SQL Project - Web Application Users Data Analysis

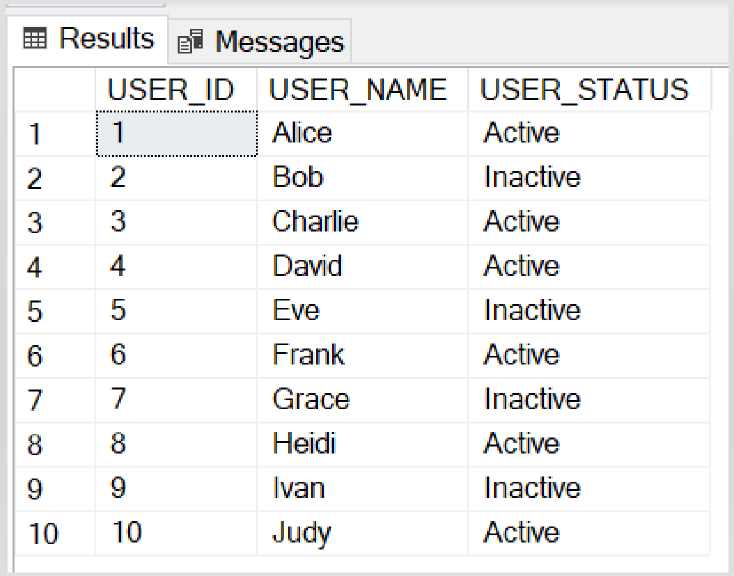
- Tables

  1. Users: USER_ID, USER_NAME,USER_STATUS

  2. Logins: USER_ID, LOGIN_TIMESTAMP,SESSION_ID,SESSION_SCORE

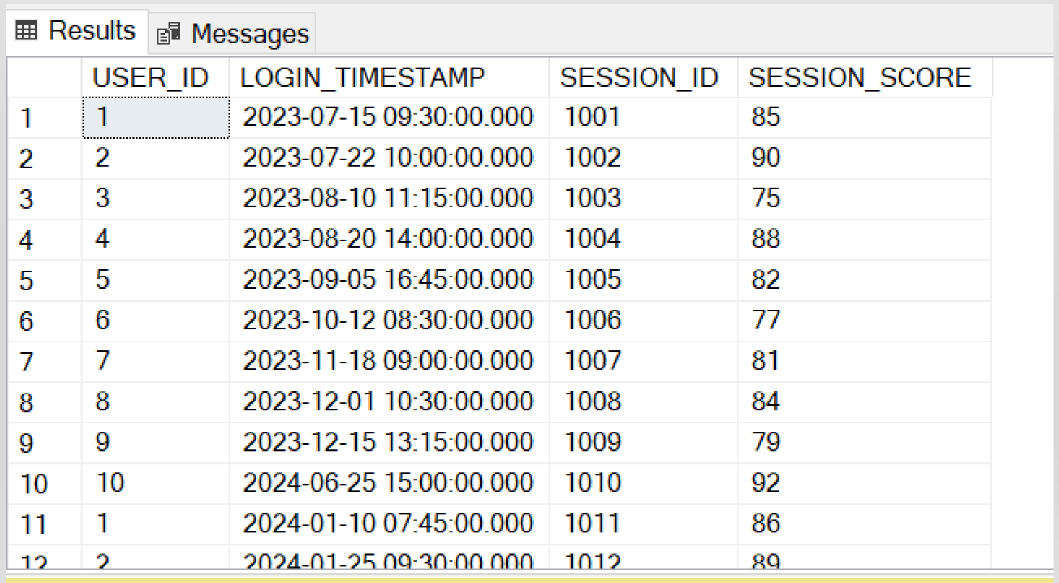
- Objective:

  Analyzing the web application users and their login activity to answer specific questions posed by management, aiding in business decision-making.



	USER_ID	USER_NAME	USER_STATUS
1	1	Alice	Active
2	2	Bob	Inactive
3	3	Charlie	Active
4	4	David	Active
5	5	Eve	Inactive
6	6	Frank	Active
7	7	Grace	Inactive
8	8	Heidi	Active
9	9	Ivan	Inactive
10	10	Judy	Active

