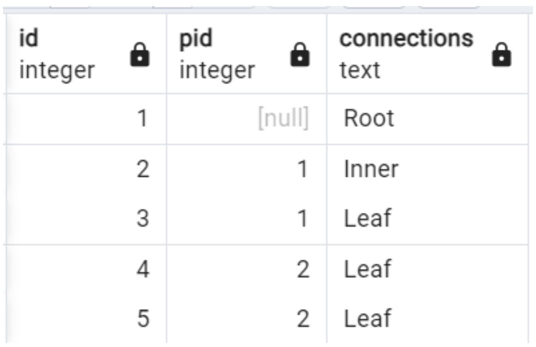
FAANG Questions - Part III

Author:- Azarudhin Usman

Reference:- LeetCode

--Q Find the Leaf, inner and root from the table.

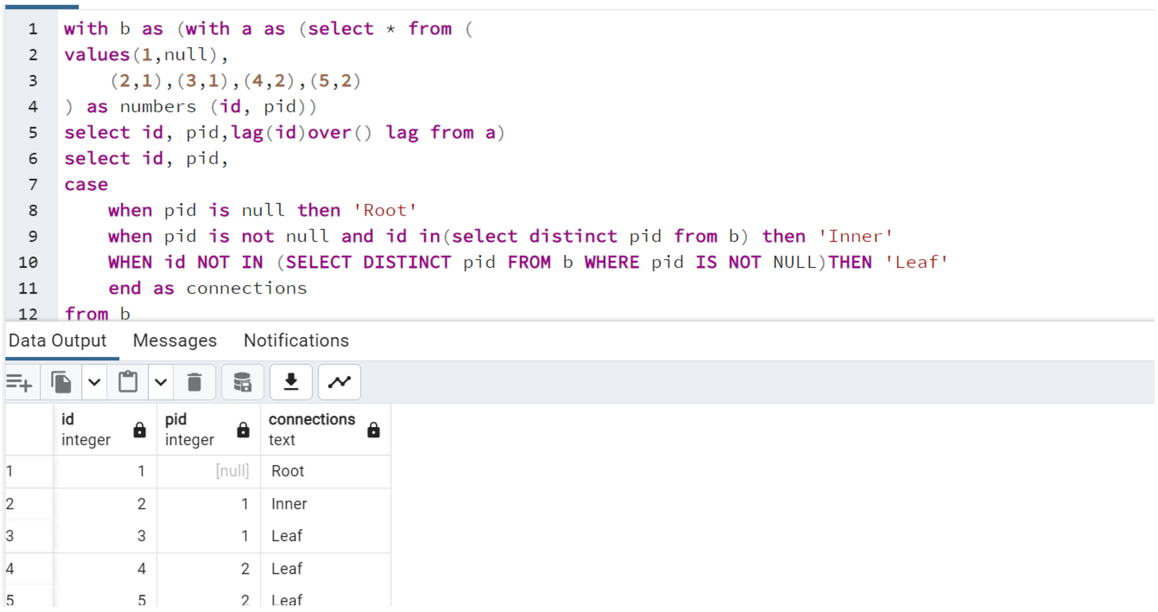
-- Sample output



id integer	pid integer	connections text
1	[null]	Root
2	1	Inner
3	1	Leaf
4	2	Leaf
5	2	Leaf

-- Query

```
with b as (with a as (select * from (
values(1,null),
      (2,1),(3,1),(4,2),(5,2)
) as numbers (id, pid))
select id, pid,lag(id)over() lag from a)
select id, pid,
case
      when pid is null then 'Root'
      when pid is not null and id in(select distinct pid from b) then 'Inner'
      WHEN id NOT IN (SELECT DISTINCT pid FROM b WHERE pid IS NOT NULL)THEN
'Leaf'
      end as connections
from b
```

