MEESHO Business Analyst Assessment

Ashwath J

1.Employee Tracker

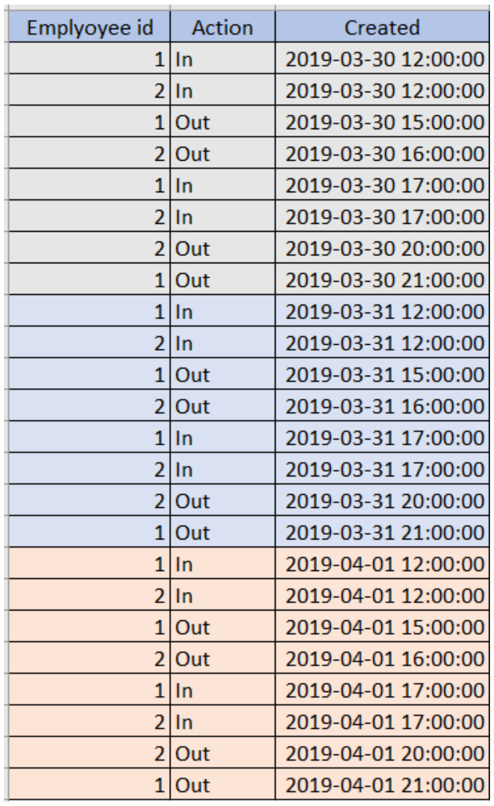
Q1) Find the number of employees inside the office at the current time.

Assumptions:
For simplicity I have assumed there are 2 employees in the office and I have considered the data for the past 3 days.

In order to find number of employees at the current time:
Let's assume the current timestamp - "2019-04-01 21:30:00" (Instead CURRENT_TIMESTAMP keyword can be used).

If the status is 'IN' then the employee is in the office else if the status is 'OUT' the employee is out of the office.

Raw data assumed:-

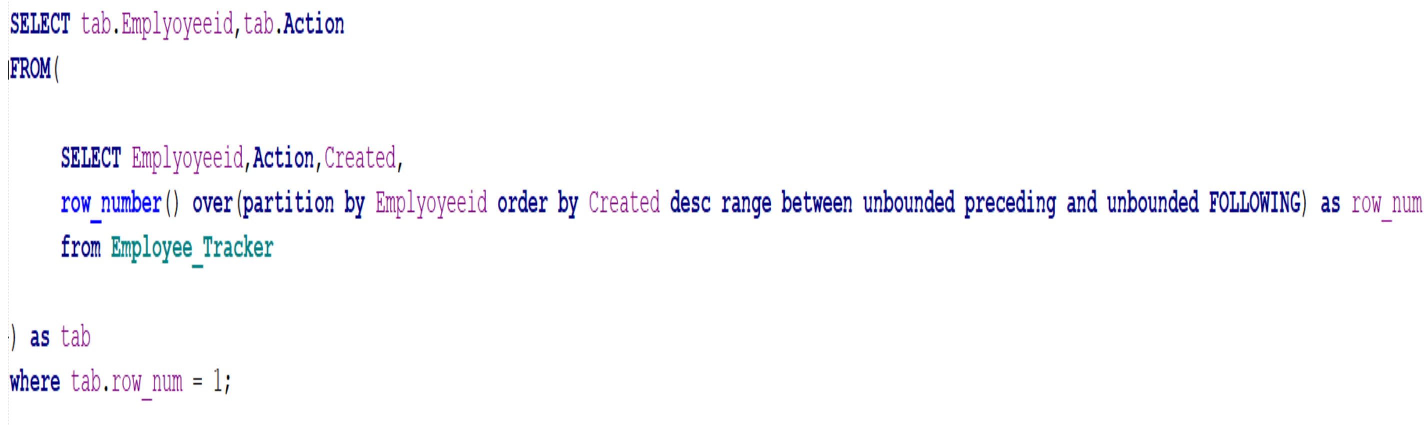


Emplyoyee id	Action	Created
1	In	2019-03-30 12:00:00
2	In	2019-03-30 12:00:00
1	Out	2019-03-30 15:00:00
2	Out	2019-03-30 16:00:00
1	In	2019-03-30 17:00:00
2	In	2019-03-30 17:00:00
2	Out	2019-03-30 20:00:00
1	Out	2019-03-30 21:00:00
1	In	2019-03-31 12:00:00
2	In	2019-03-31 12:00:00
1	Out	2019-03-31 15:00:00
2	Out	2019-03-31 16:00:00
1	In	2019-03-31 17:00:00
2	In	2019-03-31 17:00:00
2	Out	2019-03-31 20:00:00
1	Out	2019-03-31 21:00:00
1	In	2019-04-01 12:00:00
2	In	2019-04-01 12:00:00
1	Out	2019-04-01 15:00:00
2	Out	2019-04-01 16:00:00
1	In	2019-04-01 17:00:00
2	In	2019-04-01 17:00:00
2	Out	2019-04-01 20:00:00
1	Out	2019-04-01 21:00:00

MEESHO Business Analyst Assessment

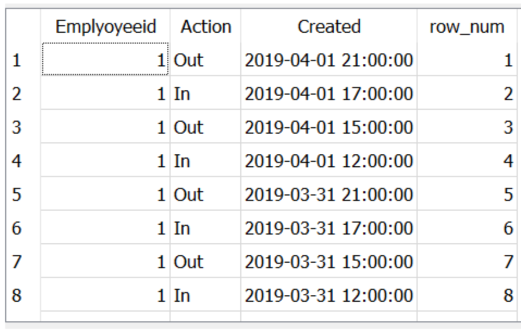
Ashwath J

Query :



