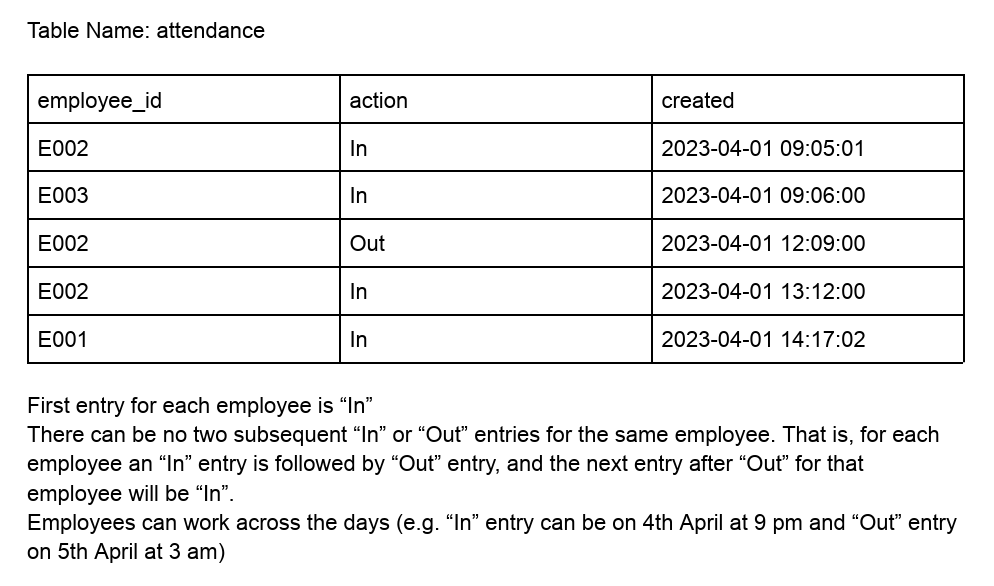
SQL

Q1

In an HR Management Software, employees going into and out of office are recorded in the following format (The rows data given is only for illustration purposes. Assume that this kind of data is there in the table since the **beginning of the company for all employees. So, it may run into years.)**

**Given this table, write SQL queries to find the**

**● number of employees inside the Office at current time**

**DDL Command for the above table:**

create table attendance (

employee\_id varchar(20),

action\_ varchar(10),

created datetime);

insert into attendance

values

("E002", "In", "2023-04-01 09:05:01"),

("E003", "In", "2023-04-01 09:06:00"),

("E002", "Out", "2023-04-01 12:09:00"),

("E002", "In", "2023-04-01 13:12:00"),

("E001", "In", "2023-04-01 14:17:02");

**DML Query:**

**1st Method**

with cte

as

(select \*,

lead(action\_)over(partition by employee\_id order by created) as subsequent\_action

from attendance

order by employee\_id, created)

select sum(case when

subsequent\_action = "Out" and action\_ = "In" then -1 else 1 end) as present\_employee

from cte;



**Explanation:** To answer this question I have used a common table expression(cte) along with a short query to extract information from the cte. First in order to arrange the action column to get subsequent “In” and “Out”, I arranged the employee\_id column in ascending order and did secondary sort on created column. Then used the lead function to arrange the “Out” action for every “In” action. Used this query as CTE.

By using the 1st method, I have built a logic using the case when statement that if subsequent\_action is “Out” and action is “In” then give a value of -1 and otherwise give a value of 1. Thus if a employee leaves and checks “Out” then the value would turn -1 and if he/she doesn’t leave the value will be 1. Thus if we add up this newly created column, we will get the head count of the employees present.

**2nd Method:**

with cte as

(select \*,

lead(action\_)over(partition by employee\_id order by created) as subsequent\_action,

lead(created)over(partition by employee\_id order by created) as end\_time

from attendance

order by employee\_id, created)

select count(distinct employee\_id) as present\_employee

from

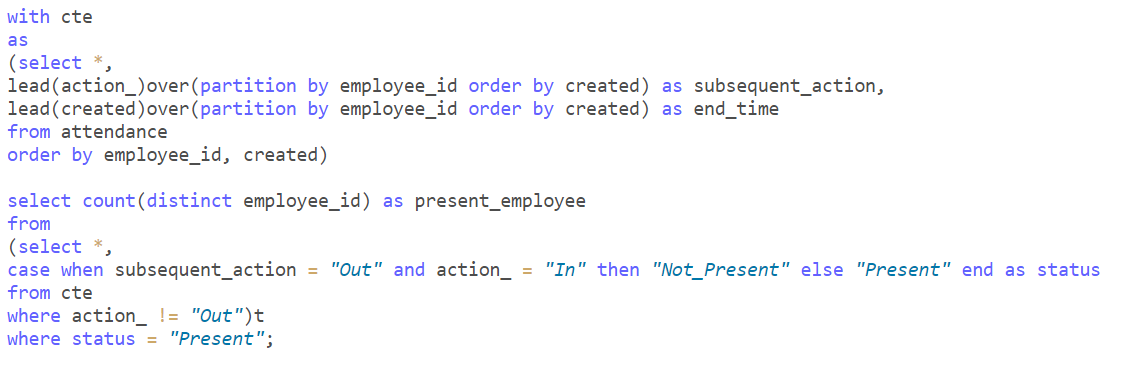
(select \*,

case when subsequent\_action = "Out" and action\_ = "In" then "Not\_Present" else "Present" end as status

