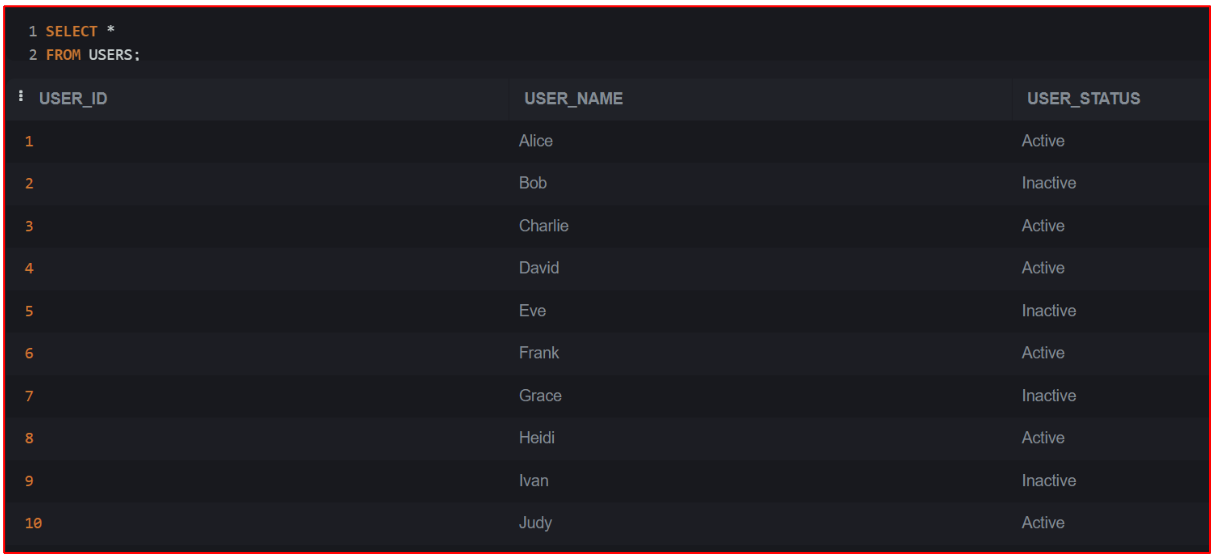
User Activity Analysis Using SQL

Script:

