Day 17 - Users logged in for 5 consecutive days

Problem Statement:

User login table shows the date when each user logged in to the system. Identify the users who logged in for 5 or more consecutive days. Return the user id, start date, end date and no of consecutive days.

Please remember a user can login multiple times during a day but only consider users whose consecutive logins spanned 5 days or more.

Table: user_login

USER_ID	LOGIN_DATE
1	2024-03-01
1	2024-03-02
1	2024-03-03
1	2024-03-04
1	2024-03-06
1	2024-03-10
1	2024-03-11
1	2024-03-12
1	2024-03-13
1	2024-03-14
1	2024-03-20
1	2024-03-25
1	2024-03-26
1	2024-03-27
1	2024-03-28
1	2024-03-29
1	2024-03-30
2	2024-03-01
2	2024-03-02
2	2024-03-03
2	2024-03-04
3	2024-03-01
3	2024-03-02
3	2024-03-03
3	2024-03-04
3	2024-03-04
3	2024-03-04
3	2024-03-05
4	2024-03-01
4	2024-03-02
4	2024-03-03
4	2024-03-04
4	2024-03-04

Expected Output:

USER_ID	START_DATE	END_DATE	CONSECUTIVE_DAYS
1	2024-03-10	2024-03-14	5
1	2024-03-25	2024-03-30	6
3	2024-03-01	2024-03-05	5

SOLUTION BREAKDOWN

This is similar to **Gaps and Islands** Problem statement.
Solving **Islands** Problem statement refers to finding the range of ***existing*** values in a sequence.

STEP 1: Need to group consecutive login dates by user_id. To do this, when we subtract each login date from the sequential row_num, they will result in same date if login dates are consecutive. [login_date – rn = groups]

Example:

2024-03-10 – rownum(6) = 2024-03-04
2024-03-11 – rownum(7) = 2024-03-04
2024-03-20 – rownum(11) = 2024-03-09

```
select user_id,
      login_date,
      ROW_NUMBER() over(partition by user_id order by login_date) as
rn,
      DATEADD(day,-ROW_NUMBER() over(partition by user_id order by
      login_date),login_date) as grp_Set
from user_login
```

	user_id	login_date	rn	grp_Set
1	1	2024-03-01	1	2024-02-29
2	1	2024-03-02	2	2024-02-29
3	1	2024-03-03	3	2024-02-29
4	1	2024-03-04	4	2024-02-29
5	1	2024-03-06	5	2024-03-01
6	1	2024-03-10	6	2024-03-04
7	1	2024-03-11	7	2024-03-04
8	1	2024-03-12	8	2024-03-04
9	1	2024-03-13	9	2024-03-04
10	1	2024-03-14	10	2024-03-04
11	1	2024-03-20	11	2024-03-09
12	1	2024-03-25	12	2024-03-13
13	1	2024-03-26	13	2024-03-13
14	1	2024-03-27	14	2024-03-13
15	1	2024-03-28	15	2024-03-13

Why ROW_NUM() and not other ranking functions?

As per our requirement, user can login multiple times during a day but we only need to consider users whose consecutive logins spanned 5 days or more.
ROW_NUM will assign unique row numbers to same login_date.

	user_id	login_date	rn	grp_Set
19	2	2024-03-02	2	2024-02-29
20	2	2024-03-03	3	2024-02-29
21	2	2024-03-04	4	2024-02-29
22	3	2024-03-01	1	2024-02-29
23	3	2024-03-02	2	2024-02-29
24	3	2024-03-03	3	2024-02-29
25	3	2024-03-04	4	2024-02-29
26	3	2024-03-04	5	2024-02-28
27	3	2024-03-04	6	2024-02-27
28	3	2024-03-05	7	2024-02-27
29	4	2024-03-01	1	2024-02-29

STEP 2: Now we just need to fetch minimum and maximum date in each group and check if the total user_ids in each group is greater than or equal to 5.

```
-- FINAL SOLUTION
With cte as (
      select user_id,login_date,
            DATEADD (day,-ROW_NUMBER() over (partition by user_id
            order by login_date), login_date) as grp_Set
      from user_login
)
select user_id,
      MIN(login_date) as start_date,
      MAX(login_date) as end_date,
      COUNT(user_id) as consecutive_days
from cte
group by user_id, grp_set
having COUNT(user_id) >=5
order by user_id, grp_set
```

	user_id	start_date	end_date	consecutive_days
1	1	2024-03-10	2024-03-14	5
2	1	2024-03-25	2024-03-30	6
3	4	2024-03-01	2024-03-05	5