JENSON USA

ANALYZING JENSON DATA

SQL Queries for Key Insight

Rahul Nagra

ABOUT JENSON USA

JENSON USA

Jenson USA is a prominent online retailer specializing in bicycles, bike components, accessories, and gear for cycling enthusiasts. Established in 1994, the company is known for its wide product selection, competitive pricing, and excellent customer service. Catering to mountain bikers, road cyclists, and casual riders, Jenson USA also offers expert advice and resources to help customers find the right products for their needs. It has a strong reputation in the cycling community for quality and reliability.

TO LOAD THE DATA

Rahul Nagra

QUERIES

1	-- Find the total number of products sold by each store along with the store name.
2	-- Calculate the cumulative sum of quantities sold for each product over time.
3	-- Find the product with the highest total sales (quantity * price) for each category.
4	-- Find the customer who spent the most money on orders.
5	-- Find the highest-priced product for each category name.
6	-- Find the total number of orders placed by each customer per store.

QUERIES

7	-- Find the names of staff members who have not made any sales.
8	-- Find the top 3 most sold products in terms of quantity.
9	-- Find the median value of the price list.
10	-- List all products that have never been ordered.(use Exists)
11	--List the names of staff members who have made more sales than the average number of sales by all staff members.
12	-- Identify the customers who have ordered all types of products (i.e., from every category).

Q1:-

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

Syntax

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

Result

store_id	store_name	products_sold
1	Santa Cruz Bikes	1516
2	Baldwin Bikes	4779
3	Rowlett Bikes	783

Q2:-

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

Syntax

```
SELECT
products.product_name,
order_items.product_id,
orders.order_date,
order_items.quantity,
SUM(order_items.quantity)
over(partition by order_items.product_id order by orders.order_date)
as cum_quantity
FROM
order_items
LEFT JOIN
orders USING (order_id)
LEFT JOIN
products ON order_items.product_id = products.product_id;
```

Result

product_name	product_id	order_date	quantity	cum_quantity
Ritchey Timberwolf Frameset - 2016	2	2016-01-03	2	2
Ritchey Timberwolf Frameset - 2016	2	2016-01-14	2	4
Ritchey Timberwolf Frameset - 2016	2	2016-01-18	1	5
Ritchey Timberwolf Frameset - 2016	2	2016-02-05	1	6
Ritchey Timberwolf Frameset - 2016	2	2016-02-09	1	7
Ritchey Timberwolf Frameset - 2016	2	2016-02-26	1	8
Ritchey Timberwolf Frameset - 2016	2	2016-02-28	1	10
Ritchey Timberwolf Frameset - 2016	2	2016-02-28	1	10
Ritchey Timberwolf Frameset - 2016	2	2016-03-08	1	11

Q3:-

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

Syntax

```
with category_products_by_sales as (
SELECT
    products.category_id,
    categories.category_name,
    products.product_id,
    products.product_name,
    SUM(order_items.quantity * order_items.list_price) AS total_sales,
    rank() over(partition by products.category_id order by SUM(order_items.quantity * order_items.list_price) desc) as product_sales_rank
FROM
    order_items
        LEFT JOIN
    products USING (product_id)
    left join
    categories on products.category_id = categories.category_id
GROUP BY products.category_id, products.product_id)
select category_id, category_name, product_id, product_name, total_sales from category_products_by_sales where product_sales_rank = 1;
```

Result

category_id	category_name	product_id	product_name	total_sales
1	Children Bicycles	23	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	4619846.00
2	Comfort Bicycles	26	Electra Townie Original 7D EQ - 2016	8039866.00
3	Cruisers Bicycles	16	Electra Townie Original 7D EQ - 2016	9359844.00
4	Cyclocross Bicycles	11	Surly Straggler 650b - 2016	25382949.00
5	Electric Bikes	9	Trek Conduit+ - 2016	43499855.00
6	Mountain Bikes	7	Trek Slash 8 275 - 2016	61599846.00
7	Road Bikes	56	Trek Domane SLR 6 Disc - 2017	23649957.00

Q4:-

```
-- Find the customer who spent the most money on orders.
```

Syntax

```
SELECT
    customers.customer_id,
    CONCAT(customers.first_name, ' ', customers.last_name) name,
    SUM(order_items.list_price * order_items.quantity) total_sales
FROM
    order_items
        LEFT JOIN
    orders USING (order_id)
        LEFT JOIN
    customers ON orders.customer_id = customers.customer_id
GROUP BY customers.customer_id , CONCAT(customers.first_name, ' ', customers.last_name)
ORDER BY total_sales DESC
LIMIT 1;
```

Result

customer_id	name	total_sales
10	Pamelia Newman	3780184.00

Q5:-

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

Syntax

```
with product_rank_cte as (
SELECT
    c.category_id,
    c.category_name,
    p.product_id,
    p.product_name,
    p.list_price,
    rank() over (partition by c.category_id order by p.list_price desc) as prod_rank
FROM
    products p
        JOIN
    categories c USING (category_id))
SELECT category_id, category_name, product_id, product_name, list_price
FROM product_rank_cte
WHERE prod_rank = 1;
```

Result

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
5	Electric Bikes	203	Trek Powerfly 7 FS - 2018	499999.00

Q6:-

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

Syntax

```
SELECT
    stores.store_id,
    stores.store_name,
    customers.customer_id,
    customers.first_name,
    customers.last_name,
    COUNT(*) AS num_orders
FROM
    orders
        LEFT JOIN
    customers USING (customer_id)
        LEFT JOIN
    stores ON orders.store_id = stores.store_id
GROUP BY customers.customer_id , customers.first_name , customers.last_name , stores.store_id , stores.store_name
ORDER BY stores.store_id , num_orders DESC , customers.customer_id;
```

