SQL Project: Online Retail Data Analysis

Project Overview

This project demonstrates a comprehensive Retail Business Analysis using SQL, applied to a structured relational database containing 9 interconnected tables: brands, categories, customers, order_items, orders, products, staffs, stocks, and stores.

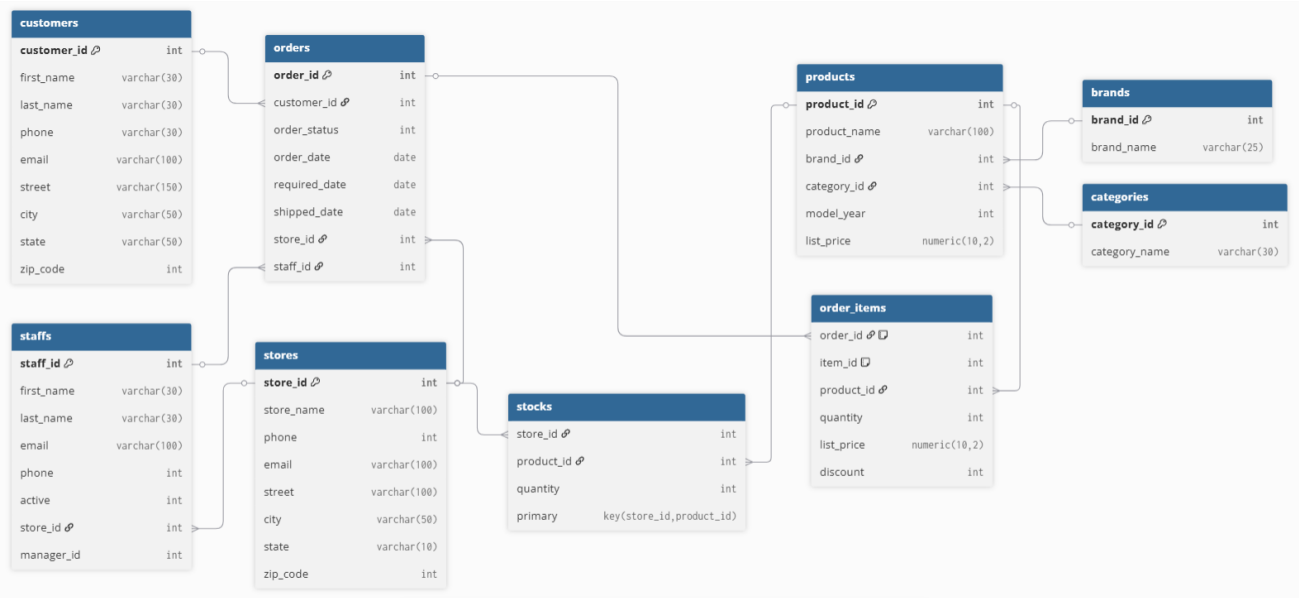
Each table represents a crucial component of the business — from customer details and product catalogs to store inventory and staff management. The project uses advanced SQL techniques to extract valuable insights, track business performance, and assist in strategic decision-making.

Key objectives of the project include:

- Understanding customer behavior and order patterns
- Analyzing product pricing, stock availability, and category trends
- Monitoring staff performance and delivery efficiency
- Segmenting customers based on their purchase value
- Identifying high-performing products and stores

The SQL queries span multiple concepts such as joins, aggregations, subqueries, window functions, CTEs, CASE logic, and set operations, making this project a powerful demonstration of data analysis capabilities in PostgreSQL.

Entity Relationship Diagram (ERD)



List of Analytical SQL Queries

1. Customer Order Date

Query

```sql
-- 1. Fetch customers with their order dates
SELECT c.first_name, c.last_name, o.order_date
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
ORDER BY c.first_name;
```

Query Output

	first_name character varying (30)	last_name character varying (30)	order_date date
1	Aaron	Knapp	2016-02-20
2	Abbey	Pugh	2016-04-08
3	Abby	Gamble	2017-12-27
4	Abby	Gamble	2018-04-08
5	Abram	Copeland	2017-06-05

Business Insight: Displays each customer's order date to understand when they made their purchases and identify buying behavior.

2. Product, Brand & Category Mapping

Query

```sql
-- 2. List product names with brand and category
SELECT p.product_name, b.brand_name, c.category_name
FROM products p
JOIN brands b ON p.brand_id = b.brand_id
JOIN categories c ON p.category_id = c.category_id;
```

Query Output

	product_name character varying (100)	brand_name character varying (25)	category_name character varying (30)
1	Trek 820 - 2016	Trek	Mountain Bikes
2	Ritchey Timberwolf Frameset - 2016	Ritchey	Mountain Bikes
3	Surly Wednesday Frameset - 2016	Surly	Mountain Bikes
4	Trek Fuel EX 8 29 - 2016	Trek	Mountain Bikes
5	Heller Shagamaw Frame - 2016	Heller	Mountain Bikes

Business Insight: Shows how each product is linked to its brand and category — useful for catalog organization and brand-level analysis.

