



Exploring customer and sales data  
through SQL queries

# BIKE STORE SQL ANALYSIS

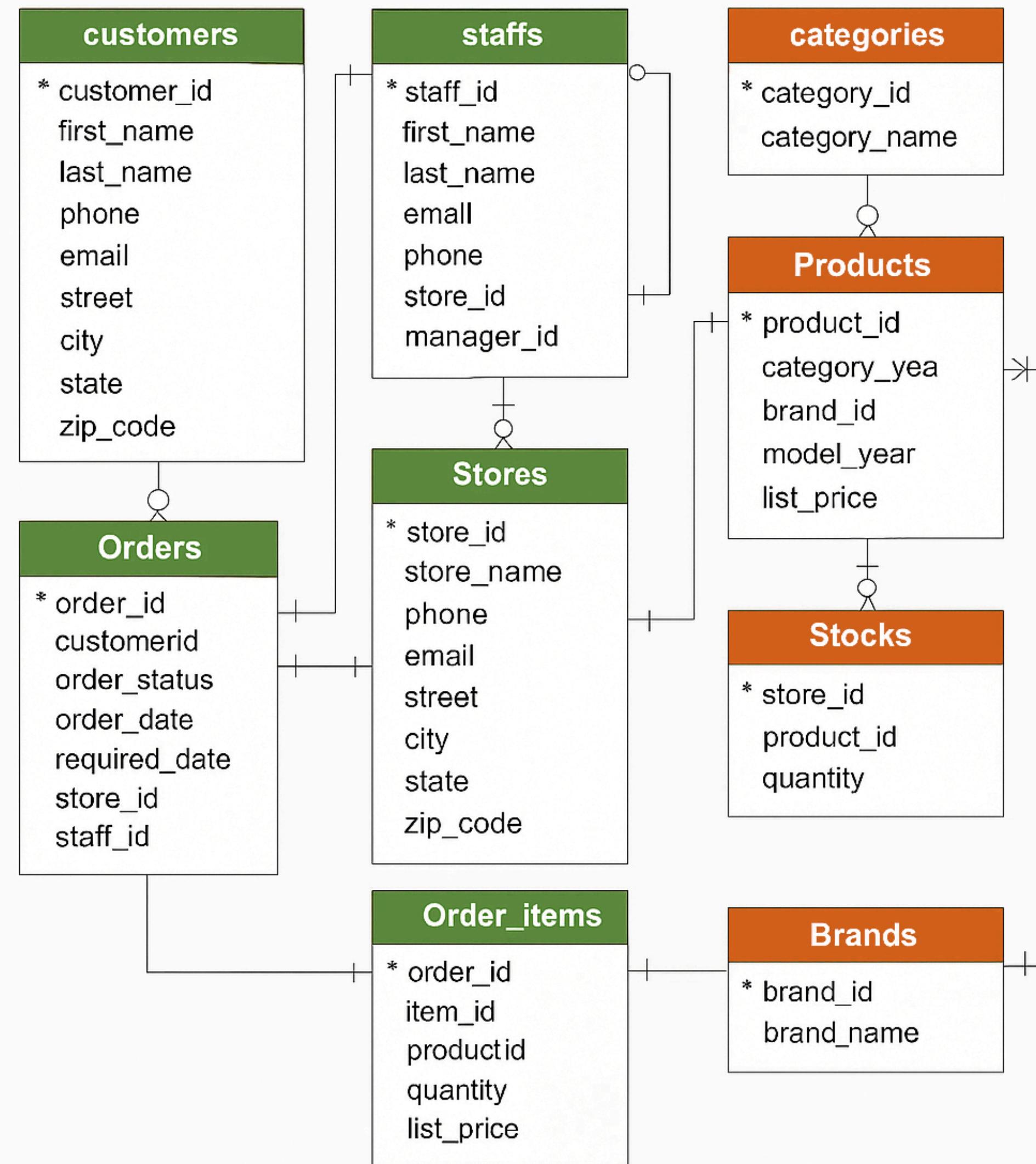


# DATABASE & STRUCTURE



Tables included in the project:

- 🏷️ **Brands** – Information about bike manufacturers and suppliers
- 🧩 **Categories** – Product groupings such as mountain, road, and electric bikes
- 🚲 **Products** – Detailed product data including price and category
- 👤 **Customers** – Customer details and purchasing behavior
- 📋 **Orders** – Transactional sales data
- 📦 **Order\_Items** – Product-level details for each order
- 👔 **Staffs** – Employee information related to sales and management
- 🏢 **Stores** – Store locations and performance data
- 🏗️ **Stocks** – Product inventory per store



# TOP-SELLING PRODUCT PER CATEGORY (BY REVENUE)

```
WITH ProductSales AS (
    SELECT p.product_id, p.product_name, p.category_id,
           SUM(oi.quantity * oi.list_price * (1 - oi.discount)) AS total_revenue
    FROM "Products" p
    JOIN "Order_Items" oi ON p.product_id = oi.product_id
    GROUP BY p.product_id, p.product_name, p.category_id
)
SELECT c.category_name, ps.product_name, ps.total_revenue,
       RANK() OVER(PARTITION BY c.category_name ORDER BY ps.total_revenue DESC) AS sales_rank_in_category
FROM ProductSales ps
JOIN "Categories" c ON ps.category_id = c.category_id
ORDER BY c.category_name, sales_rank_in_category;
```

	category_name	product_name	total_revenue	sales_rank_in_category
1	Children Bicycles	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	41011.6329	1
2	Children Bicycles	Electra Girl's Hawaii 1 (16-inch) - 2015/2016	34728.8137	2
3	Children Bicycles	Electra Cruiser 1 (24-Inch) - 2016	34078.1378	3
4	Children Bicycles	Electra Townie 3i EQ (20-inch) - Boys' - 2017	17125.1505	4
5	Children Bicycles	Electra Girl's Hawaii 1 16" - 2017	12785.5738	5
6	Children Bicycles	Trek Precaliber 24 (21-Speed) - Girls - 2017	12179.652	6
7	Children Bicycles	Electra Townie 7D (20-inch) - Boys' - 2017	12079.8447	7



# MOST FREQUENTLY BOUGHT PRODUCT PAIRS

```
WITH OrderProducts AS (
    SELECT oi.order_id, p.product_id, p.product_name
    FROM "Order_Items" oi
    JOIN "Products" p ON oi.product_id = p.product_id
),
ProductPairs AS (
    SELECT op1.product_name AS product_1, op2.product_name AS product_2,
           COUNT(*) AS times_ordered_together
    FROM OrderProducts op1
    JOIN OrderProducts op2 ON op1.order_id = op2.order_id
    AND op1.product_id < op2.product_id
    GROUP BY op1.product_name, op2.product_name
)
SELECT *
FROM ProductPairs
WHERE times_ordered_together > 15
ORDER BY times_ordered_together DESC;
```

	product_1	product_2	times_ordered_together
1	Electra Cruiser 1 (24-Inch) - 2016	Electra Girl's Hawaii 1 (16-inch) - 2015/2016	27
2	Trek Slash 8 27.5 - 2016	Electra Townie Original 21D - 2016	22
3	Electra Townie Original 21D - 2016	Electra Cruiser 1 (24-Inch) - 2016	22
4	Electra Girl's Hawaii 1 (16-inch) - 2015/2016	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	22
5	Electra Cruiser 1 (24-Inch) - 2016	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	21