USERS



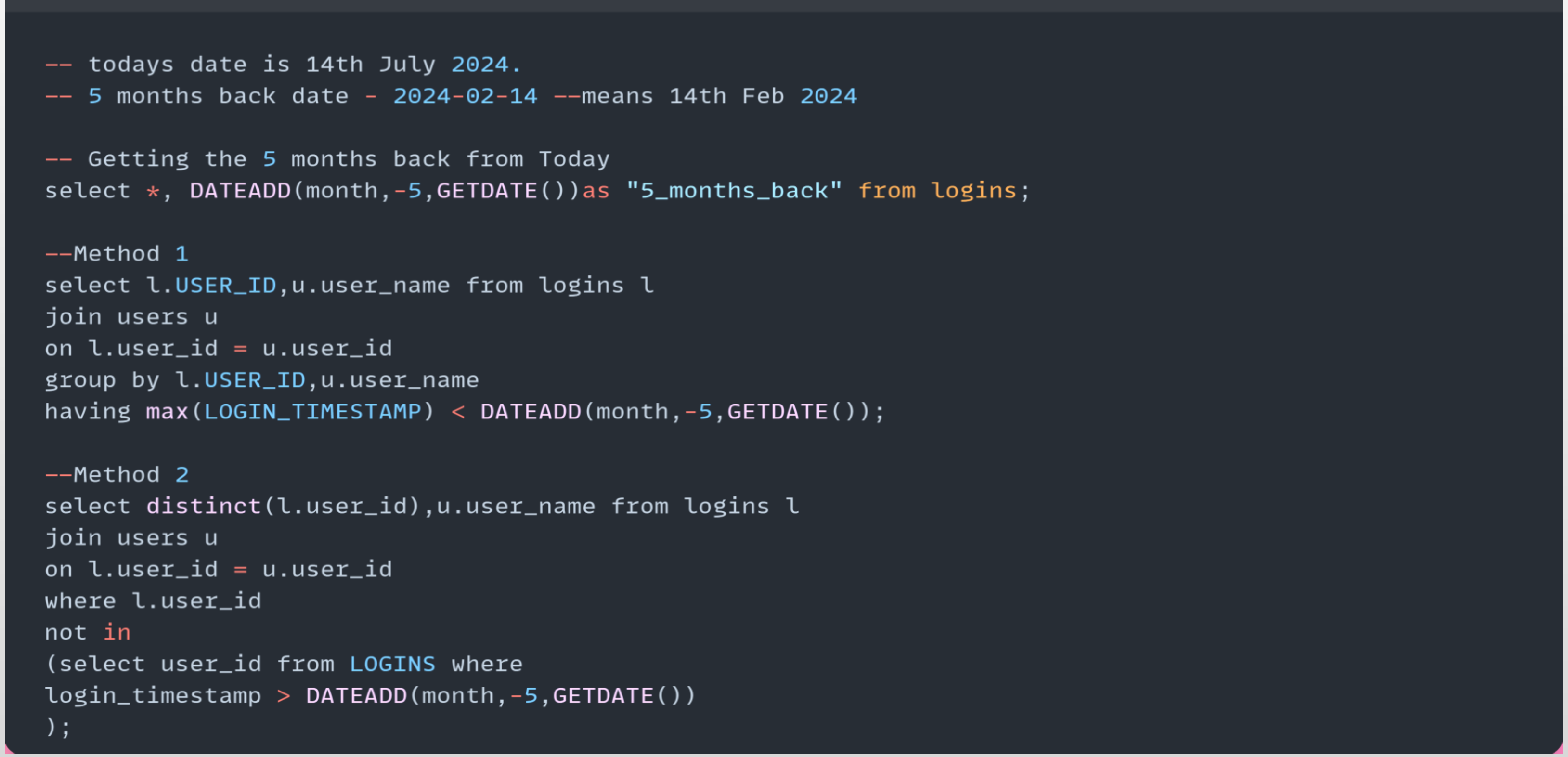
	USER_ID	LOGIN_TIMESTAMP	SESSION_ID	SESSION_SCORE
1	1	2023-07-15 09:30:00.000	1001	85
2	2	2023-07-22 10:00:00.000	1002	90
3	3	2023-08-10 11:15:00.000	1003	75
4	4	2023-08-20 14:00:00.000	1004	88
5	5	2023-09-05 16:45:00.000	1005	82
6	6	2023-10-12 08:30:00.000	1006	77
7	7	2023-11-18 09:00:00.000	1007	81
8	8	2023-12-01 10:30:00.000	1008	84
9	9	2023-12-15 13:15:00.000	1009	79
10	10	2024-06-25 15:00:00.000	1010	92
11	1	2024-01-10 07:45:00.000	1011	86
12	2	2024-01-25 09:30:00.000	1012	89

LOGINS

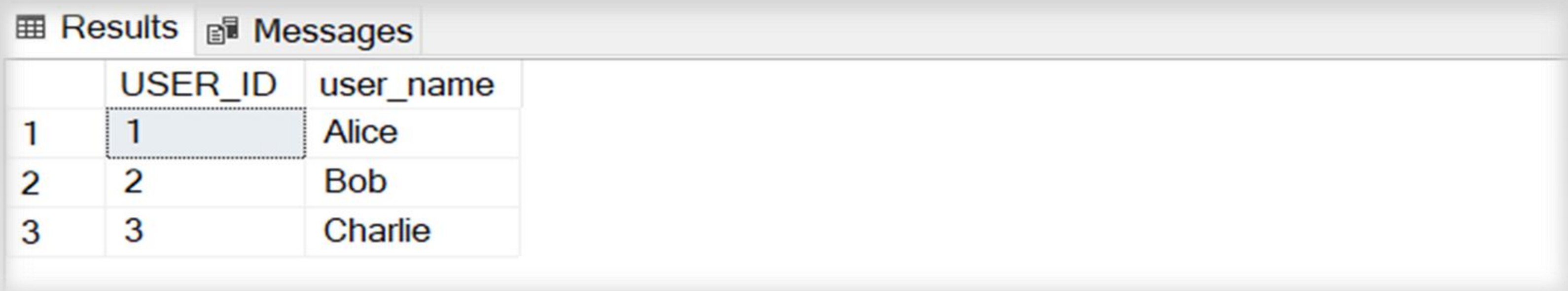
Vinayak_Hiremath_LinkedIn

Q1. Management wants to see all the users who did not login in the past 5 months.
-- return: username.

