# SQL ANALYSIS FOR JENSON USA

# PROJECT

Presented by: K. K. Sahani

# OVERVIEW

01 About project

02 SQL query

03 result

04 contact

# About Project

## Overview:

This project focuses on analyzing a dataset from Jenson USA to derive actionable insights. Using SQL, it answers key questions about customer purchasing patterns, product sales, staff performance, and inventory management, helping to identify trends for better business decisions.

## Key Analysis Areas:

**Customer Behavior**: Identifying top spenders and understanding their patterns.

**Staff Performance**: Highlighting top performers and underperformers.

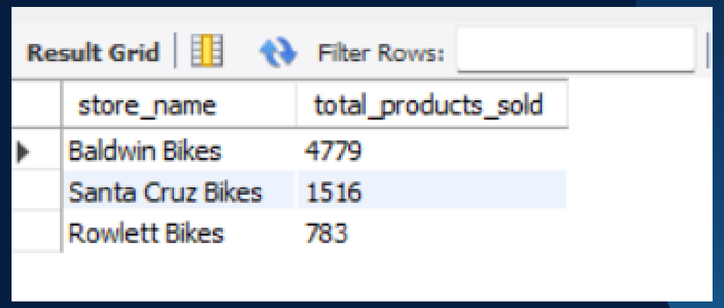
**Inventory Management:** Evaluating product sales, stock levels, and slow-moving items.

**Store Operations:** Analyzing store performance based on sales and orders.

SQL queries will provide insights to optimize operations, increase sales, and enhance customer satisfaction.

# Q1. Find the total number of products sold by each store along with the store name.

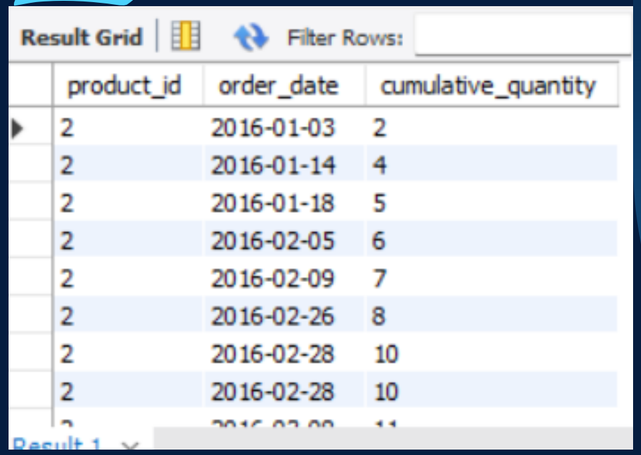
```sql
-- Q1.  Find the total number of products sold by each store along with the store name.  write sql querry
SELECT
    s.store_name,
    SUM(oi.quantity) AS total_products_sold
FROM
    stores s
JOIN
    orders o ON s.store_id = o.store_id
JOIN
    order_items oi ON o.order_id = oi.order_id
GROUP BY
    s.store_name
ORDER BY
    total_products_sold DESC; -- Optional: Add ordering for better insight
```




| store_name | total_products_sold |
|---|---|
| Baldwin Bikes | 4779 |
| Santa Cruz Bikes | 1516 |
| Rowlett Bikes | 783 |

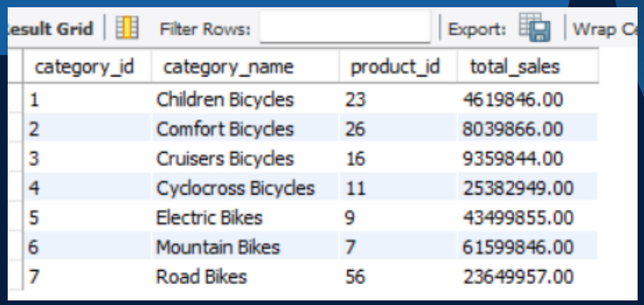
# Q2. Calculate the cumulative sum of quantities sold for each product over time

```sql
--- Q2.Calculate the cumulative sum of quantities sold for each product over time.
SELECT
    oi.product_id,
    o.order_date,
    SUM(oi.quantity) OVER (
        PARTITION BY oi.product_id
        ORDER BY o.order_date
    ) AS cumulative_quantity
FROM
    order_items oi
JOIN
    orders o ON oi.order_id = o.order_id
ORDER BY
    oi.product_id, o.order_date;
```