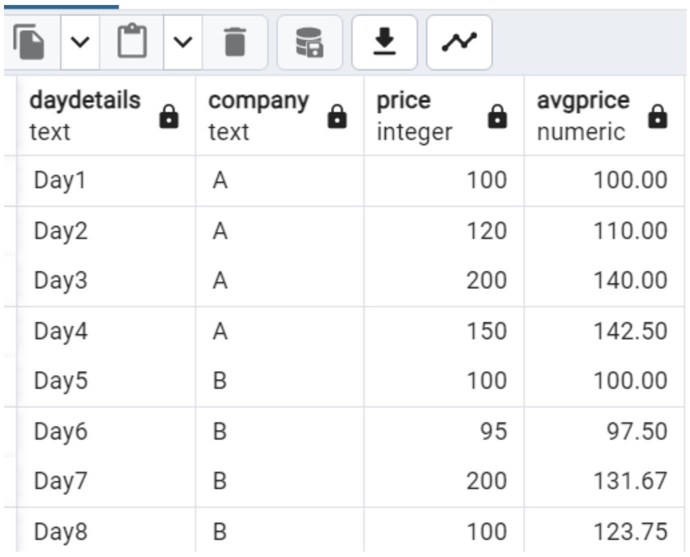


```
with b as (with a as (select * from (
values(1,null),
    (2,1),(3,1),(4,2),(5,2)
) as numbers (id, pid))
select id, pid,lag(id)over() lag from a)
select id, pid,
case
    when pid is null then 'Root'
    when pid is not null and id in(select distinct pid from b) then 'Inner'
    WHEN id NOT IN (SELECT DISTINCT pid FROM b WHERE pid IS NOT NULL)THEN 'Leaf'
    end as connections
from b
```

Data Output   Messages   Notifications

	id integer	pid integer	connections text
1	1	[null]	Root
2	2	1	Inner
3	3	1	Leaf
4	4	2	Leaf
5	5	2	Leaf

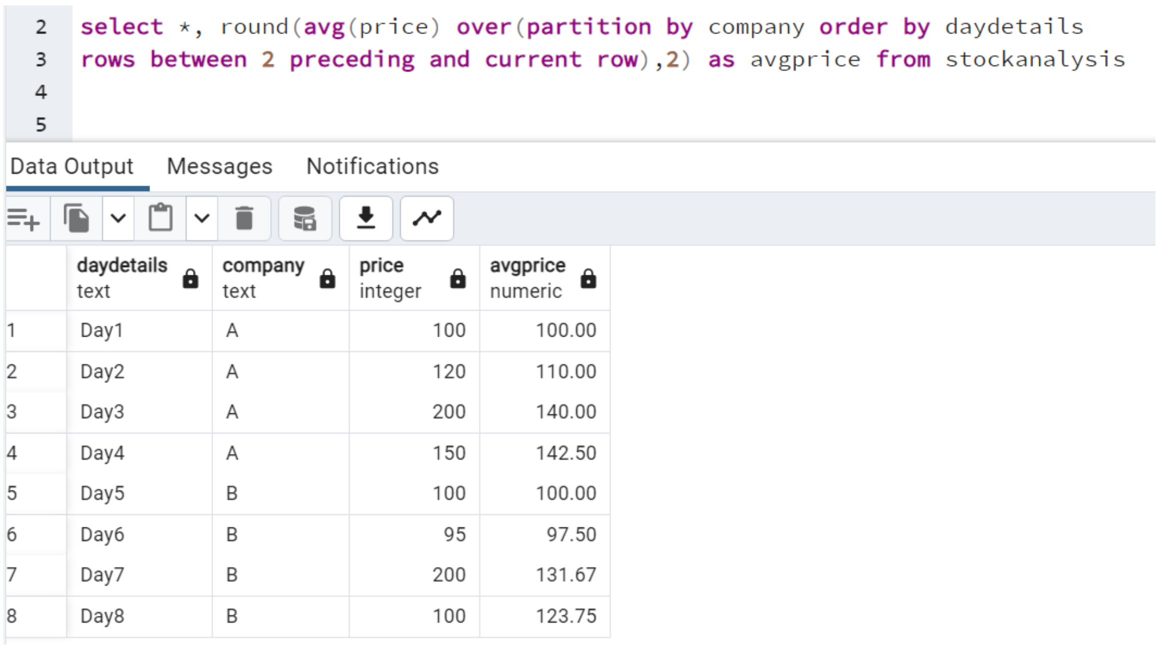
```
*************************************************************

--Q Find the 3 days moving average of stocks

select * into stockanalysis from (select * from (
values('Day1','A',100),('Day2','A',120),('Day3','A',200),
      ('Day4','A',150),('Day5','B',100),('Day6','B',95),('Day7','B',200
),('Day8','B',100)
) as numbers(Daydetails,company,price))

-- Sample output
```



daydetails text	company text	price integer	avgprice numeric
Day1	A	100	100.00
Day2	A	120	110.00
Day3	A	200	140.00
Day4	A	150	142.50
Day5	B	100	100.00
Day6	B	95	97.50
Day7	B	200	131.67
Day8	B	100	123.75

```
-- Query
```

```
select *, round(avg(price) over(partition by company order by daydetails
rows between 2 preceding and current row),2) as avgprice from stockanalysis
```

