Promo Email

Create a right sub-table for recently viewed events.

```sql
select user_id,
    item_id,
    event_time,
    row_number() over (partition by user_id
             order by event_time desc) as view_number
from dsv1069.view_item_events
```

Exercise 1: Create the right subtable for recently viewed events using the view_item_events table.

```sql
select *
from
  ( select user_id,
        item_id,
        event_time,
        row_number() over (partition by user_id
                 order by event_time desc) as view_number
  from dsv1069.view_item_events ) recent_views
join dsv1069.users
 on users.id = recent_views.user_id
join dsv1069.items
 on items.id = recent_views.item_id
```

Exercise 2: Check your joins. Join your tables together recent_views, users, items Starter Code: The result from Ex1.

```sql
select users.id as user_id,
    users.email_address,
    items.id as item_id,
    items.name as item_name,
    items.category as item_category
from
  ( select user_id,
        item_id,
        event_time,
        row_number() over (partition by user_id
                 order by event_time desc) as view_number
  from dsv1069.view_item_events ) recent_views
join dsv1069.users
 on users.id = recent_views.user_id
join dsv1069.items
 on items.id = recent_views.item_id
```

Exercise 3: Clean up your columns. The goal of all this is to return all of the information we'll need to send users an email about the item they viewed more recently. Clean up this query outline from the outline in EX2 and pull only the columns you need. Make sure they are named appropriately so that another human can read and understand their contents.

```sql
select coalesce(users.parent_user_id, users.id) as user_id,
    users.email_address,
    items.id as item_id,
    items.name as item_name,
    items.category as item_category
from
  ( select user_id,
        item_id,
        event_time,
        row_number() over (partition by user_id
                 order by event_time desc) as view_number
  from dsv1069.view_item_events
  where event_time >= '2017-12-01' ) recent_views
join dsv1069.users
 on users.id = recent_views.user_id
join dsv1069.items
 on items.id = recent_views.item_id
left outer join dsv1069.orders
 on orders.item_id = recent_views.item_id
 and orders.user_id = recent_views.user_id
where view_number = 1
 and users.deleted_at is not null
 and orders.item_id is null
```

Exercise 4: Consider any edge cases. If we sent an email to everyone in the results of this query, what would we want to filter out. Add in any extra filtering that you think would make this email better. For example should we include deleted users? Should we send this email to users who already ordered the item they viewed most recently?

```sql
select coalesce(users.parent_user_id, users.id) as user_id,
    users.email_address,
    recent_views.item_id,
    items.name as item_name,
    items.category as item_category
from
  (select user_id,
      item_id,
      event_time,
      row_number() over (partition by user_id
                 order by event_time desc) as view_number
  from dsv1069.view_item_events
  where event_time > '2018-12-01' ) recent_views
join dsv1069.users
 on users.id = recent_views.user_id
join dsv1069.items
 on items.id = recent_views.item_id
left outer join dsv1069.orders
 on orders.item_id = recent_views.item_id
 and orders.user_id = recent_views.user_id
where view_number = 1
 and users.deleted_at is null
 and orders.item_id is null
```

Mode Report Link: https://app.mode.com/sum14/reports/f900b0a21054