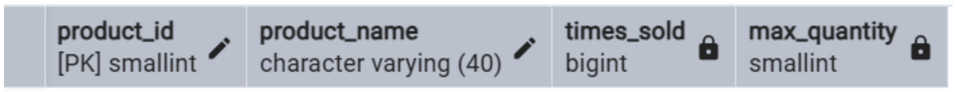


product_id [PK] smallint	product_name character varying (40)	total_sales double precision	sales_percentile numeric
38	Côte de Blaye	141396.7356273254	0.01
29	Thüringer Rostbratwurst	80368.6724385033	0.03
59	Raclette Courdavault	71155.69990943	0.04
62	Tarte au sucre	47234.969978504174	0.05

```
\* Identifying Products Sold Only Once Per Order
 Problem:
Inventory managers suspect some products are always sold in quantities of 1.
These might not be bulk-sale friendly or are just low-demand items *\

QUERY:
SELECT p.product_id, p.product_name, COUNT(*) AS times_sold
FROM products AS p
JOIN order_details AS od
ON p.product_id = od.product_id
GROUP BY p.product_id, p.product_name
HAVING MIN(od.quantity) = 1 AND MAX(od.quantity) = 1;

RESULT:
```



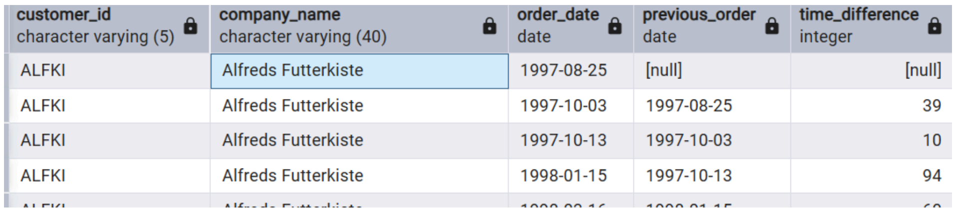
product_id [PK] smallint	product_name character varying (40)	times_sold bigint	max_quantity smallint

```
\* 2. Returning Customers - Time Between Orders
   Problem:
Marketing wants to know how often customers return — i.e., the time between
their current and previous orders.

  Project Goal:
Show each customer's order date alongside the date of their previous order *\

QUERY:
WITH customer_orders AS(
SELECT c.customer_id, c.company_name, o.order_date,
LAG(order_date) OVER (PARTITION BY c.customer_id ORDER BY o.order_date
ASC) AS previous_order
FROM customers AS c
JOIN orders AS o
ON c.customer_id = o.customer_id
),
return_period AS(
SELECT *, (order_date - previous_order) AS time_difference
FROM customer_orders
)
SELECT *
FROM return_period;

RESULT:
```



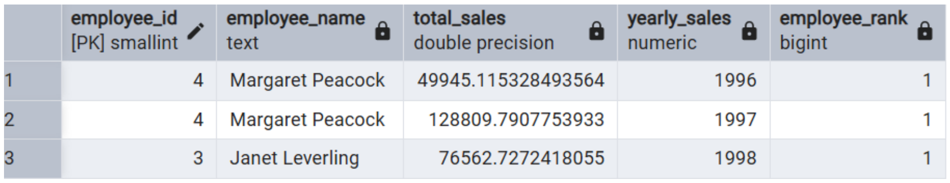
customer_id character varying (5)	company_name character varying (40)	order_date date	previous_order date	time_difference integer
ALFKI	Alfreds Futterkiste	1997-08-25	[null]	[null]
ALFKI	Alfreds Futterkiste	1997-10-03	1997-08-25	39
ALFKI	Alfreds Futterkiste	1997-10-13	1997-10-03	10
ALFKI	Alfreds Futterkiste	1998-01-15	1997-10-13	94

```
\*  Employee Sales Ranking per Year
   Problem:
The management wants to identify the top-performing sales employees by ye
ar. They need each employee’s sales rank within each year based on total sale
s.
*/

QUERY:
WITH
employee_year_sales AS (
SELECT e.employee_id, first_name || ' ' || last_name AS employee_name, SUM
(od.unit_price * od.quantity * (1-od.discount)) AS total_sales,
EXTRACT(YEAR FROM order_date) AS yearly_sales
FROM employees AS e
JOIN orders AS o
ON e.employee_id = o.employee_id
JOIN order_details AS od
ON od.order_id = o.order_id
GROUP BY EXTRACT(YEAR FROM order_date), e.employee_id
),
ranked_sales AS(
SELECT *,
RANK() OVER (PARTITION BY yearly_sales ORDER BY total_sales DESC) AS e
mployee_rank
FROM employee_year_sales
)
SELECT*
FROM ranked_sales
```

```
WHERE employee_rank = 1;

RESULT:
```

