**SOLUTION: Time delta between consecutive events**

```sql
SELECT user_id,
       unix_timestamp - previous_time AS Delta_SecondLastOne_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;
```

<table>
<thead>
<tr>
<th>user_id</th>
<th>Delta_SecondLastOne_LastOne</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

**SOLUTION: Segment by mobile, web, and cross-device users**

```sql
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;
```

<table>
<thead>
<tr>
<th>WEB_ONLY</th>
<th>MOBILE_ONLY</th>
<th>BOTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>31</td>
<td>52</td>
</tr>
</tbody>
</table>
**SOLUTION: Identify power users based on their history**

```
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;
```

<table>
<thead>
<tr>
<th>user_id</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2015-10-21 06:20:14</td>
</tr>
<tr>
<td>3</td>
<td>2015-10-29 22:41:00</td>
</tr>
<tr>
<td>4</td>
<td>2015-09-25 12:36:10</td>
</tr>
<tr>
<td>5</td>
<td>2015-09-18 06:25:40</td>
</tr>
<tr>
<td>6</td>
<td>2015-12-30 00:53:59</td>
</tr>
</tbody>
</table>

**SOLUTION: Estimate total and running values**

**Query 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;
```

<table>
<thead>
<tr>
<th>user_id</th>
<th>total_amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>67</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>156</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
</tr>
</tbody>
</table>
Query 2. Solution

```sql
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;
```

<table>
<thead>
<tr>
<th>user_id</th>
<th>date</th>
<th>total_amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2015-03-13</td>
<td>67</td>
</tr>
<tr>
<td>3</td>
<td>2015-03-31</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>2015-03-28</td>
<td>63</td>
</tr>
<tr>
<td>4</td>
<td>2015-04-20</td>
<td>156</td>
</tr>
<tr>
<td>5</td>
<td>2015-03-01</td>
<td>45</td>
</tr>
</tbody>
</table>

⚠️ **SOLUTION: Summary stats - Custom implementation of the median vs average**

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

```sql
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 b.
        to_date(transaction_date) AS date_only
    FROM query_five_transactions
    ) b
    ON a.user_id = b.user_id AND a.signup_date = b.date_only
    ) tmp;
```

<table>
<thead>
<tr>
<th>average</th>
<th>median</th>
</tr>
</thead>
<tbody>
<tr>
<td>49.25</td>
<td>49</td>
</tr>
</tbody>
</table>
SOLUTION: Rank users within groups

Query 1. Solution

```sql
SELECT country,
       user_count
FROM
  (SELECT *
   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;
```

<table>
<thead>
<tr>
<th>country</th>
<th>user_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>China</td>
<td>18350</td>
</tr>
<tr>
<td>Vietnam</td>
<td>1</td>
</tr>
</tbody>
</table>

Query 2. Solution

```sql
SELECT user_id,
       created_at,
       country
FROM (SELECT *
       FROM (SELECT *
            FROM (SELECT *
            FROM (SELECT *
            FROM (SELECT *
            FROM query_six
            ) tmp
            ) tmp
            ) tmp
            ) tmp
            ) tmp
WHERE count_asc = 1 OR count_desc = 1
LIMIT 5;
```

<table>
<thead>
<tr>
<th>user_id</th>
<th>date</th>
<th>total_amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>999103</td>
<td>2015-09-29 13:32:07</td>
<td>Bangladesh</td>
</tr>
<tr>
<td>155</td>
<td>2015-02-28 16:59:45</td>
<td>Bangladesh</td>
</tr>
<tr>
<td>999671</td>
<td>22015-09-29 16:12:39</td>
<td>Brazil</td>
</tr>
<tr>
<td>234</td>
<td>2015-02-28 17:39:27</td>
<td>Brazil</td>
</tr>
<tr>
<td>999806</td>
<td>2015-09-29 16:34:59</td>
<td>China</td>
</tr>
</tbody>
</table>