# Q3.Find the product with the highest total sales (quantity * price) for each category.

```
-- Q3.Find the product with the highest total sales (quantity * price) for each category.
WITH ProductSales AS (
    SELECT
        c.category_id,
        c.category_name,
        p.product_id,
        SUM(oi.quantity * oi.list_price) AS total_sales
    FROM
        order_items oi
    JOIN
        products p ON oi.product_id = p.product_id
    JOIN
        categories c ON p.category_id = c.category_id
    GROUP BY
        c.category_id, c.category_name, p.product_id
),
RankedSales AS (
    SELECT
        category_id,
        category_name,
        product_id,
        total_sales,
        ROW_NUMBER() OVER (
            PARTITION BY category_id
            ORDER BY total_sales DESC
        ) AS sales_rank
    FROM
        ProductSales
)
SELECT
    category_id,
    category_name,
    product_id,
    total_sales
FROM
    RankedSales
WHERE
    sales_rank = 1;
```

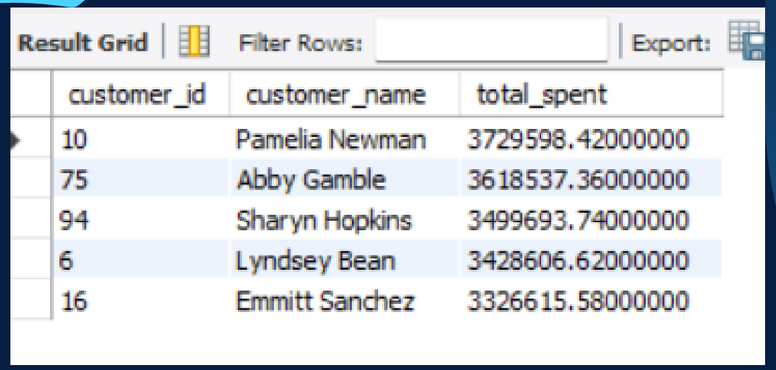


Result Grid | Filter Rows: | Export: | Wrap C

| category_id | category_name | product_id | total_sales |
|---|---|---|---|
| 1 | Children Bicycles | 23 | 4619846.00 |
| 2 | Comfort Bicycles | 26 | 8039866.00 |
| 3 | Cruisers Bicycles | 16 | 9359844.00 |
| 4 | Cyclocross Bicycles | 11 | 25382949.00 |
| 5 | Electric Bikes | 9 | 43499855.00 |
| 6 | Mountain Bikes | 7 | 61599846.00 |
| 7 | Road Bikes | 56 | 23649957.00 |

# Q4. Find the customer who spent the most money on orders

```
--
Q4.Find the customer who spent the most money on orders.
WITH CustomerSpending AS (
    SELECT
        o.customer_id,
        CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
        SUM((oi.quantity * oi.list_price) - (oi.quantity * oi.list_price * oi.discount / 100)) AS total_spent
    FROM
        orders o
    JOIN
        order_items oi ON o.order_id = oi.order_id
    JOIN
        customers c ON o.customer_id = c.customer_id
    GROUP BY
        o.customer_id, c.first_name, c.last_name
)
SELECT
    customer_id,
    customer_name,
    total_spent
FROM
    CustomerSpending
ORDER BY
    total_spent DESC
LIMIT 5;
```



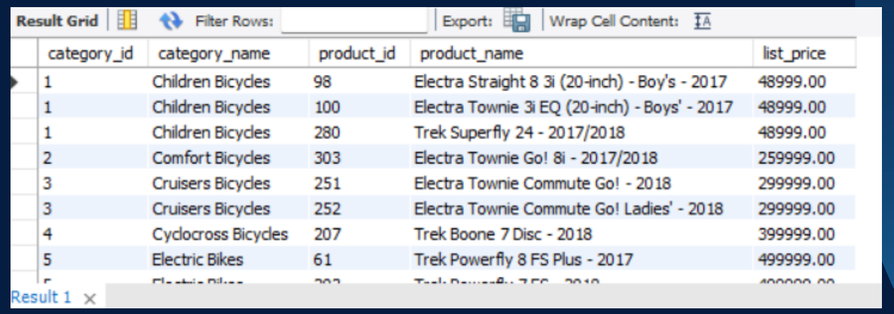
Result Grid | Filter Rows: | Export:

| customer_id | customer_name | total_spent |
|---|---|---|
| 10 | Pamelia Newman | 3729598.42000000 |
| 75 | Abby Gamble | 3618537.36000000 |
| 94 | Sharyn Hopkins | 3499693.74000000 |
| 6 | Lyndsey Bean | 3428606.62000000 |
| 16 | Emmitt Sanchez | 3326615.58000000 |

# Q5. Find the highest-priced product for each category name.

Q5.Find the highest-priced product for each category name.

```
SELECT
    c.category_id,
    c.category_name,
    p.product_id,
    p.product_name,
    p.list_price
FROM
    products p
JOIN
    categories c ON p.category_id = c.category_id
WHERE
    p.list_price = (
        SELECT
            MAX(p2.list_price)
        FROM
            products p2
        WHERE
            p2.category_id = p.category_id
    );
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| category_id | category_name | product_id | product_name | list_price |
|---|---|---|---|---|
| 1 | Children Bicycles | 98 | Electra Straight 8 3i (20-inch) - Boy's - 2017 | 48999.00 |
| 1 | Children Bicycles | 100 | Electra Townie 3i EQ (20-inch) - Boys' - 2017 | 48999.00 |
| 1 | Children Bicycles | 280 | Trek Superfly 24 - 2017/2018 | 48999.00 |
| 2 | Comfort Bicycles | 303 | Electra Townie Go! 8i - 2017/2018 | 259999.00 |
| 3 | Cruisers Bicycles | 251 | Electra Townie Commute Go! - 2018 | 299999.00 |
| 3 | Cruisers Bicycles | 252 | Electra Townie Commute Go! Ladies' - 2018 | 299999.00 |
| 4 | Cyclocross Bicycles | 207 | Trek Boone 7 Disc - 2018 | 399999.00 |
| 5 | Electric Bikes | 61 | Trek Powerfly 8 FS Plus - 2017 | 499999.00 |

Result 1

# Q6.Find the total number of orders placed by each customer per store.

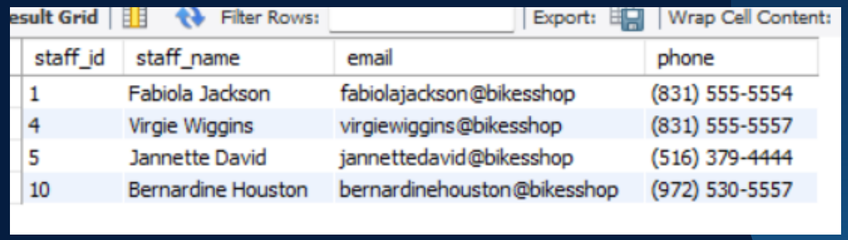
```sql
-- Q6.Find the total number of orders placed by each customer per store.
SELECT
    o.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    o.store_id,
    s.store_name,
    COUNT(o.order_id) AS total_orders
FROM
    orders o
JOIN
    customers c ON o.customer_id = c.customer_id
JOIN
    stores s ON o.store_id = s.store_id
GROUP BY
    o.customer_id, c.first_name, c.last_name, o.store_id, s.store_name
ORDER BY
    o.customer_id, o.store_id;
```

esult Grid | Filter Rows: | Export: | Wrap Cell C

| customer_id | customer_name | store_id | store_name | total_orders |
|---|---|---|---|---|
| 1 | Debra Burks | 2 | Baldwin Bikes | 3 |
| 2 | Kasha Todd | 1 | Santa Cruz Bikes | 3 |
| 3 | Tameka Fisher | 1 | Santa Cruz Bikes | 3 |
| 4 | Daryl Spence | 2 | Baldwin Bikes | 3 |
| 5 | Charolette Rice | 1 | Santa Cruz Bikes | 3 |
| 6 | Lyndsey Bean | 2 | Baldwin Bikes | 3 |
| 7 | Latasha Hays | 2 | Baldwin Bikes | 3 |
| 8 | Jacquline Duncan | 2 | Baldwin Bikes | 3 |

# Q7.Find the names of staff members who have not made any sales

