Window functions _ 2

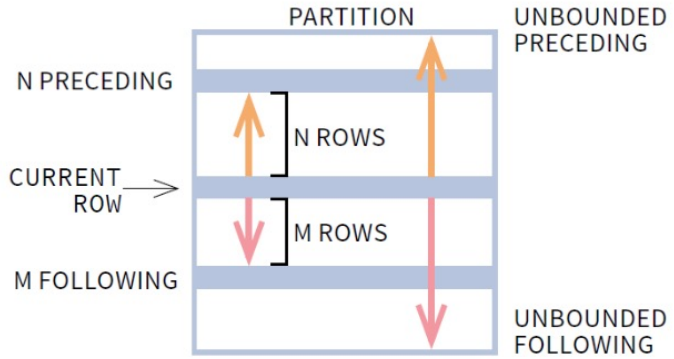
Quiz

- Assume a table contains marks, students and score. When we want to filter the students that have scored the 4th highest mark, which is the ideal function to use?
  - Rank()
  - DenseRank() - corrrect
  - Row_number()
- Sum(sales) over(partition by vendor_id). In this example the entire table is considered as one partition?
  - True
  - False - correct
- When have not mentioned any PARTITION BY then entire table is considered as one partition?
  - True - correct
  - False
- Ntile create based only based on the number of rows and the ntile value and not based on the values in a particular column?
  - True - correct
  - False
- Assume a table of 5 rows with 2 unique categories in the category column and sales as another column. How many rows we get if
  1. groupby on the category and sum(sales)
  2. Sum(sales) using window functions
  - Groupby – 5, Window_fn. - 2
  - Groupby – 3, Window_fn. - 3
  - Groupby – 2, Window_fn. - 5 - correct
  - Groupby – 3, Window_fn. - 4

NTILE – Split the data into 10 buckets within each product_id ordered by price in descending order

```
SELECT
vendor_id,
market_date,
product_id,
original_price,
NTILE(10) OVER (ORDER BY original_price DESC) AS price_ntile
FROM farmers_market.vendor_inventory
ORDER BY original_price DESC;
```

Understanding Window frames




```sql
select Date, Sales,
sum(s.sales) over() as ovr_sales,
sum(s.sales) over(order by date range between unbounded preceding and unbounded following) as ovr_sales_1,
sum(s.sales) over(order by date) as cum_sum,
sum(s.sales) over(order by date range between unbounded preceding and current row) as cum_sum_1,
sum(s.sales) over(order by date rows between unbounded preceding and current row) as unb_pre_cur_row,
sum(s.sales) over(order by date rows between 1 preceding and current row) as pre_1_cur_row,
sum(s.sales) over(order by date rows between 1 preceding and 1 following) as pre_1_folo_1,
sum(s.sales) over(order by date rows between current row and 1 following) as cur_row_folo_1
from temp_sales.sales s
```

Date	Sales	ovr_sales	ovr_sales_1	cum_sum	cum_sum_1	unb_pre_cur_row	pre_1_cur_row	pre_1_folo_1	cur_row_folo_1
2017-03-01	200	2600	2600	600	600	200	200	600	600
2017-03-01	400	2600	2600	600	600	600	600	900	700
2017-04-01	300	2600	2600	1200	1200	900	700	1000	600
2017-04-01	300	2600	2600	1200	1200	1200	600	1100	800
2017-05-01	500	2600	2600	2600	2600	1700	800	1200	900
2017-05-01	400	2600	2600	2600	2600	2100	900	1400	900
2017-05-01	500	2600	2600	2600	2600	2600	900	900	500

Question: Using the **vendor_booth_assignments** table in the Farmer's Market database, display each vendor's booth assignment for each ***market_date*** alongside their previous booth assignments.

```
select *,
lag(booth_number, 1) over (partition by vendor_id
                           order by market_date) as prev_booth
from `farmers_market.vendor_booth_assignments`
```

Question: The Market manager may want to filter these query results to a specific market date to determine which vendors are new or changing booths that day, so we can contact them and ensure setup goes smoothly.

```
select * from
(
select vendor_id, market_date, booth_number as current_booth,
lag(booth_number, 1) over (partition by vendor_id order by market_date) as prev_booth
from `farmers_market.vendor_booth_assignments`
) X
where (current_booth<>prev_booth) or (prev_booth is null)
```

Question: Let's say you want to find out if the total sales on each market date are higher or lower than they were on the previous market date.

```
select * from
(
select
	market_date,
	sum(quantity*cost_to_customer_per_qty) as current_sales,
	lag(sum(quantity*cost_to_customer_per_qty), 1) over (order by market_date) as
prev_day_sales
from farmers_market.customer_purchases
group by market_date
order by market_date
)X
where X.current_sales>X.prev_day_sales
```

Question: Calculate the moving average of sales on a window frame of 1 preceding and 1 following.
Assuming the one row is equal to one day only

```
SELECT date, sale,
    AVG(sale) OVER (ORDER BY date
                RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS sliding_avg
    FROM sales;
```

How will you calculate 5 - DAY moving average for Stock prices?

```
SELECT
      date, sale,
      AVG(sale) OVER (ORDER BY date RANGE 4 PRECEDING and current row) AS
sliding_avg
FROM sales;
```

 # This is also correct

```
SELECT
      date, sale,
      AVG(sale) OVER (ORDER BY date RANGE 4 PRECEDING) AS sliding_avg
FROM sales;
```