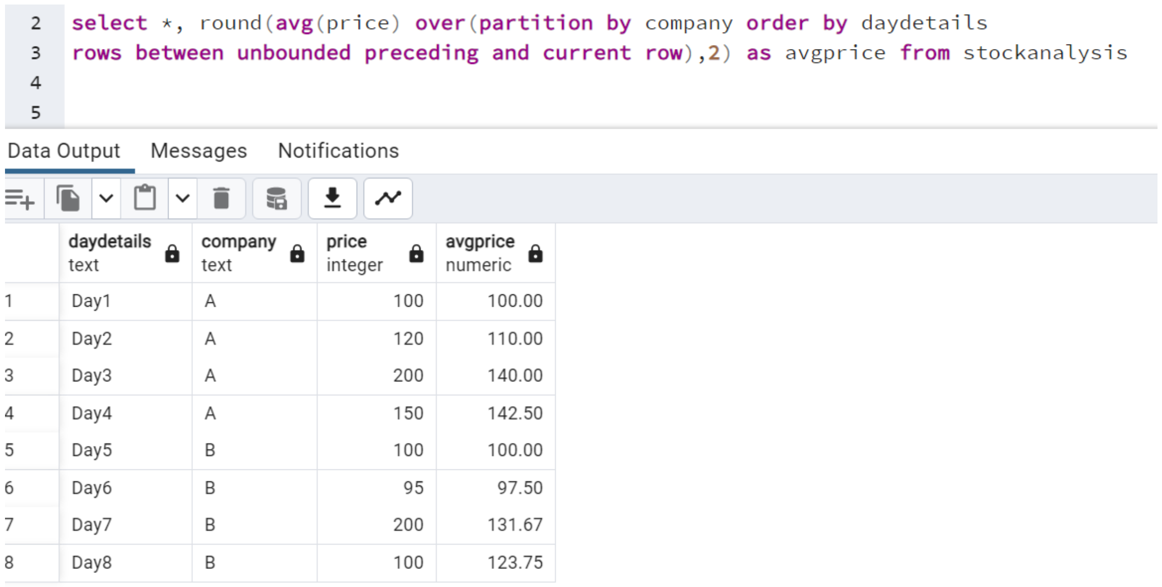


```
select *, round(avg(price) over(partition by company order by daydetails
rows between 2 preceding and current row),2) as avgprice from stockanalysis
```

Data Output Messages Notifications

	daydetails text	company text	price integer	avgprice numeric
1	Day1	A	100	100.00
2	Day2	A	120	110.00
3	Day3	A	200	140.00
4	Day4	A	150	142.50
5	Day5	B	100	100.00
6	Day6	B	95	97.50
7	Day7	B	200	131.67
8	Day8	B	100	123.75

```
-- Dispaly average value of a stock till date.

-- Query

select *, round(avg(price) over(partition by company order by daydetails
rows between unbounded preceding and current row),2) as avgprice from
stockanalysis
```

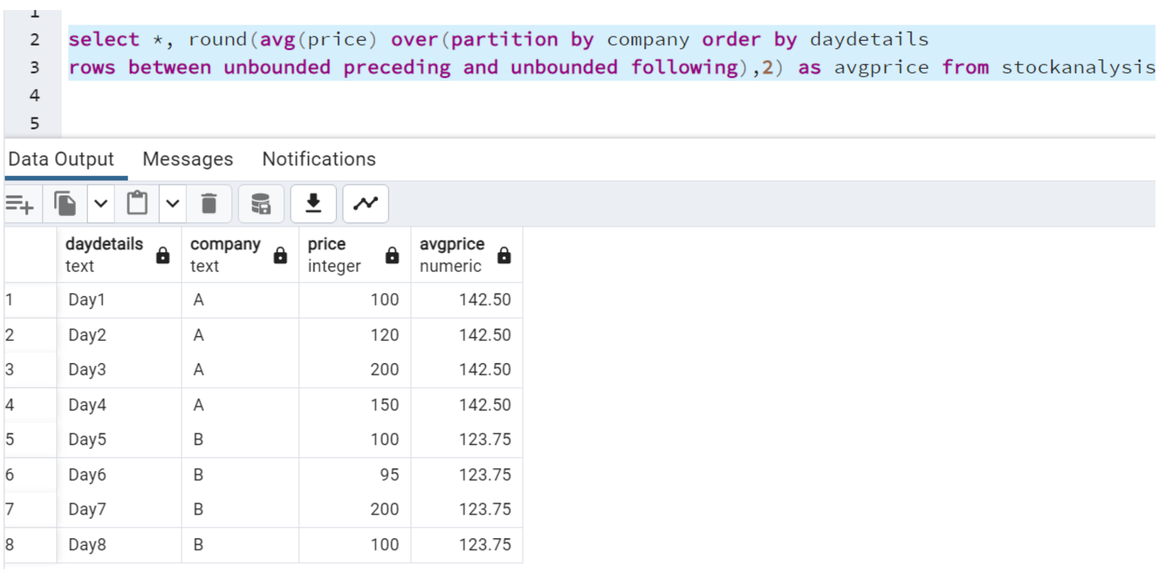


```
select *, round(avg(price) over(partition by company order by daydetails
rows between unbounded preceding and current row),2) as avgprice from stockanalysis
```