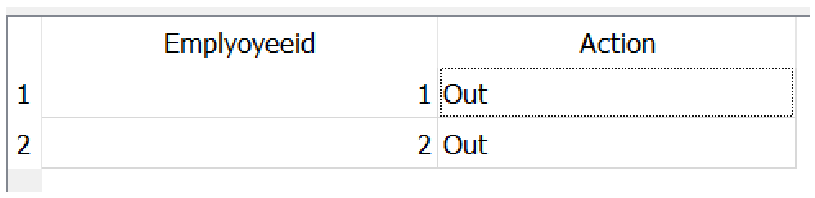
```
SELECT tab.Emplyoyeeid,tab.Action
FROM(

    SELECT Emplyoyeeid,Action,Created,
    row_number() over(partition by Emplyoyeeid order by Created desc range between unbounded preceding and unbounded FOLLOWING) as row_num
    from Employee_Tracker

) as tab
where tab.row_num = 1;
```

Methodology:
Row number windows function is used to partition the employee ids and since our requirement is to find the count of employees at the current time.
Last created status for each employee is considered, and when this status is IN then the employee is inside the office at the current timestamp else he is out of the office.



	Emplyoyeeid	Action	Created	row_num
1	1	Out	2019-04-01 21:00:00	1
2	1	In	2019-04-01 17:00:00	2
3	1	Out	2019-04-01 15:00:00	3
4	1	In	2019-04-01 12:00:00	4
5	1	Out	2019-03-31 21:00:00	5
6	1	In	2019-03-31 17:00:00	6
7	1	Out	2019-03-31 15:00:00	7
8	1	In	2019-03-31 12:00:00	8

Output :
On 2019 - 04 - 01 both our employees left the office before 21:30:00 PM so the status for both is OUT.

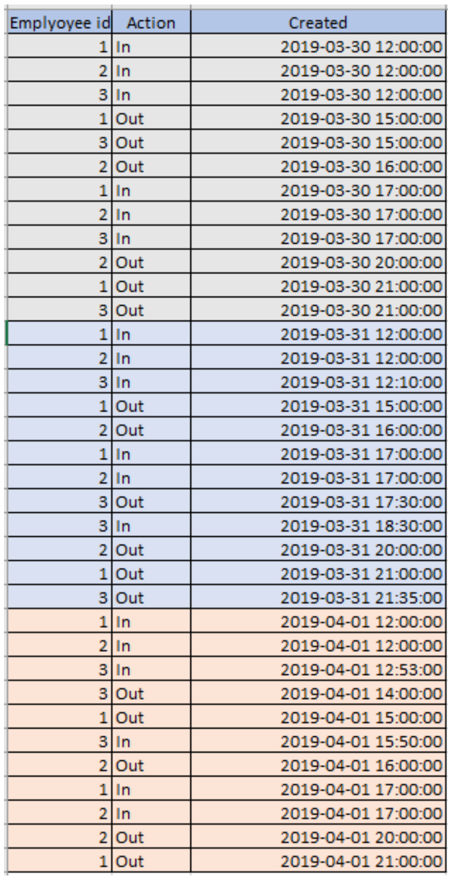


	Emplyoyeeid	Action
1	1	Out
2	2	Out

MEESHO BUSINESS ANALYST ASSESSMENT

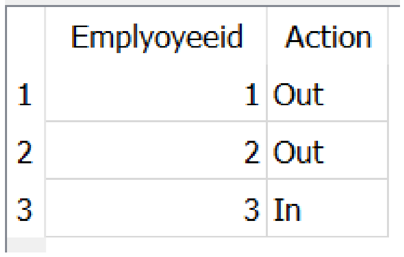
Ashwath J

Verifying the query:(Test data set)
To verify the query I'm adding another employee details who hasn't left the office yet.



Emplyoyee id	Action	Created
1	In	2019-03-30 12:00:00
2	In	2019-03-30 12:00:00
3	In	2019-03-30 12:00:00
1	Out	2019-03-30 15:00:00
3	Out	2019-03-30 15:00:00
2	Out	2019-03-30 16:00:00
1	In	2019-03-30 17:00:00
2	In	2019-03-30 17:00:00
3	In	2019-03-30 17:00:00
2	Out	2019-03-30 20:00:00
1	Out	2019-03-30 21:00:00
3	Out	2019-03-30 21:00:00
1	In	2019-03-31 12:00:00
2	In	2019-03-31 12:00:00
3	In	2019-03-31 12:10:00
1	Out	2019-03-31 15:00:00
2	Out	2019-03-31 16:00:00
1	In	2019-03-31 17:00:00
2	In	2019-03-31 17:00:00
3	Out	2019-03-31 17:30:00
3	In	2019-03-31 18:30:00
2	Out	2019-03-31 20:00:00
1	Out	2019-03-31 21:00:00
3	Out	2019-03-31 21:35:00
1	In	2019-04-01 12:00:00
2	In	2019-04-01 12:00:00
3	In	2019-04-01 12:53:00
3	Out	2019-04-01 14:00:00
1	Out	2019-04-01 15:00:00
3	In	2019-04-01 15:50:00
2	Out	2019-04-01 16:00:00
1	In	2019-04-01 17:00:00
2	In	2019-04-01 17:00:00
2	Out	2019-04-01 20:00:00
1	Out	2019-04-01 21:00:00

