MySQL Documentation

Thanmughil D K
71772118150

Problem Statement - 1

Question 1 :

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

```
select
    product_id, title, sum(total) as sales
from
    orders o, products p
where
    o.product_id = p.id  and year(o.created_at) = 2019
group by
    product_id
order by
    sales desc LIMIT 5;
```

Grouping by product_id in the orders table and finding the sum(total) returns the sale of each product.

Then the result set is ordered by the aggregated sales value and a lookup is done in the products table to get the title and the top 5 is displayed using LIMIT.

Question 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, monthname(o.created_at);
```

Join products and orders to associate ordered items with category, then perform double group by to get both category-wise and month-wise.

Then perform avg() aggregate function to get the ATV for the respective months.

Question 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 user_id end)/
    count(user_id))*100 as ConRate
from
    orders
group by
    user_id
having
    ConRate > 0
order by
    ConRate desc, user_id asc;
```

Find count of orders placed in 2017 for each user, and count of all orders placed by each user, then find the ratio to get the conversion rate.

Question 4 :

// Rank users based on their order count

```sql
select
    u.name, count(user_id) as OrdersPlaced, rank() over
    (order by count(user_id) desc) as UserRank
from
    orders o inner join users u on o.user_id = u.id
group by
    user_id
order by
    UserRank;
```

Find order count of each user using group by and count(), assign rank to users based on count() in desc, then order by the rank.

Problem Statement - 2

Question 1 :

// Top 10 users with the highest order count in November 2018

```sql
select
    u.name, count(user_id) 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
    user_id
order by
    ordersplaced desc,
    u.name asc
limit 10;
```

Join orders and user table, then group by user_id in orders, count orders for each user in November 2018, then sort by the calculated order_count. Use dense_rank() for better results.

Question 2 :

// Percentage growth in Organic Sales

```
with Sales as (
select
    category,
    sum(case when year(o.created_at) = 2018 then total end) as Sales_2018,
    sum(case when year(o.created_at) = 2019 then total end) as Sales_2019
from
    orders o inner join users u on o.user_id = u.id
    inner join products p on o.product_id = p.id
where
    u.source = "Organic"
group by
    category
order by
    category )

select
    category, Sales_2018, Sales_2019,
    ((Sales_2019-Sales_2018)*100/Sales_2019) as Growth
from
    Sales
order by
    category;
```

Create a CTE to compute total sales from Organic sources from a joined table of orders and products, then use it to calculate the growth percentage.

Question 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
LIMIT 3;
```

Group the joined table of orders and users by source, count the number of orders contributed by each source, then order them by the order_count. Use dense_rank() for better results.

Question 4 :

// Percentage of customers who have made more than 1 purchase

```
with repeatOrders as (
select
    user_id, count(user_id) as Orders
from
    orders
group by
    user_id
having
    Orders > 1 )

select
    ((select count(user_id) from repeatOrders)*100 /
    count(distinct user_id)) as RPR
from
    orders;
```

Use CTE to find the number of users who have ordered more than once by grouping by user_id in orders and count of orders greater than 1, then use it to find the percentage.

Problem Statement - 3

Question 1 :

// Top 5 Expensive products in each category

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

Partition by category and order by prices, assign dense_rank over the partitions, then select items with seq_no less than or equal to 5.

Question 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;
```

Use group by to split into categories, then perform the aggregate functions min, max and avg over price column in the products table.

Question 3 :

// Find the top selling vendors in terms of order count

```
select
    vendor, count(o.id) as Orders
from
    orders o inner join products p on o.product_id = p.id
group by
    vendor
order by
    Orders desc;
```

Join orders and products table on product_id and then group by vendors and count the number of orders, and display it in descending order.

Question 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
    orders o inner join products p on o.product_id = p.id
    left join reviews r on o.product_id = r.product_id
group by
    o.product_id;
```

Left join orders and reviews, group by the product_id, and perform aggregate functions min, max and avg over the ratings column.

Question 5 :

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

```
select
    p.title
from
    products p
where
    p.id not in (
        select
            product_id
        from
            orders
        where
            year(created_at) = 2019 and
            (
                month(created_at) = 11 or
                month(created_at) = 12
            )
    );
```

Find products not sold in 2019 in the months 11 or 12 by using the NOT IN operator in products table

Problem Statement - 4

Question 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
        from (
            select
                year(created_at) as Year,
                count(id) as UsersSignedUp
            from
                users
            group by
                Year
            order by
                UsersSignedUp desc
            LIMIT 1
        ) as SignUpTable
    )
group by
    source
order by
    UsersBroughtIn desc;
```

Find the year with highest user sign-up by grouping by year in users table and counting the number of users in each year, then group by source and count number of users brought in from the users table where the year is the calculated highest sign-up year

Question 2 :

// Find number of users in each age group specified

```sql
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"
        when timestampdiff(year,birth_date,curdate()) > 55
            then "55+"
    end
    ) as ageGroup, count(id) as Users
from
    users
group by
    ageGroup
order by
    Users desc
LIMIT 1;
```

Use case and timestampdiff to calculate age and find the category, then group by the ageGroup and find the number of users in each group and sort it in descending order.

Question 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
    o.user_id
order by
    OrderAmount desc;
```

Join the order table with users, then group by user_id in order table with sum aggregate on total to find purchase value, then sort the users by the purchase value.

Question 4 :

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

```
select
    u.name, u.email, sum(total) as OrderAmount
from
    (
        select * from products
        where vendor = "Fisher-Kemmer"
    ) as p
    inner join orders o on o.product_id = p.id
    inner join users u on o.user_id = u.id
group by
    u.id
order by
    OrderAmount desc;
```

Find the products sold by the specified vendor, then join it with the order table and group by user_id with sum aggregate on total to find purchase value, and then sort by the purchase value and find the eMail and other specified details with a join on the user table.