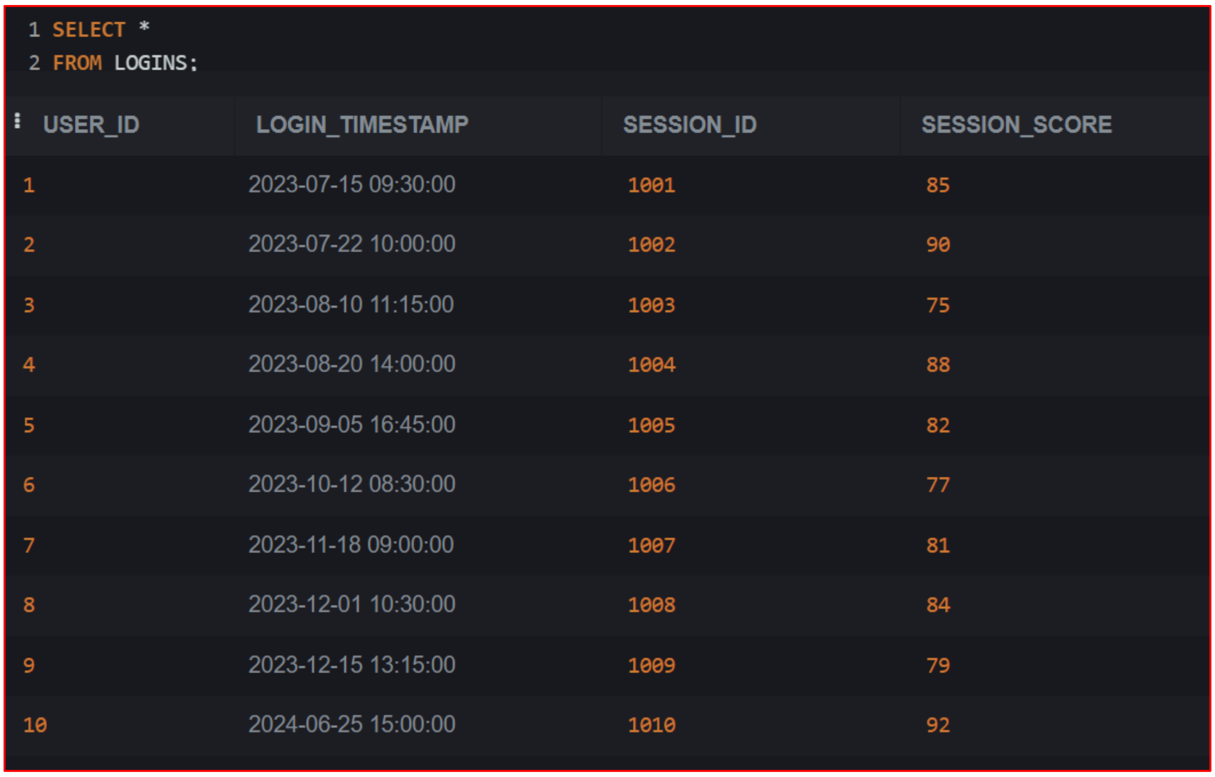
```
CREATE TABLE users (
   USER_ID INT PRIMARY KEY,
   USER_NAME VARCHAR(20) NOT NULL,
   USER_STATUS VARCHAR(20) NOT NULL
);

CREATE TABLE logins (
   USER_ID INT,
   LOGIN_TIMESTAMP DATETIME NOT NULL,
   SESSION_ID INT PRIMARY KEY,
   SESSION_SCORE INT,
   FOREIGN KEY (USER_ID) REFERENCES USERS(USER_ID)
);

-- Users Table
INSERT INTO USERS VALUES (1, 'Alice', 'Active');
INSERT INTO USERS VALUES (2, 'Bob', 'Inactive');
INSERT INTO USERS VALUES (3, 'Charlie', 'Active');
INSERT INTO USERS  VALUES (4, 'David', 'Active');
INSERT INTO USERS  VALUES (5, 'Eve', 'Inactive');
INSERT INTO USERS  VALUES (6, 'Frank', 'Active');
INSERT INTO USERS  VALUES (7, 'Grace', 'Inactive');
INSERT INTO USERS  VALUES (8, 'Heidi', 'Active');
INSERT INTO USERS VALUES (9, 'Ivan', 'Inactive');
INSERT INTO USERS VALUES (10, 'Judy', 'Active');

-- Logins Table

INSERT INTO LOGINS  VALUES (1, '2023-07-15 09:30:00', 1001, 85);
INSERT INTO LOGINS VALUES (2, '2023-07-22 10:00:00', 1002, 90);
INSERT INTO LOGINS VALUES (3, '2023-08-10 11:15:00', 1003, 75);
INSERT INTO LOGINS VALUES (4, '2023-08-20 14:00:00', 1004, 88);
INSERT INTO LOGINS  VALUES (5, '2023-09-05 16:45:00', 1005, 82);

INSERT INTO LOGINS  VALUES (6, '2023-10-12 08:30:00', 1006, 77);
INSERT INTO LOGINS  VALUES (7, '2023-11-18 09:00:00', 1007, 81);
INSERT INTO LOGINS VALUES (8, '2023-12-01 10:30:00', 1008, 84);
INSERT INTO LOGINS  VALUES (9, '2023-12-15 13:15:00', 1009, 79);


-- 2024 Q1
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (1, '2024-01-10
07:45:00', 1011, 86);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (2, '2024-01-25
09:30:00', 1012, 89);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (3, '2024-02-05
11:00:00', 1013, 78);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (4, '2024-03-01
14:30:00', 1014, 91);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (5, '2024-03-15
16:00:00', 1015, 83);
```

INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (6, '2024-04-12 08:00:00', 1016, 80);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (7, '2024-05-18 09:15:00', 1017, 82);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (8, '2024-05-28 10:45:00', 1018, 87);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (9, '2024-06-15 13:30:00', 1019, 76);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (10, '2024-06-25 15:00:00', 1010, 92);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (10, '2024-06-26 15:45:00', 1020, 93);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (10, '2024-06-27 15:00:00', 1021, 92);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (10, '2024-06-28 15:45:00', 1022, 93);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (1, '2024-01-10 07:45:00', 1101, 86);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (3, '2024-01-25 09:30:00', 1102, 89);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (5, '2024-01-15 11:00:00', 1103, 78);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (2, '2023-11-10 07:45:00', 1201, 82);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (4, '2023-11-25 09:30:00', 1202, 84);
INSERT INTO LOGINS (USER_ID, LOGIN_TIMESTAMP, SESSION_ID, SESSION_SCORE) VALUES (6, '2023-11-15 11:00:00', 1203, 80);



```sql
SELECT *
FROM USERS;
```

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



```sql
SELECT *
FROM LOGINS;
```