Data Output Messages Notifications

	daydetails text	company text	price integer	avgprice numeric
1	Day1	A	100	100.00
2	Day2	A	120	110.00
3	Day3	A	200	140.00
4	Day4	A	150	142.50
5	Day5	B	100	100.00
6	Day6	B	95	97.50
7	Day7	B	200	131.67
8	Day8	B	100	123.75

```
-- What is Unbounded Preceding and unbounded following?
-- Query

select *, round(avg(price) over(partition by company order by daydetails
rows between unbounded preceding and unbounded following),2) as avgprice
from stockanalysis
```



```
select *, round(avg(price) over(partition by company order by daydetails
rows between unbounded preceding and unbounded following),2) as avgprice from stockanalysis
```

	daydetails text	company text	price integer	avgprice numeric
1	Day1	A	100	142.50
2	Day2	A	120	142.50
3	Day3	A	200	142.50
4	Day4	A	150	142.50
5	Day5	B	100	123.75
6	Day6	B	95	123.75
7	Day7	B	200	123.75
8	Day8	B	100	123.75

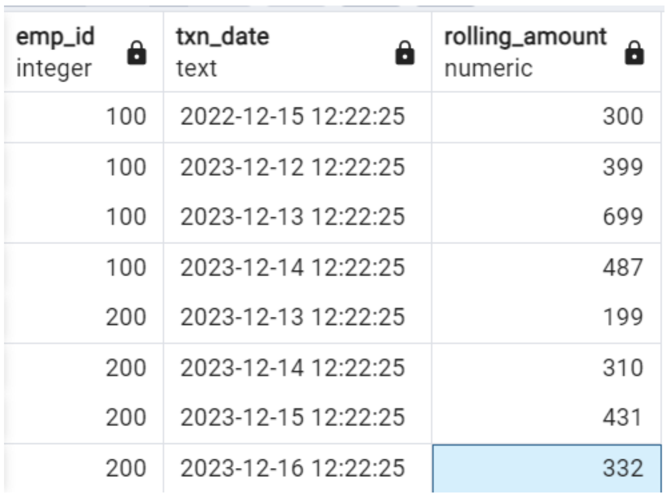
Note:

ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING: Includes all rows in the partition from the first to the last row.

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW: Includes all rows in the partition from the first row up to and including the current row.

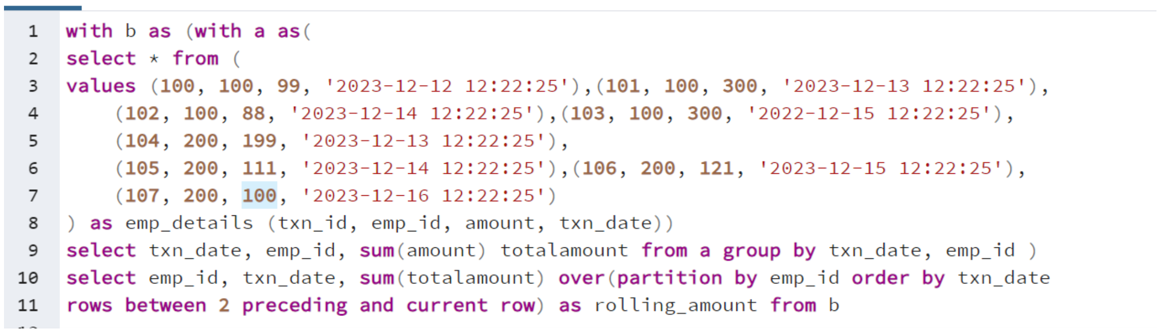
****

```
--Q Display the rolling salary of each customer_id

-- Sample Output
```

