Problem Statement

1. Write an SQL query to report how many units in each category have been ordered on each day of the week.

-- You are the business owner and would like to obtain a sales report
for category items and day of the week.

-- Write an SQL query to report how many units in each category
have been ordered on each day of the week.

-- Return the result table ordered by category.

-- The query result format is in the following example:

-- Orders table:
-- +------------+--------------+-------------+--------------+-------------+
-- | order_id | customer_id | order_date | item_id | quantity |
-- +------------+--------------+-------------+--------------+-------------+
-- | 1 | 1 | 2020-06-01 | 1 | 10 |
-- | 2 | 1 | 2020-06-08 | 2 | 10 |
-- | 3 | 2 | 2020-06-02 | 1 | 5 |
-- | 4 | 3 | 2020-06-03 | 3 | 5 |
-- | 5 | 4 | 2020-06-04 | 4 | 1 |
-- | 6 | 4 | 2020-06-05 | 5 | 5 |
-- | 7 | 5 | 2020-06-05 | 1 | 10 |
-- | 8 | 5 | 2020-06-14 | 4 | 5 |
-- | 9 | 5 | 2020-06-21 | 3 | 5 |
-- +------------+--------------+-------------+--------------+-------------+

-- Items table:
-- +------------+---------------+---------------+
-- | item_id | item_name | item_category |
-- +------------+---------------+---------------+
-- | 1 | LC Alg. Book | Book |
-- | 2 | LC DB. Book | Book |
-- | 3 | LC SmarthPhone| Phone |
-- | 4 | LC Phone 2020 | Phone |
-- | 5 | LC SmartGlass | Glasses |
-- | 6 | LC T-Shirt XL | T-Shirt |
-- +------------+---------------+---------------+

-- Result table:
-- +------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
-- | Category | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
Sunday |

-- +------------+----------+----------+----------+----------+----------+----------+----------+
-- | Book | 20 | 5 | 0 | 0 | 10 | 0 | 0 |
-- | Glasses | 0 | 0 | 0 | 0 | 5 | 0 | 0 |
-- | Phone | 0 | 0 | 5 | 1 | 0 | 0 | 10 |
-- | T-Shirt | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-- +------------+----------+----------+----------+----------+----------+----------+----------+
-- On Monday (2020-06-01, 2020-06-08) were sold a total of 20 units (10 + 10)
in the category Book (ids: 1, 2).
-- On Tuesday (2020-06-02) were sold a total of 5 units
in the category Book (ids: 1, 2).
-- On Wednesday (2020-06-03) were sold a total of 5 units
in the category Phone (ids: 3, 4).
-- On Thursday (2020-06-04) were sold a total of 1 unit
in the category Phone (ids: 3, 4).
-- On Friday (2020-06-05) were sold 10 units in the category
Book (ids: 1, 2) and 5 units in Glasses (ids: 5).
-- On Saturday there are no items sold.
-- On Sunday (2020-06-14, 2020-06-21) were sold a total of
10 units (5 +5) in the category Phone (ids: 3, 4).
-- There are no sales of T-Shirt.

2. Write an SQL query to find employees who earn the top three salaries in each of the departments. For the above tables, your SQL query should return the following rows (order of rows does not matter).

-- The Employee table holds all employees. Every employee has an Id, and
There is also a column for the department Id.

-- +----+-------+--------+--------------+
-- | Id | Name | Salary | DepartmentId |
-- +----+-------+--------+--------------+
-- | 1 | Joe | 85000 | 1 |
-- | 2 | Henry | 80000 | 2 |
-- | 3 | Sam | 60000 | 2 |
-- | 4 | Max | 90000 | 1 |
-- | 5 | Janet | 69000 | 1 |
-- | 6 | Randy | 85000 | 1 |
-- | 7 | Will | 70000 | 1 |
-- +----+-------+--------+--------------+
-- The Department table holds all departments of the company.