SQL Query



```sql
-- todays date is 14th July 2024.
-- 5 months back date - 2024-02-14 --means 14th Feb 2024

-- Getting the 5 months back from Today
select *, DATEADD(month,-5,GETDATE())as "5_months_back" from logins;

--Method 1
select l.USER_ID,u.user_name from logins l
join users u
on l.user_id = u.user_id
group by l.USER_ID,u.user_name
having max(LOGIN_TIMESTAMP) < DATEADD(month,-5,GETDATE());

--Method 2
select distinct(l.user_id),u.user_name from logins l
join users u
on l.user_id = u.user_id
where l.user_id
not in
(select user_id from LOGINS where
login_timestamp > DATEADD(month,-5,GETDATE())
);
```



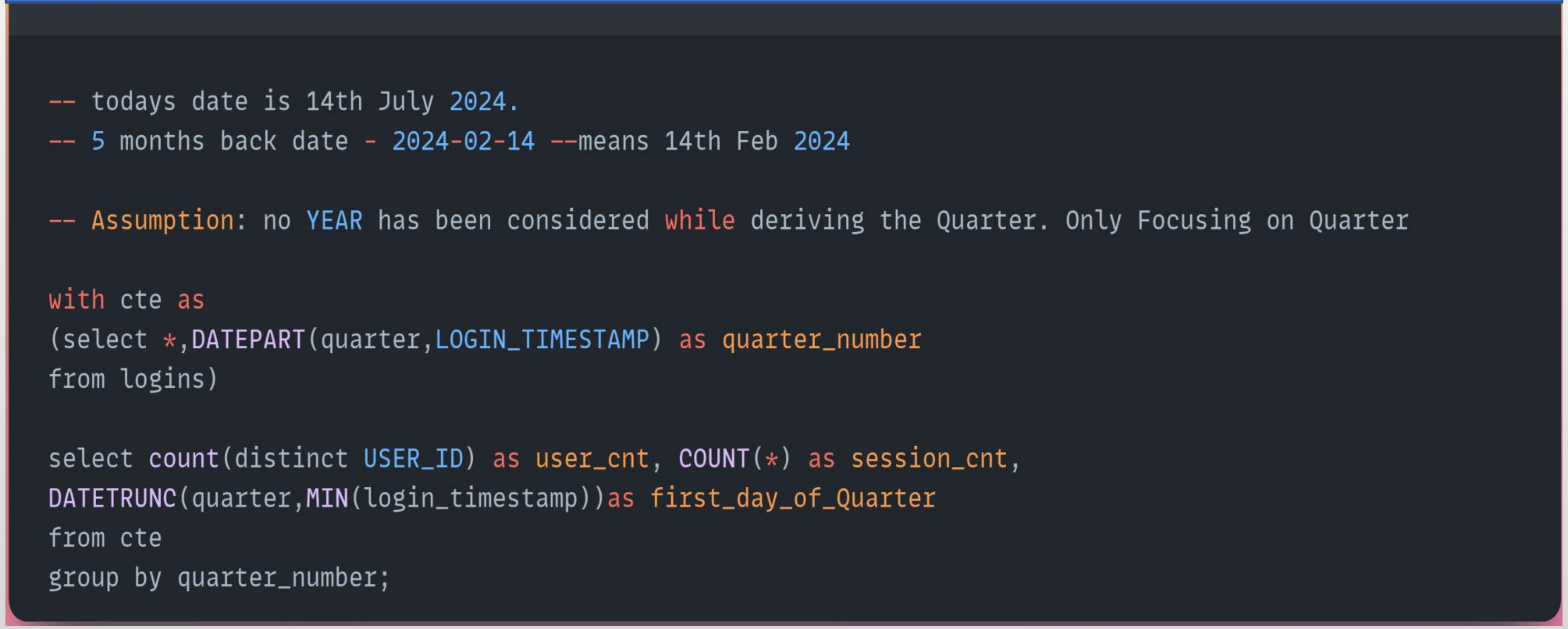
Results Messages

	USER_ID	user_name
1	1	Alice
2	2	Bob
3	3	Charlie

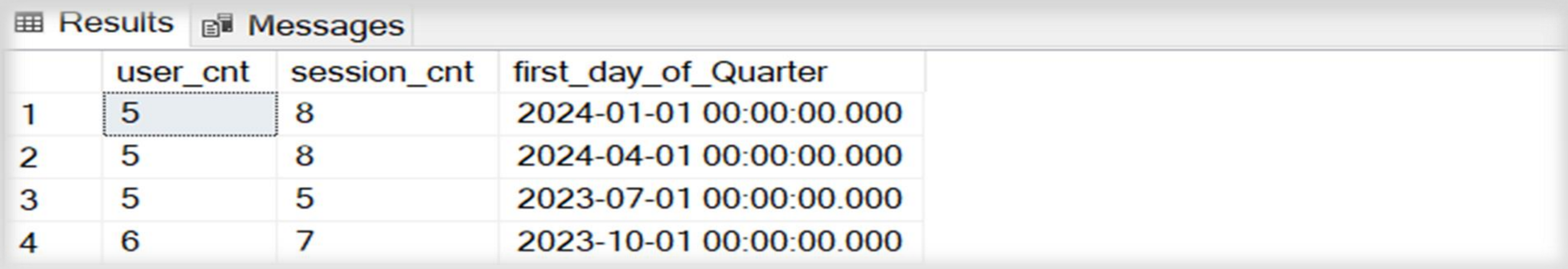
Vinayak Hiremath LinkedIn

Q2. For the Business Units quarterly analysis, calculate how many users and how many sessions were at each quarter.
→ Order by quarter from newest to oldest
→ Return: first day of the quarter,user_cnt, session_cnt.

