Brice CHIVU
Friday 6 November 2020

Data Analyst - written test

Answers using MySQL grammar

1. Social Networks

In social networks, people send friend requests and accept others' requests as well.

Table **request_accepted**

requester_id	accepter_id	accept_date
1	2	2016-06-03
1	3	2016-06-08
2	3	2016-06-08
3	4	2016-06-09

This table holds the data of friend acceptance, while requester_id and accepter_id both are the id of a person.

Write a query to find the people who has most friends and the most friends number under the following rules:

- It is guaranteed that there is only 1 people having the most friends.
- The friend request could only been accepted once, which means there is no multiple records with the same requester_id and accepter_id value.

For the sample data above, the result is:

Result table:

id	num
3	3

The person with id '3' is a friend of people '1', '2' and '4', so he has 3 friends in total, which is the most number than any others.

#Write your MySQL query statement below

Brice's answer:

I am assuming that a user can't send a request to himself.

```
SELECT cte.id,
COUNT(cte.id) AS num
FROM (
        SELECT requester_id AS id
        FROM request_accepted
        UNION ALL                          ## combines the result of the 2 'SELECT'
        SELECT accepter_id AS id
        FROM request_accepted
) AS cte
GROUP BY cte.id
ORDER BY num desc
LIMIT 1;                                   ## limits the result to 1 row only. Since we
                                           ## have ordered by num in a descending way,
                                           ## 'LIMIT 1' will give us only the max
```

2. Office Furniture

The Office Furniture company sells 6 different products, listed in the product table.

Table **product**

product_id	name	price
123A	Desk chair	80
2446	Conference chair	70
312B	Desk	150
4481	Sit/stand desk	250
5552	Filing cabinet	80
6111	Shelving	30

Orders received by the company are stored in the orders table.

Table **orders**

order_num	order_date	product_id
1	2018-01-23	123A
2	2018-01-23	312B
3	2018-01-23	5552
4	2018-01-24	2446
5	2018-01-25	2446

A. Produce the information required for the invoices by listing the order_num, order date, product name and price and estimated delivery date which is 2 months after the order date, order the results by the estimated delivery date.

B. List the product codes (only once) that have exactly 5 characters and no letters. Include the product name.

#Write your MySQL query statement below

Brice's answer:

A.

```
SELECT
o.order_num,
o.order_date,
p.name AS product_name,
p.price,
DATE_ADD(o.order_date, INTERVAL 2 MONTH) AS estimated_delivery_date      ## adds 2 months
FROM product p                                                            ## to the dates
INNER JOIN orders o on p.product_id = o.product_id       ## joins orders with product
ORDER BY estimated_delivery_date;
```

B.

```
SELECT DISTINCT                          ## selects product code + name only once
product_id,
name
FROM product
WHERE product_id REGEXP '^[0-9]{5}$';
## RegExp expression to select only 5 digits and no letter
```

3. Consecutive Numbers.

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

Table **consecutive**

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.

#Write your MySQL query statement below

Brice's answer:

```
SET @counter := 0;          ## sets a variable 'counter' to count the cumulative number
SET @precedent := 0;        ## sets a variable 'precedent' to store the previous value of 'num'

SELECT DISTINCT             ## selects only the distinct 'ConsecutiveNums'
num AS ConsecutiveNums
FROM(
        SELECT
        num,
        MAX(counting) AS maximum    ## selects the maximum of 'counting'
        FROM(
                SELECT
                id,
                num,
                (@counter := IF(num=@precedent, @counter+1, 1)) AS counting,
                ## adds 1 to '@counter' if the previous 'num' is equal to the current 'num',
                ## otherwise '@counter' restarts to 1
                (@precedent := num)
                ## updates '@precedent' to the current 'num'
                from consecutive) AS cte
        GROUP BY num
        HAVING maximum >= 3
        ## selects only the 'num' that repeats at least 3 times consecutively
) AS cte2;
```

4. Seat of Students.

Mary is a teacher in a middle school and she has a table seat storing students' names and their corresponding seat ids.
The column id is continuous increment.
Mary wants to change seats for the adjacent students.
Can you write a SQL query to output the result for Mary?

Table **school**

id	student
1	Abbot
2	Doris
3	Emerson
4	Green
5	Jeames

For the sample input, the output is:

id	student
1	Doris
2	Abbot
3	Green
4	Emerson
5	Jeames

Note:
If the number of students is odd, there is no need to change the last one's seat.

#Write your MySQL query statement below

Brice's answer:

```
SET @id2 := 0;    ## sets a variable

SELECT
DENSE_RANK() OVER (ORDER BY new_id) AS id,
## creates a rank so that we don't end up with a gap in the id when the number of
## students is odd
student
FROM(
        SELECT
        (@id2 := id + IF(MOD(id,2)=0, -1, 1)) AS new_id,
        ## subtracts 1 to @id when the id is even
        ## adds 1 to @id when the id is odd
        student
        FROM school
) AS cte;
```

Thank you for reading.

Brice CHIVU