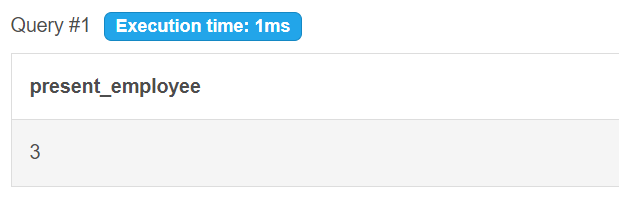
from cte

where action\_ != "Out")t

where status = "Present"

****

**Query ->**



**Result ->**

**Explanation:** Here, the first CTE remains the same. In the second part, I have eliminated the “Out” action from the action column as I have already used lead to populate “Out” in subsequent\_action column. Then I have created a column “status” using case when and gave the logic that if subsequent\_action is “Out” and action\_ is “In” then the employee is “Not Present” else he/she is “Present”. Then used a subquery to count the distinct employee\_id where the status is “Present”.

**● number of employees inside the Office at “2023-06-15 19:05:00”**

**Note\*:** Here for the given dataset, the date column doesn’t match with the given date criteria, so I have added few dummy data rows so that they fall under the given criteria

**DDL:**

create table attendance (

employee\_id varchar(20),

action\_ varchar(10),

created datetime);

insert into attendance

values

("E002", "In", "2023-04-01 09:05:01"),

("E003", "In", "2023-04-01 09:06:00"),

("E002", "Out", "2023-04-01 12:09:00"),

("E005", "In", "2023-04-01 13:12:00"),

("E001", "In", "2023-04-01 14:17:02"),

("E005", "Out", "2023-04-01 18:30:00"),

("E001", "Out", "2023-04-01 21:30:00"),

("E006", "In", "2023-04-01 21:30:00"),

("E007", "In", "2023-04-01 07:30:00"),

("E002", "In", "2023-04-01 23:05:01"),

("E002", "Out", "2023-04-02 03:05:01"),

("E001", "In", "2023-05-01 10:05:01"),

("E007", "Out", "2023-04-01 15:30:00"),

("E001", "Out", "2023-05-01 20:05:01"),

("E007", "In", "2023-05-01 11:30:00"),

("E007", "Out", "2023-05-01 17:30:00"),

("E005", "In", "2023-06-15 03:30:00"),

("E007", "In", "2023-06-15 13:30:00"),

("E007", "Out", "2023-06-15 20:00:00"),

("E008", "In", "2023-06-16 17:30:00"),

("E009", "In", "2023-06-15 12:00:00"),

("E009", "Out", "2023-06-15 20:30:00");

**DML:**

with cte as

(select \*,

lead(action\_)over(partition by employee\_id order by created) as subsequent\_action,

lead(created)over(partition by employee\_id order by created) as end\_time

from attendance

order by employee\_id, created)

select count(distinct employee\_id) as no\_of\_employees\_inside\_office

from

(select \*,

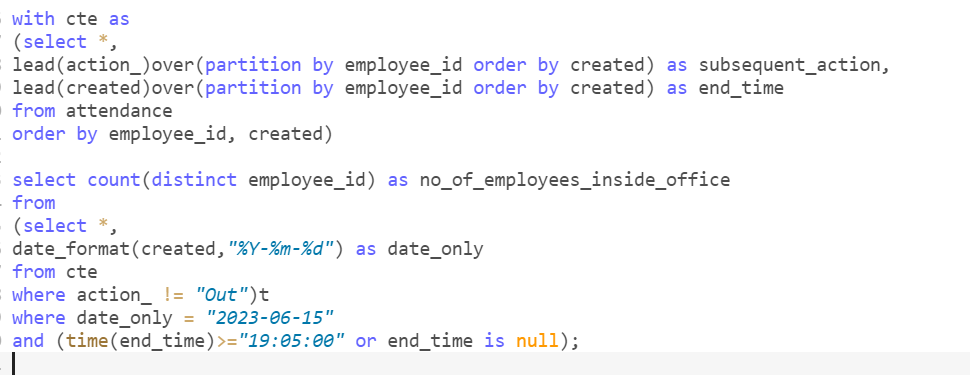
date\_format(created,"%Y-%m-%d") as date\_only

from cte

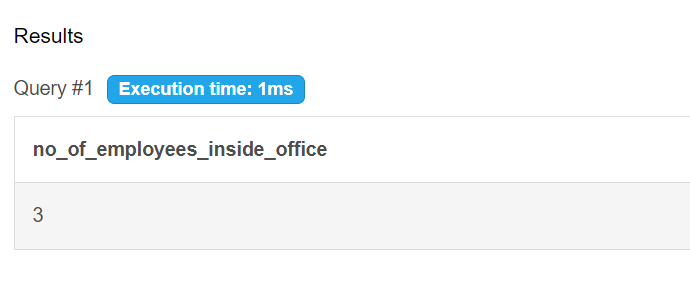
where action\_ != "Out")t

where date\_only = "2023-06-15"

and (time(end\_time)>="19:05:00" or end\_time is null);