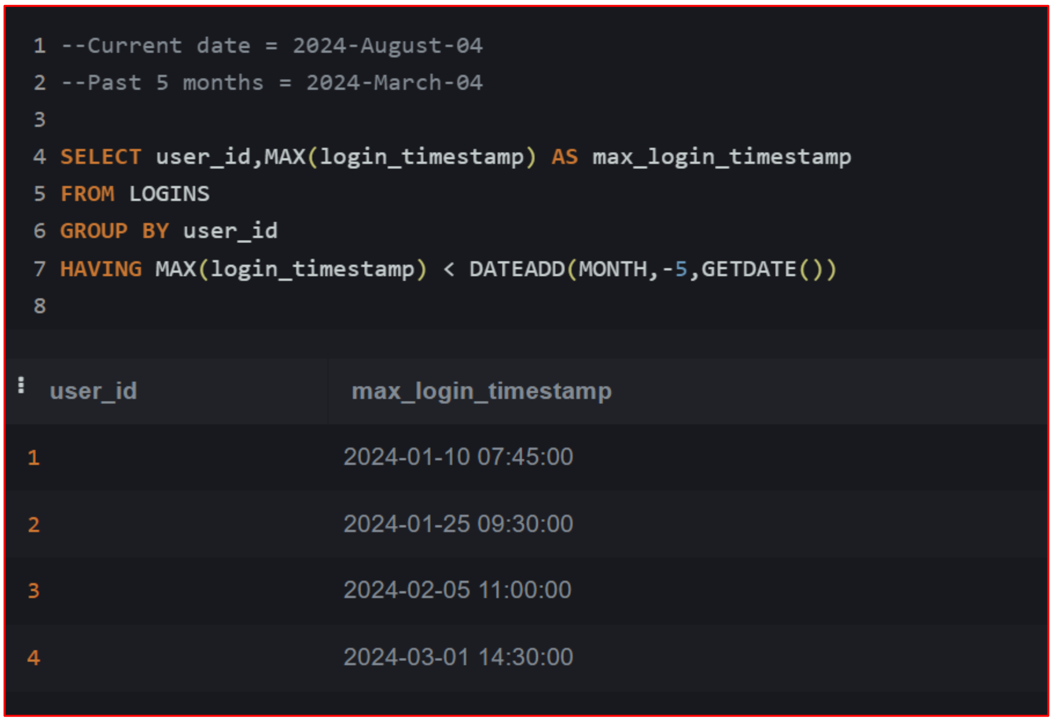
USER_ID	LOGIN_TIMESTAMP	SESSION_ID	SESSION_SCORE
1	2023-07-15 09:30:00	1001	85
2	2023-07-22 10:00:00	1002	90
3	2023-08-10 11:15:00	1003	75
4	2023-08-20 14:00:00	1004	88
5	2023-09-05 16:45:00	1005	82
6	2023-10-12 08:30:00	1006	77
7	2023-11-18 09:00:00	1007	81
8	2023-12-01 10:30:00	1008	84
9	2023-12-15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

Que 1 : Which users did not log in during the past 5 months?

Today's date – 2024-AUGUST-04

Past 5 months date – 2024-MAR-04

SELECT user_id,MAX(login_timestamp)
FROM LOGINS
GROUP BY user_id
HAVING MAX(login_timestamp) < DATEADD(MONTH,-5,GETDATE());