Result

store_id	store_name	customer_id	first_name	last_name	num_orders
1	Santa Cruz Bikes	2	Kasha	Todd	3
1	Santa Cruz Bikes	3	Tameka	Fisher	3
1	Santa Cruz Bikes	5	Charolette	Rice	3
1	Santa Cruz Bikes	24	Corene	Wall	3
1	Santa Cruz Bikes	30	Jamaal	Albert	3
1	Santa Cruz Bikes	31	Williemae	Holloway	3
1	Santa Cruz Bikes	32	Araceli	Golden	3
1	Santa Cruz Bikes	33	Deloris	Burke	3

Q7:-

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

Syntax

```
SELECT
    staff_id, first_name, last_name
FROM
    staffs
WHERE
    staff_id NOT IN (SELECT DISTINCT staff_id FROM orders);
```

Result

staff_id	first_name	last_name
1	Fabiola	Jackson
4	Virgie	Wiggins
5	Jannette	David
10	Bernardine	Houston
NULL	NULL	NULL

Q8:-

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

Syntax

```
SELECT
    p.product_id, product_name, SUM(quantity) total_quantity
FROM
    order_items o
        LEFT JOIN
    products p USING (product_id)
GROUP BY product_id
ORDER BY total_quantity DESC
LIMIT 3;
```

Result

product_id	product_name	total_quantity
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.

Syntax

```
with list_price_cte as (
SELECT
    list_price,
    ROW_NUMBER() OVER (ORDER BY list_price) AS price_rank,
    count(*) OVER() AS total_count
FROM
    products )
SELECT
    CASE
        WHEN total_count%2 = 0 THEN ( SELECT AVG(list_price)
                                      FROM list_price_cte
                                      WHERE price_rank IN ( total_count/2, total_count/2 + 1  )  )
        ELSE ( SELECT list_price
               FROM list_price_cte
               WHERE price_rank = (total_count + 1)/2  )
    END as list_price_median_value
FROM
    list_price_cte
limit 1;
```

Result

list_price_median_value
74999.00

Q10:-

```
-- List all products that have never been ordered.(use Exists)
```

Syntax

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

Result

product_id	product_name
1	Trek 820 - 2016
121	Surly Krampus Frameset - 2018
125	Trek Kids' Dual Sport - 2018
154	Trek Domane SLR 6 Disc Women's - 2018
195	Electra Townie Go! 8i Ladies' - 2018
267	Trek Precaliber 12 Girl's - 2018
284	Electra Savannah 1 (20-inch) - Girl's - 2018
291	Electra Sweet Ride 1 (20-inch) - Girl's - 2018
316	Trek Checkpoint ALR 4 Women's - 2019
317	Trek Checkpoint ALR 5 - 2019

Q11:-

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

Syntax

```
WITH staff_sales AS (
SELECT
    staff_id, SUM(quantity* list_price) AS sales
FROM
    orders o
        JOIN
    order_items ot USING (order_id)
GROUP BY staff_id
ORDER BY sales DESC )
SELECT
    staffs.staff_id, first_name, last_name, sales
FROM
    staff_sales
        LEFT JOIN
    staffs USING (staff_id)
WHERE
sales > (SELECT AVG(sales) FROM staff_sales);
```

Result

staff_id	first_name	last_name	sales
6	Marcelene	Boyer	293888873.00
7	Venita	Daniel	288735348.00

Q12:-

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

Syntax

```
SELECT
    orders.customer_id,
    customers.first_name,
    customers.last_name
FROM
    order_items
        JOIN
    orders USING (order_id)
        LEFT JOIN
    products ON order_items.product_id = products.product_id
        LEFT JOIN
    customers ON orders.customer_id = customers.customer_id
GROUP BY orders.customer_id , customers.first_name , customers.last_name
HAVING COUNT(DISTINCT products.category_id) = (SELECT
        COUNT(category_id)
    FROM
        categories)
ORDER BY orders.customer_id;
```

Result

customer_id	first_name	last_name
9	Genoveva	Baldwin

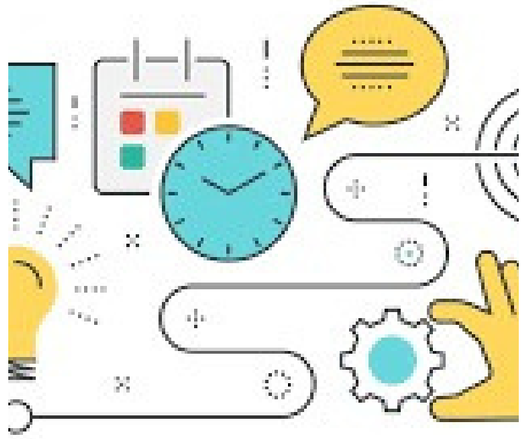
MEET WHAT WE HAVE LEARNED

In this project, we advanced our SQL skills with complex filtering and aggregate functions, sharpening our ability to analyze product attributes and customer reviews. We applied these skills to real-world data, improving our data-driven decision-making and optimizing queries to uncover valuable insights.

Advanced SQL Query Techniques

Practical Applications

Data Analysis Skills



Enhanced Query Optimization

Find this useful? like and share this post with your friends.

Connect with me.

Email
rahul.nagra.001@gmail.com

LinkedIn
Rahul Nagra

Call
7973193965