# TOP PERFORMING STORE BY MONTHLY REVENUE

```
WITH MonthlyTop AS (
    SELECT
        TO_CHAR(O.Order_date, 'YYYY-MM') AS MONTH,
        S.Store_name,
        ROUND(SUM(Oi.Quantity * Oi.List_price * (1 - Oi.Discount)), 2) AS Total_Revenue,
        COUNT(DISTINCT O.Order_id) AS Total_Orders,
        ROUND(AVG(Oi.Quantity * Oi.List_price * (1 - Oi.Discount)), 2) AS Avg_Revenue,
        ROW_NUMBER() OVER (PARTITION BY TO_CHAR(O.Order_date, 'YYYY-MM'))
    ORDER BY SUM(Oi.Quantity * Oi.List_price * (1 - Oi.Discount)) DESC) AS rn
    FROM "Orders" O
    JOIN "Stores" S ON O.Store_id = S.Store_id
    JOIN "Order_Items" Oi ON O.Order_id = Oi.Order_id
    GROUP BY MONTH, S.Store_name
)
SELECT MONTH, Store_name, Total_Orders, Total_Revenue, Avg_Revenue
FROM MonthlyTop
WHERE rn = 1
ORDER BY MONTH;
```

	month	store_name	total_orders	total_revenue	avg_revenue
1	2016-01	Baldwin Bikes	34	132894.3	1265.66
2	2016-02	Baldwin Bikes	35	102201.91	1032.34
3	2016-03	Baldwin Bikes	39	110338.79	1161.46



# YEARLY SALES AND YOY GROWTH

```
WITH YearlySales AS (
    SELECT EXTRACT(YEAR FROM o.order_date) AS order_year,
    ROUND(SUM(oi.quantity * oi.list_price * (1 - oi.discount)), 2) AS total_sales
    FROM "Orders" o
    JOIN "Order_Items" oi ON o.order_id = oi.order_id
    GROUP BY order_year
)
SELECT order_year, total_sales,
    COALESCE(
        ROUND((total_sales - LAG(total_sales) OVER (ORDER BY order_year)) /
        LAG(total_sales) OVER (ORDER BY order_year) * 100, 2), 0) AS yoy_growth
FROM YearlySales
ORDER BY order_year;
```

	order_year	total_sales	yoy_growth
1	2016	2427378.53	0
2	2017	3447208.24	42.01
3	2018	1814529.79	-47.36



# THE 15 LOWEST STOCKED PRODUCTS FOR EACH STORE

```
WITH StockRanking AS (
    SELECT
        Stk.Store_id, Stk.Product_id, Stk.Quantity,
        ROW_NUMBER() OVER (PARTITION BY Stk.Store_id ORDER BY Stk.Quantity ASC) AS RowNum
    FROM "Stocks" Stk
)
SELECT
    St.Store_name, P.Product_name, SR.Quantity
FROM StockRanking SR
JOIN "Stores" St ON SR.Store_id = St.Store_id
JOIN "Products" P ON SR.Product_id = P.Product_id
WHERE SR.RowNum <= 15
ORDER BY St.Store_name, SR.Quantity DESC;
```

	store_name	product_name	quantity
1	Baldwin Bikes	Trek Remedy 29 Carbon Frameset - 2016	1
2	Baldwin Bikes	Electra Townie Original 21D Ladies' - 2018	1
3	Baldwin Bikes	Trek 820 - 2018	1
4	Baldwin Bikes	Heller Shagamaw GX1 - 2018	1
5	Baldwin Bikes	Heller Shagamaw Frame - 2016	1
6	Baldwin Bikes	Electra Townie Commute Go! - 2018	0



# TOTAL ORDERS BY DAY OF THE WEEK

```
SELECT
    MOD(EXTRACT(DOW FROM order_date)::INT + 6, 7) + 1 AS day_num,
    TO_CHAR(order_date, 'FMDay') AS day_of_week,
    COUNT(order_id) AS total_orders
FROM "Orders"
GROUP BY day_num, TO_CHAR(order_date, 'FMDay')
ORDER BY day_num;
```

	day_num	day_of_week	total_orders
1	1	Monday	237
2	2	Tuesday	213
3	3	Wednesday	219
4	4	Thursday	232
5	5	Friday	219
6	6	Saturday	229
7	7	Sunday	266