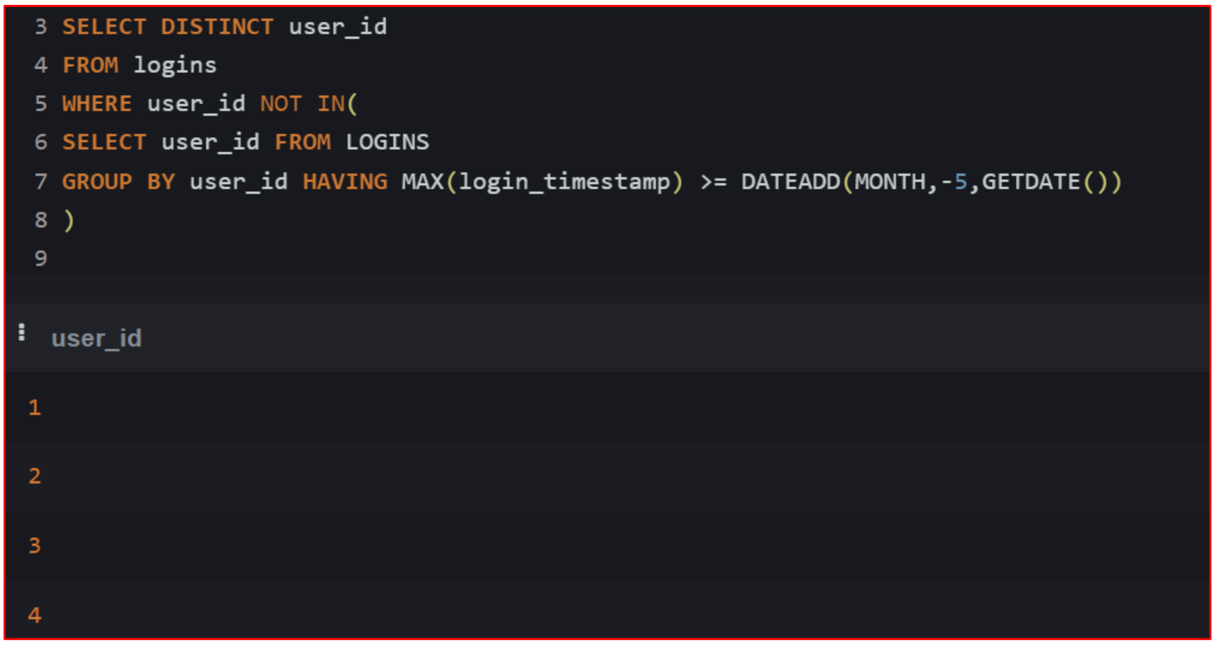


```sql
--Current date = 2024-August-04
--Past 5 months = 2024-March-04

SELECT user_id,MAX(login_timestamp) AS max_login_timestamp
FROM LOGINS
GROUP BY user_id
HAVING MAX(login_timestamp) < DATEADD(MONTH,-5,GETDATE())

```

user_id	max_login_timestamp
1	2024-01-10 07:45:00
2	2024-01-25 09:30:00
3	2024-02-05 11:00:00
4	2024-03-01 14:30:00

2ND Method:

SELECT DISTINCT user_id, login_timestamp
FROM logins
WHERE user_id NOT IN(
SELECT user_id FROM LOGINS
GROUP BY user_id HAVING MAX(login_timestamp) >= DATEADD(MONTH,-5,GETDATE())
)



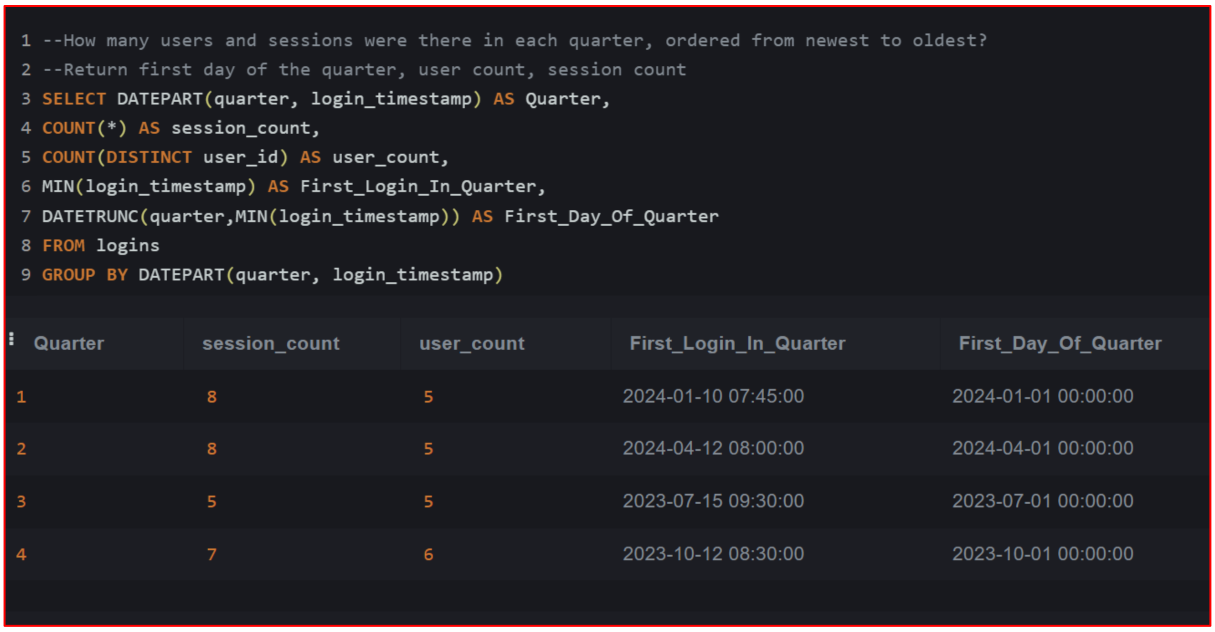
```
SELECT DISTINCT user_id
FROM logins
WHERE user_id NOT IN(
SELECT user_id FROM LOGINS
GROUP BY user_id HAVING MAX(login_timestamp) >= DATEADD(MONTH,-5,GETDATE())
)
```

```
user_id
1
2
3
4
```

Que 2) How many users and sessions were there in each quarter, ordered from newest to oldest?

