Project: User Engagement & Retention Analysis

"Analysed login patterns, inactive users, and high-value customers to drive retention strategies"

**Overview**

The User Login Activity Analysis System is a database-driven project designed to manage and analyse user login activities for a business application. The system tracks user information and their login sessions, enabling the business to derive insights into user engagement, session trends, and activity patterns. The project involves two primary tables: users and logins, with SQL queries developed to address specific business requirements such as identifying inactive users, analysing quarterly session trends, and recognizing top-performing users based on session scores and consistency.

**Database Schema:**

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)

);

**Key Business Questions & Analysis**

* Inactive Users (No Logins in Past 5 Months)
  + Retrieve users who have not logged in since January 28, 2024.
* Quarterly User & Session Analysis
  + Calculate the number of unique users and total sessions per quarter, ordered from newest to oldest.
  + Return: Quarter start date, user count, session count.
* Users Active in January 2024 but Not in November 2023
  + Identify users who logged in January 2024 but had no logins in November 2023.
* Quarterly Session Growth Rate
  + Extend the query from Question 2 to include:
    - Session count
    - Previous quarter’s session count
    - Percentage change in sessions
  + Return: Quarter start date, session count, previous session count, % change.
* Daily Top Performers (Highest Session Score per Day)
  + For each day, find the user with the highest session score.
  + Return: Date, username, max score.
* Most Consistent Users (Logged in Every Day Since First Login)
  + Identify users who logged in every single day since their first login (assume no breaks).
  + Return: User ID.
* Dates with Zero Logins
  + Find dates where no logins occurred at all.
  + Return: Login date (with zero activity).

**Expected Outcomes**

* Identification of inactive users for re-engagement campaigns.
* Insights into quarterly user & session trends for business planning.
* Recognition of top-performing users based on session scores.
* Detection of platform downtime (dates with no logins).
* Understanding user retention patterns (consistent vs. sporadic logins).

**Technologies & Methods Used**

* SQL (Window functions, subqueries, aggregations, date filtering)
* Time-Series Analysis (Quarterly trends, daily activity)
* Data-Driven Decision Making (User engagement strategies, retention analysis)

1. Managment wants to see all the users that did not login in past 5 months

select max(LOGIN\_TIMESTAMP) from logins --2024-06-28 --2024-01-28

--WAY 1

select USER\_ID, MAX(LOGIN\_TIMESTAMP) as MAX\_DATE

from logins

group by USER\_ID

having max(LOGIN\_TIMESTAMP)<'2024-01-28'

--WAY 2

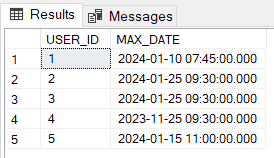
select distinct USER\_ID

from logins

where USER\_ID not in(select USER\_ID

from logins

where LOGIN\_TIMESTAMP>'2024-01-28')



2.For the business units quaterly analysis, calculate how many users and how many sessions were at each quarter,

order by quater from newest to oldest.

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

assumption: considered quater only irrespective of year

select \* from logins

select DATEPART(QUARTER,LOGIN\_TIMESTAMP) as QUATER\_NO

,min(LOGIN\_TIMESTAMP) as QTR\_FIRST\_LOGIN

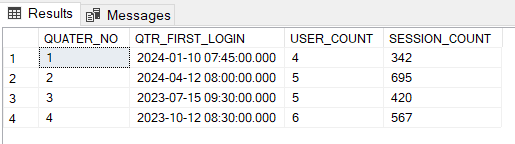
,count(distinct USER\_ID) as USER\_COUNT

,sum(SESSION\_SCORE) as SESSION\_COUNT

--,DATETRUNC(QUARTER,min(LOGIN\_TIMESTAMP)) as FIRST\_QUATER\_DATE

from logins

group by DATEPART(QUARTER,LOGIN\_TIMESTAMP



3. Display user id that log-in in January 2024 and did not login in on November 2023

--january 2024 1 2 3 5

--novemeber 2023 2 4 6 7

select USER\_ID

from logins

where MONTH(LOGIN\_TIMESTAMP)=1 and YEAR(LOGIN\_TIMESTAMP)=2024

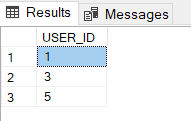
and user\_id not in (select USER\_ID

from logins

where MONTH(LOGIN\_TIMESTAMP)=11 and YEAR(LOGIN\_TIMESTAMP)=2023

group by USER\_ID)

group by USER\_ID



4. Add to the query from question 2 the percentage change in the session form the last quater

return- first day of the quater, session cnt,

session cnt previous, session percentage change.

