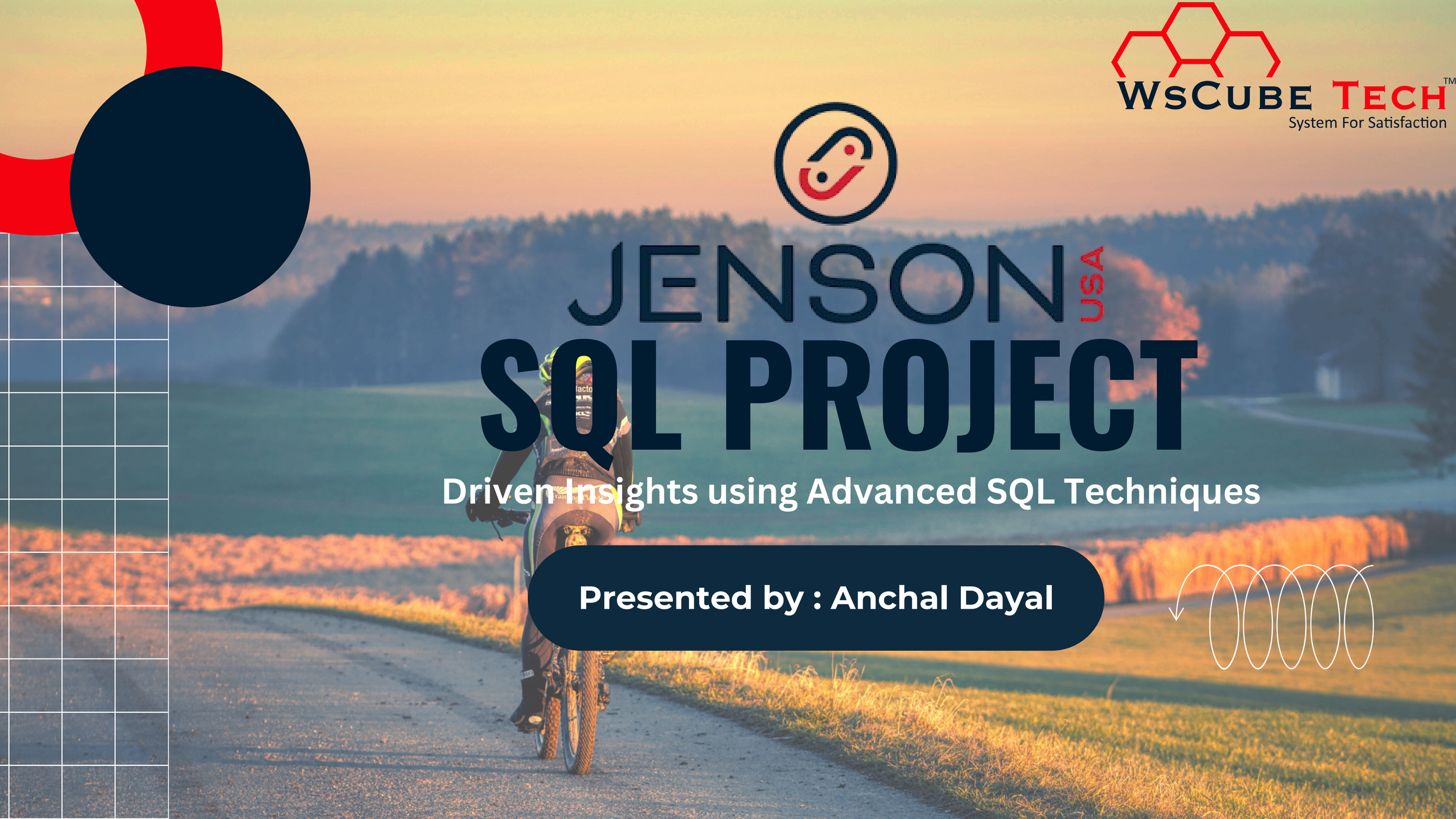
WsCube Tech™
System For Satisfaction



JENSON USA SQL PROJECT

Driven Insights using Advanced SQL Techniques

Presented by : Anchal Dayal

PROJECT : JENSON USA

My job has been to study and optimize the database to analyze data and derive actionable insights for improving business performance

