Q1) Given an Order Table with the schema (id, user\_id, total, created). Write a SQL Query to create a retention plot. The format for the raw data and output is given. Week Start Date is the 1st Week in which the User\_Id Placed the order, Week 0 is Unique User ids who placed their 1st Order in this week. Out of those ids, Week 1 is unique for users who placed an order in 1st Week + 1, Then Week 2 is 1st Week + 2, and so on till Week 10.

Solution :

create table #order

( id int identity not null ,

userid int,

total float,

created date);

insert into #order values (10,9076,'2020-01-01')

insert into #order values (12,2698,'2020-01-02')

insert into #order values (15,5301,'2020-01-03')

insert into #order values (11,5944,'2020-01-04')

insert into #order values (13,3920,'2020-01-05')

insert into #order values (9,6144,'2020-01-06')

insert into #order values (21,5248,'2020-01-07')

insert into #order values (10,6645,'2020-01-08')

insert into #order values (10,4722,'2020-01-09')

insert into #order values (2,2927,'2020-01-10')

insert into #order values (16,7002,'2020-01-11')

insert into #order values (12,7258,'2020-01-12')

insert into #order values (10,7825,'2020-01-13')

insert into #order values (18,6803,'2020-01-14')

insert into #order values (15,5953,'2020-01-15')

insert into #order values (12,9124,'2020-01-16')

insert into #order values (10,2588,'2020-01-17')

insert into #order values (18,6015,'2020-01-18')

insert into #order values (14,8994,'2020-01-19')

insert into #order values (16,7528,'2020-01-20')

insert into #order values (12,8059,'2020-01-21')

insert into #order values (21,9831,'2020-01-22')

insert into #order values (13,7832,'2020-01-23')

insert into #order values (15,5407,'2020-01-24')

insert into #order values (20,4783,'2020-01-25')

insert into #order values (14,9682,'2020-01-26')

insert into #order values (16,6994,'2020-01-27')

insert into #order values (15,6488,'2020-01-28')

insert into #order values (10,9512,'2020-01-29')

insert into #order values (18,7795,'2020-01-30')

GO

select first\_week,

SUM(CASE WHEN week\_number = 0 THEN 1 ELSE 0 END) AS week\_0,

SUM(CASE WHEN week\_number = 1 THEN 1 ELSE 0 END) AS week\_1,

SUM(CASE WHEN week\_number = 2 THEN 1 ELSE 0 END) AS week\_2,

SUM(CASE WHEN week\_number = 3 THEN 1 ELSE 0 END) AS week\_3,

SUM(CASE WHEN week\_number = 4 THEN 1 ELSE 0 END) AS week\_4,

SUM(CASE WHEN week\_number = 5 THEN 1 ELSE 0 END) AS week\_5,

SUM(CASE WHEN week\_number = 6 THEN 1 ELSE 0 END) AS week\_6,

SUM(CASE WHEN week\_number = 7 THEN 1 ELSE 0 END) AS week\_7,

SUM(CASE WHEN week\_number = 8 THEN 1 ELSE 0 END) AS week\_8,

SUM(CASE WHEN week\_number = 9 THEN 1 ELSE 0 END) AS week\_9

from (

select a.userid,a.order\_placed,b.first\_week ,

a.order\_placed-first\_week as week\_number from

(SELECT

userid,

DATEPART(WEEK, created) -

DATEPART(WEEK, DATEADD(MM, DATEDIFF(MM,0,created), 0))+ 1 AS order\_placed

FROM #order

GROUP BY userid,

(DATEPART(WEEK, created) -

DATEPART(WEEK, DATEADD(MM, DATEDIFF(MM,0,created), 0))+ 1) )a,

(SELECT

userid,

min(DATEPART(WEEK, created) -

DATEPART(WEEK, DATEADD(MM, DATEDIFF(MM,0,created), 0))+ 1) AS first\_week

FROM #order

GROUP BY userid) b

where a.userid=b.userid

) as with\_week\_number

group by first\_week

order by first\_week;