Return first day of the quarter, user count, session count

SELECT
count(distinct user_id) as user_count,
DATETRUNC(quarter,MIN(login_timestamp)) AS First_Day_Of_Quarter
FROM logins
GROUP BY DATEPART(quarter, login_timestamp);

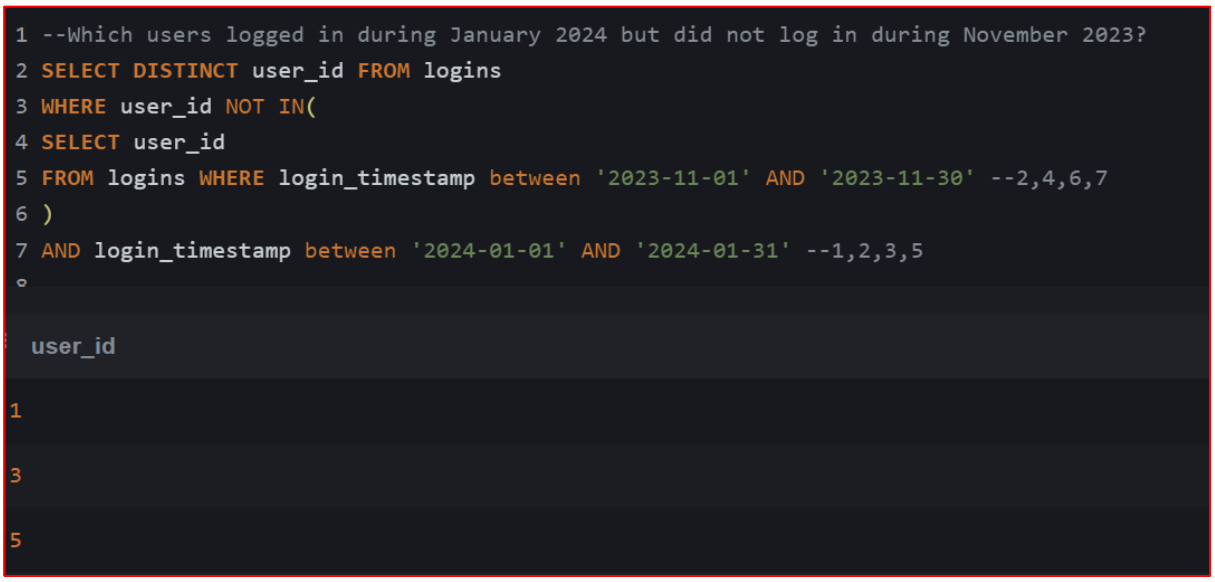


```
--How many users and sessions were there in each quarter, ordered from newest to oldest?
--Return first day of the quarter, user count, session count
SELECT DATEPART(quarter, login_timestamp) AS Quarter,
COUNT(*) AS session_count,
COUNT(DISTINCT user_id) AS user_count,
MIN(login_timestamp) AS First_Login_In_Quarter,
DATETRUNC(quarter,MIN(login_timestamp)) AS First_Day_Of_Quarter
FROM logins
GROUP BY DATEPART(quarter, login_timestamp)
```

Quarter	session_count	user_count	First_Login_In_Quarter	First_Day_Of_Quarter
1	8	5	2024-01-10 07:45:00	2024-01-01 00:00:00
2	8	5	2024-04-12 08:00:00	2024-04-01 00:00:00
3	5	5	2023-07-15 09:30:00	2023-07-01 00:00:00
4	7	6	2023-10-12 08:30:00	2023-10-01 00:00:00

Que 3) Which users logged in during January 2024 but did not log in during November 2023?