Result Output for test case:



	Emplyoyeeid	Action
1	1	Out
2	2	Out
3	3	In

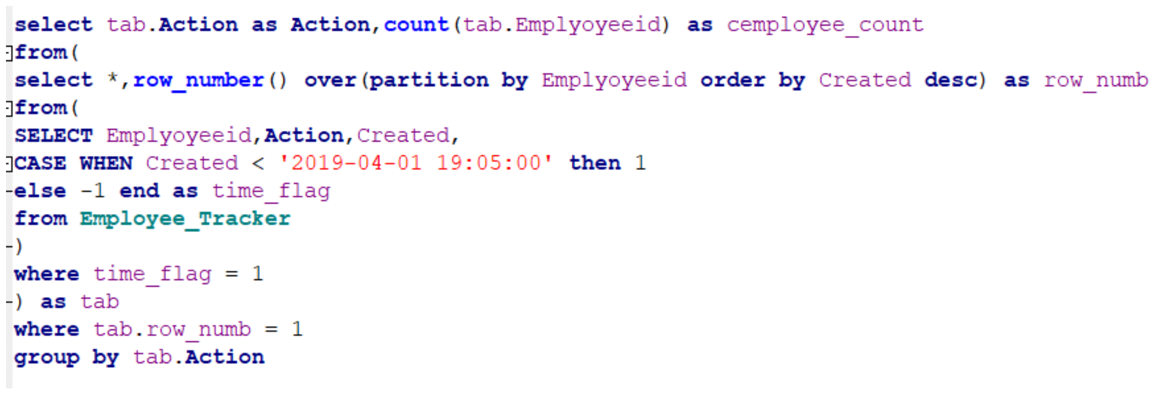
MEESHO BUSINESS ANALYST ASSESSMENT

Ashwath J

Q2) Find number of employees inside the Office at “2019-05-01 19:05:00”.
METHODOLOGY:
Inorder to find the number of employees in the office at “2019-05-01 19:05:00”,we need to find the last Created status row for each employee before the given timestamp and if the status is ‘IN’ we are supposed to count them.

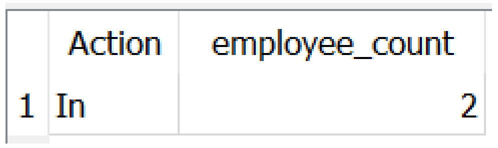
1) A calculated field is calculated using case statement.(if Created is less than the timestamp then 1 else -1).
2) Filter the data using the new calculated field (where the column values equal to 1) using where clause.
3) Create a row_number window function to fetch the last status for the individual employees.

Query:

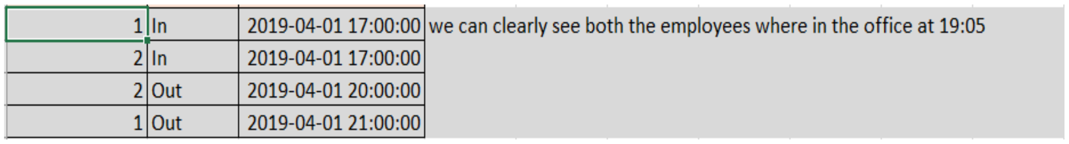


```
select tab.Action as Action,count(tab.Emplyoyeeid) as cemployee_count
from(
select *,row_number() over(partition by Emplyoyeeid order by Created desc) as row_numb
from(
SELECT Emplyoyeeid,Action,Created,
CASE WHEN Created < '2019-04-01 19:05:00' then 1
else -1 end as time_flag
from Employee_Tracker
)
where time_flag = 1
) as tab
where tab.row_numb = 1
group by tab.Action
```

Output:



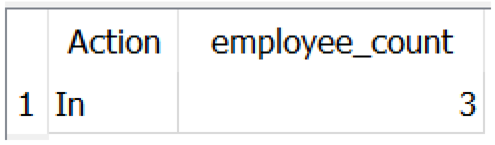
	Action	employee_count
1	In	2



1	In	2019-04-01 17:00:00	we can clearly see both the employees where in the office at 19:05
2	In	2019-04-01 17:00:00	
2	Out	2019-04-01 20:00:00	
1	Out	2019-04-01 21:00:00	

Excel ss to support the previous answer.

Verifying the query:(test data set with 3 employees)