Here are the findings, actions taken for improvement, and insights

Data Source:
Jenson USA Data

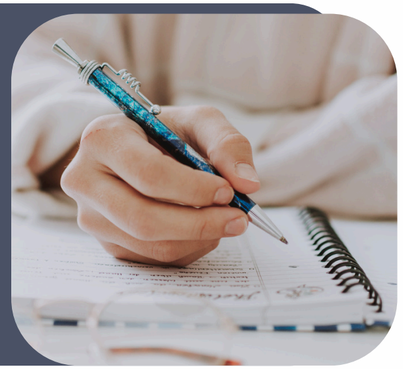
Tools Used:
MySQL

Skills enforced
- SQL Query Crafting
- Data Analysis
- Insights Derivation

What We Uncover

01

Customer Behavior

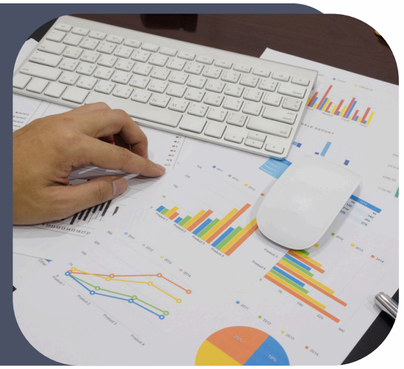


02

Inventory Management

03

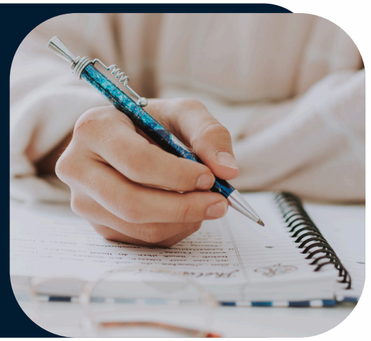
Staff Performance



04

Store Operations

Customer Behavior



Find the customer who spent the most money on orders

```sql
WITH CustomerSpending AS (
    SELECT c.customer_id,
    SUM(oi.quantity * oi.list_price)
    AS total_spent
    FROM
        customers c JOIN orders o ON c.customer_id = o.customer_id
        JOIN order_items oi ON o.order_id = oi.order_id
        GROUP BY c.customer_id
)
SELECT c.first_name, c.last_name, cs.total_spent
FROM
  CustomerSpending cs JOIN customers c ON cs.customer_id = c.customer_id
  ORDER BY cs.total_spent DESC LIMIT 1;
```

Customer Behavior

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
WHERE NOT EXISTS (
    SELECT category_id
    FROM
        categories cat
    EXCEPT
    SELECT p.category_id
    FROM
        orders o JOIN
        order_items oi ON o.order_id = oi.order_id JOIN
        products p ON oi.product_id p.product_id
    WHEREo.customer_id = c.customer_id );
```

Inventory Management

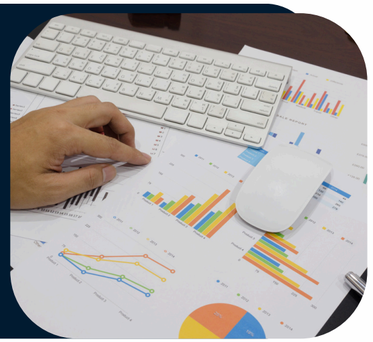
Find the names of staff members who have not made any sales

```sql
SELECT
    st.first_name,
    st.last_name
FROM
    staffs st
LEFT JOIN
    orders o ON st.staff_id = o.staff_id
WHERE
    o.order_id IS NULL;
```

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 staff_id, COUNT(order_id) AS total_sales
    FROM  orders
    GROUP BY  staff_id ),
AvgSales AS (
    SELECT   AVG( total_sales) AS
 avg_sales FROM    StaffSales )
SELECT st.first_name, st.last_name
FROM StaffSales ss
JOIN  staffs st ON ss.staff_id = st.staff_id
JOIN   AvgSales avg ON ss.total_sales > avg.avg_sales;
```

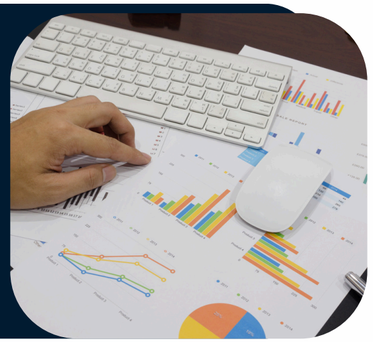
Staff Performance



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

```
select
order_items.product_id, orders.order_date, sum(order_items.quantity)
over(partition by order_items.product_id order by
orders.order_date) as cumulative_sum
from order_items
join orders on order_items.order_id = orders.order_id;
```

