Intro

Things to keep in mind regarding SQL shared screen coding challenges:

- Make sure you are familiar with window functions.
- As always with shared screen interviews, break down the problem as much as possible. This is typically easily doable with SQL. And then solve each small chunk. The goal of shared screen interviews is to check that you can code. Focus on sending that message more than trying to look super smart.
- Some companies will ask you to solve the problem with just one query, i.e. you are not allowed to create temporary tables. Even in that case, break down the problem into smaller pieces and solve them independently. Then put these parts together into just one query via subqueries.
- The data sets attached to this pdf are just for you to practice and verify that your query works. Almost never in a shared screen interview you will have to work with actual data. Here there are no wrong data to find, nothing related to the preprocessing steps you usually have to take care of in a takehome challenge or insights to find. Datasets are just to make sure the query is good.
- Solutions below are in Hive, but they work for pretty much all SQL-like languages commonly used by data scientists.

Query 1

Goal

For each user_id, find the difference between the last action and the second last action. Action here is defined as visiting a page. If the user has just one action, you can either remove her from the final results or keep that user_id and have NULL as time difference between the two actions.

The table below shows for each user all the pages she visited and the corresponding timestamp.

Data

You can download the data set **here**

head (query_one,1)

Column Name	Value	Description
user_id	6684	this is id of the user
page	home_page	the page visited
unix_timestamp	1451640067	unix timestamp in seconds

Query 1 Solution

```
SELECT user_id,
       unix_timestamp - previous_time AS Delta_SecondLast0ne_LastOne
FROM
     (SELECT user_id,
              unix_timestamp,
              LAG(unix_timestamp, 1) OVER (PARTITION BY user_id ORDER BY
unix_timestamp) AS previous_time,
              ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY
unix_timestamp DESC) AS order_desc
     FROM query_one
     ) tmp
WHERE order_desc = 1
ORDER BY user_id
LIMIT 5;
```

user_id	Delta_SecondLast0ne_LastOne
2	
3	2
4	4
5	
7	

Query 2

Goal

We have two tables. One table has all mobile actions, i.e. all pages visited by the users on mobile. The other table has all web actions, i.e. all pages visited on web by the users.
Write a query that returns the percentage of users who only visited mobile, only web and both. That is, the percentage of users who are only in the mobile table, only in the web table and in both tables. The sum of the percentages should return 1.

Data

You can download the data sets here.

head (data_mobile,1)

Column Name	Value	Description
user_id	128	this is id of the user who visited a given page on mobile
page	page_5_mobile	page visited by that user on mobile

---

head (data_web,1)

Column Name	Value	Description
user_id	1210	this is id of the user who visited a given page on web
page	page_1_web	page visited by that user on web

Query 2 Solution

```
SELECT 100*SUM(CASE WHEN m.user_id IS null THEN 1 ELSE 0 END)/COUNT(*) as WEB_ONLY,
       100*SUM(CASE WHEN w.user_id IS null THEN 1 ELSE 0 END)/COUNT(*) as MOBILE_ONLY,
       100*SUM(CASE WHEN m.user_id IS NOT null AND w.user_id IS NOT null THEN 1 ELSE 0 END)/COUNT(*) as BOTH
FROM
(SELECT distinct user_id FROM query_two_web ) w
FULL OUTER JOIN
(SELECT distinct user_id FROM query_two_mobile ) m
ON m.user_id = w.user_id;
```

WEB_ONLY	MOBILE_ONLY	BOTH
16	31	52

Query 3

Goal

We define as power users those users who bought at least 10 products. Write a query that returns for each user on which day they became a power user. That is, for each user, on which day they bought the 10th item.

The table below represents transactions. That is, each row means that the corresponding user has bought something on that date.

Data

You can download the data set **here**.

head (data,1)

Column Name	Value	Description
user_id	675	this is id of the user
date	2014-12-31 16:16:12	user 675 bought something on Dec 31, 2014 at 4:16:12 PM

Query 3 Solution

```
SELECT user_id,
       date
FROM
(SELECT *, ROW_NUMBER() over(PARTITION BY user_id ORDER BY date) row_num FROM query_three ) tmp
WHERE row_num = 10
LIMIT 5;
```

user_id	date
1	2015-10-21 06:20:14
3	2015-10-29 22:41:00
4	2015-09-25 12:36:10
5	2015-09-18 06:25:40
6	2015-12-30 00:53:59

Query 4

Goal

We have two tables. One table has all $ transactions from users during the month of March and one for the month of April.

- Write a query that returns the total amount of money spent by each user. That is, the sum of the column transaction_amount for each user over both tables.
- Write a query that returns day by day the cumulative sum of money spent by each user. That is, each day a user had a transcation, we should have how much money she has spent in total until that day. Obviously, the last day cumulative sum should match the numbers from the previous bullet point.