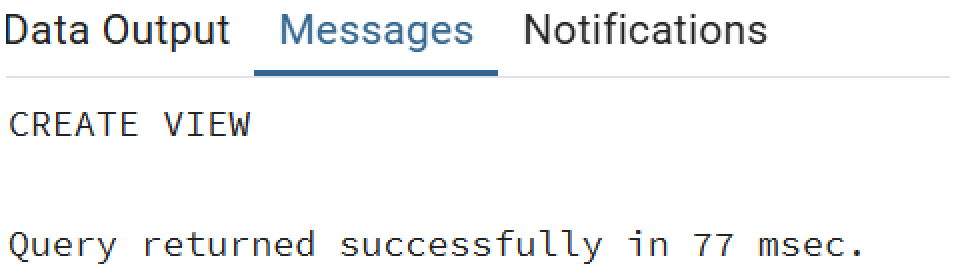


	employee_id [PK] smallint	employee_name text	total_sales double precision	yearly_sales numeric	employee_rank bigint
1	4	Margaret Peacock	49945.115328493564	1996	1
2	4	Margaret Peacock	128809.7907753933	1997	1
3	3	Janet Leverling	76562.7272418055	1998	1

```
--  Revenue by Category (CREATE A VIEW)
-- Goal: Calculate total revenue grouped by product category.

QUERY:
CREATE VIEW revenue_per_category AS
SELECT c.category_id, c.category_name, SUM(od.unit_price * od.quantity * (1
-od.discount)) AS revenue
FROM products AS p
JOIN order_details AS od
ON p.product_id = od.product_id
JOIN categories AS c
ON p.category_id = c.category_id
GROUP BY c.category_id, c.category_name;

RESULT:
```



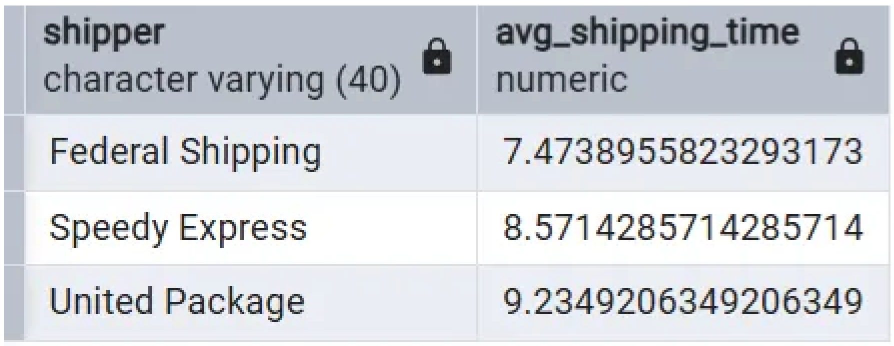
```
CREATE VIEW

Query returned successfully in 77 msec.
```

```
-- Problem : Products Never Ordered
-- Goal: List all products that have never been included in any order.

QUERY:
SELECT product_name
  FROM products AS p
  LEFT JOIN order_details AS od
    ON p.product_id = od.product_id
 WHERE od.product_id IS NULL;

RESULT:
```

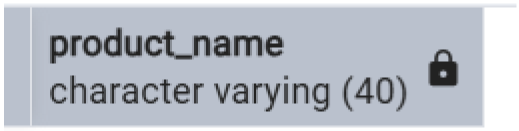


shipper character varying (40)	avg_shipping_time numeric
Federal Shipping	7.4738955823293173
Speedy Express	8.5714285714285714
United Package	9.2349206349206349

```
-- Problem: Average Shipping Time per Shipper
-- Calculate the average number of days it takes each shipper (company) to s
hip an order.

QUERY:
SELECT s.company_name AS shipper, avg(o.shipped_date - o.order_date) AS
avg_shipping_time
FROM shippers AS s
JOIN orders AS o
ON s.shipper_id = o.ship_via
WHERE shipped_date IS NOT NULL AND order_date IS NOT NULL
GROUP BY s.company_name;
```

```
RESULT:
```

