MAS DSE 201 Homework: Sales Cube

Milestone II [due Feb 18 midnight]

Write the following queries:

1. *Show the total sales (quantity sold and dollar value) for each customer.*

```sql
-- 1. Total sales for each customer
select c.customer_id, c.first_name, c.last_name, coalesce(sum(s.quantity),0)
    as "quantity sold", coalesce(sum(s.price_paid), cast(0 as decimal(36,2))) as "dollar value"
from customers c
left join sales s
on s.customer_id = c.customer_id
group by c.customer_id
order by "dollar value" desc
;
```

2. *Show the total sales for each state.*

```sql
-- 2. Total sales for each state
select st.state_id, st.name as "state name", coalesce(sum(s.quantity), 0)
    as "quantity sold", coalesce(sum(s.price_paid), cast(0 as decimal(36,2))) as "dollar value"
from sales s
join customers c
on s.customer_id = c.customer_id
right join states st
on c.state_id = st.state_id
group by st.name, st.state_id
order by "dollar value" desc
;
```

3. *Show the total sales for each product, for a given customer. Only products that were actually bought by the given customer are listed. Order by dollar value.*

```sql
-- 3. Total sales for each product for a given customer
select c.customer_id, c.first_name, c.last_name, p.name as "product name",
    sum(s.quantity) as "quantity sold", sum(s.price_paid) as "dollar value"
from sales s
join customers c
on s.customer_id = c.customer_id
join products p
on s.product_id = p.product_id
-- Define customer and exclude zero entries for qty and price
where (c.customer_id = 25) and (s.quantity != 0) and (s.price_paid != 0)
group by p.name, c.customer_id
order by "dollar value" desc
;
```

4. *Show the total sales for each product and customer. Order by dollar value.*

```sql
-- 4. Total sales for each product and customer
select c.customer_id, c.first_name, c.last_name, p.product_id, p.name
    as "product name", coalesce(sum(s.quantity),0) as "quantity sold",
    coalesce(sum(s.price_paid), cast(0 as decimal(36,2))) as "dollar value"
from products p
cross join customers c
left join sales s
on (s.product_id = p.product_id) and (c.customer_id = s.customer_id)
group by p.product_id, c.customer_id
order by "dollar value" desc
;
```

5. *Show the total sales for each product category and state.*

```
-- 5. Total sales for each product category and state
select cat.category_id, cat.name as "category name", st.state_id, st.name
    as "state name", coalesce(sum(s.quantity),0) as "quantity sold",
    coalesce(sum(s.price_paid), cast(0 as decimal(36,2))) as "dollar value"
from categories cat
cross join states st
left join products p
on p.category_id = cat.category_id
left join customers c
on st.state_id = c.state_id
left join sales s
on (s.product_id = p.product_id) and (c.customer_id = s.customer_id)
group by cat.category_id, st.state_id
order by "dollar value" desc
;
```

6. *For each one of the top 20 product categories and top 20 customers, return a tuple (top product, top customer, quantity sold, dollar value)*

```sql
-- 6. Tuples for each of the top 20 product categories and top 20 customers
select t5.category_id, t5.name as "top category", t5.customer_id, t5.first_name
    as "top customer first name", t5.last_name as "top customer last name",
    coalesce(sum(s.quantity),0) as "quantity sold", coalesce(sum(s.price_paid),0) as "dollar value"
from sales s
join products p
on s.product_id = p.product_id
join categories cat
on cat.category_id = p.category_id
right join (
    select * from (
        -- select top 20 for categories
        select *
        from (
            select cat.category_id, cat.name, dense_rank() over(order by sum(s.price_paid) desc) as rn
            from sales s
            join products p
            on s.product_id = p.product_id
            join categories cat
            on cat.category_id = p.category_id
            group by cat.category_id, cat.name
            ) as t1
            where rn <= 20
            ) as t2
            cross join ( -- find all possible combinations ~400 with cross join
            -- select top 20 for customers
            select *
            from (
            select c.customer_id, c.first_name, c.last_name, dense_rank() over(order by sum(s.price_paid) desc) as rn
            from sales s
            join customers c
            on s.customer_id = c.customer_id
            group by c.customer_id
            ) as t3
            where rn <= 20
            ) as t4
        ) as t5
-- join on categories and customers only
on cat.category_id = t5.category_id and s.customer_id = t5.customer_id
group by t5.category_id, t5.name, t5.customer_id, t5.first_name, t5.last_name
order by "dollar value" desc
;
```

MAS DSE 201 Homework: 201Cats

Milestone II [due Feb 18 at midnight]

201Cats offers to the logged-in user X many options for controlling how the Top-10 cat videos are selected.