**Query ->**



**Result ->**

**Explanation:** I have order the main table by employee\_id so that the action column gets sorted correctly in terms of “In” and “Out”. Used the lead function on both action\_ and created column, so that for every employee their next subsequent column values comes up. In the later query part I have filtered on the action\_ column to not include “Out” as it it already included in the subsequent\_action then from this table I have used a subquery to filter out the day given **“2023-06-15”** then again filterted on the end\_time with the logic that (if the end\_time is greater than **“19:05:00”** that means that person was in the office at the given time) or (end\_time is null on that day – that means that person hasn’t logged out yet on that day, so he/she will still be in office)

**● number of hours spent by each employee inside the office since the day they started**

**\*\* I have used the given dataset for this**

**DML:**

with cte as

(select \*,

lead(action\_)over(partition by employee\_id order by created) as subsequent\_action,

lead(created)over(partition by employee\_id order by created) as end\_time

from attendance

order by employee\_id, created)

select employee\_id,

date\_format(created, "%W") as start\_day,

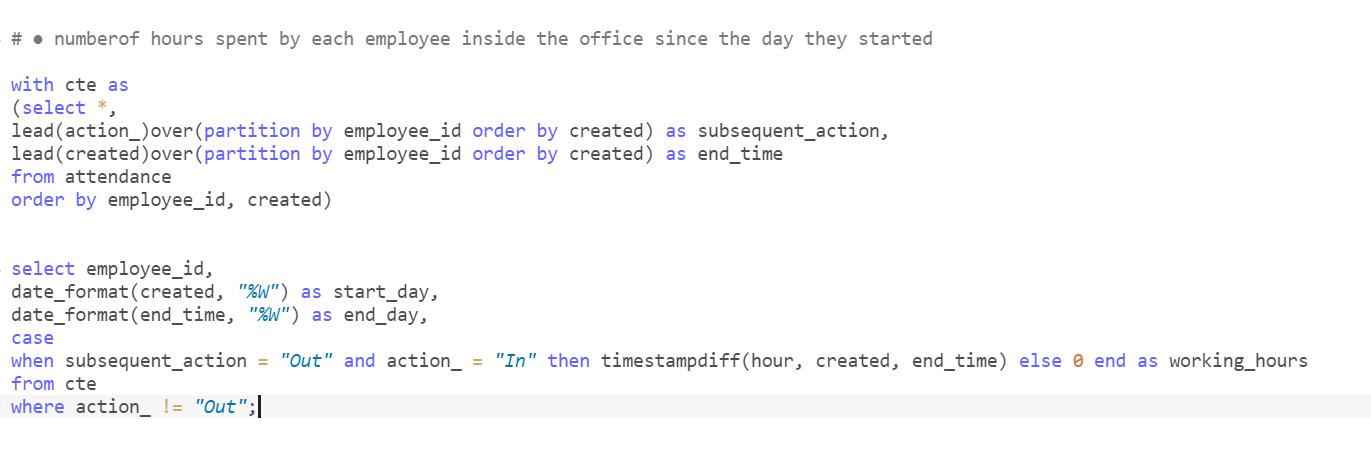
date\_format(end\_time, "%W") as end\_day,

case

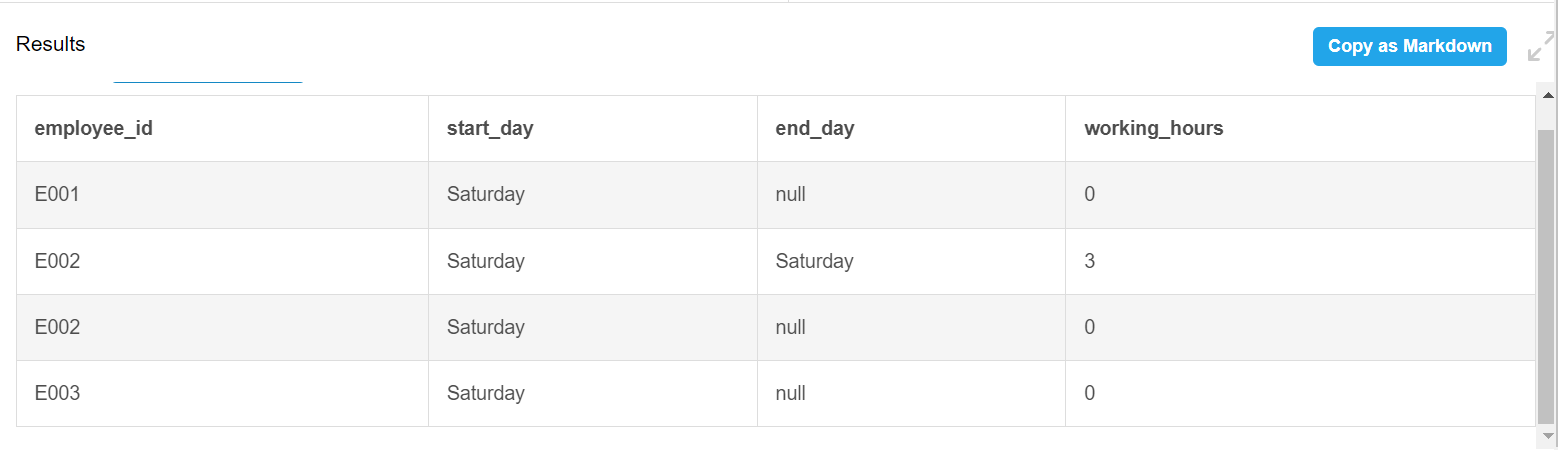
when subsequent\_action = "Out" and action\_ = "In" then timestampdiff(hour, created, end\_time) else 0 end as working\_hours

from cte

where action\_ != "Out";



