SQL Window functions and Date and Time

Quiz

- Sum(sales) over() is same as sum(sales) over(order by date range between unbounded preceding and unbounded following)
  - True - correct
  - False
- Sum(sales) over(order by date) is same as sum(sales) over(order by date range between current row and unbounded following)
  - True
  - False – correct
- When we have not mentioned any PARTITION BY then entire table is considered as one partition?
  - True - correct
  - False
- Sum(sales) over(partition by vendor_id). In this example all the records within each vendor is considered as 1 frame?
  - True - correct
  - False
- Assume we have monthly sales data. Which is the right query to get the running avg of sales by month.
  - Avg(sales) over (order by month) - correct
  - Avg(sales) over (partition by month)
  - Avg(sales) over (partition by month order by month range between unbounded preceding and current row)
  - Avg(sales) over (order by month range between unbounded preceding and unbounded following)
- Find the wrong answer
  - Order by date – has access to curr_row, all rows above curr.row and all rows after curr.row - correct
  - Order by date range between unbounded preceding and unbounded following – has access to curr_row, all rows above it and below it
  - Order by date range between curr.row and unbounded following – has access to curr_row and rows after it
  - Order by date range between 1 preceding and 2 following – has access to curr_row, 1 rows above it and 2 rows after it

Assume there is an employee table, create three new columns as
* Find the name of the employee with the highest salary within each dept
* Find the name of the employee with the lowest salary within each dept
* Name of the employee in the 3rd row in each dept with salary sorted in asc order.
and show the result in each row

```
select * ,
first_value(employee) over (partition by dept order by salary) lowest_salary_employee,
last_value(employee) over (partition by dept order by salary) Highest_salary_employee,
nth_value(employee, 3) over (partition by dept order by salary) nth_salary_employee
from `temp_sales.employee_salary`
```

dept	date	employee	salary	lowest_salary_employee	Highest_salary_employee	nth_salary_employee
dept_A	2015-03-01	B	200	B	B	*null*
dept_A	2015-03-01	C	300	B	C	*null*
dept_A	2015-01-01	A	1000	B	A	A
dept_B	2015-05-01	G	200	G	H	*null*
dept_B	2015-05-01	H	200	G	H	*null*
dept_B	2015-07-01	I	250	G	I	I
dept_B	2015-04-01	D	300	G	D	I

Note: first_value, last_value, nth_value may not be the right function to use logically when 2 or employees have the same salary. As these function only pullout the value in that particular row and not look at the value logically.

How can we modify the previous query such that last_value/nth_value has access to all the rows?

```
select * ,
last_value(employee) over (partition by dept order by salary
                           rows between unbounded preceding
                           and unbounded following) Highest_salary_employee,
nth_value(employee, 3) over (partition by dept order by salary
                           rows between unbounded preceding
                           and unbounded following) nth_row_employee
from `temp_sales.employee_salary`
```

Row	dept	date	employee	salary	Highest_salary_employee	nth_row_employee
1	dept_A	2015-03-01	B	200	A	A
2	dept_A	2015-03-01	C	300	A	A
3	dept_A	2015-01-01	A	1000	A	A
4	dept_B	2015-05-01	G	200	D	I
5	dept_B	2015-05-01	H	200	D	I
6	dept_B	2015-07-01	I	250	D	I
7	dept_B	2015-04-01	D	300	D	I

Creating a new date_time_demo table

```
# Big Query version
CREATE TABLE farmer_market.datetime_demo AS
(
SELECT market_date,
      market_start_time,
      market_end_time
      , parse_datetime('%Y-%m-%d %I:%M %P', CONCAT(market_date, ' ', market_start_time))AS market_start_datetime
      , parse_datetime('%Y-%m-%d %I:%M %P',CONCAT(market_date, ' ', market_end_time)) AS market_end_datetime
FROM farmer_market.market_date_info
)
```