	Action	employee_count
1	In	3

MEESHO BUSINESS ANALYST ASSESSMENT

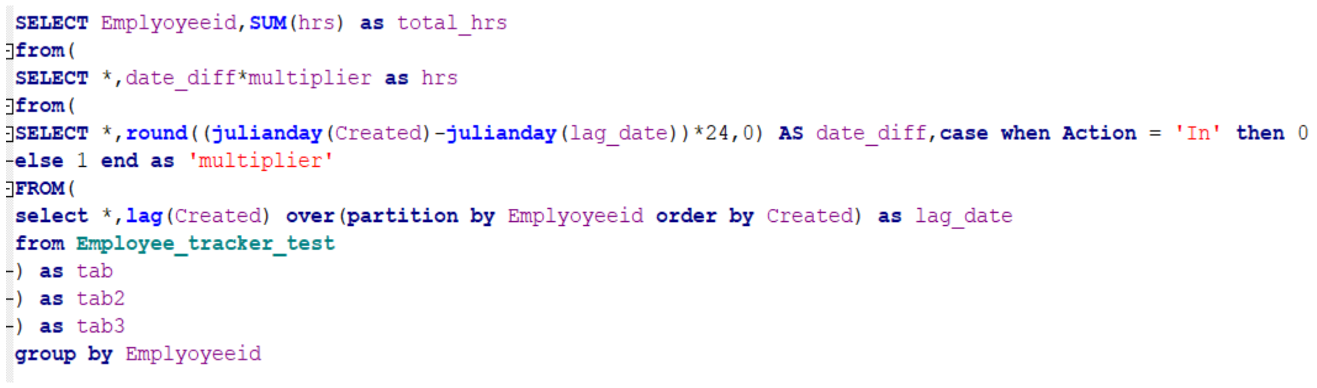
Ashwath J

Q3) Measure the amount of hours spent by each employee inside the office since the day they started (Account for current shift if she/he is working).

Methodology:

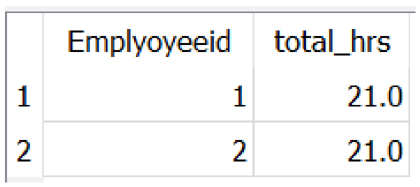
1) A calculated field of lag dates is created using lag windows functions.
2) Difference between the Created date and Lag date is calculated.
3) In order to find the sum of total hours spent inside the office,I am creating an additional column “multiplier”,if the Action status is ‘IN’ then multiplier is 0 else 1.
4) Multiply the multiplier column * date_difference column.(If the Action is IN the lag would be OUT which means this difference in time is spent by the employee outside the office so we are multiplying it with 0 to neglect this value in the sum) and name this column as total_hours.
5) Finally aggregate the total_hours column → SUM(total_hours) employee wise.

Query:



```
SELECT Emplyoyeeid,SUM(hrs) as total_hrs
from(
SELECT *,date_diff*multiplier as hrs
from(
SELECT *,round((julianday(Created)-julianday(lag_date))*24,0) AS date_diff,case when Action = 'In' then 0
else 1 end as 'multiplier'
FROM(
select *,lag(Created) over(partition by Emplyoyeeid order by Created) as lag_date
from Employee_tracker_test
) as tab
) as tab2
) as tab3
group by Emplyoyeeid
```

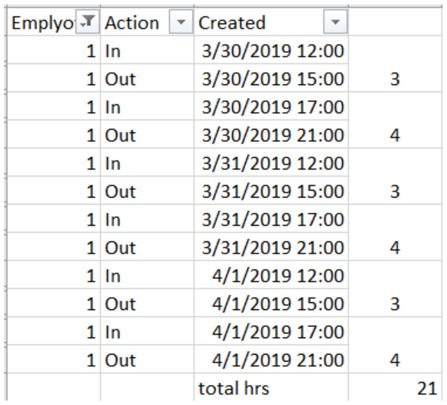
Output:



	Emplyoyeeid	total_hrs
1	1	21.0
2	2	21.0

MEESHO Business Analyst Assessment

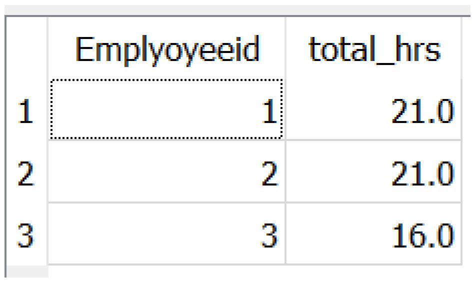
Ashwath J



Emplyo	Action	Created	
1	In	3/30/2019 12:00	
1	Out	3/30/2019 15:00	3
1	In	3/30/2019 17:00	
1	Out	3/30/2019 21:00	4
1	In	3/31/2019 12:00	
1	Out	3/31/2019 15:00	3
1	In	3/31/2019 17:00	
1	Out	3/31/2019 21:00	4
1	In	4/1/2019 12:00	
1	Out	4/1/2019 15:00	3
1	In	4/1/2019 17:00	
1	Out	4/1/2019 21:00	4
		total hrs	21

Excel solution to support the previous answer.

Verifying the query:(test data set with 3 employees)



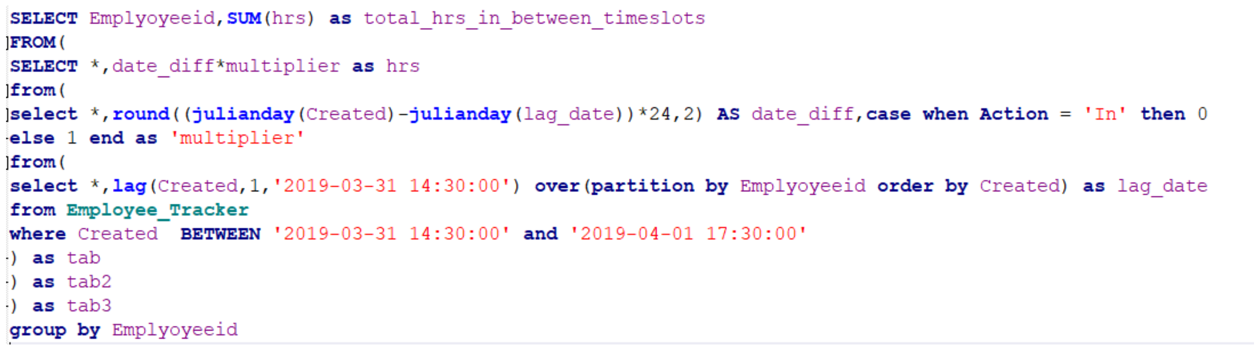
	Emplyoyeeid	total_hrs
1	1	21.0
2	2	21.0
3	3	16.0

Q4) Measure the amount of hours spent by each employee inside the office between “2019-04-01 14:00:00” and “2019-04-02 10:00:00”.