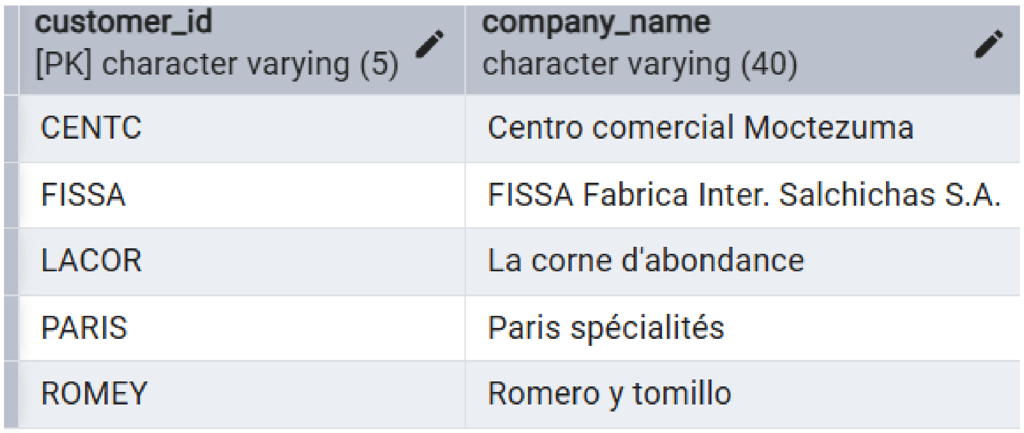


product_name character varying (40)

```
-- Problem: Customers with No Orders in 1997
-- Goal: Identify customers who did not place any orders during the year 1997.

QUERY:
SELECT customer_id, company_name
FROM customers
WHERE customer_id NOT IN
(SELECT customer_id
FROM orders
WHERE EXTRACT(YEAR FROM order_date) = 1997);

RESULT:
```

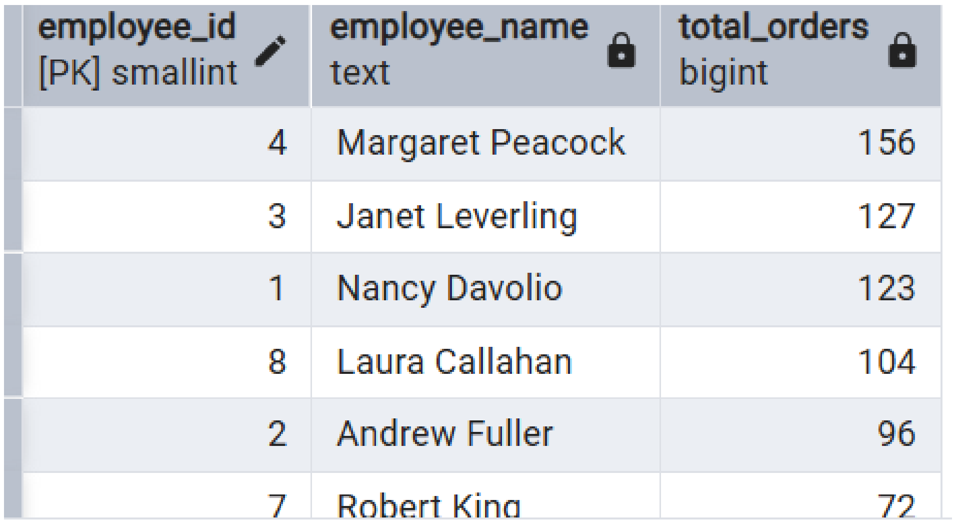


customer_id [PK] character varying (5)	company_name character varying (40)
CENTC	Centro comercial Moctezuma
FISSA	FISSA Fabrica Inter. Salchichas S.A.
LACOR	La corne d'abondance
PARIS	Paris spécialités
ROMEY	Romero y tomillo

```
-- Problem 5: Employees with Highest Number of Orders Handled
-- Goal: Identify which employees have handled the most orders.

QUERY:
SELECT e.employee_id, e.first_name || ' ' || e.last_name AS employee_name,
COUNT(DISTINCT od.order_id) AS total_orders
FROM employees AS e
JOIN orders AS o
ON e.employee_id = o.employee_id
JOIN order_details AS od
ON o.order_id = od.order_id
GROUP BY e.employee_id
ORDER BY total_orders DESC;

RESULT:
```