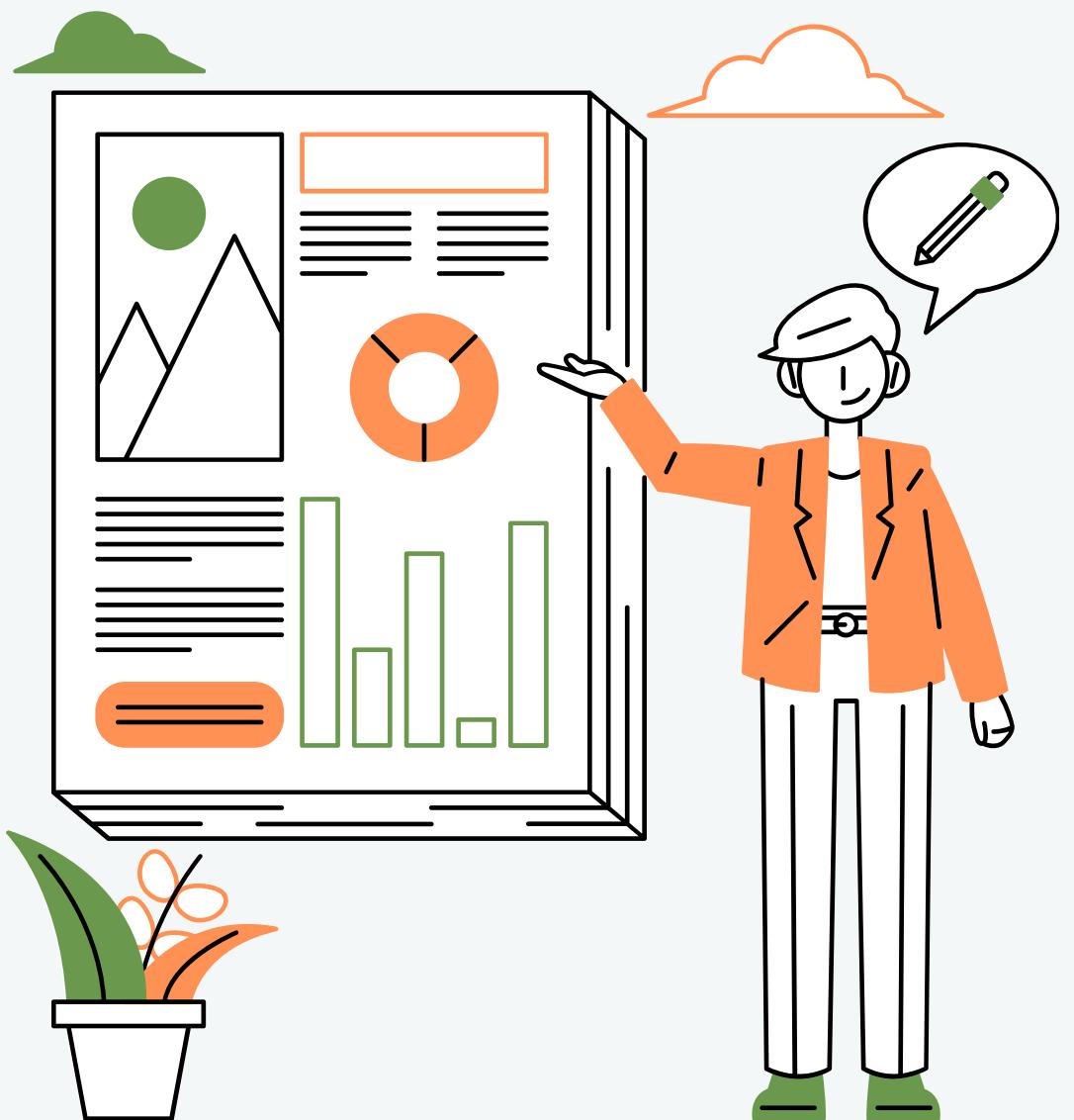
# MOST EXPENSIVE PRODUCT PER BRAND

```
SELECT brand_name, product_name, list_price
FROM (
    SELECT b.brand_name, p.product_name, p.list_price,
           RANK() OVER (PARTITION BY b.brand_id ORDER BY p.list_price DESC) AS rnk
    FROM "Brands" b
    JOIN "Products" p ON b.brand_id = p.brand_id
)
WHERE rnk = 1;
```