**Query ->**



**Result ->**

**Explanation:** I have order the main table by employee\_id so that the action column gets sorted correctly in terms of “In” and “Out”. Used the lead function on both action\_ and created column, so that for every employee their next subsequent column values comes up. In the second query, I have used the date\_format function to name the weekday of both the created and end\_day, so that if employees have records of different days in the attendance sheet then it will be easier to see that. Then I have used a case statement that if the subsequent\_action is “Out” and action is “In” then find the timestampdiff between created and end\_time as working\_hours else put 0.

**● number of hours spent by each employee inside the office between “2023-05-01 09:00:00” and “2023-05-02 00:00:00”**

**Note\*:** Here for the given dataset, the date column doesn’t match with the given date criteria, so I have added few dummy data rows so that they fall under the given criteria

**DDL:**

create table attendance (

employee\_id varchar(20),

action\_ varchar(10),

created datetime);

insert into attendance

values

("E002", "In", "2023-04-01 09:05:01"),

("E003", "In", "2023-04-01 09:06:00"),

("E002", "Out", "2023-04-01 12:09:00"),

("E005", "In", "2023-04-01 13:12:00"),

("E001", "In", "2023-04-01 14:17:02"),

("E005", "Out", "2023-04-01 18:30:00"),

("E001", "Out", "2023-04-01 21:30:00"),

("E006", "In", "2023-04-01 21:30:00"),

("E007", "In", "2023-04-01 07:30:00"),

("E002", "In", "2023-04-01 23:05:01"),

("E002", "Out", "2023-04-02 03:05:01"),

("E001", "In", "2023-05-01 10:05:01"),

("E007", "Out", "2023-04-01 15:30:00"),

("E001", "Out", "2023-05-01 20:05:01"),

("E007", "In", "2023-05-01 11:30:00"),

("E007", "Out", "2023-05-01 17:30:00")

;

**DML:**

with cte as

(select \*,

lead(action\_)over(partition by employee\_id order by created) as subsequent\_action,

lead(created)over(partition by employee\_id order by created) as end\_time

from attendance

order by employee\_id, created)