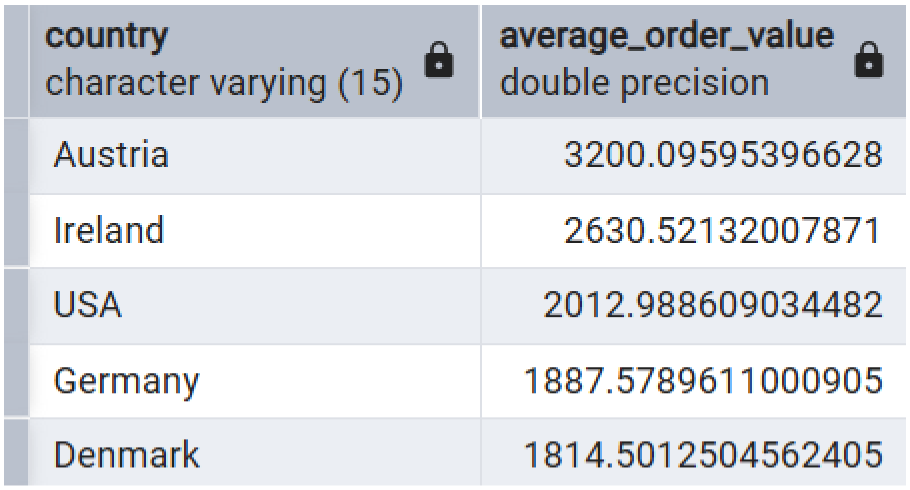


employee_id [PK] smallint	employee_name text	total_orders bigint
4	Margaret Peacock	156
3	Janet Leverling	127
1	Nancy Davolio	123
8	Laura Callahan	104
2	Andrew Fuller	96
7	Robert King	72

```
-- Problem: Average Order Value by Country
-- Goal: Calculate the average order value for each customer country to under stand market value by region.
```

```
QUERY:
WITH total_orders AS(
SELECT od.order_id, c.country, SUM(od.unit_price * od.quantity * (1 -od.disco
unt)) AS total_order_value
FROM customers AS c
JOIN orders AS o
ON c.customer_id = o.customer_id
JOIN order_details AS od
ON o.order_id = od.order_id
GROUP BY od.order_id, c.country
)
SELECT country, AVG(total_order_value) AS average_order_value
FROM total_orders
GROUP BY country
ORDER BY AVG(total_order_value) DESC;

RESULT:
```



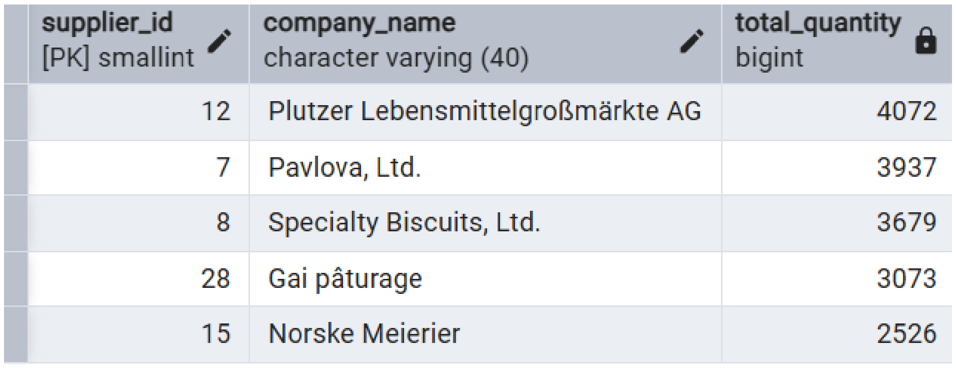
country character varying (15)	average_order_value double precision
Austria	3200.09595396628
Ireland	2630.52132007871
USA	2012.988609034482
Germany	1887.5789611000905
Denmark	1814.5012504562405

```
-- Top 5 Suppliers by Product Quantity Supplied

QUERY:
SELECT s.supplier_id, s.company_name, SUM(od.quantity) AS total_quantity
```

```
FROM order_details AS od
JOIN products AS p
ON od.product_id = p.product_id
JOIN suppliers AS s
ON p.supplier_id = s.supplier_id
GROUP BY s.supplier_id, s.company_name
ORDER BY SUM(quantity) DESC
LIMIT 5;

RESULT:
```