Methodology:

1) The data is filtered between the given time slots(using where and between).
2) A lag window function is used instead of a general version of lag function,a function with the starting time slot is passed to capture the initial time spent(time between time Action captured and the initial time).
3) In order to find the time spent after the final Action captured and the ending time slot,a new table is queried which can join using CTE.

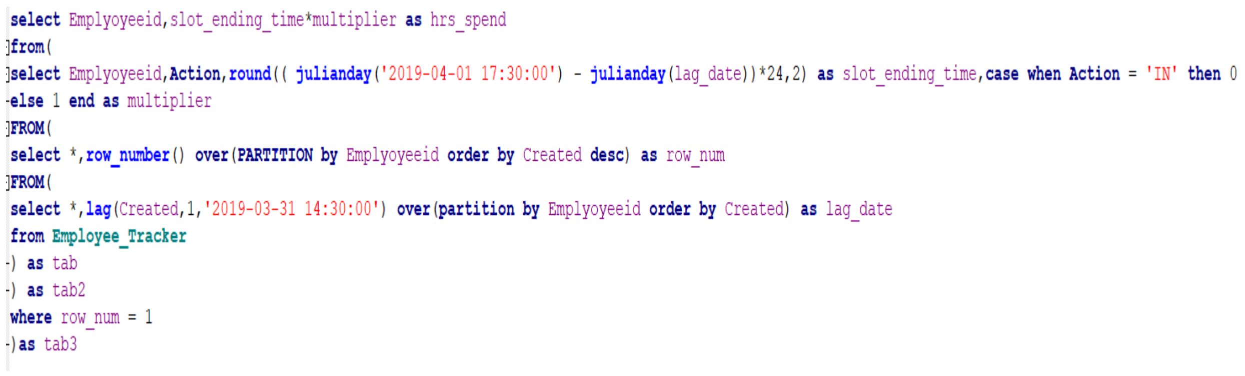
Query:



```
SELECT Emplyoyeeid,SUM(hrs) as total_hrs_in_between_timeslots
FROM(
SELECT *,date_diff*multiplier as hrs
from(
select *,round((julianday(Created)-julianday(lag_date))*24,2) AS date_diff,case when Action = 'In' then 0
else 1 end as 'multiplier'
from(
select *,lag(Created,1,'2019-03-31 14:30:00') over(partition by Emplyoyeeid order by Created) as lag_date
from Employee_Tracker
where Created  BETWEEN '2019-03-31 14:30:00' and '2019-04-01 17:30:00'
) as tab
) as tab2
) as tab3
group by Emplyoyeeid
```

MEESHO BUSINESS ANALYST ASSESSMENT

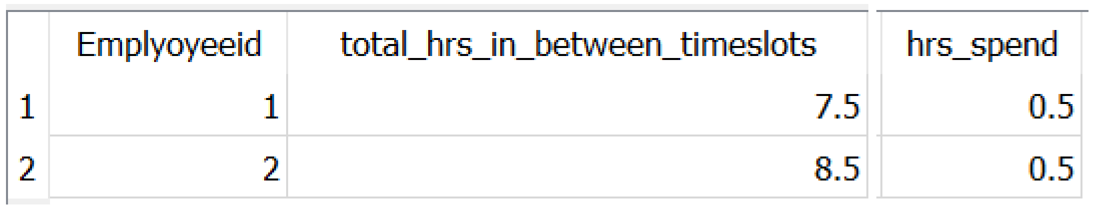
Ashwath J



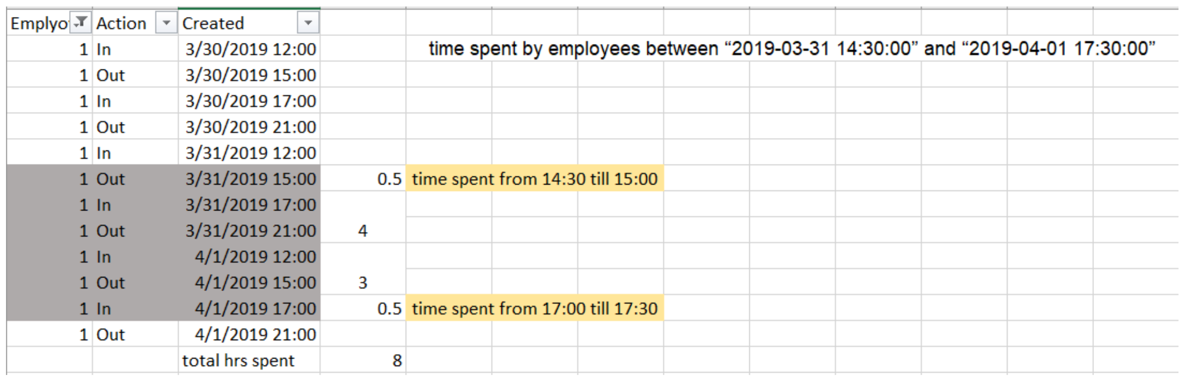
```
select Emplyoyeeid,slot_ending_time*multiplier as hrs_spend
from(
select Emplyoyeeid,Action,round(( julianday('2019-04-01 17:30:00') - julianday(lag_date))*24,2) as slot_ending_time,case when Action = 'IN' then 0
else 1 end as multiplier
FROM(
select *,row_number() over(PARTITION by Emplyoyeeid order by Created desc) as row_num
FROM(
select *,lag(Created,1,'2019-03-31 14:30:00') over(partition by Emplyoyeeid order by Created) as lag_date
from Employee_Tracker
) as tab
) as tab2
where row_num = 1
)as tab3
```

.