SQL Query



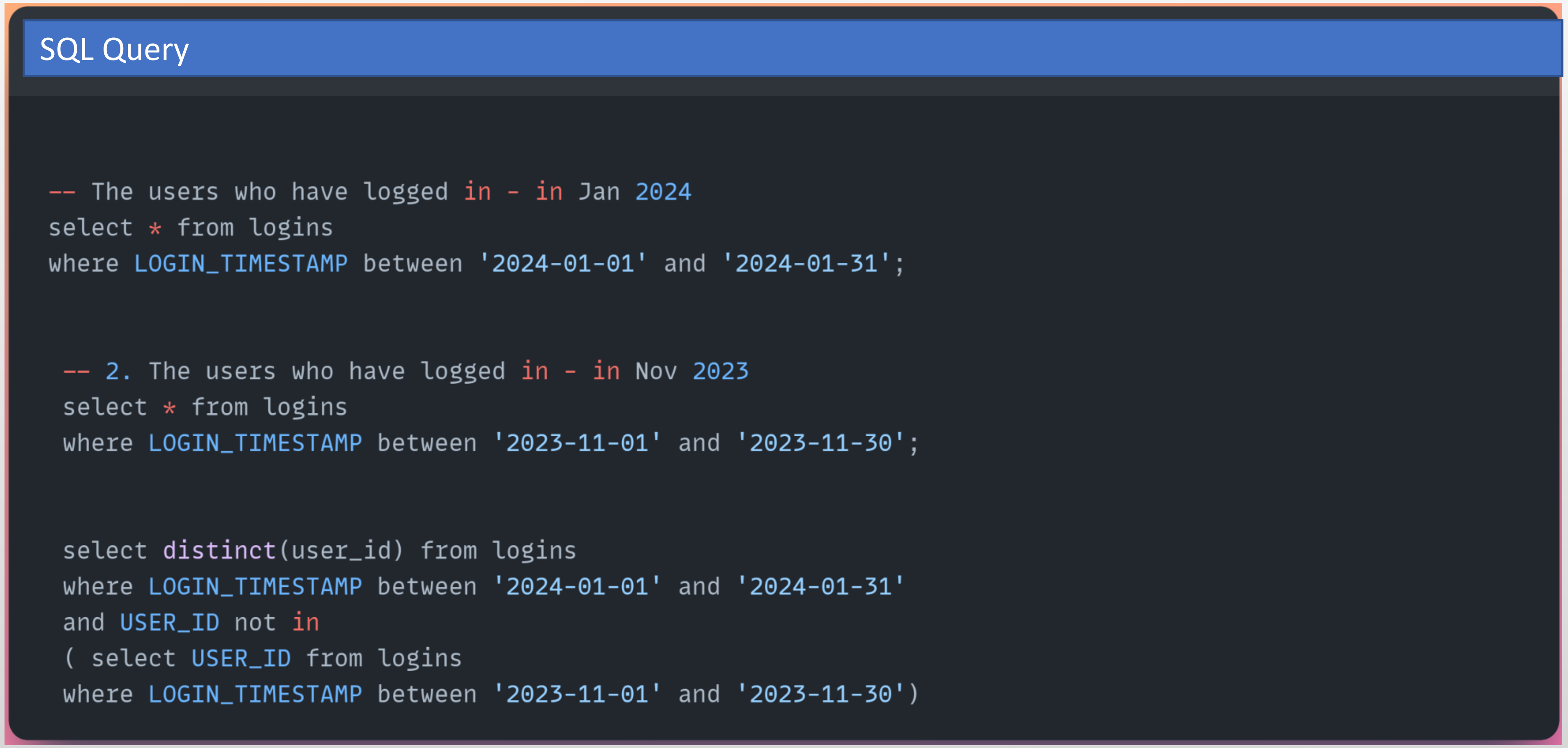
```sql
-- todays date is 14th July 2024.
-- 5 months back date - 2024-02-14 --means 14th Feb 2024

-- Assumption: no YEAR has been considered while deriving the Quarter. Only Focusing on Quarter

with cte as
(select *,DATEPART(quarter,LOGIN_TIMESTAMP) as quarter_number
from logins)

select count(distinct USER_ID) as user_cnt, COUNT(*) as session_cnt,
DATETRUNC(quarter,MIN(login_timestamp))as first_day_of_Quarter
from cte
group by quarter_number;
```



	user_cnt	session_cnt	first_day_of_Quarter
1	5	8	2024-01-01 00:00:00.000
2	5	8	2024-04-01 00:00:00.000
3	5	5	2023-07-01 00:00:00.000
4	6	7	2023-10-01 00:00:00.000

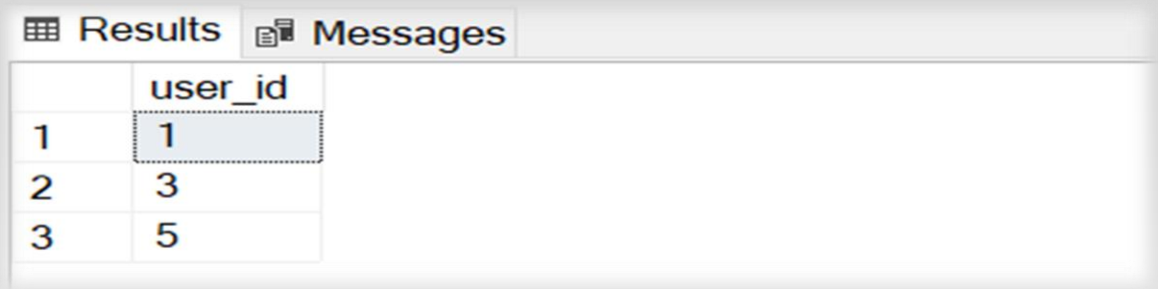
Vinayak_Hiremath_LinkedIn

Q3. Display user_id's that login in Jan 2024 and did not login in Nov 2023.
-- Return: user_id