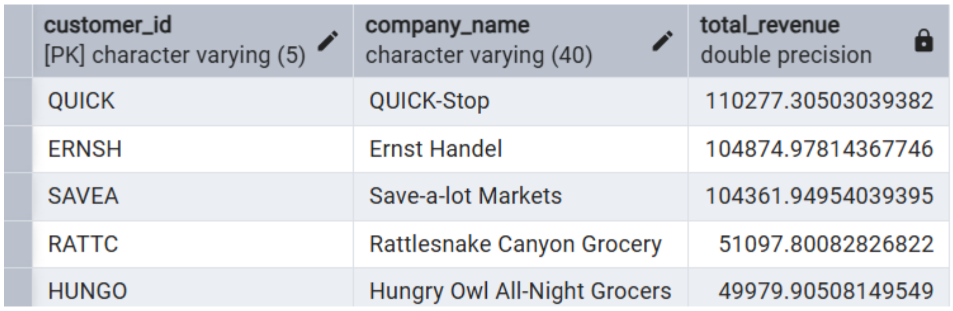
supplier_id [PK] smallint	company_name character varying (40)	total_quantity bigint
12	Plutzer Lebensmittelgroßmärkte AG	4072
7	Pavlova, Ltd.	3937
8	Specialty Biscuits, Ltd.	3679
28	Gai pâturage	3073
15	Norske Meierier	2526

```
-- Top 5 Customers by Revenue
→ Who are the highest revenue-generating customers?

QUERY:
SELECT
  c.customer_id,
  c.company_name,
  SUM(od.unit_price * od.quantity * (1 - od.discount)) AS total_revenue
FROM
  orders AS o
JOIN
  customers AS c ON o.customer_id = c.customer_id
```

```
JOIN
  order_details AS od ON o.order_id = od.order_id
GROUP BY
  c.customer_id, c.company_name
ORDER BY
  total_revenue DESC
LIMIT 5;

RESULT:
```



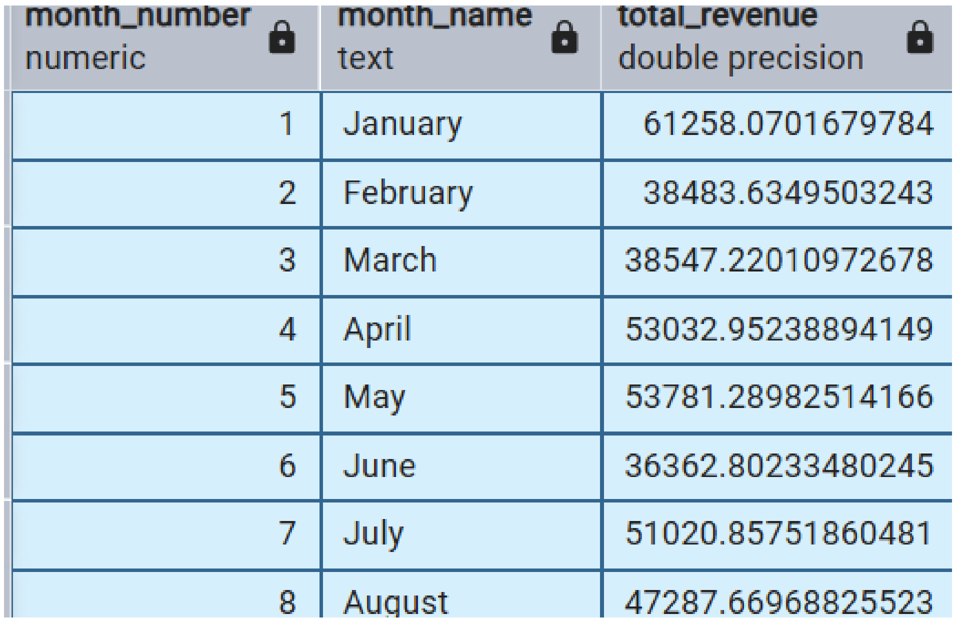
customer_id [PK] character varying (5)	company_name character varying (40)	total_revenue double precision
QUICK	QUICK-Stop	110277.30503039382
ERNSH	Ernst Handel	104874.97814367746
SAVEA	Save-a-lot Markets	104361.94954039395
RATTC	Rattlesnake Canyon Grocery	51097.80082826822
HUNGO	Hungry Owl All-Night Grocers	49979.90508149549

