MySQL Documentation

SRIDHAR E
71772118145

Problem Statement -1

Qn 1:

Top 5 products with highest sales revenue (year changed to 2019 since first order is in 2016)

```
SELECT p.id AS product_id, p.title, SUM(o.total) AS sales
FROM orders o
JOIN products p ON o.product_id = p.id
WHERE YEAR(o.created_at) = 2019
GROUP BY p.id, p.title
ORDER BY sales DESC
LIMIT 5;
```

Finds the top 5 products sold in 2019. It combines data from the "orders" and "products" tables, sums the total sales for each product, sorts them from highest to lowest sales, and then shows only the top 5 products.

Qn 2:

Compare ATV for each category in Month over Month (Jan/Feb)

```
SELECT
    category,
    MONTHNAME(o.created_at) AS Month,
    AVG(total) AS ATV
FROM
    orders o
INNER JOIN products p ON o.product_id = p.id
WHERE
    YEAR(o.created_at) = 2017
    AND MONTH(o.created_at) IN (1, 2)
GROUP BY
    category,
    MONTHNAME(o.created_at)
ORDER BY
    category,
    MONTH(o.created_at);
```

Fetches the average total sales (ATV) for each category in January and February 2017. It joins the "orders" and "products" tables, groups the results by category and month name, then sorts them by category and month.

Qn 3:

Conversion rate of users who have placed at least 1 order in 2017

```
SELECT
    user_id,
    (COUNT(CASE WHEN YEAR(created_at) = 2017 THEN 1 END) /
COUNT(*)) * 100 AS ConRate
FROM
    orders
GROUP BY
    user_id
HAVING
    ConRate > 0
ORDER BY
    ConRate DESC,
    user_id ASC;
```

Calculates the conversion rate of users who placed at least one order in 2017. It counts the number of orders made by each user in 2017 and divides it by the total number of orders made by that user across all years.

The result is multiplied by 100 to get the percentage. Users with a conversion rate greater than 0 are selected and ordered by conversion rate in descending order, with ties broken by user ID in ascending order.

Qn 4:

Rank users based on their order count

```
SELECT
    u.name,
    COUNT(o.user_id) AS OrdersPlaced,
    RANK() OVER (ORDER BY COUNT(o.user_id) DESC) AS UserRank
FROM
    orders o
INNER JOIN
    users u ON o.user_id = u.id
GROUP BY
    u.id
ORDER BY
    UserRank;
```

Lists users by their names alongside the total number of orders each user has placed. It assigns a rank to each user based on their order count, with the user having the most orders receiving the highest rank. The results are ordered by user rank.

Problem Statement -2

Qn 1

Top 10 users with the highest order count in November 2018

```
SELECT
    u.name,
    COUNT(*) AS ordersplaced
FROM
    orders o
INNER JOIN
    users u ON o.user_id = u.id
WHERE
    YEAR(o.created_at) = 2018
    AND MONTH(o.created_at) = 11
GROUP BY
    u.id, u.name
ORDER BY
    ordersplaced DESC,
    u.name ASC
LIMIT 10;
```

First it lists the top 10 users with the highest number of orders placed in November 2018. It retrieves each user's name and counts the total number of orders placed by them within that month. The results are sorted in descending order of order count, and if there are ties, they are broken alphabetically by user name.

Qn 2

Percentage growth in Organic Sales

```
WITH Sales AS (
    SELECT
        category,
        SUM(CASE WHEN YEAR(created_at) = 2018 THEN total ELSE 0 END) AS Sales_2018,
        SUM(CASE WHEN YEAR(created_at) = 2019 THEN total ELSE 0 END) AS Sales_2019
    FROM
        orders
    WHERE
        user_id IN (SELECT id FROM users WHERE source = 'Organic')
    GROUP BY
        category
)
SELECT
    category,
    Sales_2018,
    Sales_2019,
    CASE
        WHEN Sales_2019 = 0 THEN NULL
        ELSE ((Sales_2019 - Sales_2018) * 100.0 / Sales_2019)
    END AS Growth
FROM
    Sales
ORDER BY
    category;
```

Computes the growth rate in organic sales by comparing 2018 and 2019 totals for each product category, safeguarding against division by zero errors when 2019 sales are null.

Qn 3

Top 3 channels contributing to orders since 2017

```
SELECT
    u.source,
    COUNT(*) AS OrdersContributed
FROM
    orders o
INNER JOIN
    users u ON o.user_id = u.id
WHERE
    YEAR(o.created_at) >= 2017
GROUP BY
    u.source
ORDER BY
    OrdersContributed DESC
LIMIT 3;
```

This identifies the top 3 channels that have contributed to orders since 2017, counting the number of orders associated with each channel and listing them in descending order of contribution.

Qn 4

Percentage of customers who have made more than 1 purchase

```
WITH RepeatOrders AS (
    SELECT
        user_id
    FROM
        orders
    GROUP BY
        user_id
    HAVING
        COUNT(*) > 1
)
SELECT
    (COUNT(RO.user_id) * 100.0 / (SELECT COUNT(DISTINCT
user_id) FROM orders)) AS RPR
FROM
    RepeatOrders RO;
```

It calculates the percentage of customers who have made more than one purchase. It counts the number of distinct users with multiple orders and divides it by the total number of distinct users, then expresses the result as a percentage.