SELECT DISTINCT user_id FROM logins
WHERE user_id NOT IN(
SELECT user_id
from logins WHERE login_timestamp between '2023-11-01' AND '2023-11-30' --2,4,6,7
)
AND login_timestamp between '2024-01-01' AND '2024-01-31' --1,2,3,5;



```
--Which users logged in during January 2024 but did not log in during November 2023?
SELECT DISTINCT user_id FROM logins
WHERE user_id NOT IN(
SELECT user_id
FROM logins WHERE login_timestamp between '2023-11-01' AND '2023-11-30' --2,4,6,7
)
AND login_timestamp between '2024-01-01' AND '2024-01-31' --1,2,3,5
```

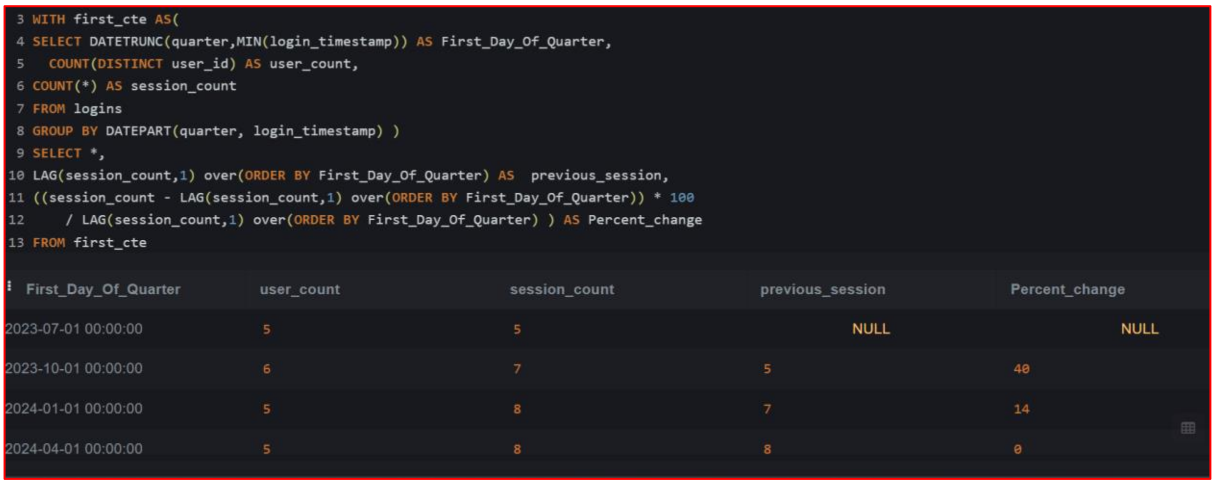
user_id
1
3
5

Que4) What is the percentage change in sessions from the last quarter? Add to question 2 above

--Return first day of the quarter, user count, session count, previous session count, percentage change

with first_cte AS(
SELECT DATETRUNC(quarter,MIN(login_timestamp)) AS First_Day_Of_Quarter,
count(distinct user_id) as user_count,

```
COUNT(*) AS session_count
FROM logins
GROUP BY DATEPART(quarter, login_timestamp) )
SELECT *,
LAG(session_count,1) over(order by First_Day_Of_Quarter) AS  previous_session,
((session_count - LAG(session_count,1) over(order by First_Day_Of_Quarter)) * 100
            / LAG(session_count,1) over(order by First_Day_Of_Quarter) ) AS Percent_change
FROM first_cte
```

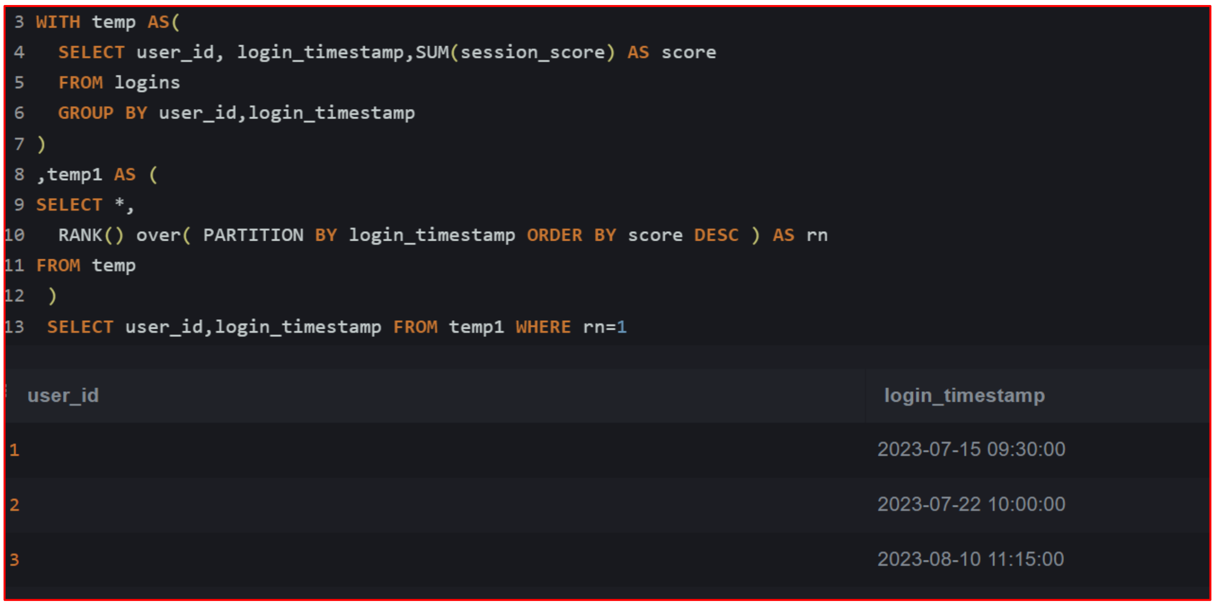


```
WITH first_cte AS(
SELECT DATETRUNC(quarter,MIN(login_timestamp)) AS First_Day_Of_Quarter,
  COUNT(DISTINCT user_id) AS user_count,
COUNT(*) AS session_count
FROM logins
GROUP BY DATEPART(quarter, login_timestamp) )
SELECT *,
LAG(session_count,1) over(ORDER BY First_Day_Of_Quarter) AS  previous_session,
((session_count - LAG(session_count,1) over(ORDER BY First_Day_Of_Quarter)) * 100
    / LAG(session_count,1) over(ORDER BY First_Day_Of_Quarter) ) AS Percent_change
FROM first_cte
```