SQL Query

```sql
-- The users who have logged in - in Jan 2024
select * from logins
where LOGIN_TIMESTAMP between '2024-01-01' and '2024-01-31';


 -- 2. The users who have logged in - in Nov 2023
 select * from logins
 where LOGIN_TIMESTAMP between '2023-11-01' and '2023-11-30';


 select distinct(user_id) from logins
 where LOGIN_TIMESTAMP between '2024-01-01' and '2024-01-31'
 and USER_ID not in
 ( select USER_ID from logins
 where LOGIN_TIMESTAMP between '2023-11-01' and '2023-11-30')
```



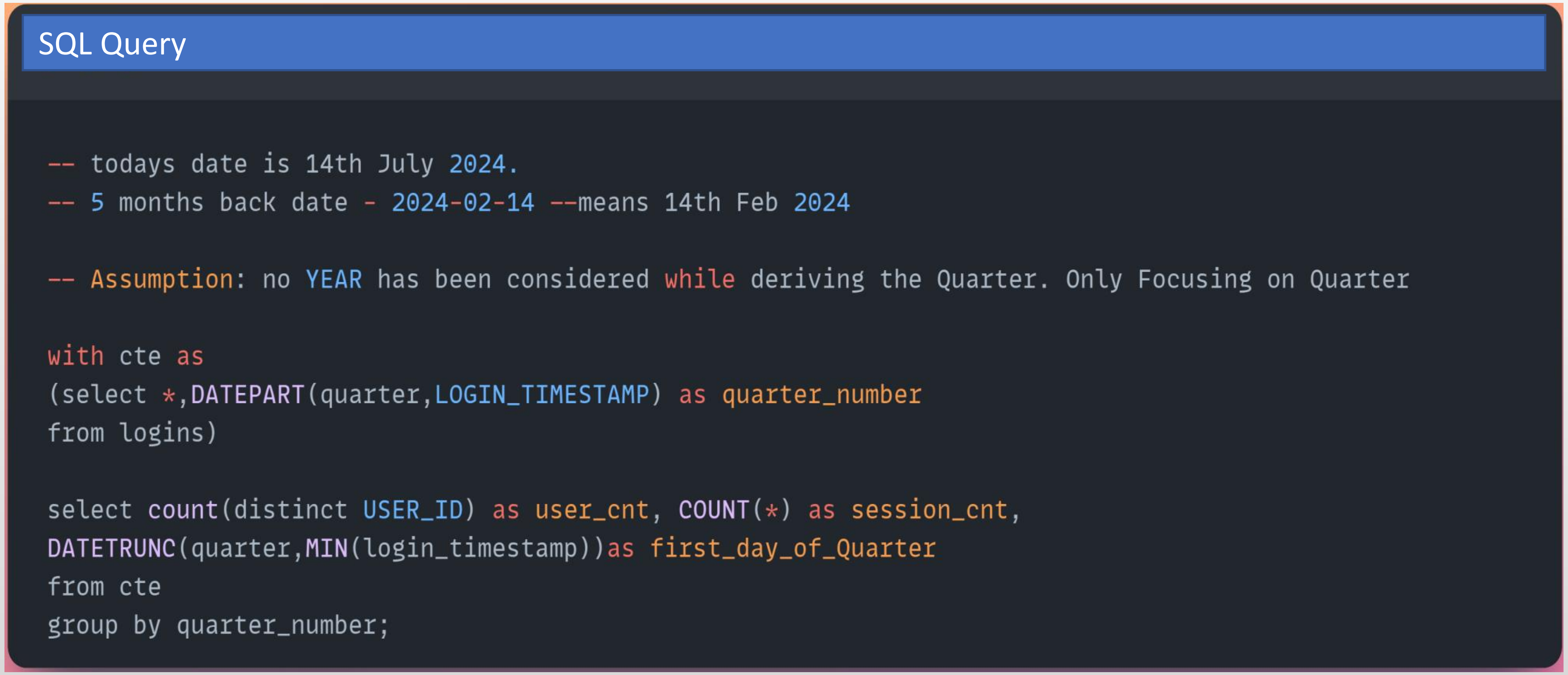
Results Messages

	user_id
1	1
2	3
3	5

Vinayak_Hiremath_LinkedIn

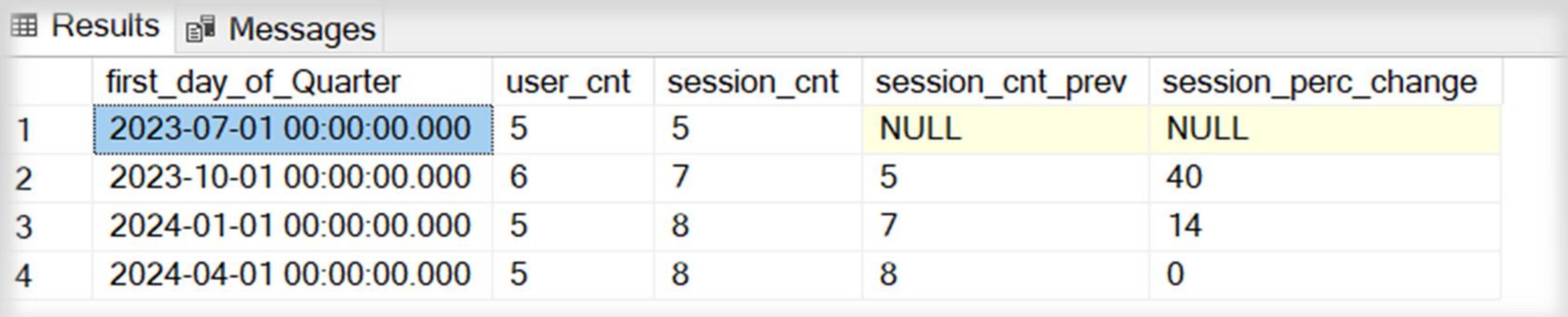
Q4. Add the query from 2 the percentage change in sessions from last quarter.