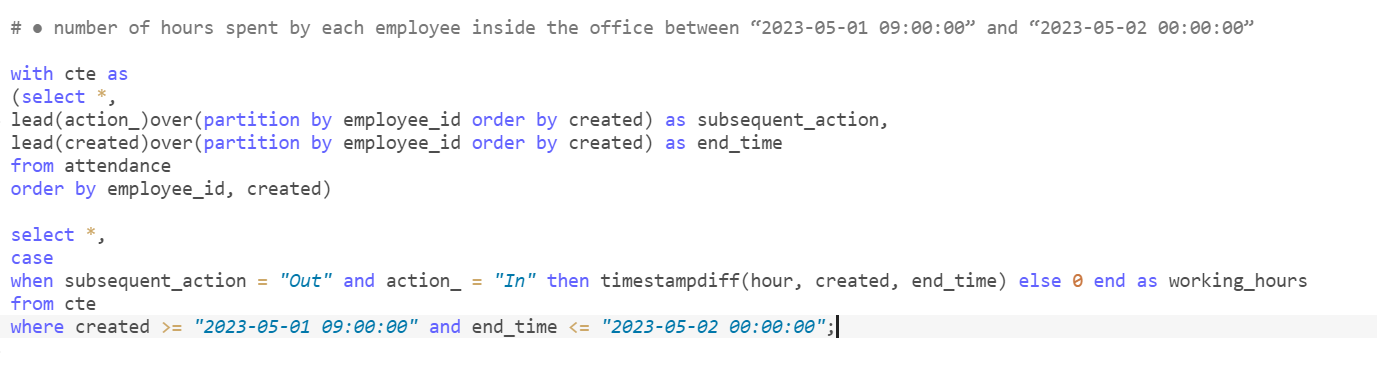
select \*,

case

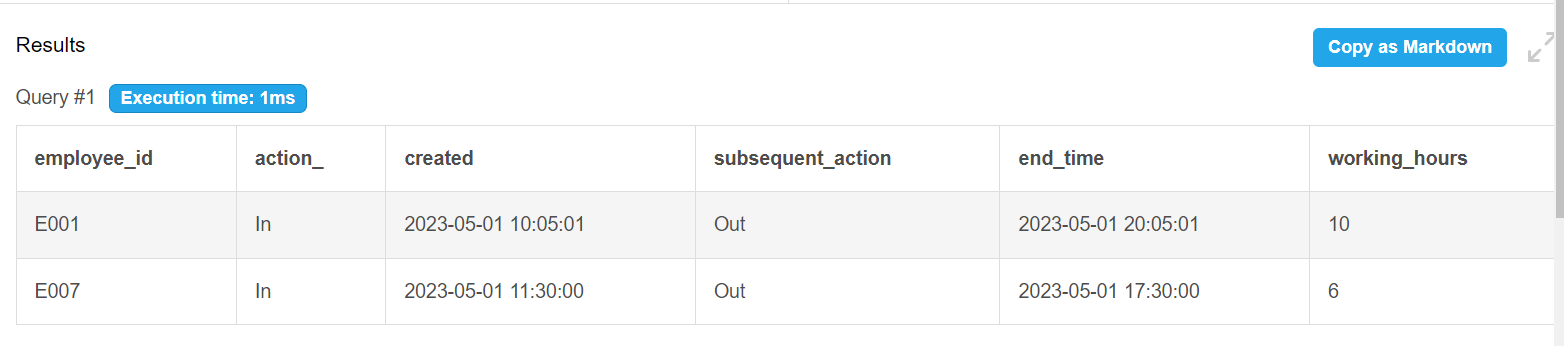
when subsequent\_action = "Out" and action\_ = "In" then timestampdiff(hour, created, end\_time) else 0 end as working\_hours

from cte

where created >= "2023-05-01 09:00:00" and end\_time <= "2023-05-02 00:00:00";



**Query ->**

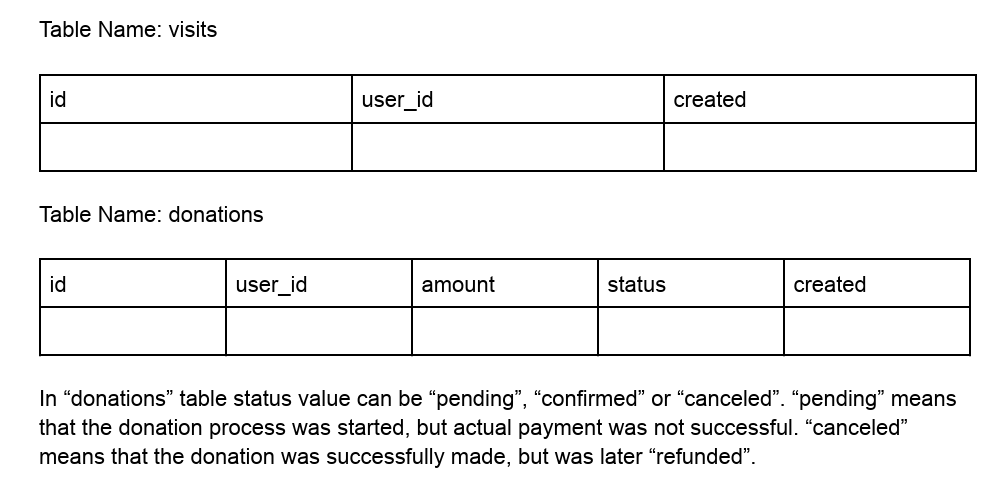


**Result ->**

**Explanation:** I have order the main table by employee\_id so that the action column gets sorted correctly in terms of “In” and “Out”. Used the lead function on both action\_ and created column, so that for every employee their next subsequent column values comes up. In the second part of the query, I have filtered on the end\_time and created as mentioned in the question. Then I have used a case statement that if the subsequent\_action is “Out” and action is “In” then find the timestampdiff between created and end\_time as working\_hours else put 0.

Q2

Consider the following two tables



**Write SQL queries to find**

1. **The average number of visits before the 1st confirmed donation**

\*Note: I have created a dummy dataset to work on this problem and to validate and optimize the query.

**DDL:**

CREATE TABLE visits (

id INT PRIMARY KEY,

user\_id INT,

created DATE

);

INSERT INTO visits (id, user\_id, created) VALUES

(1, 101, '2022-01-01'),

(2, 102, '2022-01-02'),

(3, 103, '2022-01-03'),

(4, 101, '2022-01-04'),

(5, 104, '2022-01-05'),

(6, 105, '2022-01-06'),

(7, 102, '2022-01-07'),

(8, 106, '2022-01-08'),

(9, 107, '2022-01-09'),

(10, 108, '2022-01-10'),

(11, 109, '2022-01-11'),

(12, 110, '2022-01-12'),

(13, 111, '2022-01-13'),

(14, 112, '2022-01-14'),

(15, 113, '2022-01-15'),

(16, 114, '2022-01-16');

-- Create the "donations" table

CREATE TABLE donations (

id INT PRIMARY KEY,

user\_id INT,

amount INT,

status VARCHAR(10),

created DATE

);

INSERT INTO donations (id, user\_id, amount, status, created) VALUES

(1, 101, 500, 'canceled', '2022-01-01'),

(2, 102, 1000, 'pending', '2022-01-02'),

(3, 103, 250, 'canceled', '2022-01-03'),

(4, 101, 750, 'pending', '2022-01-04'),

(5, 104, 1200, 'pending', '2022-01-05'),

(6, 105, 300, 'canceled', '2022-01-06'),

(7, 102, 800, 'canceled', '2022-01-07'),

(8, 106, 600, 'canceled', '2022-01-08'),

(9, 107, 1100, 'pending', '2022-01-09'),

(10, 108, 400, 'confirmed', '2022-01-10'),

(11, 109, 950, 'pending', '2022-01-11'),

(12, 110, 550, 'confirmed', '2022-01-12'),

(13, 111, 200, 'canceled', '2022-01-13'),

(14, 112, 800, 'pending', '2022-01-14'),

(15, 113, 300, 'confirmed', '2022-01-15'),

(16, 114, 700, 'canceled', '2022-01-16');