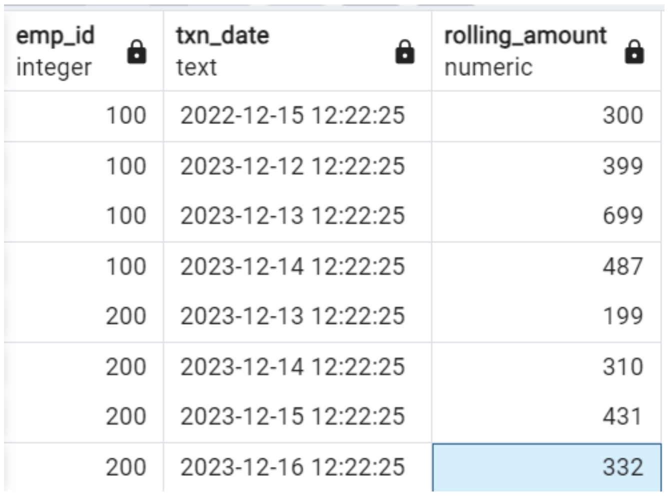


emp_id integer	txn_date text	rolling_amount numeric
100	2022-12-15 12:22:25	300
100	2023-12-12 12:22:25	399
100	2023-12-13 12:22:25	699
100	2023-12-14 12:22:25	487
200	2023-12-13 12:22:25	199
200	2023-12-14 12:22:25	310
200	2023-12-15 12:22:25	431
200	2023-12-16 12:22:25	332

```
-- Query

with b as (with a as(
select * from (
values (100, 100, 99, '2023-12-12 12:22:25'),(101, 100, 300, '2023-12-13
12:22:25'),
      (102, 100, 88, '2023-12-14 12:22:25'),(103, 100, 300, '2022-12-15
12:22:25'),
      (104, 200, 199, '2023-12-13 12:22:25'),
      (105, 200, 111, '2023-12-14 12:22:25'),(106, 200, 121, '2023-12-15
12:22:25')
) as emp_details (txn_id, emp_id, amount, txn_date))
select txn_id, emp_id, sum(amount) totalamount from a
group by txn_id, emp_id )
select emp_id, txn_id, sum(totalamount) over(partition by emp_id order by
txn_id
range between 2 preceding and current row) as rolling_amount from b
```

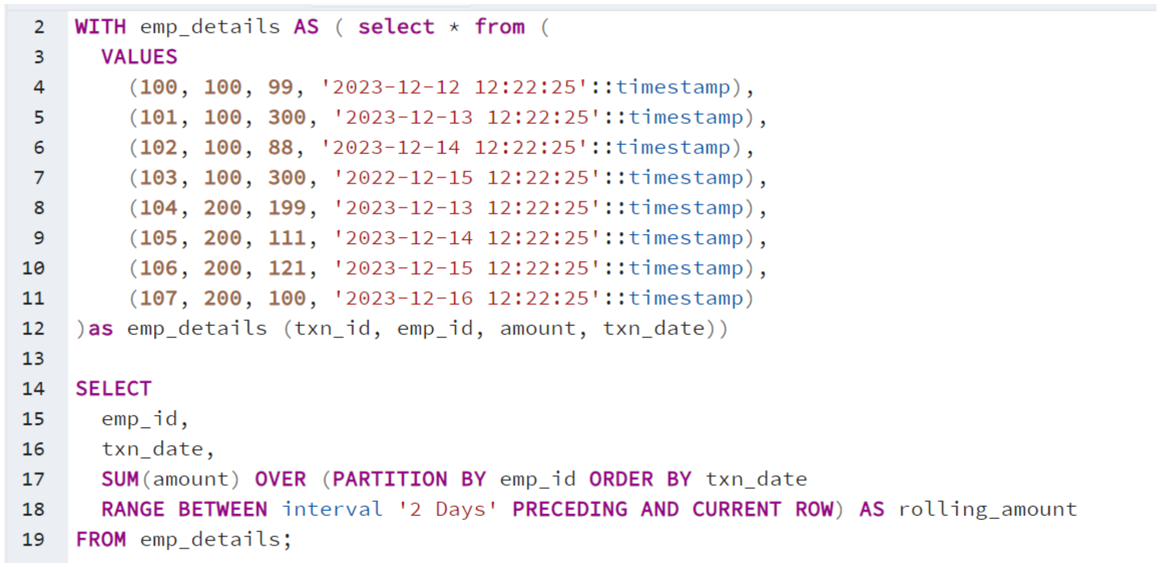


```
with b as (with a as(
select * from (
values (100, 100, 99, '2023-12-12 12:22:25'),(101, 100, 300, '2023-12-13 12:22:25'),
    (102, 100, 88, '2023-12-14 12:22:25'),(103, 100, 300, '2022-12-15 12:22:25'),
    (104, 200, 199, '2023-12-13 12:22:25'),
    (105, 200, 111, '2023-12-14 12:22:25'),(106, 200, 121, '2023-12-15 12:22:25'),
    (107, 200, 100, '2023-12-16 12:22:25')
) as emp_details (txn_id, emp_id, amount, txn_date))
select txn_date, emp_id, sum(amount) totalamount from a group by txn_date, emp_id )
select emp_id, txn_date, sum(totalamount) over(partition by emp_id order by txn_date
rows between 2 preceding and current row) as rolling_amount from b
```