→ Return: first day of the quarter, session_cnt, session_cnt_prev, session_perc_change



SQL Query

```
-- todays date is 14th July 2024.
-- 5 months back date - 2024-02-14 --means 14th Feb 2024

-- Assumption: no YEAR has been considered while deriving the Quarter. Only Focusing on Quarter

with cte as
(select *,DATEPART(quarter,LOGIN_TIMESTAMP) as quarter_number
from logins)

select count(distinct USER_ID) as user_cnt, COUNT(*) as session_cnt,
DATETRUNC(quarter,MIN(login_timestamp))as first_day_of_Quarter
from cte
group by quarter_number;
```



Results Messages

	first_day_of_Quarter	user_cnt	session_cnt	session_cnt_prev	session_perc_change
1	2023-07-01 00:00:00.000	5	5	NULL	NULL
2	2023-10-01 00:00:00.000	6	7	5	40
3	2024-01-01 00:00:00.000	5	8	7	14
4	2024-04-01 00:00:00.000	5	8	8	0

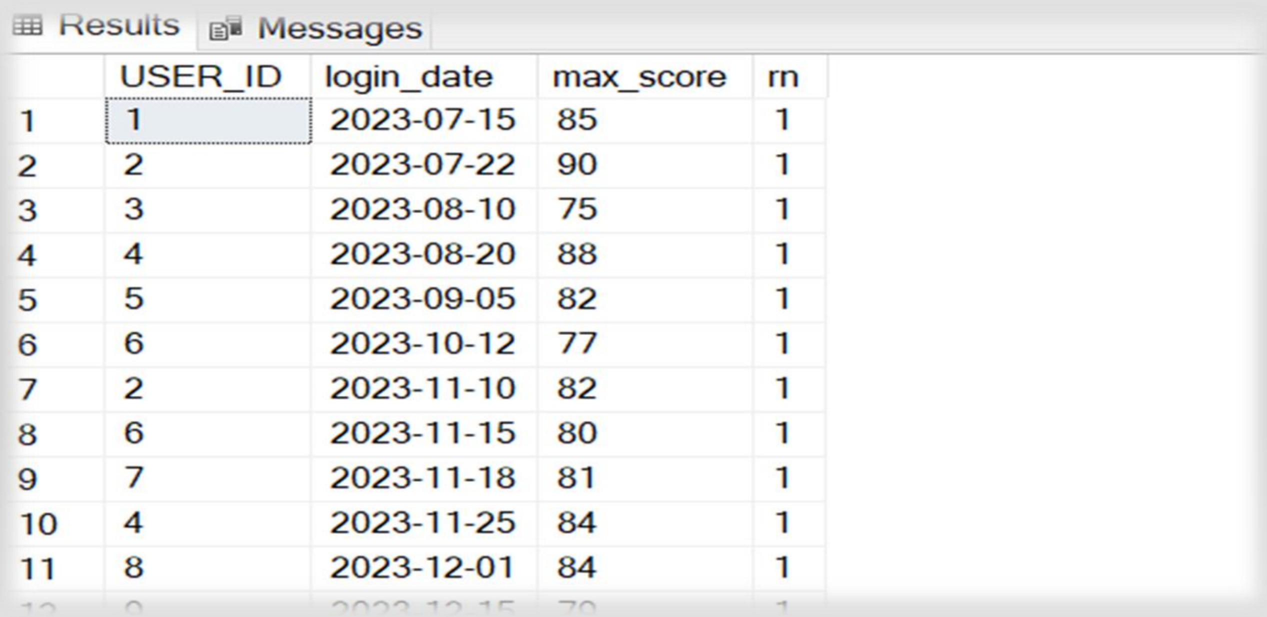
Vinayak_Hiremath_LinkedIn

Q5. Display the user that had the highest session score for each day.

→ Return: date , username, and score

SQL Query

```
with cte as
(
select USER_ID, cast(login_timestamp as date) as login_date,sum(session_score) as max_score
from logins
group by user_id,cast(login_timestamp as date)
)
select * from
(select *, ROW_NUMBER() over (partition by login_date order by max_score desc) as rn
from cte) as a
where rn=1;
```

