Case Study - Tiny Shop Sales

Ad-Hoc-Requests

Overview

This case study uses PostgreSQL. To successfully answer all the questions you should have been exposed to the following areas of SQL:

- Basic aggregations
- CASE WHEN statements
- Window Functions
- Joins
- Date time functions
- CTEs

Questions (Ad Hoc Requests)

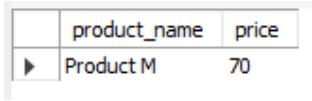
1. Which product has the highest price? Only return a single row.
2. Which customer has made the most orders?
3. What’s the total revenue per product?
4. Find the day with the highest revenue.
5. Find the first order (by date) for each customer.
6. Find the top 3 customers who have ordered the most distinct products
7. Which product has been bought the least in terms of quantity?
8. What is the median order total?
9. For each order, determine if it was ‘Expensive’ (total over 300), ‘Affordable’ (total over 100), or ‘Cheap’.
10. Find customers who have ordered the product with the highest price.

> **Question 1. Identify the product with the highest price. Provide a single-row answer.**

Query

```
SELECT
    product_name, price
FROM
    products
ORDER BY price DESC
LIMIT 1;
```

Here's your output:

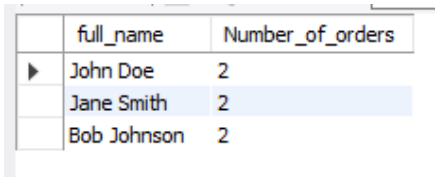


	product_name	price
▶	Product M	70

> **Question 2 - Which customer has made the highest number of orders?**

```
SELECT
    concat(first_name, ' ', last_name) AS full_name,
    count(o.order_id) AS Number_of_orders
FROM
    customers c
        JOIN
    orders o ON c.customer_id = o.customer_id
GROUP BY 1
HAVING number_of_orders > 1;
```

Here’s your output:



	full_name	Number_of_orders
▶	John Doe	2
	Jane Smith	2
	Bob Johnson	2

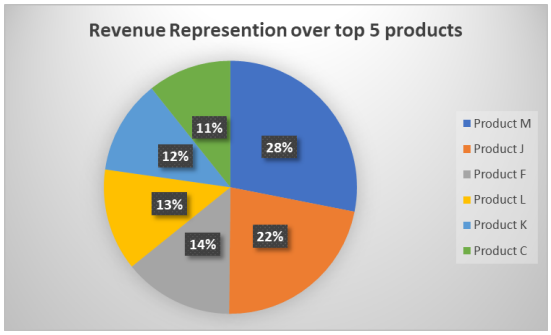
Que 3. - What’s the total revenue per product?

```
SELECT
  p.product_id,
    p.product_name,
    p.price,
    SUM(ot.quantity) AS Total_Qty,
    SUM(ot.quantity * p.price) AS Total_Revenue
FROM
    products p
        JOIN
    order_items ot ON ot.product_id = p.product_id
 GROUP BY 1
 ORDER BY Total_Revenue DESC;
```

Here’s your output

product_id	product_name	price	Total_Qty	Total_Revenue
13	Product M	70	6	420
10	Product J	55	6	330
6	Product F	35	6	210
12	Product L	65	3	195
11	Product K	60	3	180
3	Product C	20	8	160
9	Product I	50	3	150
2	Product B	15	9	135
8	Product H	45	3	135
7	Product G	40	3	120
5	Product E	30	3	90
4	Product D	25	3	75
1	Product A	10	5	50

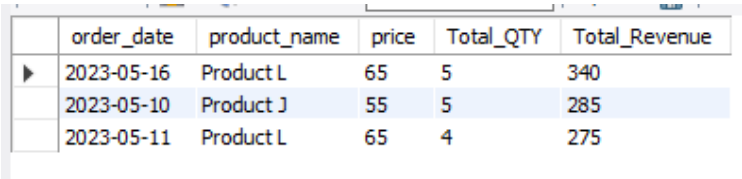
Visualization Revenue over Products



Que. 4 - Find the day with the highest revenue.

```
SELECT
    distinct o.order_date,
    p.product_name,
    p.price,
    SUM(oi.quantity) AS Total_QTY,
    SUM(p.price * oi.quantity) AS Total_Revenue
FROM
    order_items oi
        JOIN
    products p ON p.product_id = oi.product_id
        JOIN
    orders o ON o.order_id = oi.order_id
GROUP BY 1
ORDER BY Total_Revenue DESC
LIMIT 3;
```