Result:



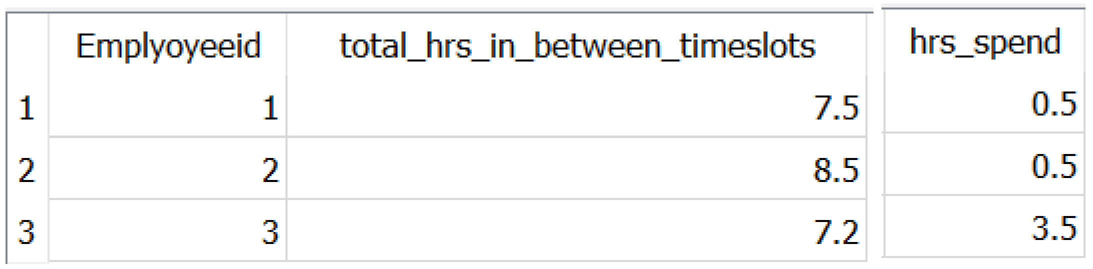
	Emplyoyeeid	total_hrs_in_between_timeslots	hrs_spend
1	1	7.5	0.5
2	2	8.5	0.5



Emplyo	Action	Created		
1	In	3/30/2019 12:00		time spent by employees between "2019-03-31 14:30:00" and "2019-04-01 17:30:00"
1	Out	3/30/2019 15:00		
1	In	3/30/2019 17:00		
1	Out	3/30/2019 21:00		
1	In	3/31/2019 12:00		
1	Out	3/31/2019 15:00	0.5	time spent from 14:30 till 15:00
1	In	3/31/2019 17:00		
1	Out	3/31/2019 21:00	4	
1	In	4/1/2019 12:00		
1	Out	4/1/2019 15:00	3	
1	In	4/1/2019 17:00	0.5	time spent from 17:00 till 17:30
1	Out	4/1/2019 21:00		
		total hrs spent	8	

Query result is equal to the excel answer.

Verifying the query:(test data set with 3 employees)



	Emplyoyeeid	total_hrs_in_between_timeslots	hrs_spend
1	1	7.5	0.5
2	2	8.5	0.5
3	3	7.2	3.5

MEESHO BUSINESS ANALYST ASSESSMENT

Ashwath J

2) RANKER SYSTEM

A company wants to build a recommendation system for its users, they plan to show 50 products to the customers which they are most likely to buy (highest score) . To improve diversity they want to keep products from all categories 5, in the system

Given table with columns user_id, catalog_id, score, category

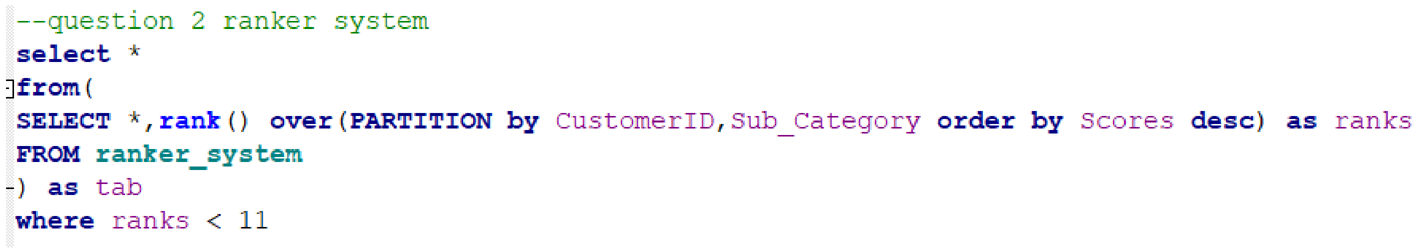
Write a query that gives 50 catalogs, top 10 for each category for each user.

Assumption: Since the raw data was not present I have assumed the data and came up with the query.