-- +----+----------+
-- | Id | Name |
-- +----+----------+
-- | 1 | IT |
-- | 2 | Sales |
-- +----+----------+

-- Write a SQL query to find employees who earn the top three salaries in each of the department. For the above tables, your SQL query should return the following rows (order of rows does not matter).

```
-- +------------+----------+--------+
-- | Department | Employee | Salary |
-- +------------+----------+--------+
-- | IT         | Max      | 90000  |
-- | IT         | Randy    | 85000  |
-- | IT         | Joe      | 85000  |
-- | IT         | Will     | 70000  |
-- | Sales      | Henry    | 80000  |
-- | Sales      | Sam      | 60000  |
-- +------------+----------+--------+
```

-- Explanation:

-- In IT department, Max earns the highest salary, both Randy and Joe earn the second highest salary,
-- and Will earns the third highest salary.
-- There are only two employees in the Sales department,
-- Henry earns the highest salary while Sam earns the second highest salary.

3. Write an SQL query to compute moving average of how much customer paid in a 7 days window (current day + 6 days before)

-- You are the restaurant owner and you want to analyze a possible expansion (there will be at least one customer every day).

-- Write an SQL query to compute moving average of how much customer paid in a 7 days window (current day + 6 days before) .

-- The query result format is in the following example:

-- Return result table ordered by visited_on.

-- average_amount should be rounded to 2 decimal places, all dates are in the format ('YYYY-MM-DD').

-- Customer table:

```
-- +-------------+--------------+--------------+-------------+
-- | customer_id | name         | visited_on   | amount      |
-- +-------------+--------------+--------------+-------------+
-- | 1           | Jhon         | 2019-01-01   | 100         |
-- | 2           | Daniel       | 2019-01-02   | 110         |
-- | 3           | Jade         | 2019-01-03   | 120         |
-- | 4           | Khaled       | 2019-01-04   | 130         |
-- | 5           | Winston      | 2019-01-05   | 110         |
-- | 6           | Elvis        | 2019-01-06   | 140         |
```

```
-- | 7          | Anna        | 2019-01-07   | 150        |
-- | 8          | Maria       | 2019-01-08   | 80         |
-- | 9          | Jaze        | 2019-01-09   | 110        |
-- | 1          | Jhon        | 2019-01-10   | 130        |
-- | 3          | Jade        | 2019-01-10   | 150        |
-- +------------+-------------+--------------+------------+

-- Result table:
-- +--------------+--------------+----------------+
-- | visited_on   | amount       | average_amount |
-- +--------------+--------------+----------------+
-- | 2019-01-07   | 860          | 122.86         |
-- | 2019-01-08   | 840          | 120            |
-- | 2019-01-09   | 840          | 120            |
-- | 2019-01-10   | 1000         | 142.86         |
-- +--------------+--------------+----------------+

-- 1st moving average from 2019-01-01 to 2019-01-07 has an average_amount of
(100 + 110 + 120 + 130 + 110 + 140 + 150)/7 = 122.86
-- 2nd moving average from 2019-01-02 to 2019-01-08 has an average_amount of
(110 + 120 + 130 + 110 + 140 + 150 + 80)/7 = 120
-- 3rd moving average from 2019-01-03 to 2019-01-09 has an average_amount of
(120 + 130 + 110 + 140 + 150 + 80 + 110)/7 = 120
-- 4th moving average from 2019-01-04 to 2019-01-10 has an average_amount of
(130 + 110 + 140 + 150 + 80 + 110 + 130 + 150)/7 = 142.86
```

4. Write a query to find the shortest distance between these points rounded to 2 decimals.

```
-- Write a query to find the shortest distance between these points rounded to 2 decimals.


-- | x  | y  |
-- |----|----|
-- | -1 | -1 |
-- | 0  | 0  |
-- | -1 | -2 |




-- The shortest distance is 1.00 from point (-1,-1) to (-1,2). So the output should be:


-- | shortest |
-- |----------|
-- | 1.00     |
```

-- Note: The longest distance among all the points is less than 10000.

5. Write an SQL query to find all numbers that appear at least three times consecutively.

-- Write a SQL query to find all numbers that appear at least three times consecutively.

```
-- +----+-----+
-- | Id | Num |
-- +----+-----+
-- | 1  |  1  |
-- | 2  |  1  |
-- | 3  |  1  |
-- | 4  |  2  |
-- | 5  |  1  |
-- | 6  |  2  |
-- | 7  |  2  |
-- +----+-----+
```

-- For example, given the above Logs table, 1 is the only number that appears consecutively for at least three times.

```
-- +-----------------+
-- | ConsecutiveNums |
-- +-----------------+
-- | 1               |
-- +-----------------+
```