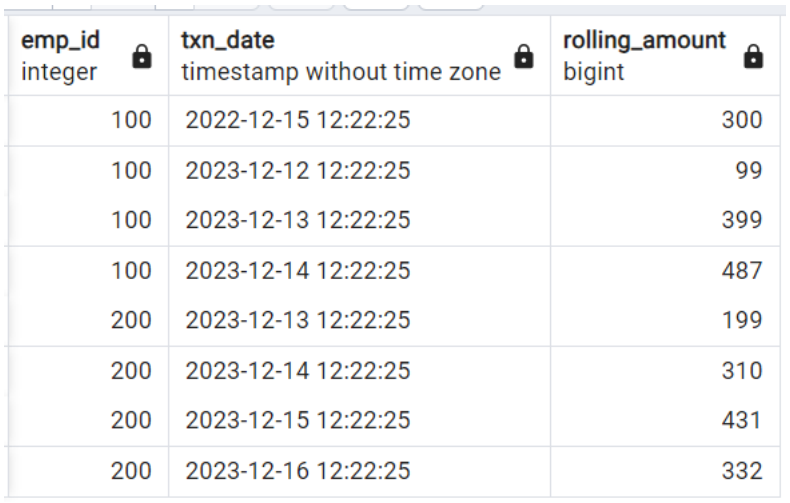


emp_id integer	txn_date text	rolling_amount numeric
100	2022-12-15 12:22:25	300
100	2023-12-12 12:22:25	399
100	2023-12-13 12:22:25	699
100	2023-12-14 12:22:25	487
200	2023-12-13 12:22:25	199
200	2023-12-14 12:22:25	310
200	2023-12-15 12:22:25	431
200	2023-12-16 12:22:25	332

```
***************************************************************

-- Display the rolling 2 days salary

-- Query

WITH emp_details AS ( select * from (
  VALUES
    (100, 100, 99, '2023-12-12 12:22:25'::timestamp),
    (101, 100, 300, '2023-12-13 12:22:25'::timestamp),
    (102, 100, 88, '2023-12-14 12:22:25'::timestamp),
    (103, 100, 300, '2022-12-15 12:22:25'::timestamp),
    (104, 200, 199, '2023-12-13 12:22:25'::timestamp),
    (105, 200, 111, '2023-12-14 12:22:25'::timestamp),
    (106, 200, 121, '2023-12-15 12:22:25'::timestamp),
    (107, 200, 100, '2023-12-16 12:22:25'::timestamp)
)as emp_details (txn_id, emp_id, amount, txn_date))

SELECT
  emp_id,
  txn_date,
  SUM(amount) OVER (PARTITION BY emp_id ORDER BY txn_date
  RANGE BETWEEN interval '2 Days' PRECEDING AND CURRENT ROW) AS rolling_amount
FROM emp_details;
```



```sql
WITH emp_details AS ( select * from (
  VALUES
    (100, 100, 99, '2023-12-12 12:22:25'::timestamp),
    (101, 100, 300, '2023-12-13 12:22:25'::timestamp),
    (102, 100, 88, '2023-12-14 12:22:25'::timestamp),
    (103, 100, 300, '2022-12-15 12:22:25'::timestamp),
    (104, 200, 199, '2023-12-13 12:22:25'::timestamp),
    (105, 200, 111, '2023-12-14 12:22:25'::timestamp),
    (106, 200, 121, '2023-12-15 12:22:25'::timestamp),
    (107, 200, 100, '2023-12-16 12:22:25'::timestamp)
)as emp_details (txn_id, emp_id, amount, txn_date))

SELECT
  emp_id,
  txn_date,
  SUM(amount) OVER (PARTITION BY emp_id ORDER BY txn_date
  RANGE BETWEEN interval '2 Days' PRECEDING AND CURRENT ROW) AS rolling_amount
FROM emp_details;
```



emp_id integer	txn_date timestamp without time zone	rolling_amount bigint
100	2022-12-15 12:22:25	300
100	2023-12-12 12:22:25	99
100	2023-12-13 12:22:25	399
100	2023-12-14 12:22:25	487
200	2023-12-13 12:22:25	199
200	2023-12-14 12:22:25	310
200	2023-12-15 12:22:25	431
200	2023-12-16 12:22:25	332

What is the differenct between below 2 functions:

Range and Rows

RANGE BETWEEN interval '2 Days' PRECEDING AND CURRENT ROW: Considers rows within a time interval of 2 days preceding the current row based on the specified timestamp column.