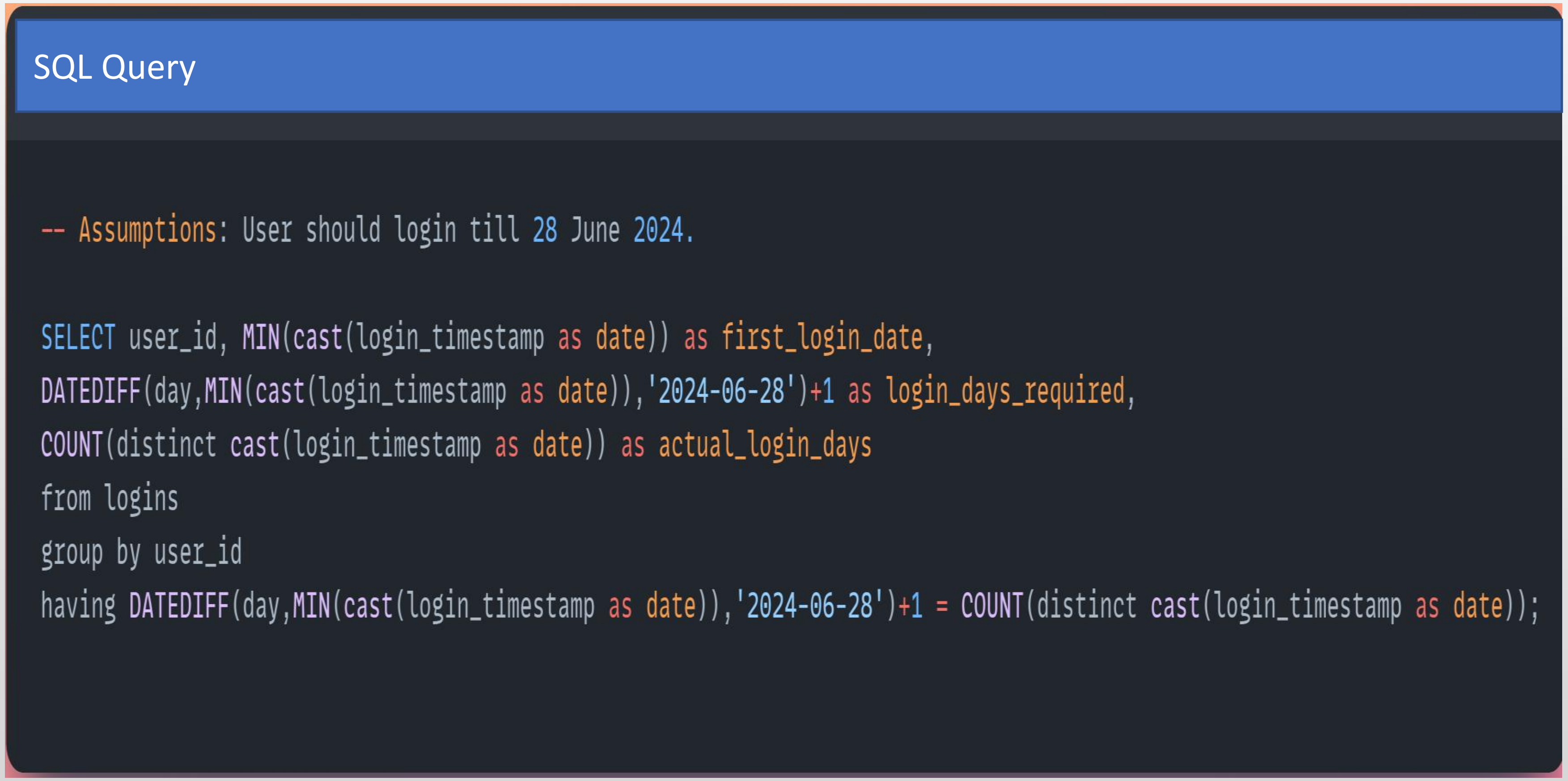

Results Messages

	USER_ID	login_date	max_score	rn
1	1	2023-07-15	85	1
2	2	2023-07-22	90	1
3	3	2023-08-10	75	1
4	4	2023-08-20	88	1
5	5	2023-09-05	82	1
6	6	2023-10-12	77	1
7	2	2023-11-10	82	1
8	6	2023-11-15	80	1
9	7	2023-11-18	81	1
10	4	2023-11-25	84	1
11	8	2023-12-01	84	1
12	9	2023-12-15	79	1

Vinayak_Hiremath_LinkedIn

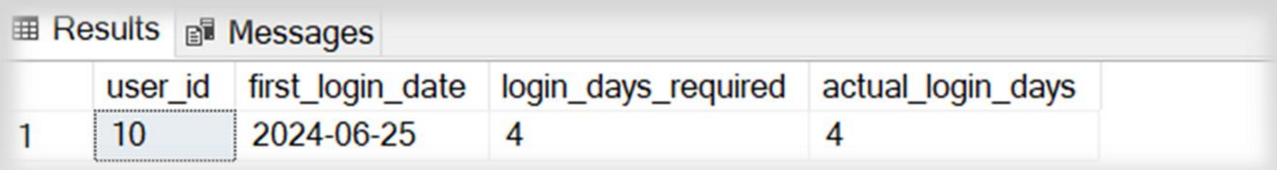
Q6. To identify our best users - Return the users that had a session on every single day since their first login.

- → Make Assumptions if needed
- → Return: User_id



SQL Query

```sql
-- Assumptions: User should login till 28 June 2024.

SELECT user_id, MIN(cast(login_timestamp as date)) as first_login_date,
DATEDIFF(day,MIN(cast(login_timestamp as date)),'2024-06-28')+1 as login_days_required,
COUNT(distinct cast(login_timestamp as date)) as actual_login_days
from logins
group by user_id
having DATEDIFF(day,MIN(cast(login_timestamp as date)),'2024-06-28')+1 = COUNT(distinct cast(login_timestamp as date));
```