First_Day_Of_Quarter	user_count	session_count	previous_session	Percent_change
2023-07-01 00:00:00	5	5	NULL	NULL
2023-10-01 00:00:00	6	7	5	40
2024-01-01 00:00:00	5	8	7	14
2024-04-01 00:00:00	5	8	8	0

Que 5) Which user had the highest session score each day?

```
WITH temp AS(
  SELECT user_id, login_timestamp,SUM(session_score) AS score
  FROM logins
  GROUP BY user_id,login_timestamp
)
,temp1 AS (
SELECT *,
  RANK() over( PARTITION by login_timestamp order BY score DESC ) AS rn
FROM temp
 )
 SELECT user_id,login_timestamp FROM temp1 WHERE rn=1
```

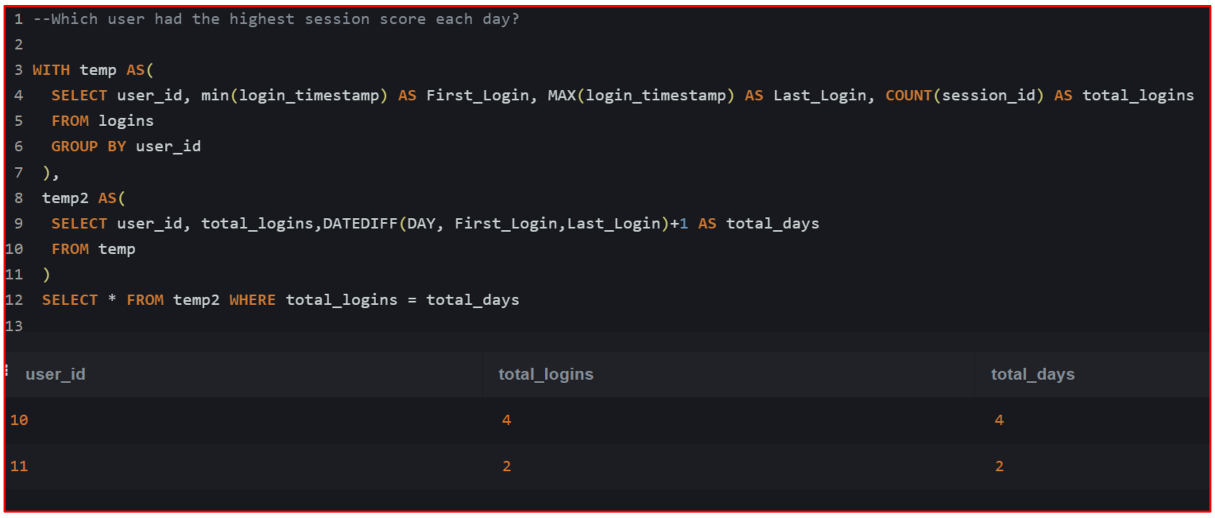


```sql
WITH temp AS(
  SELECT user_id, login_timestamp,SUM(session_score) AS score
  FROM logins
  GROUP BY user_id,login_timestamp
)
,temp1 AS (
SELECT *,
  RANK() over( PARTITION BY login_timestamp ORDER BY score DESC ) AS rn
FROM temp
 )
 SELECT user_id,login_timestamp FROM temp1 WHERE rn=1
```

user_id	login_timestamp
1	2023-07-15 09:30:00
2	2023-07-22 10:00:00
3	2023-08-10 11:15:00

Que 6) Which users have had a session every single day since their first login?