Here’s your output

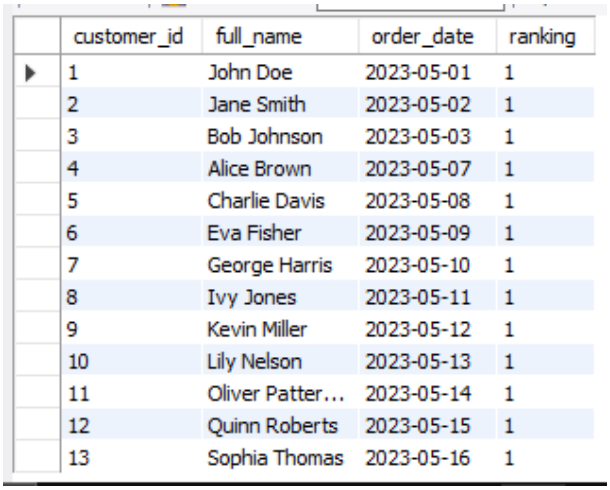


order_date	product_name	price	Total_QTY	Total_Revenue
2023-05-16	Product L	65	5	340
2023-05-10	Product J	55	5	285
2023-05-11	Product L	65	4	275

Que. 5 - Find the first order (by date) for each customer.

```
WITH ranking_order AS
  (
    SELECT
      c.customer_id,
            concat(first_name, ' ', last_name) AS full_name,
            o.order_date,
      dense_rank() over(partition by c.customer_id order by o.order_date ASC) AS ranking
        FROM
      customers c
        JOIN
      orders o
            ON c.customer_id = o.customer_id
       )
       SELECT
      *
       FROM
      ranking_order
       WHERE ranking = 1;
```

Here’s your output

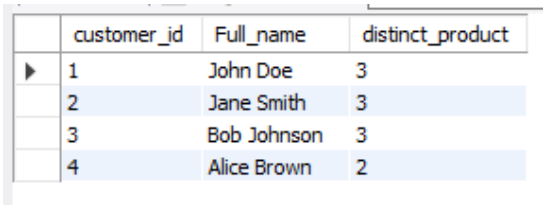


customer_id	full_name	order_date	ranking
1	John Doe	2023-05-01	1
2	Jane Smith	2023-05-02	1
3	Bob Johnson	2023-05-03	1
4	Alice Brown	2023-05-07	1
5	Charlie Davis	2023-05-08	1
6	Eva Fisher	2023-05-09	1
7	George Harris	2023-05-10	1
8	Ivy Jones	2023-05-11	1
9	Kevin Miller	2023-05-12	1
10	Lily Nelson	2023-05-13	1
11	Oliver Patter…	2023-05-14	1
12	Quinn Roberts	2023-05-15	1
13	Sophia Thomas	2023-05-16	1

> ## Que. 6 - Find the top 3 customers who have ordered the most distinct products.

```
SELECT
    c.customer_id,
    CONCAT(first_name, ' ', last_name) AS Full_name,
    COUNT(DISTINCT oi.product_id) AS distinct_product
FROM
    customers c
        JOIN
    orders o ON o.customer_id = c.customer_id
        JOIN
    order_items oi ON oi.order_id = o.order_id
GROUP BY c.customer_id , Full_name
ORDER BY distinct_product DESC
LIMIT 4;
```

Here’s your output

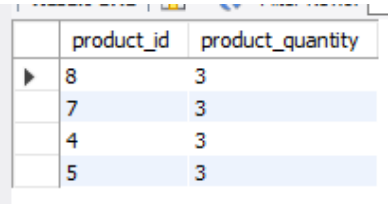


customer_id	Full_name	distinct_product
1	John Doe	3
2	Jane Smith	3
3	Bob Johnson	3
4	Alice Brown	2

