Product Analysis

Exercise 0: Count how many users we have.

```
select count(*) from dsv1069.users
```

Exercise 1: Find out how many users have ever ordered.

```
select count(distinct user_id) as users_with_orders
from
 dsv1069.orders
```

Exercise 2: --Goal find how many users have reordered the same item.

```
select *
from
 ( select user_id,
      item_id,
      count(distinct line_item_id) as times_user_ordered
  from dsv1069.orders
  group by user_id,
      item_id ) user_level_orders
where times_user_ordered > 1
order by times_user_ordered desc
limit 5
```

```
select count(distinct user_id) as users_who_reordered
from
 (select user_id,
     item_id,
     count(distinct line_item_id) as times_user_ordered
  from dsv1069.orders
  group by user_id,
      item_id) user_level_orders
where times_user_ordered > 1
```

Exercise 3: --Do users even order more than once?

```
select count(distinct user_id)
from
 (select user_id,
      count(distinct invoice_id) as order_count
  from dsv1069.orders
  group by user_id ) user_level
where order_count > 1
```

Exercise 4: --Orders per item.

```
select item_id,
    count(line_item_id) as times_ordered
from dsv1069.orders
group by item_id
order by 2 desc
limit 5
```

Exercise 5: --Orders per category.

```
select item_category,
    count(line_item_id) as times_ordered
from dsv1069.orders
group by item_category
order by 2 desc
```

```
6.  limit 5
```

Exercise 6: --Goal: Do user order multiple things from the same category?

```
1.  select item_category,
2.      round(avg(times_category_ordered), 2) as avg_times_category_ordered
3.  from
4.   (select user_id,
5.       item_category,
6.       count(distinct line_item_id) as times_category_ordered
7.    from dsv1069.orders
8.    group by user_id,
9.         item_category) user_level
10. group by item_category
11. order by 2 desc
```

Exercise 7: --Goal: Find the average time between orders --Decide if this analysis is necessary.

```
1.  select first_orders.user_id,
2.      date(first_orders.paid_at) as first_order_date,
3.      date(second_orders.paid_at) as second_order_date,
4.      date(second_orders.paid_at) -date(first_orders.paid_at) as date_diff
5.  from
6.    (select user_id,
7.        invoice_id,
8.        paid_at,
9.        dense_rank() over (partition by user_id
10.                  order by paid_at asc) as order_num
11.   from dsv1069.orders) first_orders
12. join
13.   (select user_id,
14.      invoice_id,
15.      paid_at,
16.      dense_rank() over (partition by user_id
17.                 order by paid_at asc) as order_num
18.   from dsv1069.orders) second_orders
19. on first_orders.user_id = second_orders.user_id
20. where first_orders.order_num = 1
21. and second_orders.order_num = 2
22. limit 5
```

Mode Report Link: https://app.mode.com/sum14/reports/ba84f1e5e142