Methodology:

1) **A calculated field is created using the 'RANK' window function.**
2) **In order to get category wise top 10 products for individuals, I am partitioning the data on both CustomerIDs and Subcategory level and then ranking them from highest scores to the least scores.**
3) **Finally the data is filtered using the calculated field (where cal field <= 10).**

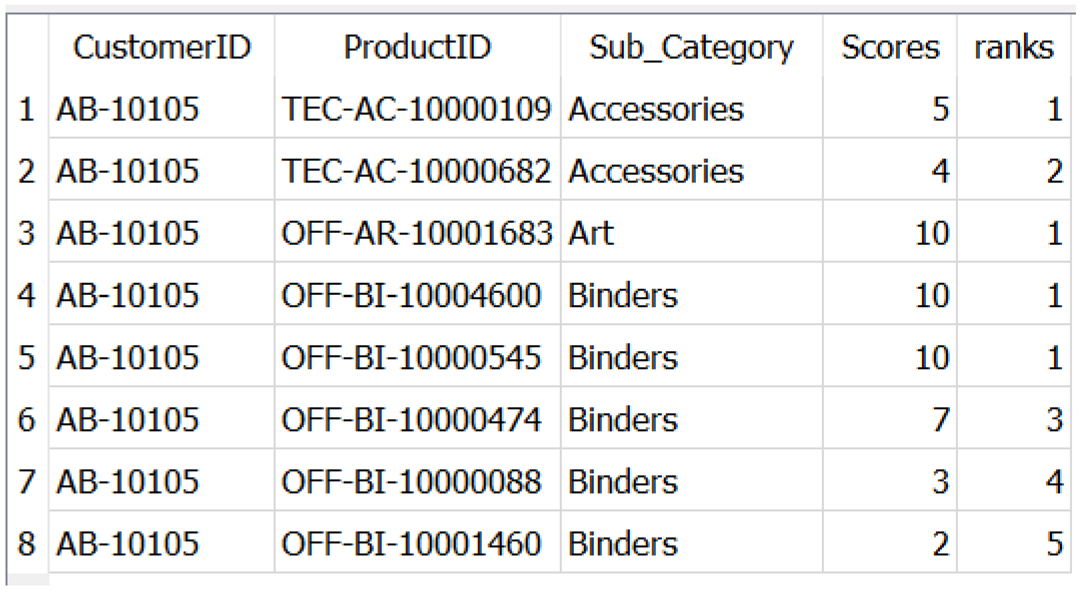
Query:



```
--question 2 ranker system
select *
from(
SELECT *,rank() over(PARTITION by CustomerID,Sub_Category order by Scores desc) as ranks
FROM ranker_system
) as tab
where ranks < 11
```

Output of the query:

We get product ids based on their highest rating and to maintain diversity we consider all the subcategories.



	CustomerID	ProductID	Sub_Category	Scores	ranks
1	AB-10105	TEC-AC-10000109	Accessories	5	1
2	AB-10105	TEC-AC-10000682	Accessories	4	2
3	AB-10105	OFF-AR-10001683	Art	10	1
4	AB-10105	OFF-BI-10004600	Binders	10	1
5	AB-10105	OFF-BI-10000545	Binders	10	1
6	AB-10105	OFF-BI-10000474	Binders	7	3
7	AB-10105	OFF-BI-10000088	Binders	3	4
8	AB-10105	OFF-BI-10001460	Binders	2	5

MEESHO Business Analyst Assessment

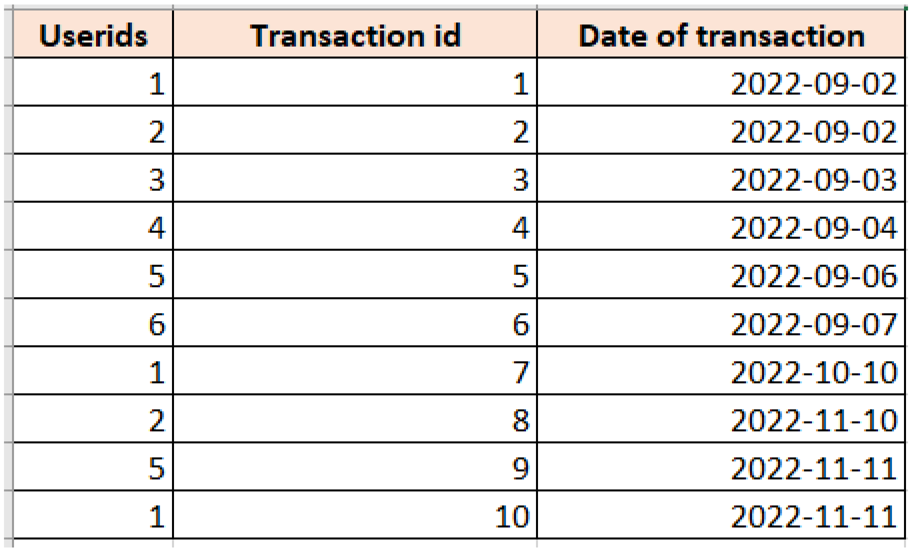
Ashwath J

3)<u>Retention Cohort</u>

A company have the transaction data for its ecommerce site stored in a table which contains columns/fields (users_id ,transaction_id, timestamp)

For users acquired in a given week (first transaction for that user was in that week), week starting 29 st Sep 2022. Find the D+7, D+30 and D+60 retained transactors.
D+i → Retained transactors are users who transacted at least i days after there onboarding date
So for eg. if a user was on boarded on 31st Nov 2022 and did his 2nd transaction and last transaction on 1st Jan, that seller will be D+7 and D+30 retained but not D+60 retained

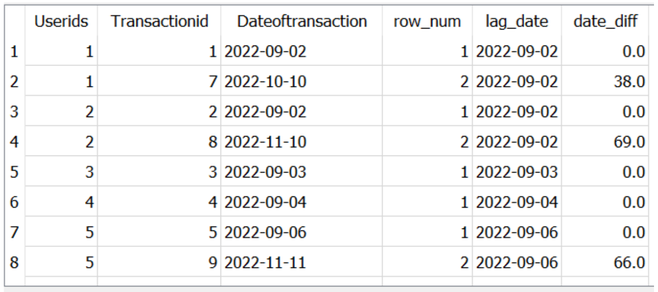
Write a query to get the desired output
RAW DATA: (Since raw data wasn't present the data is assumed)