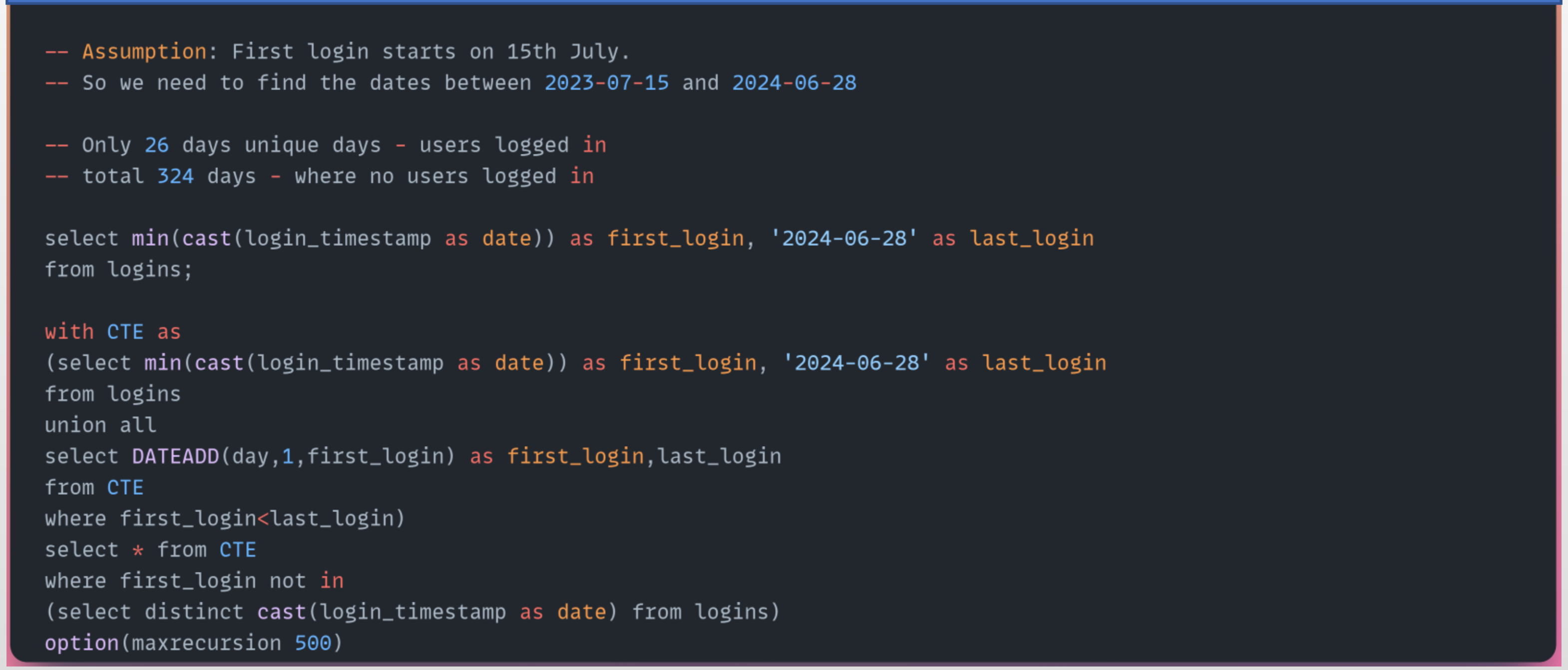
Results | Messages

	user_id	first_login_date	login_days_required	actual_login_days
1	10	2024-06-25	4	4

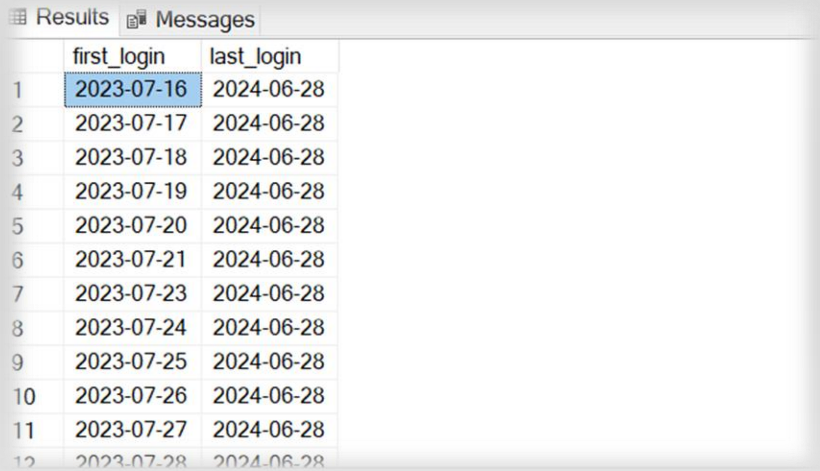
Vinayak_Hiremath_LinkedIn

Q7. on what dates there were no login at all
→ Return: Login Dates

SQL Query



```
-- Assumption: First login starts on 15th July.
-- So we need to find the dates between 2023-07-15 and 2024-06-28

-- Only 26 days unique days - users logged in
-- total 324 days - where no users logged in

select min(cast(login_timestamp as date)) as first_login, '2024-06-28' as last_login
from logins;

with CTE as
(select min(cast(login_timestamp as date)) as first_login, '2024-06-28' as last_login
from logins
union all
select DATEADD(day,1,first_login) as first_login,last_login
from CTE
where first_login<last_login)
select * from CTE
where first_login not in
(select distinct cast(login_timestamp as date) from logins)
option(maxrecursion 500)
```



Results Messages

	first_login	last_login
1	2023-07-16	2024-06-28
2	2023-07-17	2024-06-28
3	2023-07-18	2024-06-28
4	2023-07-19	2024-06-28
5	2023-07-20	2024-06-28
6	2023-07-21	2024-06-28
7	2023-07-23	2024-06-28
8	2023-07-24	2024-06-28
9	2023-07-25	2024-06-28
10	2023-07-26	2024-06-28
11	2023-07-27	2024-06-28
12	2023-07-28	2024-06-28

Vinayak_Hiremath_LinkedIn