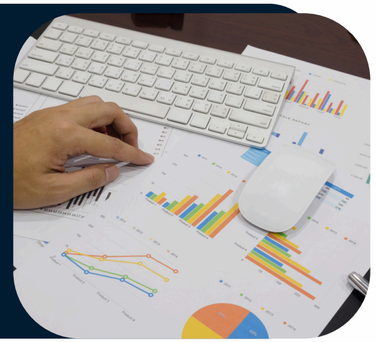
Staff Performance



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

```sql
WITH SalesByProduct AS (
    SELECT  p.category_id, oi.product_id, SUM(p.list_price * oi.quantity) AS total_sales
    FROM products p
    JOIN  order_items oi ON p.product_id = oi.product_id
    GROUP BY  p.category_id, oi.product_id
),
RankedSales AS (
    SELECT sbp.category_id, sbp.product_id,  sbp.total_sales,
        DENSE_RANK() OVER (PARTITION BY sbp.category_id ORDER BY sbp.total_sales DESC) AS rank_
    FROM SalesByProduct sbp
)
SELECT rs.category_id, rs.product_id, rs.total_sales FROM RankedSales rs WHERE rs.rank_ = 1;
```

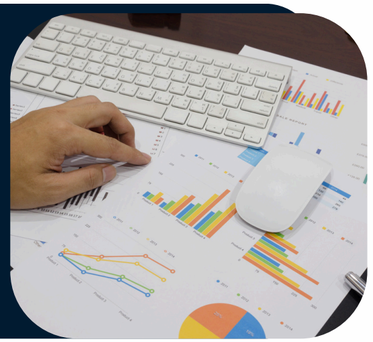
Staff Performance



Find the highest-priced product for each category name

```
WITH MaxPricePerCategory AS (
    SELECT category_id,
MAX(list_price) AS max_price
    FROM products
    GROUP BY category_id
)
SELECT  c.category_name, p.product_name, p.list_price
FROM MaxPricePerCategory mp
JOIN products p ON mp.category_id = p.category_id AND mp.max_price = p.list_price
JOIN
    categories c ON p.category_id = c.category_id;
```

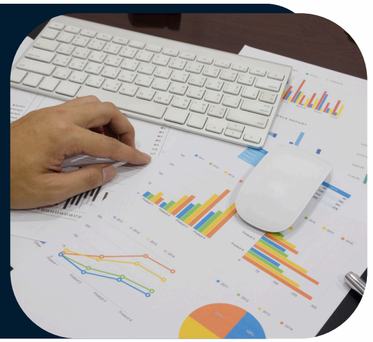
Staff Performance



List all products that have never been ordered. (Use EXISTS)

```
SELECT p.product_name
FROM products p
WHERE
    NOT EXISTS (
        SELECT 1
        FROM order_items oi
        WHERE oi.product_id = p.product_id
 );
```

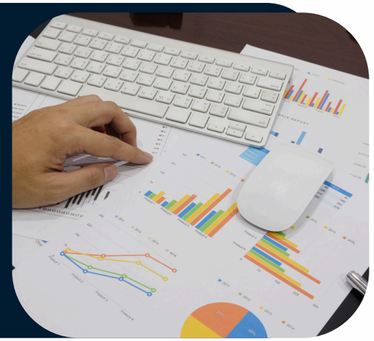
Staff Performance



Find the top 3 most sold products in terms of quantity

```
select * from( select
product_id, sum(quantity),
rank() over (order by sum(quantity) desc) rank_
from order_items
group by product_id ) as rank_t
where rank_ <= "3" ;
```

Staff Performance



Find the median value of the price list

```
WITH m AS ( SELECT
        list_price,
        ROW_NUMBER() OVER (ORDER BY list_price) AS rn,
        COUNT(*) OVER () AS cn
    FROM
        order_items
)
SELECT
    CASE
        WHEN cn % 2 = 0 THEN (SELECT AVG(list_price)
            FROM m
            WHERE rn IN (cn / 2, cn / 2 + 1))
        ELSE (SELECT list_price
            FROM m
            WHERE rn = (cn + 1) / 2)
    END AS median
    FROM m
    LIMIT 1;
```

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

Store Operations

```sql
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;
```

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

```sql
SELECT
    customer_id,
     store_id,
     COUNT(order_id) AS total_orders
FROM  orders
GROUP BY customer_id,  store_id
ORDER BY  customer_id, store_id;
```

Social media *creates communities*, not markets.