```sql
-- Q7.Find the names of staff members who have not made any sales.
SELECT
    s.staff_id,
    CONCAT(s.first_name, ' ', s.last_name) AS staff_name,
    s.email,
    s.phone
FROM
    staffs s
LEFT JOIN
    orders o ON s.staff_id = o.staff_id
WHERE
    o.order_id IS NULL;
```

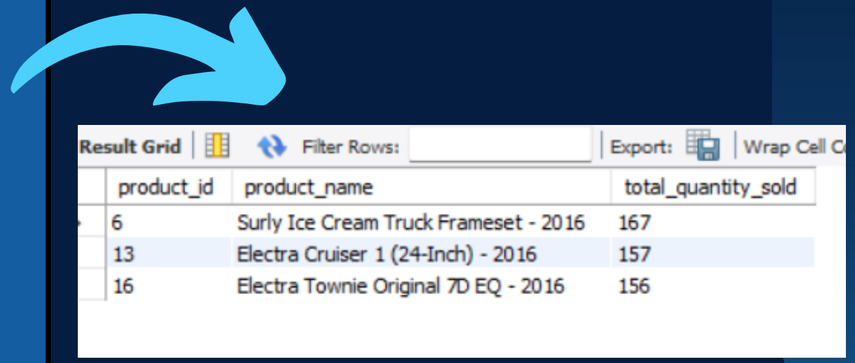


esult Grid | Filter Rows: | Export: | Wrap Cell Content:

| staff_id | staff_name | email | phone |
|---|---|---|---|
| 1 | Fabiola Jackson | fabiolajackson@bikesshop | (831) 555-5554 |
| 4 | Virgie Wiggins | virgiewiggins@bikesshop | (831) 555-5557 |
| 5 | Jannette David | jannettedavid@bikesshop | (516) 379-4444 |
| 10 | Bernardine Houston | bernardinehouston@bikesshop | (972) 530-5557 |

# Q8.Find the top 3 most sold products in terms of quantity.

```
-- Q8.Find the top 3 most sold products in terms of quantity.
 SELECT
    p.product_id,
    p.product_name,
    SUM(oi.quantity) AS total_quantity_sold
FROM
    order_items oi
JOIN
    products p ON oi.product_id = p.product_id
GROUP BY
    p.product_id, p.product_name
ORDER BY
    total_quantity_sold DESC
LIMIT 3;
```

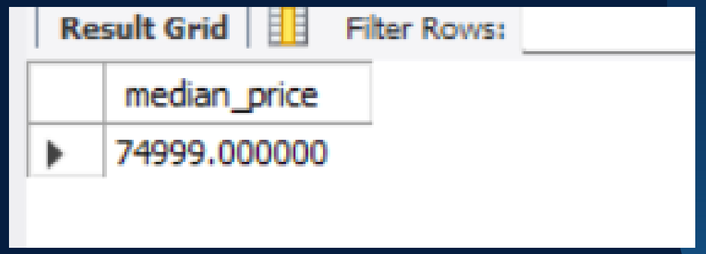


Result Grid | Filter Rows: | Export: | Wrap Cell C

| product_id | product_name | total_quantity_sold |
|---|---|---|
| 6 | Surly Ice Cream Truck Frameset - 2016 | 167 |
| 13 | Electra Cruiser 1 (24-Inch) - 2016 | 157 |
| 16 | Electra Townie Original 7D EQ - 2016 | 156 |

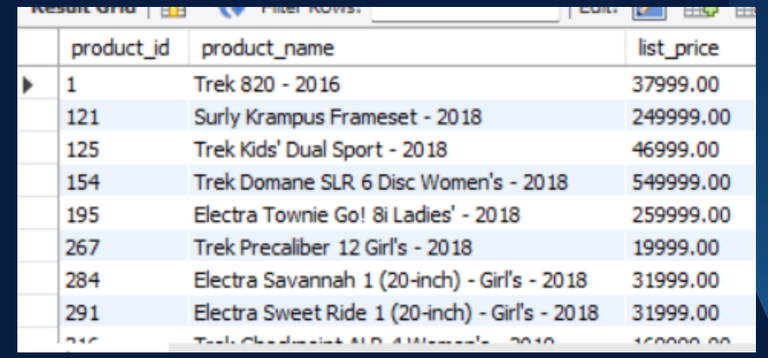
# Q9.Find the median value of the price list.