- *Option "Overall Likes": The Top-10 cat videos are the ones that have collected the highest numbers of likes, overall.*

```
-- Overall Likes
select video_id, "Video Name", "Overall Likes"
from (
    -- select videos with ranked likes
    select l.video_id, v.name as "Video Name", count(l.user_id)
        as "Overall Likes", dense_rank() over(order by count(l.user_id) desc) as rn
    from likes l
    join videos v
    on l.video_id = v.video_id
    group by l.video_id, v.name
    ) as t
-- filter for top videos
where rn <= 10
;
```

- *Option "Friend Likes": The Top-10 cat videos are the ones that have collected the highest numbers of likes from the friends of X.*

```
-- Friend Likes
select video_id, "Video Name", "Overall Likes"
from (
    -- select videos with ranked likes
    select l.video_id, v.name as "Video Name", count(l.user_id)
        as "Overall Likes", dense_rank() over(order by count(l.user_id) desc) as rn
    from likes l
    join videos v
    on l.video_id = v.video_id
    where user_id in (
        -- select list for friends
        select friend_id
        from friends
        where user_id = 3 -- specify user X
        )
    group by l.video_id, v.name
) as t
-- filter for top videos
where rn <= 10
;
```

- *Option "Friends-of-Friends Likes": The Top-10 cat videos are the ones that have collected the highest numbers of likes from friends and friends-of-friends.*

```
-- Friends of Friends Likes
select video_id, "Video Name", "Overall Likes"
from (
    -- select videos with ranked likes
    select l.video_id, v.name as "Video Name", count(l.user_id) as "Overall Likes",
        dense_rank() over(order by count(l.user_id) desc) as rn
    from likes l
    join videos v
    on l.video_id = v.video_id
    where user_id in (
        -- select list for friends
        select friend_id
        from friends
        where user_id = 3 -- specify user X
        union -- union both lists with user ids
        -- select list for friends of friends
        select friend_id
        from friends
        where user_id in (
            select friend_id
            from friends
            where user_id = 3 -- specify user X
        )
    )
    group by l.video_id, v.name
) as t
-- filter for top videos
where rn <= 10
;
```

- *Option "My kind of cats": The Top-10 cat videos are the ones that have collected the most likes from users who have liked at least one cat video that was liked by X.*

```sql
-- My kind of cats
select video_id, "Video Name", "Overall Likes"
from (
    -- select videos with ranked likes
    select l.video_id, v.name as "Video Name", count(l.user_id) as "Overall Likes",
        dense_rank() over(order by count(l.user_id) desc) as rn
    from likes l
    join videos v
    on l.video_id = v.video_id
    where user_id in (
        -- select users who have liked at least one cat video liked by X
        select  distinct user_id
        from likes
        where video_id in (
            -- select videos liked by user X
            select video_id
            from likes
            where user_id = 3 -- specify user X
            )
        )
        -- exclude videos liked by user X
        and user_id != 3 -- specify user X
    group by l.video_id, v.name
) as t
-- filter for top videos
where rn <= 10
;
```

- *Option "My kind of cats – with preference (to cat aficionados that have the same tastes)": The Top-10 cat videos are the ones that have collected the highest sum of weighted likes from every other user Y (i.e., given a cat video, each like on it, is multiplied by a weight).*

  The weight is the log cosine lc(X,Y) defined as follows: Conceptually, there is a vector v_U for each user U, including the logged-in user X. The vector has as many elements as the number of cat videos. Element i is 1 if U liked the ith cat video; it is 0 otherwise. For example, if 201Cats has five cat videos and user 21 liked only the 1^{st} and the 4^{th}, then v21=<1,0,0,1,0>, i.e., v21[1]=v21[4]=1 and v21[2]= v21[3]=v21[5]=0. Assuming there are N cat videos, the log cosine lc(X, Y) is

$$Ic(X,Y) = log\ (1 + \sum_{\{i=1,\ldots,N\}} (v_X[i] \times v_Y[i]))$$

Each of the options is essentially a query that finds top-10 cat videos. Write and submit the queries.

```sql
-- My kind of cats - with preference
select video_id, "Video Name", "Sum Weighted Likes"
from (
    -- selectect videos with ranked log cosine
    select l.video_id, v.name as "Video Name", cast(sum(t4.lc) as decimal(36,4)) as "Sum Weighted Likes",
        dense_rank() over(order by sum(t4.lc) desc) as rn
    from likes l
    join(
        -- calculate inner product and log cosine
        select user_y, log(sum(product)+1) as lc
        from(
            -- calculate and select multiplication from inner product
            select t1.video_id, t1.user_id as user_x, t2.user_id as user_y, t1.liked as liked_x, t2.liked
                as liked_y, (t1.liked * t2.liked) as product
            from (
                -- calculate and select the vector for parameterized user X
                select v.video_id, u.user_id, (case when l.like_id is null then 0 else 1 end) as liked
                from videos v
                cross join users u
                left join likes l
                on u.user_id = l.user_id and  v.video_id = l.video_id
                where u.user_id = 3 -- specify user X
                ) as t1
            left join (
                -- calculate and select the vector for all users Y
                select v.video_id, u.user_id, (case when l.like_id is null then 0 else 1 end) as liked
                from videos v
                cross join users u
                left join likes l
                on u.user_id = l.user_id and  v.video_id = l.video_id
                where u.user_id != 3 -- exclude user X
                ) as t2
            on t1.video_id = t2.video_id
            ) as t3
        group by user_y
        ) as t4
    on l.user_id = t4.user_y
    join videos v
    on l.video_id = v.video_id
    group by  l.video_id, v.name
) as t5
-- filter for top videos
where rn <= 10
;
```