WITH temp AS(
SELECT user_id, min(login_timestamp) AS First_Login, MAX(login_timestamp) AS Last_Login, COUNT(session_id) as total_logins
FROM logins
GROUP BY user_id
),
temp2 AS(
SELECT user_id, total_logins,DATEDIFF(DAY, First_Login,Last_Login)+1 AS total_days
FROM temp
)
*SELECT * FROM temp2 WHERE total_logins = total_days*

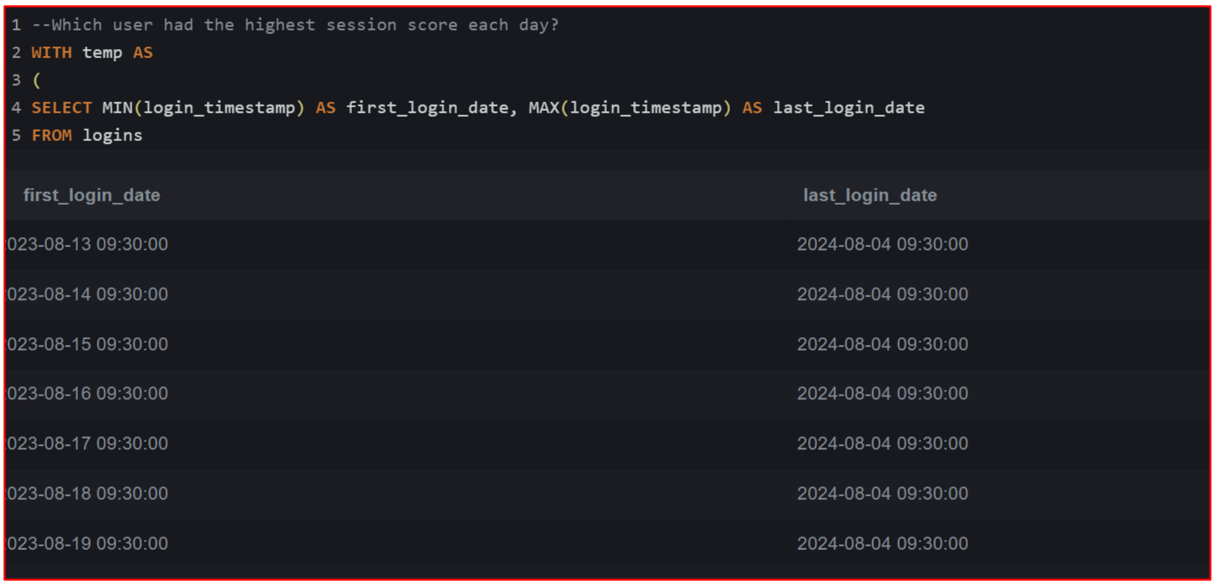


```sql
--Which user had the highest session score each day?

WITH temp AS(
  SELECT user_id, min(login_timestamp) AS First_Login, MAX(login_timestamp) AS Last_Login, COUNT(session_id) AS total_logins
  FROM logins
  GROUP BY user_id
 ),
 temp2 AS(
  SELECT user_id, total_logins,DATEDIFF(DAY, First_Login,Last_Login)+1 AS total_days
  FROM temp
 )
 SELECT * FROM temp2 WHERE total_logins = total_days

```

user_id	total_logins	total_days
10	4	4
11	2	2

Que 7) On what dates were there no logins at all?

```
WITH temp AS
(
select MIN(login_timestamp) AS first_login_date, MAX(login_timestamp) AS last_login_date
from logins
UNION ALL
select DATEADD(DAY,1, first_login_date) AS first_login_date, last_login_date
from temp
WHERE first_login_date < last_login_date
)
SELECT *
FROM temp WHERE first_login_date NOT IN (SELECT DISTINCT login_timestamp from logins)
option(maxrecursion 400)
```



```
--Which user had the highest session score each day?
WITH temp AS
(
SELECT MIN(login_timestamp) AS first_login_date, MAX(login_timestamp) AS last_login_date
FROM logins
```

first_login_date	last_login_date
023-08-13 09:30:00	2024-08-04 09:30:00
023-08-14 09:30:00	2024-08-04 09:30:00
023-08-15 09:30:00	2024-08-04 09:30:00
023-08-16 09:30:00	2024-08-04 09:30:00
023-08-17 09:30:00	2024-08-04 09:30:00
023-08-18 09:30:00	2024-08-04 09:30:00
023-08-19 09:30:00	2024-08-04 09:30:00