3. Orders with Staff and Store Info

Query

```sql
-- 3. Show orders with staff names and store names
SELECT o.order_id, s.first_name || ' ' || s.last_name
AS staff_name, st.store_name
FROM orders o
JOIN staffs s ON o.staff_id = s.staff_id
JOIN stores st ON o.store_id = st.store_id;
```

Query Output

	order_id integer	staff_name text	store_name character varying (100)
1	1	Mireya Copela...	Santa Cruz Bikes
2	2	Marcelene Boyer	Baldwin Bikes
3	3	Venita Daniel	Baldwin Bikes
4	4	Genna Serrano	Santa Cruz Bikes
5	5	Marcelene Boyer	Baldwin Bikes

Business Insight: Retrieves orders along with the staff who handled them and the store they came from — helps in operational tracking.

4. All Customers (With or Without Orders)

Query

```sql
-- 4. Show all customers with or without orders
SELECT c.customer_id, c.first_name, o.order_id
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;
```

Query Output

	customer_id integer	first_name character varying (30)	order_id integer
1	259	Johnathan	1
2	1212	Jaqueline	2
3	523	Joshua	3
4	175	Nova	4
5	1324	Arla	5

Business Insight: Lists all registered customers, including those who have never placed an order — valuable for targeting inactive users.

5. Product Stock by Store

Query

```sql
-- 5. List products with stock quantity per store
SELECT p.product_name, s.store_id, st.quantity
FROM products p
LEFT JOIN stocks st ON p.product_id = st.product_id
LEFT JOIN stores s ON st.store_id = s.store_id;
```

Query Output

	product_name character varying (100)	store_id integer	quantity integer
1	Trek 820 - 2016	1	27
2	Ritchey Timberwolf Frameset - 2016	1	5
3	Surly Wednesday Frameset - 2016	1	6
4	Trek Fuel EX 8 29 - 2016	1	23
5	Heller Shagamaw Frame - 2016	1	22

Business Insight: Displays available stock of each product in every store — supports inventory management and restocking decisions.

6. Total Orders per Store

Query

```sql
-- 6. Count total orders placed at each store
SELECT store_id, COUNT(*) AS total_orders
FROM orders
GROUP BY store_id;
```

Query Output

	store_id integer	total_orders bigint
1	1	348
2	3	174
3	2	1093

Business Insight: Shows the total number of orders handled by each store — helps evaluate store performance.

7. Average Product Price by Category

Query

```sql
-- 7. Calculate average product price by category
SELECT c.category_name, AVG(p.list_price) AS average_price
FROM products p
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name;
```

Query Output

	category_name character varying (30)	average_price numeric
1	Electric Bikes	3281.6566666666666667
2	Comfort Bicycles	682.1233333333333333
3	Cruisers Bicycles	730.4123076923076923
4	Mountain Bikes	1649.7573333333333333
5	Cyclocross Bicycles	2542.7930000000000000
6	Road Bikes	3175.3573333333333333
7	Children Bicycles	287.7866101694915254

Business Insight: Calculates the average price of products within each category — useful for pricing analysis and strategy.

8. Top 5 Most Ordered Products

Query

```
-- 8. Show top 5 most ordered products
SELECT p.product_name, SUM(oi.quantity) AS total_ordered
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_ordered DESC
LIMIT 5;
```

Query Output

	product_name character varying (100)	total_ordered bigint
1	Electra Cruiser 1 (24-Inch) - 2016	296
2	Electra Townie Original 7D EQ - 2016	290
3	Electra Townie Original 21D - 2016	289
4	Electra Girl's Hawaii 1 (16-inch) - 2015/2016	269
5	Surly Ice Cream Truck Frameset - 2016	167

Business Insight: Identifies the top 5 most frequently ordered products — helps prioritize high-demand items for promotions.

9. Customers with No Orders

Query

```
-- 9. List customers who haven't placed any orders
SELECT customer_id, first_name, last_name
FROM customers
WHERE customer_id NOT IN
(SELECT DISTINCT customer_id FROM orders);
```

Query Output

	customer_id [PK] integer	first_name character varying (30)	last_name character varying (30)

Business Insight: Finds customers who haven't placed any orders — useful for customer re-engagement or marketing efforts.

