JENSON USA

ADVANCE SQL PROJECT

PRESENTED BY: AMANJOT SINGH

PROJECT INTRODUCTION

Jenson USA, America’s leading online and retail cycling store, strives to deliver exceptional value to cycling enthusiasts while optimizing operations through data-driven insights. This project analyzes Jenson USA’s SQL dataset to uncover patterns in customer behavior, staff performance, inventory trends, and store efficiency. The findings will guide Jenson USA in enhancing customer satisfaction, improving operational effectiveness, and solidifying its position as a trusted name in the cycling industry.

JENSON USA

SQL QURIES USED FOR ANALYSIS

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

```sql
SELECT
    stores.store_name,
    SUM(order_items.quantity) AS total_products_sold
FROM
    order_items
JOIN
    orders ON order_items.order_id = orders.order_id
JOIN
    stores ON orders.store_id = stores.store_id
GROUP BY
    stores.store_name;
```

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

```sql
WITH product_sales AS (
    SELECT products.product_id AS product_id,
    orders.order_date AS order_date,
    SUM(order_items.quantity) AS quantity
    FROM orders
    JOIN order_items ON orders.order_id = order_items.order_id
    JOIN products ON order_items.product_id = products.product_id
    GROUP BY products.product_id, orders.order_date
)
SELECT
product_id, order_date, quantity, SUM(quantity) OVER
(PARTITION BY product_id ORDER BY order_date) AS cumulative_quantity
FROM product_sales;
```

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

```sql
WITH category_sales AS (
    SELECT products.category_id, products.product_id,
    SUM(order_items.quantity * (products.list_price - order_items.discount))
    AS total_sales FROM products
    JOIN order_items ON products.product_id = order_items.product_id
    GROUP BY products.category_id, products.product_id
)
SELECT
    category_id, product_id, total_sales
FROM (
    SELECT
    *,
    RANK() OVER (PARTITION BY category_id ORDER BY total_sales DESC) AS rnk
    FROM
    category_sales
) ranked_sales WHERE rnk = 1;
```

Find the customer who spent the most money on orders.

```sql
WITH customer_sales AS (
    SELECT
    orders.customer_id,
    CONCAT(customers.first_name, " ", customers.last_name) AS customer_name,
    SUM(order_items.quantity * (order_items.list_price - order_items.discount))
    AS total_spent FROM orders
    JOIN order_items ON orders.order_id = order_items.order_id
    JOIN customers ON orders.customer_id = customers.customer_id
    GROUP BY orders.customer_id, customer_name
)
SELECT
    customer_id, customer_name, total_spent
FROM (SELECT *,
    RANK() OVER (ORDER BY total_spent DESC) AS rnk
    FROM customer_sales) ranked_sales
    WHERE rnk = 1;
```

Find the highest-priced product for each category.

```
SELECT
    category_id, category_name,
    product_name, list_price
FROM (
    SELECT categories.category_id,
    categories.category_name,
    products.product_name,
    products.list_price,
    RANK() OVER (PARTITION BY categories.category_id
    ORDER BY products.list_price DESC) AS rnk
    FROM categories
    JOIN products ON products.category_id = categories.category_id
)   ranked_products WHERE
    rnk = 1;
```

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

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

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

```sql
SELECT
    staffs.staff_id,
    CONCAT(staffs.first_name, " ", staffs.last_name) AS staff_name
FROM
    staffs
WHERE
    NOT EXISTS (
        SELECT staff_id
        FROM orders
        WHERE staffs.staff_id = orders.staff_id
    );
```

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

```sql
SELECT
    product_name
FROM (
    SELECT
        products.product_id,
        products.product_name,
        SUM(order_items.quantity) AS total_quantity,
        RANK() OVER (ORDER BY SUM(order_items.quantity) DESC) AS rnk
    FROM products
    JOIN order_items ON products.product_id = order_items.product_id
    GROUP BY products.product_id, products.product_name
) ranked_products
WHERE rnk <= 3;
```

Find the median value of the price list.

```sql
WITH price_data AS (
    SELECT list_price,
    ROW_NUMBER() OVER (ORDER BY list_price) AS rn,
    COUNT(*) OVER () AS total_count
    FROM products)
SELECT
    CASE
        WHEN total_count % 2 = 0 THEN
            (SELECT AVG(list_price)
             FROM price_data
             WHERE rn IN (total_count / 2, (total_count / 2) + 1))
        ELSE
            (SELECT list_price
             FROM price_data
             WHERE rn = (total_count + 1) / 2)
    END AS median_price FROM
    price_data LIMIT 1;
```

List all products that have never been ordered.

```sql
SELECT
    products.product_id,
    products.product_name
FROM
    products
WHERE
    NOT EXISTS (
        SELECT
            product_id
        FROM
            order_items
        WHERE
            products.product_id = order_items.product_id
    );
```

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

```sql
WITH staff_sales AS (SELECT
        staffs.staff_id,
        CONCAT(staffs.first_name, " ", staffs.last_name) AS staff_name,
        SUM(order_items.quantity) AS total_sales
    FROM staffs
    JOIN orders ON staffs.staff_id = orders.staff_id
    JOIN order_items ON orders.order_id = order_items.order_id
    GROUP BY staffs.staff_id, staff_name),
average_sales AS (
    SELECT AVG(total_sales) AS avg_sales
    FROM staff_sales
)
SELECT staff_name, total_sales
FROM staff_sales, average_sales
WHERE staff_sales.total_sales > average_sales.avg_sales;
```

Identify the customers who have ordered all types of products

```sql
WITH category_counts AS (
    SELECT customer_id, COUNT(DISTINCT category_id) AS category_count
    FROM orders
    JOIN order_items ON orders.order_id = order_items.order_id
    JOIN products ON order_items.product_id = products.product_id
    GROUP BY customer_id),
total_categories AS (
    SELECT COUNT(DISTINCT category_id) AS total_category_count
    FROM products)
SELECT
    customers.customer_id,
    CONCAT(customers.first_name, " ", customers.last_name) AS customer_name
FROM customers
JOIN category_counts ON customers.customer_id = category_counts.customer_id
JOIN total_categories
WHERE category_counts.category_count = total_categories.total_category_count;
```

THANK YOU

Like, Comment, Share, and Follow!

Connect with me on LinkedIn

Amanjot Singh