**DML:**

with cte as

(select \*

from

(select v.id as visit\_id, v.user\_id as visit\_userid,

v.created as visit\_created , d.\*,

case when status="confirmed" then 1 else 0 end as stat\_chk

from visits v

left join

donations d

on v.user\_id = d.user\_id

and v.created = d.created)t

where visit\_created <= created)

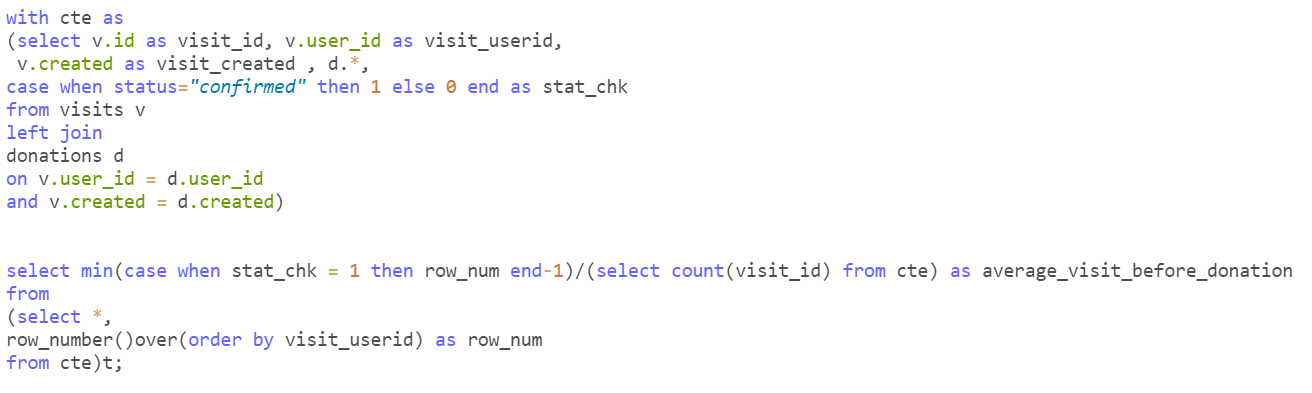
select (min(case when stat\_chk = 1 then row\_num end)-1)/(select count(visit\_id) from cte) as average\_visit\_before\_donation

from

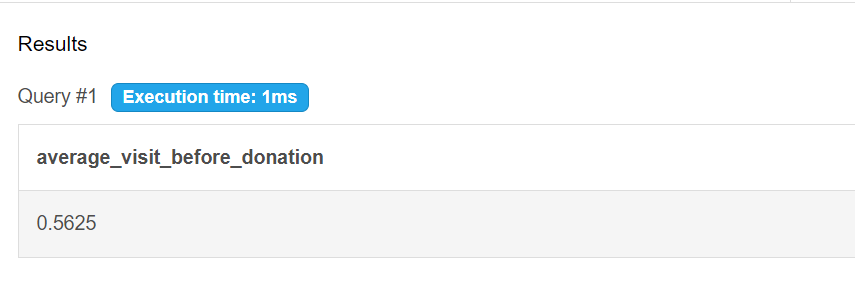
(select \*,

row\_number()over(order by visit\_userid) as row\_num

from cte)t;



**QUERY ->**



**RESULT ->**

**Explantion:** I have used a dummy dataset to deal with this problem and test my code for various scenarios. You can check the DDL command and paste it to run the same. At first I have left joined the visits table with the donations table to include all the visits even where there were no donations, I have joined the two tables on user\_id and date\_created. I have used a case statement in the same table to add a column “stat\_chk”, given the condition that when the donation is confirmed then the result will be 1 else 0. Used this whole table as a common table expression(cte). After this I have used a query to get info from the above cte.

In this query first used row number and ordered it by user\_id, then from it written a subquery and build a logic using case statement. The logic is when the when the stat\_chk becomes 1 then return the corresponding row\_number, but it will give the current row num, in order to get the number of visits before first confirmed visit, I need to deduct the row\_number by 1. This column will give all the row\_num where the stat\_chk is 1 in ascending order. To get the first confirmed donation, I have used min() to this case statement -1:

min(case when stat\_chk = 1 then row\_num end-1), this gives the number of visits before first confirmed visit. Then I have divided it by total number of visits. The full logic is :

**min(case when stat\_chk = 1 then row\_num end-1)/(select count(visit\_id) from cte) as average\_visit\_before\_donation.** This gives the average visits before first confirmed donation.