Userids	Transaction id	Date of transaction
1	1	2022-09-02
2	2	2022-09-02
3	3	2022-09-03
4	4	2022-09-04
5	5	2022-09-06
6	6	2022-09-07
1	7	2022-10-10
2	8	2022-11-10
5	9	2022-11-11
1	10	2022-11-11

Methodology:

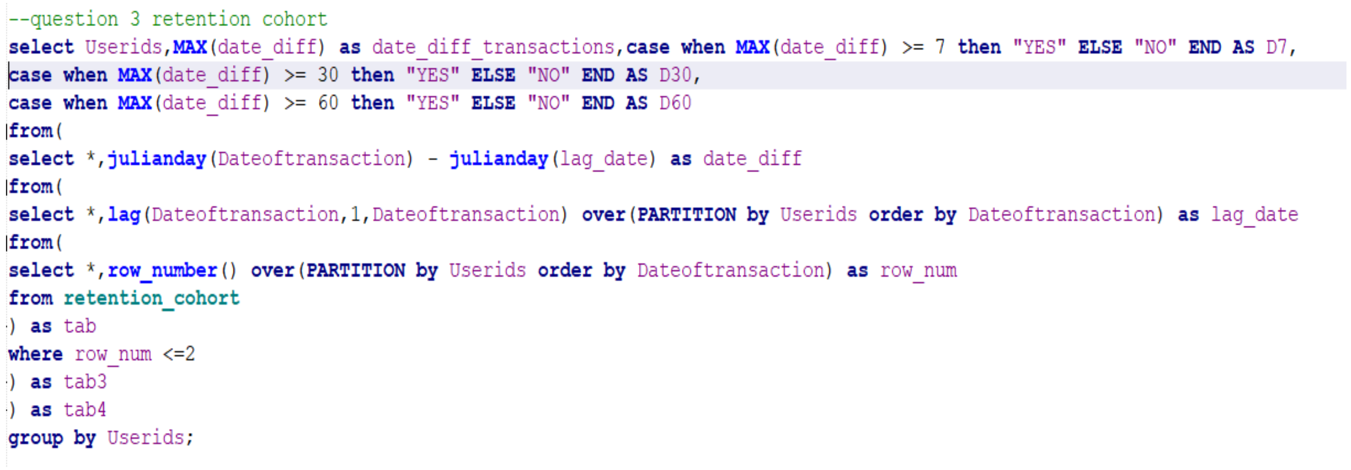
1) **A calculated field is created using row_number windows functions, the window function is partitioned using UserIDs and ordered using Date Of Transaction.**
2) **Since we need the 1st two transactions to decide the transactor status I filtered the data using the above calculated field (<=2).**
3) **A lag windows function is used to get the previous transaction date.**



	Userids	Transactionid	Dateoftransaction	row_num	lag_date	date_diff
1	1	1	2022-09-02	1	2022-09-02	0.0
2	1	7	2022-10-10	2	2022-09-02	38.0
3	2	2	2022-09-02	1	2022-09-02	0.0
4	2	8	2022-11-10	2	2022-09-02	69.0
5	3	3	2022-09-03	1	2022-09-03	0.0
6	4	4	2022-09-04	1	2022-09-04	0.0
7	5	5	2022-09-06	1	2022-09-06	0.0
8	5	9	2022-11-11	2	2022-09-06	66.0

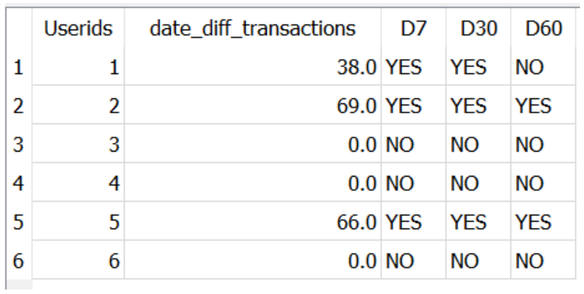
4) **Further the date difference is calculated using the Julianday function.**
5) **Finally case statements are used to categorize the date difference into required columns.**

Query:



```
--question 3 retention cohort
select Userids,MAX(date_diff) as date_diff_transactions,case when MAX(date_diff) >= 7 then "YES" ELSE "NO" END AS D7,
case when MAX(date_diff) >= 30 then "YES" ELSE "NO" END AS D30,
case when MAX(date_diff) >= 60 then "YES" ELSE "NO" END AS D60
from(
select *,julianday(Dateoftransaction) - julianday(lag_date) as date_diff
from(
select *,lag(Dateoftransaction,1,Dateoftransaction) over(PARTITION by Userids order by Dateoftransaction) as lag_date
from(
select *,row_number() over(PARTITION by Userids order by Dateoftransaction) as row_num
from retention_cohort
) as tab
where row_num <=2
) as tab3
) as tab4
group by Userids;
```

Output:



	Userids	date_diff_transactions	D7	D30	D60
1	1	38.0	YES	YES	NO
2	2	69.0	YES	YES	YES
3	3	0.0	NO	NO	NO
4	4	0.0	NO	NO	NO
5	5	66.0	YES	YES	YES
6	6	0.0	NO	NO	NO