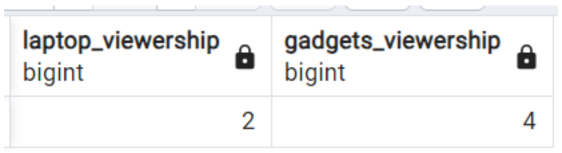
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW: Considers a fixed number of 2 rows preceding the current row based on the order defined in the ORDER BY clause, irrespective of the time interval.

****

--Q Laptop vs Mobile Viewership

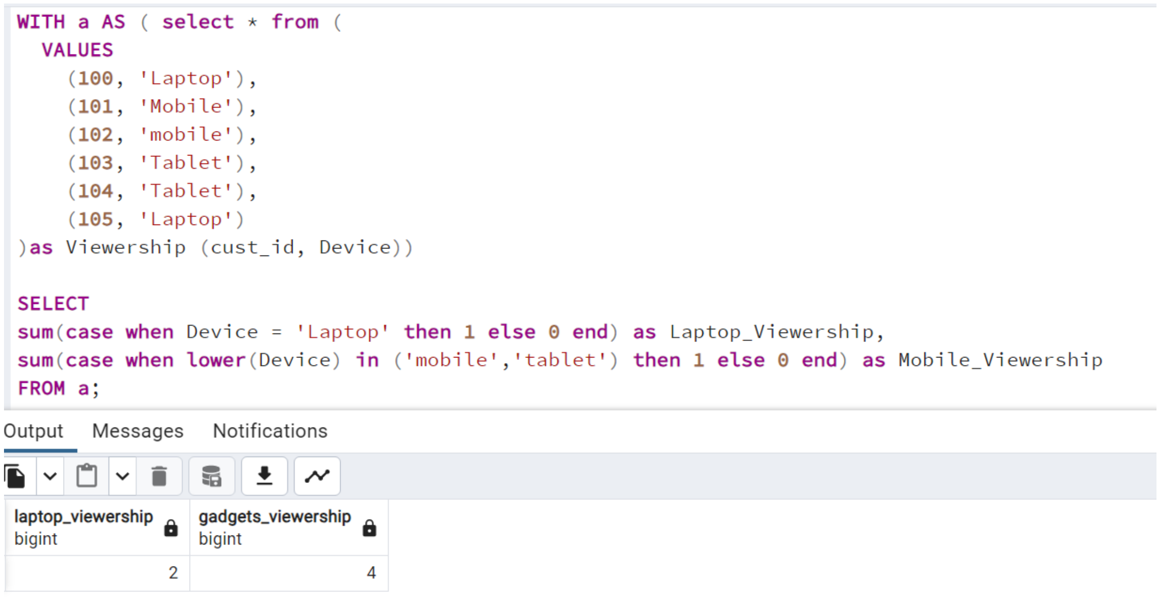
Note: Consider Mobile and Tablet viewership as Mobile viewership.

```
-- Sample Output
```



laptop_viewership bigint	gadgets_viewership bigint
2	4

```
-- Query


WITH a AS ( select * from (
  VALUES
    (100, 'Laptop'),
    (101, 'Mobile'),
    (102, 'mobile'),
    (103, 'Tablet'),
    (104, 'Tablet'),
    (105, 'Laptop')
)as Viewership (cust_id, Device))

SELECT
sum(case when Device = 'Laptop' then 1 else 0 end) as Laptop_Viewership,
sum(case when lower(Device) in ('mobile','tablet') then 1 else 0 end) as
Mobile_Viewership
FROM a;
```