Data:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| first\_week | week\_0 | week\_1 | week\_2 | week\_3 | week\_4 | week\_5 | week\_6 | week\_7 | week\_8 | week\_9 |
| 1 | 4 | 1 | 3 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 5 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Q3) A company records its employee's movement In and Out of the office on a table with 3 columns (Employee id, Action (In/Out), Created) There is NO sample data for this question. You only need to submit the queries

Employee id Action Created

1 In 2019-04-01 12:00:00

1 Out 2019-04-01 15:00:00

1 In 2019-04-01 17:00:00

1 Out 2019-04-01 21:00:00

● The first entry for each employee is “In” ● Every “In” is succeeded by an “Out” ● No data gaps and, the employee can work across days

1. Find the number of employees inside the office at the current time

Solution :

SELECT COUNT(DISTINCT a.employee\_id)

FROM emp\_record a

WHERE a.action = 'IN'

AND NOT EXISTS ( SELECT NULL

FROM emp\_record aa

WHERE a.employee\_id = aa.employee\_id

AND a.created < aa.created

AND aa.action='OUT'

);

1. Find the number of employees inside the Office at “2019-05-01 19:05:00”

Solution :

SELECT COUNT(DISTINCT a.employee\_id)

FROM emp\_record a

WHERE a.action = 'IN'

AND a.created = '2019-05-01 19:05:00' or a.created <' 2019-05-01 19:05:00'

AND NOT EXISTS ( SELECT NULL

FROM emp\_record aa

WHERE a.employee\_id = aa.employee\_id

AND a.created < aa.created

AND aa.action='OUT'

);

1. Measure the number of hours spent by each employee inside the office since the day they started (Account for a current shift if she/he is working)

Solution :

SELECT emplyee\_id,

SUM(TIMESTAMPDIFF(HOUR, created, next\_created) AS NUM\_HOURS

FROM (SELECT t.\*,

LEAD(created) OVER (PARTITION BY emplyee\_id ORDER BY created) AS next\_created

FROM emp\_record

) t

WHERE ACTION = 'IN'

GROUP BY emplyee\_id;

1. Measure the number of hours spent by each employee inside the office between “2019-04-01 14:00:00” and “2019-04-02 10:00:00

Solution :

SELECT emplyee\_id,

SUM(TIMESTAMPDIFF(HOUR, created, next\_created) AS NUM\_HOURS

FROM (SELECT t.\*,

LEAD(created) OVER (PARTITION BY emplyee\_id ORDER BY created) AS next\_created

FROM emp\_record

WHERE created between “2019-04-01 14:00:00” and “2019-04-02 10:00:00”

) t

WHERE ACTION = 'IN'

GROUP BY emplyee\_id;

Q4) An eCommerce company has the following tables table:

T1 (order id, amount, quantity, date, used ID)

Table: T2 (user ID, name, state, city)

Please write SQL queries for the below points

1. MoM (month over month) growth of order from the state of Gujarat

Solution:

Select A.\* , b.\*

Into table

From T1 a join T2 b on a.user\_id=b.user\_id;

SELECT  month,

        revenue,

        revenue - LAG (revenue) OVER (ORDER BY month ASC) AS revenue\_growth,

        (revenue - LAG (revenue) OVER (ORDER BY month ASC))/LAG (revenue) OVER (ORDER BY month ASC)\*100 AS revenue\_percentage\_growth,

        LEAD (revenue, 12) OVER (ORDER BY month ASC) AS next\_year\_revenue

FROM monthly\_revenue;

Select month(date) as month\_date,

count(quantity) as order\_count,

100\* (count(quantity)-lag(count(quantity),1) over (order by date))/lag(count(quantity),1) over(order by date) as order\_growth

From table

Where state=”Gujarat”

Group by month(date)

Order by date

1. which month of 2019 attributes the Highest % change in the total amount

Select month(date) as month\_date,

Sum(amount) as order\_amount,

100\* (Sum(amount) -lag(Sum(amount),1) over (order by date))/lag(Sum(amount),1) over(order by date) as amount\_growth

Into table2

From table

Where year(date)=”2019”

Group by month(date)

Order by date;

select month\_date

from table2

Where amount\_growth =(select max(amount\_growth) from table2)