**2nd Problem in the next page**

1. **The median number of visits before the 4th confirmed donation**

**DDL: I have added a record in both the tables so that the number of visits become even, so that I can check my code, whether or not it is working properly for the custom formula I have created for even number of visists. Please paste this DDL to check the code:**

CREATE TABLE donations (

id INT PRIMARY KEY,

user\_id INT,

amount INT,

status VARCHAR(10),

created DATE);

INSERT INTO donations (id, user\_id, amount, status, created) VALUES

(0, 100, 500, 'canceled', '2022-01-01'),

(1, 101, 500, 'canceled', '2022-01-01'),

(2, 102, 1000, 'pending', '2022-01-02'),

(3, 103, 250, 'canceled', '2022-01-03'),

(4, 101, 750, 'pending', '2022-01-04'),

(5, 104, 1200, 'pending', '2022-01-05'),

(6, 105, 300, 'canceled', '2022-01-06'),

(7, 102, 800, 'canceled', '2022-01-07'),

(8, 106, 600, 'canceled', '2022-01-08'),

(9, 107, 1100, 'pending', '2022-01-09'),

(10, 108, 400, 'confirmed', '2022-01-10'),

(11, 109, 950, 'pending', '2022-01-11'),

(12, 110, 550, 'confirmed', '2022-01-12'),

(13, 111, 200, 'canceled', '2022-01-13'),

(14, 112, 800, 'pending', '2022-01-14'),

(15, 113, 300, 'confirmed', '2022-01-15'),

(16, 114, 700, 'canceled', '2022-01-16'),

(17, 112, 850, 'confirmed', '2022-01-18');

CREATE TABLE visits (

id INT PRIMARY KEY,

user\_id INT,

created DATE);

INSERT INTO visits (id, user\_id, created) VALUES

(0, 100, '2022-01-01'),

(1, 101, '2022-01-01'),

(2, 102, '2022-01-02'),

(3, 103, '2022-01-03'),

(4, 101, '2022-01-04'),

(5, 104, '2022-01-05'),

(6, 105, '2022-01-06'),

(7, 102, '2022-01-07'),

(8, 106, '2022-01-08'),

(9, 107, '2022-01-09'),

(10, 108, '2022-01-10'),

(11, 109, '2022-01-11'),

(12, 110, '2022-01-12'),

(13, 111, '2022-01-13'),

(14, 112, '2022-01-14'),

(15, 113, '2022-01-15'),

(16, 114, '2022-01-16'),

(17, 112, '2022-01-18');

**DML:**

with cte as

(select v.id as visit\_id, v.user\_id as visit\_userid,

v.created as visit\_created , d.\*,

case when status="confirmed" then 1 else 0 end as stat\_chk

from visits v

left join

donations d

on v.user\_id = d.user\_id

and v.created = d.created),

cte2 as

(select \*,

case when stat\_chk = 1 then row\_number()over(order by visit\_userid) -1 end as no\_of\_visits\_before\_confirmed

from cte),

cte3 as

(select no\_of\_visits\_before\_confirmed as visits\_before\_4th\_confirmed\_donation

from

(select no\_of\_visits\_before\_confirmed,

dense\_rank()over(order by no\_of\_visits\_before\_confirmed) as confirmed\_donation\_rank

from cte2

where no\_of\_visits\_before\_confirmed is not null)t

where confirmed\_donation\_rank = 4)

select \*,

case

when

visits\_before\_4th\_confirmed\_donation % 2 != 0 then round((visits\_before\_4th\_confirmed\_donation+1)/2,2)