Que. 7 -Which product has been bought the least in terms of quantity?

```
SELECT
    product_id, SUM(quantity) AS product_quantity
FROM
    order_items
GROUP BY product_id
ORDER BY 2
LIMIT 4;
```

Here’s your output



product_id	product_quantity
8	3
7	3
4	3
5	3

Que. 8 - What is the median order total?

```
WITH order_total AS
  (
    SELECT
      o.order_id,
            SUM(oi.quantity * p.price) AS Total_Revenue
        FROM
      order_items oi
        JOIN
      products p ON p.product_id = oi.product_id
        JOIN
      orders o ON o.order_id = oi.order_id
        GROUP BY
      o.order_id)
      SELECT
    AVG(Total_Revenue) AS Median_Total_Revenue
      FROM  (
      SELECT
        Total_Revenue,
                NTILE(2) OVER (ORDER BY Total_Revenue) AS Quartile
             FROM
        order_total
            ) median_query
       WHERE
      Quartile = 1 OR Quartile = 2;
```

Here’s your output

	Median_Total_Revenue
▶	140.6250

> **Que 9. For each order, determine if it was ‘Expensive’ (total over 300), ‘Affordable’ (total over 100), or ‘Cheap’.**

```
WITH total_rev AS
    (
    SELECT
      p.product_name,
      SUM(oi.quantity * p.price) AS Total_Revenue
    FROM
      products p
        JOIN
      order_items oi ON oi.product_id = p.product_id
    GROUP BY p.product_name
        )
 SELECT
  product_name, Total_Revenue,
    CASE WHEN Total_Revenue > 300 THEN "Expensive"
     WHEN Total_Revenue > 100 THEN "Affordable"
         ELSE "Cheap"
    END AS price_range
 FROM
  Total_rev
 ORDER BY
  Total_Revenue DESC;
```

Here’s your output

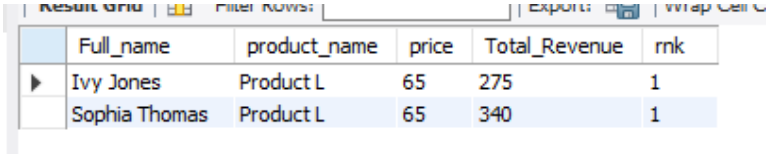
	product_name	Total_Revenue	price_range
▶	Product M	420	Expensive
	Product J	330	Expensive
	Product F	210	Affordable
	Product L	195	Affordable
	Product K	180	Affordable
	Product C	160	Affordable
	Product I	150	Affordable
	Product B	135	Affordable
	Product H	135	Affordable
	Product G	120	Affordable
	Product E	90	Cheap
	Product D	75	Cheap
	Product A	50	Cheap

Que. 10 - Find customers who have ordered the product with the highest price.

```
WITH ordered_product_price AS
(
 SELECT
  concat(first_name, ' ' , last_name) AS Full_name,
    p.product_name,
    p.price,
    SUM(oi.quantity * p.price) AS Total_Revenue,
    dense_rank() OVER( ORDER BY price DESC) AS rnk
 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 p.product_id = oi.product_id
 GROUP BY 1
 ORDER by 3 DESC)

 SELECT
  *
 FROM
  ordered_product_price
 WHERE rnk =1;
```

Here’s your output



Full_name	product_name	price	Total_Revenue	rnk
Ivy Jones	Product L	65	275	1
Sophia Thomas	Product L	65	340	1

Insights

1. **Product M** is the item with the **highest price.**
2. The **customers who have made the most orders are John Doe, Jane Smith, and Bob Johnson**. This is a great accomplishment, and it is a testament to the quality of our products and services. Keep up the good work!
3. It is a fact that **"May 16, 2023"** generated the **highest revenue** for the shop, proving that our efforts have paid off and we are on the right track to success.
4. Based on the order data, we can confidently state that the **median** transaction amount is **$140.** This insight sheds light on the typical order total and helps us better understand our customers' purchasing behavior.
5. **Ivy Jones and Sophia Thomas** are the customers who ordered **the product with the highest price.**