Refer to this link: https://cloud.google.com/bigquery/docs/reference/standard-sql/format-elements

```
# My SQL version

CREATE TABLE farmers_market.datetime_demo AS
(
SELECT market_date,
market_start_time,
market_end_time,
STR_TO_DATE(CONCAT(market_date,' ', market_start_time), '%Y-%m-%d %h:%i %p') AS market_start_datetime,
STR_TO_DATE(CONCAT(market_date,' ', market_end_time), '%Y-%m-%d %h:%i %p') AS market_end_datetime
FROM farmers_market.market_date_info
)
Refer to this link: https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html
```

Extract various date, time components (BigQuery)

```
select market_start_datetime,
   extract(date from market_start_datetime) as date,
   extract(time from market_start_datetime) as time,
   extract(year from market_start_datetime) as year_no,
   extract(quarter from market_start_datetime) as q_no,
   extract(month from market_start_datetime) as month_no,
   extract(day from market_start_datetime) as day_no,
   extract(week from market_start_datetime) as week_no,
   extract(DAYOFWEEK from market_start_datetime) as week_day,
   extract(hour from market_start_datetime) as hr,
   extract(minute from market_start_datetime) as minu,
   extract(second from market_start_datetime) as second,
   format_datetime("%B", market_start_datetime) as month_name,
   format_datetime("%A", market_start_datetime) as day_name
from farmer_market.datetime_demo
```

Create customer_purchase_date table with purchases and market_date_info tables(big query)

```
create table farmer_market.customer_purchases_date as
(
     SELECT
     c.market_date,

     PARSE_DATETIME('%Y-%m-%d %I:%M %P', CONCAT(c.market_date, " ", m.market_start_time )) AS market_start_datetime,
     PARSE_DATETIME('%Y-%m-%d %I:%M %P', CONCAT(c.market_date, " ", m.market_end_time )) AS market_end_datetime,
     PARSE_DATETIME('%Y-%m-%d %H:%M:%S', CONCAT(c.market_date, " ", c.transaction_time )) AS market_date_transaction_time,
     c.product_id,
     c.vendor_id,
     c.customer_id,
     c.quantity,
     c.cost_to_customer_per_qty
     FROM
          farmer_market.customer_purchases c
     LEFT JOIN
          farmer_market.market_date_info m
     ON
          c.market_date = m.market_date
)
```

Create customer_purchase_date table with purchases and market_date_info tables(My SQL)

```
create table farmer_market.customer_purchases_date as
(
     SELECT
     c.market_date,

     STR_TO_DATE(CONCAT(c.market_date, " ", m.market_start_time), '%Y-%m-%d %h:%i %p') AS market_start_datetime,
     STR_TO_DATE(CONCAT(c.market_date, " ", m.market_end_time),'%Y-%m-%d %h:%i %p') AS market_end_datetime,
     STR_TO_DATE(CONCAT(c.market_date, " ", c.transaction_time),'%Y-%m-%d %H:%i:%s') AS market_date_transaction_time,
     c.product_id,
     c.vendor_id,
     c.customer_id,
     c.quantity,
     c.cost_to_customer_per_qty
     FROM
          farmer_market.customer_purchases c
     LEFT JOIN
          farmer_market.market_date_info m
     ON
          c.market_date = m.market_date
)
```

Question: Suppose you wish to know from which year to which year data do we have in our database?

```
select
min(extract(year FROM market_start_datetime)) AS start_YEAR,
max(EXTRACT(YEAR FROM market_start_datetime)) AS end_YEAR,
from `farmers_market.datetime_demo`
```

Question: What if you only want to see the hour at which the market started and ended on each date?

```
SELECT
market_start_datetime,
market_end_datetime,
EXTRACT(hour FROM market_start_datetime) AS start_hr,
EXTRACT(hour FROM market_end_datetime) AS end_hr
FROM
`farmers_market.datetime_demo`
```

Question: Let's say you want to calculate how many sales occurred within the first 30 minutes after the farmer's market opened, how would you dynamically determine what cutoff time to use? (Big Query)

```
select
Count(*) from
(
    SELECT
        market_start_time,
        transaction_time,
        TIME_ADD(market_start_time, INTERVAL 30 minute) AS first_30_mins
    FROM
    `farmer_market.customer_purchases_date`
) x
where transaction_time between x.market_start_time and
x.first_30_mins
```