select \*,(SESSION\_COUNT-CHANGE\_LST\_QTR)\*100/CHANGE\_LST\_QTR as PERCENTAGE\_CHANGE from(

select \*

,LAG(SESSION\_COUNT,1,SESSION\_COUNT) over(order by QUATER\_NO) as CHANGE\_LST\_QTR

from(

select DATEPART(QUARTER,LOGIN\_TIMESTAMP) as QUATER\_NO

,min(LOGIN\_TIMESTAMP) as QTR\_FIRST\_LOGIN

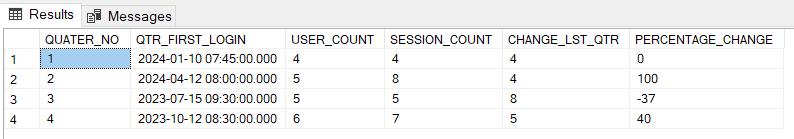
,count(distinct USER\_ID) as USER\_COUNT

,COUNT(\*) as SESSION\_COUNT

--,DATETRUNC(QUARTER,min(LOGIN\_TIMESTAMP)) as FIRST\_QUATER\_DATE

from logins

group by DATEPART(QUARTER,LOGIN\_TIMESTAMP)) A) B



5. Display the user that had highest session score(max) for each day

return: date, username, score

select LOGIN\_DATE

,MAX(SUM\_SCORE) as MAXIMUM\_SESSION\_SCORE

from(

select USER\_ID

,cast(LOGIN\_TIMESTAMP as date) as LOGIN\_DATE

,SUM(SESSION\_SCORE) as SUM\_SCORE

from logins

group by USER\_ID,cast(LOGIN\_TIMESTAMP as date)) A

group by LOGIN\_DATE

order by LOGIN\_DATE

select \*

from(

select \*

,ROW\_NUMBER() over(partition by DATEE order by SUMSCORE desc) as rn

from(

select USER\_ID

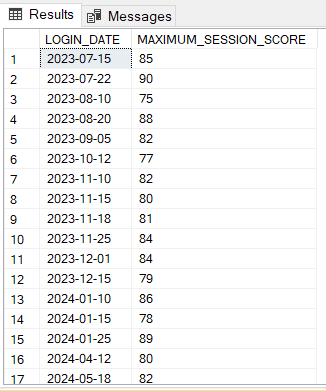
,cast(LOGIN\_TIMESTAMP as date) as DATEE

,sum(SESSION\_SCORE) as SUMSCORE

from logins

group by USER\_ID, cast(LOGIN\_TIMESTAMP as date)) A) B

where rn=1



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

(make assumption if needed)

return- userid

select \*

from(

select \*

,DATEDIFF(DAY,MIN\_LOGIN\_DATE,MAXIMUM\_LOGIN\_DATE)+1 as DAYS\_BETWEEN

from(

select USER\_ID

,MIN(LOGIN\_TIMESTAMP) as MIN\_LOGIN\_DATE

,MAX(LOGIN\_TIMESTAMP) as MAXIMUM\_LOGIN\_DATE

,COUNT(\*) as TOTAL\_NO\_OF\_LOGINS

from logins

group by USER\_ID) A)B

where DAYS\_BETWEEN=TOTAL\_NO\_OF\_LOGINS

select USER\_ID

,MIN(cast(LOGIN\_TIMESTAMP as date)) as FIRST\_LOGIN

,max(cast(LOGIN\_TIMESTAMP as date)) as LAST\_LOGIN

,DATEDIFF(DAY,MIN(cast(LOGIN\_TIMESTAMP as date))

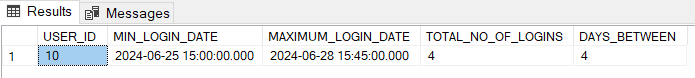
,max(cast(LOGIN\_TIMESTAMP as date)))+1 as DATE\_DIFF

,COUNT(USER\_ID) as TOTAL\_LOGIN

from logins

group by USER\_ID

having COUNT(USER\_ID)=DATEDIFF(DAY,MIN(cast(LOGIN\_TIMESTAMP as date)),max(cast(LOGIN\_TIMESTAMP as date)))+1



7. On what date there were no login at all

--return- login date

select min(LOGIN\_TIMESTAMP)as MIN\_DATE,max(LOGIN\_TIMESTAMP)as MAX\_DATE

from logins

--2023-07-15

--2024-06-28

recursive cte for creating date table from the min date to max date

with cte as (

select cast('2023-07-15' as date) as CAL\_DATE

union all

select DATEADD(day,1,CAL\_DATE) as CAL\_DATE

from cte

where CAL\_DATE<cast('2024-06-28' as date)

)

select \* into CAL\_TABLE

from cte

option(maxrecursion 500)

select \*

from CAL\_TABLE c

left join

(select cast(LOGIN\_TIMESTAMP as date)as REQ\_DATE

from logins) a

on c.CAL\_DATE=a.REQ\_DATE

where REQ\_DATE is null