10. Monthly Order Trends

Query

```sql
-- 10. Monthly order count
SELECT DATE_TRUNC('month', order_date) AS month,
COUNT(*) AS total_orders
FROM orders
GROUP BY month
ORDER BY month;
```

Query Output

	month timestamp with time zone	total_orders bigint
1	2016-01-01 00:00:00+05	50
2	2016-02-01 00:00:00+05	49
3	2016-03-01 00:00:00+05	55
4	2016-04-01 00:00:00+05	43
5	2016-05-01 00:00:00+05	51

Business Insight: Shows how order volumes vary month by month — helps spot seasonal trends and support business planning.

11. Staff-wise Order Handling

Query

```sql
-- 11. Show staff sales count
SELECT s.staff_id, s.first_name, COUNT(o.order_id)
AS total_sales FROM staffs s
LEFT JOIN orders o ON s.staff_id = o.staff_id
GROUP BY s.staff_id, s.first_name;
```

Query Output

	staff_id [PK] integer	staff_name text	units_sold_by_staff bigint
1	9	Layla Terrell	86
2	3	Genna Serrano	184
3	5	Jannette David	0
4	4	Virgie Wiggins	0
5	10	Bernardine Houston	0

Business Insight: Displays how many orders each staff member has managed — useful for evaluating staff productivity.

12. Stores Missing a Specific Product

Query

```sql
-- 12. Find stores with no stock of a specific product
SELECT store_id
FROM stores
WHERE store_id NOT IN (
  SELECT store_id FROM stocks WHERE product_id = 5
);
```

Query Output

	store_id [PK] integer

Business Insight: Finds which stores do not have a specific product in stock — supports supply chain and inventory distribution planning..

13. Highest Priced Product per Category

Query

```
77  -- 13. Highest priced product in each category
78  SELECT c.category_name, MAX(p.list_price) AS highest_price
79  FROM products p
80  JOIN categories c ON p.category_id = c.category_id
81  GROUP BY c.category_name;
```

Query Output

	category_name character varying (30)	highest_price numeric
1	Electric Bikes	4999.99
2	Comfort Bicycles	2599.99
3	Cruisers Bicycles	2999.99
4	Mountain Bikes	5299.99
5	Cyclocross Bicycles	3999.99
6	Road Bikes	11999.99
7	Children Bicycles	489.99

Business Insight: Retrieves the most expensive product in each category — useful for tracking premium items.

14. Average Quantity Sold per Product

Query

```
83  -- 14. Average quantity sold per product
84  SELECT p.product_name, AVG(oi.quantity) AS avg_quantity
85  FROM order_items oi
86  JOIN products p ON oi.product_id = p.product_id
87  GROUP BY p.product_name;
```

Query Output

	product_name character varying (100)	avg_quantity numeric
1	Trek Kids' Neko - 2018	1.6666666666666667
2	Trek Domane ALR 4 Disc - 2018	1.5000000000000000
3	Sun Bicycles Boardwalk (24-inch Wheels) - 2017	1.3125000000000000
4	Trek Powerfly 5 Women's - 2018	1.6666666666666667
5	Trek Domane ALR Frameset - 2018	1.6666666666666667

Business Insight: Calculates average number of units sold for each product — helps understand product demand levels.

15. Total Stock Value per Store

Query

```sql
-- 15. Show each store's total stock value
SELECT s.store_id, SUM(p.list_price * st.quantity)
AS total_stock_value FROM stocks st
JOIN products p ON st.product_id = p.product_id
JOIN stores s ON st.store_id = s.store_id
GROUP BY s.store_id;
```

Query Output

	store_id [PK] integer	total_stock_value numeric
1	2	6473126.28
2	3	6859214.21
3	1	6487242.16

Business Insight: Calculates the total value of stock held in each store — important for financial reporting and asset management.

Access Full Project on GitHub

For more SQL queries, additional analysis, full source code, and project files related to this project — as well as other SQL projects — please visit the GitHub repository:

☞ Click here to visit the GitHub Repository

Conclusion

This project demonstrates the power of SQL in analyzing complex retail data across multiple dimensions such as customer behavior, product trends, staff performance, and store operations. By using advanced SQL techniques including joins, subqueries, aggregations, window functions, and CASE logic, we extracted actionable insights that support strategic decision-making.

From identifying top-performing products and busiest stores to analyzing monthly trends and inactive customers, each query plays a role in understanding and improving business outcomes. This analysis highlights how data-driven decisions can enhance inventory management, optimize staff productivity, and improve customer engagement.

Overall, the project reflects a real-world approach to retail data analysis using SQL — making it a valuable asset for business intelligence and data analyst roles.