- Monthly Sales Trend for 1997
- How did sales evolve month-by-month in 1997?

```
QUERY:
SELECT
EXTRACT(MONTH FROM o.order_date) AS month_number,
  TO_CHAR(o.order_date, 'FMMonth') AS month_name, SUM(od.unit_price * o
d.quantity * (1-od.discount)) AS total_revenue
FROM
orders AS o
JOIN
order_details AS od
ON
o.order_id = od.order_id
WHERE EXTRACT(YEAR FROM o.order_date) = 1997
```

```
GROUP BY EXTRACT(MONTH FROM o.order_date), TO_CHAR(o.order_date, 'F
MMonth')
ORDER BY EXTRACT(MONTH FROM o.order_date);

RESULT:
```



month_number numeric	month_name text	total_revenue double precision
1	January	61258.0701679784
2	February	38483.6349503243
3	March	38547.22010972678
4	April	53032.95238894149
5	May	53781.28982514166
6	June	36362.80233480245
7	July	51020.85751860481
8	August	47287.66968825523

DASHBOARD ANALYSIS - KEY INSIGHTS

These are the 7 queries visualized in the Excel dashboard:

1. **Top Customers by Revenue**
    - **Insight:** A small set of customers drives the majority of revenue.
    - **Recommendation:** Focus loyalty programs and targeted marketing on top customers.
2. **Top Suppliers by Quantity Supplied**
    - **Insight:** Certain suppliers are critical for inventory maintenance.
    - **Recommendation:** Maintain strong relationships and monitor delivery performance.
3. **Top Employees by Sales**
    - **Insight:** A few employees generate the majority of sales revenue.

    - **Recommendation:** Reward top performers and encourage knowledge sharing.

4. **Top Employees by Orders Handled**
    - **Insight:** Highlights employees with high operational efficiency.
    - **Recommendation:** Balance workloads and provide support to optimize performance.
5. **Average Shipping Time per Shipper**
    - **Insight:** Some shippers are consistently faster, impacting customer satisfaction.
    - **Recommendation:** Optimize shipping timelines and consider top-performing shippers for priority deliveries.
6. **Monthly Sales Trend for 1997**
    - **Insight:** Sales peak in certain months, indicating seasonality.
    - **Recommendation:** Plan inventory, promotions, and staffing around seasonal peaks.
7. **Average Order Value by Country**
    - **Insight:** Certain countries provide higher revenue per order.
    - **Recommendation:** Tailor marketing, pricing, and promotions to high-value countries.

RECOMMENDATIONS

- Focus on high-revenue customers with loyalty and retention programs.
- Strengthen relationships with top suppliers and monitor performance.
- Recognize top-performing employees and encourage knowledge sharing.
- Optimize shipping timelines to improve customer satisfaction.
- Plan inventory, promotions, and staffing according to seasonal sales trends.
- Customize offers and pricing by country to maximize revenue.

CONCLUSION/INSIGHTS

This project demonstrated how SQL can unlock powerful insights from structured business data. From identifying top-performing products and customers to tracking employee impact and operational delays, every query contributed to a clearer picture of the business.

Key takeaways include:

- A small group of customers and products drive a majority of revenue highlighting the importance of **focused retention and inventory planning**.
- **Window functions like the lag offset function** allowed deeper, more flexible analysis of behavior over time (e.g., product trends, customer return frequency).
- The business can **make smarter, data-driven decisions** by segmenting customers and products and acting on inefficiencies.

Documents