Data

You can download the data sets **here**

head (data_march,1)

Column Name	Value	Description
user_id	13399	this is id of the user who had the corresponding transaction
date	2015-03-01	the transaction happened on March 1st.
transaction_amount	18	the user spent 18$ in that transaction

---

head (data_april,1)

Column Name	Value	Description
user_id	15895	this is id of the user who had the corresponding transaction
date	2015-04-01	the transaction happened on April 1st.
transaction_amount	66	the user spent 66$ in that transaction

Query 4 Solution

Query 4.1 Solution

```
SELECT user_id,
       SUM(transaction_amount) as total_amount
FROM
(
      SELECT * FROM query_four_march
      UNION ALL
      SELECT * FROM query_four_april
) tmp
GROUP BY user_id
ORDER BY user_id
LIMIT 5;
```

user_id	total_amount
2	67
3	26
4	156
5	45
6	32

Query 4.2 Solution

```
SELECT user_id,
       date,
       SUM(amount) over(PARTITION BY user_id ORDER BY date) as total_amount
FROM
(
      SELECT user_id, date, SUM(transaction_amount) as amount
      FROM query_four_march
      GROUP BY user_id, date
      UNION ALL
      SELECT user_id, date, SUM(transaction_amount) as amount
```

```
    FROM query_four_april
    GROUP BY user_id, date
) tmp
ORDER BY user_id, date
LIMIT 5;
```

user_id	date	total_amount
2	2015-03-13	67
3	2015-03-31	26
4	2015-03-28	63
4	2015-04-20	156
5	2015-03-01	45

Query 5

Goal

We have two tables. One is user id and their signup date. The other one shows all transactions done by those users, when the transaction happens and its corresponding dollar amount.

Find the average and median transaction amount only considering those transactions that happen on the same date as that user signed-up.

Data

You can download the data sets **here**

head (user,1)

Column Name	Value	Description
user_id	121	this is id of the user
sign_up_date	2015-01-02	user_id 121 signed up on Jan, 2.

head (transaction_table,1)

Column Name	Value	Description
user_id	856898	this is id of the user who had that transaction
transaction_date	2015-08-02 03:56:08	transaction happened on Aug, 2 at almost 4AM.
transaction_amount	49	transaction amount was 49$.

Query 5 Solution

Note: there are percentile built-in functions that can be used to estimate the median. However, estimating the median is simple enough that it is a good exercise to try to implement it without using the built-in function.

Solution

```
SELECT AVG(transaction_amount) AS average,
       AVG(CASE WHEN row_num_asc BETWEEN row_num_desc-1 and row_num_desc+1
THEN transaction_amount ELSE NULL END ) AS median
FROM
(
      SELECT transaction_amount,
             ROW_NUMBER() OVER(ORDER BY transaction_amount) row_num_asc,
             COUNT(*) OVER() - ROW_NUMBER() OVER(ORDER BY transaction_amount)
+ 1 AS row_num_desc -- need row number for median. there are many other ways
to do this
      FROM query_five_users a
      JOIN (SELECT *, to_date(transaction_date) AS date_only FROM
query_five_transactions) b
      ON a.user_id = b.user_id AND a.sign_up_date = b.date_only
      ) tmp;
```

average	median
49.25	49

Query 6

Goal

We have a table with users, their country and when they created the account. We want to find:

- The country with the largest and smallest number of users
- A query that returns for each country the first and the last user who signed up (if that country has just one user, it should just return that single user)

Data

You can download the data set **here**

head (data,1)

Column Name	Value	Description
user_id	2	this is id of the user
created_at	2015-02-28 16:00:40	user 2 created her account on Feb, 2 around 4PM
country	China	She is based in China

Query 6 Solution

Query 6.1 Solution

```
SELECT country,
       user_count
FROM
(
      SELECT *,
             ROW_NUMBER() OVER (ORDER BY user_count) count_asc,
             ROW_NUMBER() OVER (ORDER BY user_count desc) count_desc
      FROM (
             SELECT country, COUNT(distinct user_id) as user_count
             FROM query_six
             GROUP BY country
             ) a
) tmp
WHERE count_asc = 1 or count_desc = 1;
```

country	user_count
China	18350
Vietnam	1

Query 6.2 Solution

```
SELECT user_id,
       created_at,
       country
FROM
(
      SELECT *,
      ROW_NUMBER() OVER (PARTITION BY country ORDER BY created_at) count_asc,
      ROW_NUMBER() OVER (PARTITION BY country ORDER BY created_at desc)
count_desc
      FROM query_six
) tmp
```

```
WHERE count_asc = 1 or count_desc = 1
LIMIT 5;
```

user_id	date	total_amount
999103	2015-09-29 13:32:07	Bangladesh
155	2015-02-28 16:59:45	Bangladesh
999671	22015-09-29 16:12:39	Brazil
234	2015-02-28 17:39:27	Brazil
999806	2015-09-29 16:34:59	China