	brand_name	product_name	list_price
1	Electra	Electra Townie Commute Go! - 2018	2999.99
2	Electra	Electra Townie Commute Go! Ladies' - 2018	2999.99
3	Electra	Electra Townie Commute Go! - 2018	2999.99
4	Electra	Electra Townie Commute Go! Ladies' - 2018	2999.99
5	Haro	Haro Shift R3 - 2017	1469.99
6	Heller	Heller Shagamaw GX1 - 2018	2599
7	Heller	Heller Bloodhound Trail - 2018	2599



# 🔍 INSIGHTS SUMMARY



THIS PROJECT EXPLORES THE BIKESTORES DATABASE TO BETTER UNDERSTAND HOW THE COMPANY PERFORMS IN SALES, PRODUCT MANAGEMENT, CUSTOMER LOYALTY, AND STAFF EFFICIENCY.

THROUGH A SERIES OF SQL QUERIES, IT IDENTIFIES THE STORES THAT GENERATE THE MOST REVENUE, THE PRODUCTS THAT SELL THE FASTEST, AND THE CUSTOMERS WHO RETURN TO MAKE REPEAT PURCHASES. IT ALSO COMPARES STAFF PERFORMANCE AND ANALYZES DISCOUNT PRACTICES TO SEE HOW THEY AFFECT TOTAL SALES.

OVERALL, THE PROJECT PROVIDES A SIMPLE BUT COMPLETE SQL-BASED OVERVIEW OF THE COMPANY'S PERFORMANCE, HELPING IMPROVE BUSINESS PLANNING, CUSTOMER ENGAGEMENT, AND INVENTORY DECISIONS.

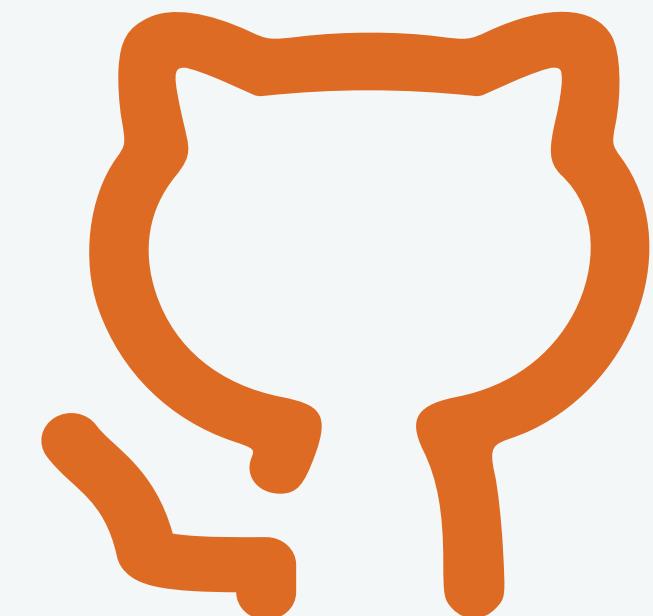


## TECH STACK USED

**TOOL:** POSTGRESQL

**SKILLS:** WINDOW FUNCTIONS, JOINS, AGGREGATIONS,  
SUBQUERIES, CTEs

**FOCUS:** SALES PERFORMANCE, INVENTORY  
OPTIMIZATION, CUSTOMER INSIGHTS



## PROJECT REPOSITORY

YOU CAN EXPLORE THE FULL SQL SCRIPTS,  
DATASET, AND DOCUMENTATION HERE:

**GITHUB:**

[HTTPS://GITHUB.COM/CFRVSLMZ/BIKESTOREANALYSISPOSTGRESQL](https://github.com/cfrvslmz/bikestoreanalysispostgresql)

