```sql
--management wants to see all the users that didn't login in the past 5 months
-- return : Username

select l.USER_ID, max(l.login_timestamp) as last_Login_Date
, DATEDIFF (month, max(l.login_timestamp), GETDATE()) as Login_Diff, u.USER_NAME
from logins l
inner join users u on
l.USER_ID = u.USER_ID
group by l.USER_ID, u.USER_NAME
having DATEDIFF (month, max(l.login_timestamp), GETDATE())>= 5;

/* 2-- for the business units' quarterly anlaysis, calculate how many users and how many
sessions were at each quarter
--order by quarter from newest to oldest
-- return: first_day_oftheQuarter, user_cnt, session_cnt
*/
Select DATEFROMPARTS(year(login_timestamp),
       case when DATEPART(QUARTER, LOGIN_TIMESTAMP) = 1 then 1
              when DATEPART(QUARTER, LOGIN_TIMESTAMP) = 2 then 4
              when DATEPART(QUARTER, LOGIN_TIMESTAMP) = 3 then 7
              when DATEPART(QUARTER, LOGIN_TIMESTAMP) = 4 then 10
       end,
       1) as first_day_of_Quarter
, count(distinct user_id) as Distinct_user_cnt
, count(session_id) as session_cnt
from Logins
group by year(LOGIN_TIMESTAMP), DATEPART(QUARTER, LOGIN_TIMESTAMP)
order by year(LOGIN_TIMESTAMP) desc, DATEPART(QUARTER, LOGIN_TIMESTAMP) desc;

--3. Display user id's that login in january 2024 and did not log-in on november 2023
--return user_id

Select distinct user_id
from logins l
where not exists (
Select USER_ID, year(login_timestamp), Month(LOGIN_TIMESTAMP)
from logins u
where year(login_timestamp) = '2023' and Month(LOGIN_TIMESTAMP) = '11'
and u.USER_ID = l.USER_ID
)
and year(login_timestamp) = '2024' and Month(LOGIN_TIMESTAMP) = '1';

----------------------------
/* 4. Add to the query from Question 2,  the percentage change in sessions from last
quarter
Return : First_Day_Quarter, Session_Cnt, Session_Cnt_Prev, Session_Per_Change

*/

with CTE as (
Select DATEFROMPARTS(year(login_timestamp),
       case when DATEPART(QUARTER, LOGIN_TIMESTAMP) = 1 then 1
              when DATEPART(QUARTER, LOGIN_TIMESTAMP) = 2 then 4
              when DATEPART(QUARTER, LOGIN_TIMESTAMP) = 3 then 7
              when DATEPART(QUARTER, LOGIN_TIMESTAMP) = 4 then 10
       end,
       1) as first_day_of_Quarter
, count(distinct user_id) as Distinct_user_cnt
```

```sql
, count(session_id) as session_cnt
from Logins
group by year(LOGIN_TIMESTAMP), DATEPART(QUARTER, LOGIN_TIMESTAMP)
--order by year(LOGIN_TIMESTAMP) desc, DATEPART(QUARTER, LOGIN_TIMESTAMP) desc
)
Select *, lag(session_cnt, 1) over(order by first_day_of_Quarter)as prev_session_cnt
, (session_cnt - (lag(session_cnt, 1) over(order by first_day_of_Quarter)) ) *100.0 /
lag(session_cnt, 1) over(order by first_day_of_Quarter) as perc_change
from CTE
order by first_day_of_Quarter desc;

/* 5. Display the user that had the highest session score for each day
return: date, user_name, score
*/

Select login_Date, USER_ID, score from (
Select * , ROW_NUMBER() over(partition by login_date order by score desc) as rn from (
Select
cast(login_timestamp as Date) as login_Date
, sum(session_score) as score
, USER_ID
from logins
group by cast(login_timestamp as Date), USER_ID )
A
)B
where rn = 1
order by login_Date;

/*
6. to indentify our best users, return the users that had a session each dayu from thier
first login date
*/

with CTE as (
select  USER_ID, cast(login_timestamp as date) as login_date
,FIRST_VALUE(cast(login_timestamp as Date)) over (partition by user_id order by
cast(login_timestamp as Date) rows between unbounded preceding and unbounded following)
as first_login
, last_VALUE(cast(login_timestamp as Date)) over (partition by user_id order by
cast(login_timestamp as Date) rows between unbounded preceding and unbounded following)
as last_login
from logins)
Select USER_ID, max(DATEDIFF(day, first_login, last_login) + 1 )as Total_days
from CTE
GROUP by user_id
having count(distinct login_date) = max(DATEDIFF(day, first_login, last_login) + 1 );

---optimized way
select user_id
from logins
group by user_id
having
count(distinct cast(login_timestamp as date)) =
DATEDIFF(day, min(cast(login_timestamp as date) ), max(cast(login_timestamp as date)) )
+1;

----7. find dates when there ws no login
with cte as (
```

```sql
Select
min(cast(login_timestamp as date)) as first_login
, max(cast(login_timestamp as date)) as last_login from logins
), r_cte as (
Select
first_login from cte
union all
Select DATEADD(day, 1, first_login)  from r_cte
where DATEADD(day, 1, first_login) <= (select last_login from cte)
)
Select *
from r_cte
where first_login not in (
Select cast(login_timestamp as Date) from logins)
option (maxrecursion 400);
```

Time Taken to Solve each Question:

Problem Statement	Start Time	End Time	Total Minutes
Q1	11:54 AM	12:11 PM	17
Q2	12:19 PM	12:26 PM	7
Q3	12:49 PM	1:02 PM	13
Q4	1:14 PM	1:20 PM	6
Q5	1:33 PM	1:44 PM	10
Q6	1:50 PM	2:00 PM	10
Q7	2:15 PM	2:22 PM	7