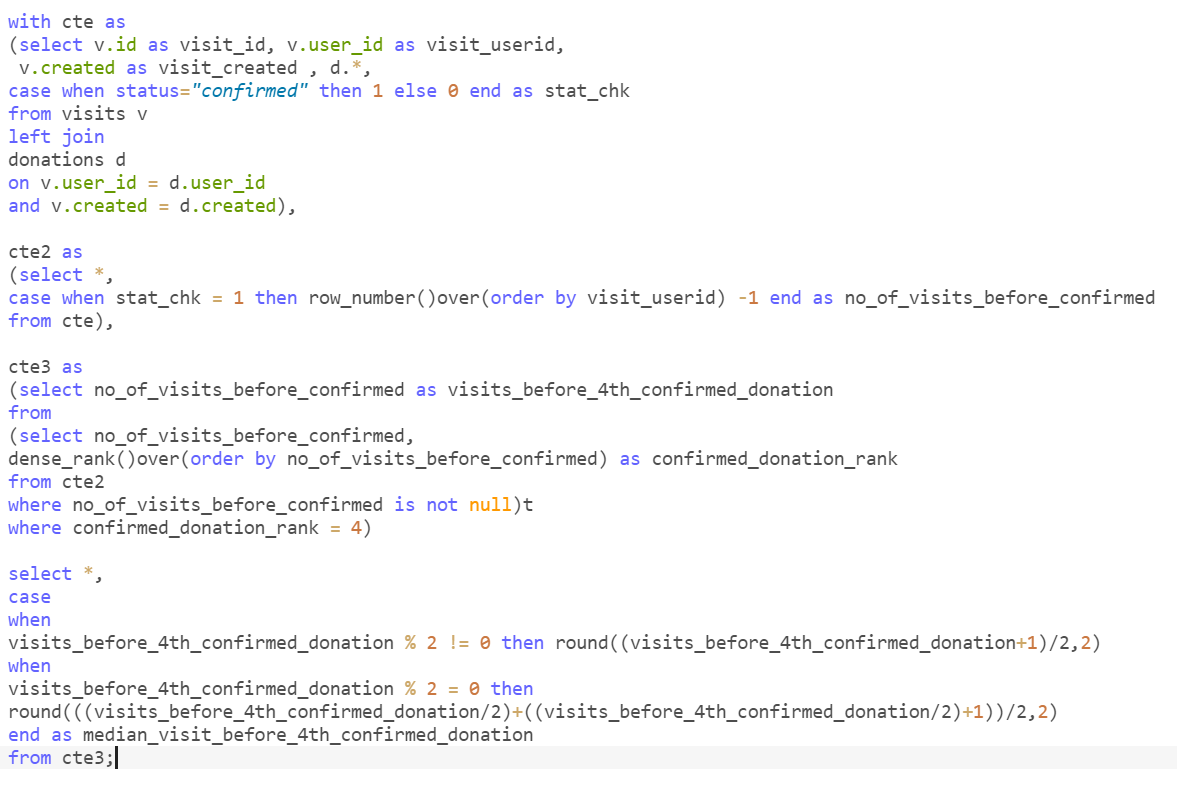
when

visits\_before\_4th\_confirmed\_donation % 2 = 0 then

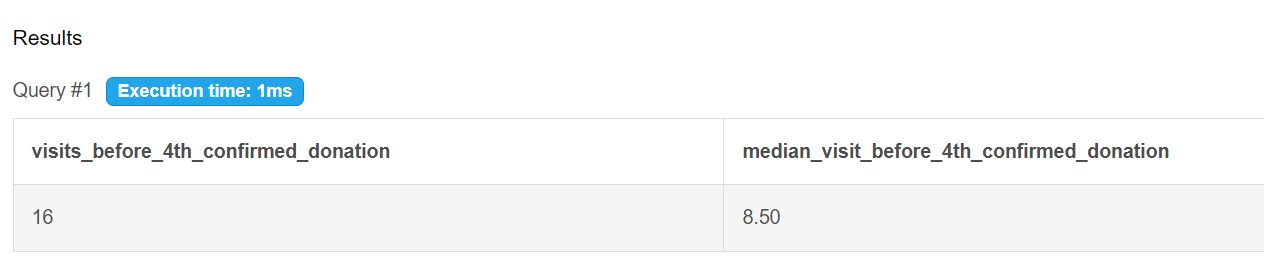
round(((visits\_before\_4th\_confirmed\_donation/2)+((visits\_before\_4th\_confirmed\_donation/2)+1))/2,2)

end as median\_visit\_before\_4th\_confirmed\_donation

from cte3;



**QUERY ->**

**RESULT ->**

**Explanation:** I have used 3 ctes and each one is extracting data from their respective above cte. In the first cte I have left joined the visits table with the donations table to include all the visits even where there were no donations, I have joined the two tables on user\_id and date\_created. I have used a case statement in the same table to add a column “stat\_chk”, given the condition that when the donation is confirmed then the result will be 1 else 0.

In the second cte, cte2. I have used a logic that when the stat\_chk is 1 then return the “row\_number()over(order by visit\_userid) -1”, this will give the no of visits before the confirmed donations.

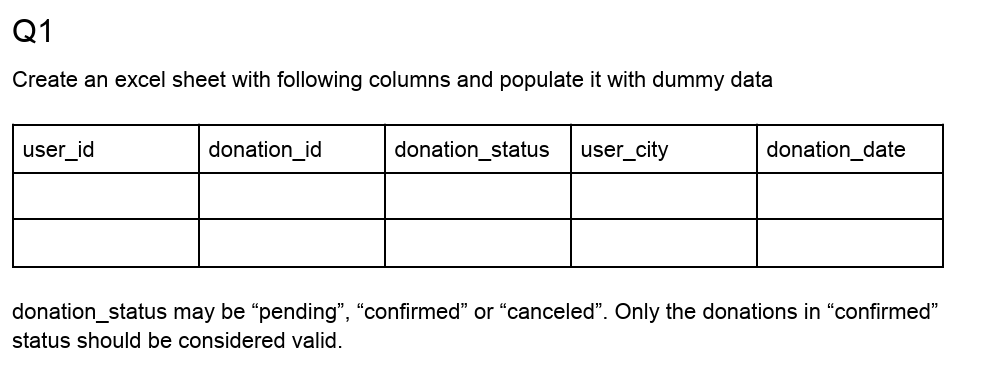
In the third cte, cte3. I have used dense rank to rank the visists before confirmed donations and filtered the data for where no of visists before confirmed donation is not null. Then to this query I have written a subquery to filter the no of visists before confirmed donations where the confirmed donation rank is 4, The confirmed donation rank column was created before when I used dense\_rank().

Now I have got the number of visists before 4th cofirmed donation. I have written a query which checks if this number is odd then it executes the formula (n+1)/2 for median. If the number is even then it executes the formula

Average of these two: (n/2) , (n/2)+1

* ((n/2)) + ((n/2)+1)/2

Now in the result I can get the median no of visits before 4th confirmed donation, both for even no of visists and odd number of visists.

**Excel**

Using excel features, find out:

1. Number of donors by city

2. Average donation received per city in the current calendar month

**I will manully send the excel solution**