Problem Statement -3

Qn 1

Top 5 Expensive products in each category

```
SELECT
    p.category,
    p.title,
    p.price
FROM
    (
    SELECT
        title,
        category,
        price,
        ROW_NUMBER() OVER (PARTITION BY category ORDER BY price
DESC) AS seq_no
    FROM
        products
    ) AS p
WHERE
    p.seq_no <= 5;
```

Lists the top 5 most expensive products in each category. It assigns a row number to each product within its category based on price, ensuring accurate ranking, and then filters to include only those with row numbers up to 5.

Qn 2

min, max and avg price of products in each category

```
SELECT
    category,
    MIN(price) AS Minimum,
    MAX(price) AS Maximum,
    AVG(price) AS Average
FROM
    products
GROUP BY
    category;
```

Calculates the minimum, maximum, and average prices of products within each category. It groups the products by category and computes these statistical measures separately for each group.

Qn 3

Find the top selling vendors in terms of order count

```
SELECT
    p.vendor,
    COUNT(o.id) AS Orders
FROM
    orders o
INNER JOIN
    products p ON o.product_id = p.id
GROUP BY
    p.vendor
ORDER BY
    Orders DESC;
```

Identifies the vendors with the highest number of orders. It counts the orders associated with each vendor and lists them in descending order based on order count.

Qn 4

Find min, max and avg rating for all ordered products

```
SELECT
    p.title AS Title,
    MIN(r.rating) AS Minimum,
    MAX(r.rating) AS Maximum,
    AVG(r.rating) AS Average
FROM
    products p
LEFT JOIN
    reviews r ON p.id = r.product_id
GROUP BY
    p.title;
```

This query calculates the minimum, maximum, and average ratings for all products, considering both reviewed and non-reviewed products. It groups the results by product title, ensuring each product's ratings are accurately represented.

Qn 5

Identify the sellers and products not sold in last 2 months of 2019

```
SELECT
    p.title
FROM
    products p
LEFT JOIN
    (
    SELECT
        DISTINCT product_id
    FROM
        orders
    WHERE
        YEAR(created_at) = 2019
        AND MONTH(created_at) IN (11, 12)
    ) AS sold_products ON p.id = sold_products.product_id
WHERE
    sold_products.product_id IS NULL;
```

Identifies products that were not sold in November and December of 2019. It compares the list of all products with the list of products sold during those months and returns the titles of products that were not sold.

Problem Statement – 4

Qn -1

Find the contributions of the marketing channels on the year of highest user sign up

```
SELECT
    source,
    COUNT(source) AS UsersBroughtIn
FROM
    users
WHERE
    YEAR(created_at) = (
        SELECT
            YEAR(created_at) AS Year
        FROM
            users
        GROUP BY
            YEAR(created_at)
        ORDER BY
            COUNT(id) DESC
        LIMIT 1
    )
GROUP BY
    source
ORDER BY
    UsersBroughtIn DESC;
```

It identifies the marketing channels that brought in users during the year with the highest number of sign-ups. It counts the users brought in by each channel and orders the results by the number of users brought in, showing the most effective channels first.

Qn -2

Find number of users in each age group specified

```
SELECT
    CASE
        WHEN TIMESTAMPDIFF(YEAR, birth_date, CURDATE()) < 18
THEN '<18'
        WHEN TIMESTAMPDIFF(YEAR, birth_date, CURDATE()) BETWEEN
18 AND 24 THEN '18-24'
        WHEN TIMESTAMPDIFF(YEAR, birth_date, CURDATE()) BETWEEN
25 AND 44 THEN '25-44'
        WHEN TIMESTAMPDIFF(YEAR, birth_date, CURDATE()) BETWEEN
45 AND 55 THEN '45-55'
        ELSE '55+'
    END AS ageGroup,
    COUNT(id) AS Users
FROM
    users
GROUP BY
    ageGroup
ORDER BY
    Users DESC;
```

Categorizes users into age groups and counts the number of users in each group. It then orders the results by the number of users in each group, showing the largest group first.

Qn -3

Find the customers having the highest amount of order values

```
SELECT
    u.name,
    u.email,
    u.address,
    SUM(o.total) AS OrderAmount
FROM
    orders o
INNER JOIN
    users u ON o.user_id = u.id
GROUP BY
    u.name,
    u.email,
    u.address
ORDER BY
    OrderAmount DESC;
```

Retrieves the names, emails, and addresses of customers, along with the total amount they spent on orders. It then organizes the results based on the highest total order amounts, showing the customers with the largest purchases first.

Qn -4

Find the eMail ID of the customer with the highest purchase from the specified vendor.

```
SELECT
    u.name,
    u.email,
    SUM(o.total) AS OrderAmount
FROM
    orders o
INNER JOIN
    users u ON o.user_id = u.id
INNER JOIN
    products p ON o.product_id = p.id
WHERE
    p.vendor = 'Fisher-Kemmer'
GROUP BY
    u.name,
    u.email
ORDER BY
    OrderAmount DESC;
```

Finds the customer name and email who made the highest purchase from a specific vendor. It calculates the total order amount for each customer and selects the one with the highest purchase from the specified vendor.