```
-- Q9.Find the median value of the price list.
WITH RankedPrices AS (
    SELECT
        list_price,
        ROW_NUMBER() OVER (ORDER BY list_price) AS row_num,
        COUNT(*) OVER () AS total_count
    FROM
        products
)
SELECT
    AVG(list_price) AS median_price
FROM
    RankedPrices
WHERE
    row_num IN (FLOOR((total_count + 1) / 2), CEIL((total_count + 1) / 2));
```



# Q10. List all products that have never been ordered.(use Exists)

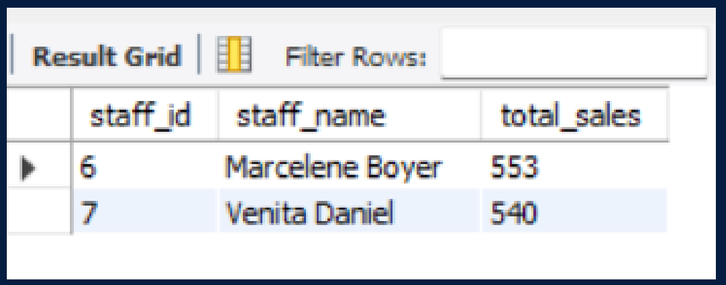
```
- Q10.List all products that have never been ordered.(use Exists)
SELECT
    p.product_id,
    p.product_name,
    p.list_price
FROM
    products p
WHERE
    NOT EXISTS (
        SELECT 1
        FROM order_items oi
        WHERE oi.product_id = p.product_id
                                                    );
```



| product_id | product_name | list_price |
|---|---|---|
| 1 | Trek 820 - 2016 | 37999.00 |
| 121 | Surly Krampus Frameset - 2018 | 249999.00 |
| 125 | Trek Kids' Dual Sport - 2018 | 46999.00 |
| 154 | Trek Domane SLR 6 Disc Women's - 2018 | 549999.00 |
| 195 | Electra Townie Go! 8i Ladies' - 2018 | 259999.00 |
| 267 | Trek Precaliber 12 Girl's - 2018 | 19999.00 |
| 284 | Electra Savannah 1 (20-inch) - Girl's - 2018 | 31999.00 |
| 291 | Electra Sweet Ride 1 (20-inch) - Girl's - 2018 | 31999.00 |

# Q11. List the names of staff members who have made more sales than the average number of sales by all staff members.

```
--
 Q11.List the names of staff members who have made more sales than the average number of sales by all staff members.
WITH StaffSales AS (
    SELECT
        s.staff_id,
        CONCAT(s.first_name, ' ', s.last_name) AS staff_name,
        COUNT(o.order_id) AS total_sales
    FROM
        staffs s
    JOIN
        orders o ON s.staff_id = o.staff_id
    GROUP BY
        s.staff_id, s.first_name, s.last_name
),
AverageSales AS (
    SELECT
        AVG(total_sales) AS avg_sales
    FROM
        StaffSales
)
SELECT
    ss.staff_id,
    ss.staff_name,
    ss.total_sales
FROM
    StaffSales ss
CROSS JOIN
    AverageSales a
WHERE
    ss.total_sales > a.avg_sales;
```



Result Grid | Filter Rows:

| staff_id | staff_name | total_sales |
|---|---|---|
| 6 | Marcelene Boyer | 553 |
| 7 | Venita Daniel | 540 |

# Q12.Identify the customers who have ordered all types of products (i.e., from every category).

```
t
-- Q12.Identify the customers who have ordered all types of products (i.e., from every category).

SELECT c.customer_id, c.first_name, c.last_name
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
JOIN categories cat ON p.category_id = cat.category_id
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(DISTINCT cat.category_id) = (SELECT COUNT(*)
FROM categories);
```

| customer_id | first_name | last_name |
|---|---|---|
| 9 | Genoveva | Baldwin |

# THANK YOU

## CONNECT WITH US.

+91-7257964940

krishnakumarsahani.bhumca22@gmail.com