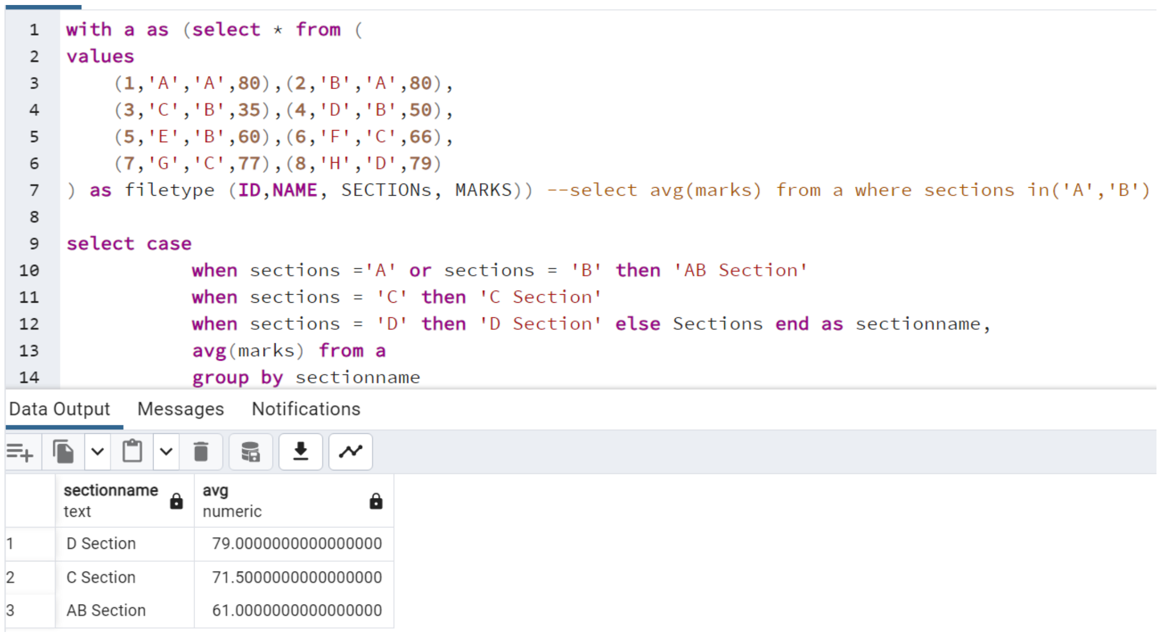
```
WITH a AS ( select * from (
  VALUES
    (100, 'Laptop'),
    (101, 'Mobile'),
    (102, 'mobile'),
    (103, 'Tablet'),
    (104, 'Tablet'),
    (105, 'Laptop')
)as Viewership (cust_id, Device))

SELECT
sum(case when Device = 'Laptop' then 1 else 0 end) as Laptop_Viewership,
sum(case when lower(Device) in ('mobile','tablet') then 1 else 0 end) as Mobile_Viewership
FROM a;
```

Output Messages Notifications

laptop_viewership bigint	gadgets_viewership bigint
2	4

```
***************************************************************

-- Display the average marks of AB section in one bucket and C, D each in
seperate bucket.
```

```
-- Method 1

with b as (with a as (select * from (
values
      (1,'A','A',80),(2,'B','A',80),
      (3,'C','B',35),(4,'D','B',50),
      (5,'E','B',60),(6,'F','C',66),
      (7,'G','C',77),(8,'H','D',79)
) as filetype (ID,NAME, SECTIONs, MARKS))

select sections, round(avg(marks),2) marks from a
group by sections)

select sum(case when sections in ('A','B') then marks else 0 end ) as
avg_mark_a_b,
sum(case when sections = 'C' then marks else 0 end) as avg_mark_C,
sum(case when sections = 'D' then marks else 0 end) as avg_mark_D
from b;

-- Method 2

with a as (select * from (
values
      (1,'A','A',80),(2,'B','A',80),
      (3,'C','B',35),(4,'D','B',50),
      (5,'E','B',60),(6,'F','C',66),
      (7,'G','C',77),(8,'H','D',79)
) as filetype (ID,NAME, SECTIONs, MARKS)) --select avg(marks) from a where
sections in('A','B')

select case
              when sections ='A' or sections = 'B' then 'AB Section'
              when sections = 'C' then 'C Section'
              when sections = 'D' then 'D Section' else Sections end as
sectionname,
              avg(marks) from a
              group by sectionname
```



```
with a as (select * from (
values
    (1,'A','A',80),(2,'B','A',80),
    (3,'C','B',35),(4,'D','B',50),
    (5,'E','B',60),(6,'F','C',66),
    (7,'G','C',77),(8,'H','D',79)
) as filetype (ID,NAME, SECTIONs, MARKS)) --select avg(marks) from a where sections in('A','B')

select case
            when sections ='A' or sections = 'B' then 'AB Section'
            when sections = 'C' then 'C Section'
            when sections = 'D' then 'D Section' else Sections end as sectionname,
            avg(marks) from a
            group by sectionname
```

Data Output Messages Notifications

	sectionname text	avg numeric
1	D Section	79.0000000000000000
2	C Section	71.5000000000000000
3	AB Section	61.0000000000000000

Note: Use round function to round off the decimal values in the output.