USE campusx;

SELECT \*, AVG(marks) OVER(PARTITION BY branch) FROM campusx.marks;

-Who has the least and who has the highest marks

SELECT \*,

MIN(marks) OVER(),

MAX(marks) OVER()

FROM marks

ORDER BY student\_id;

SELECT \*,

AVG(marks) OVER() AS 'Overall\_avg',

MIN(marks) OVER(),

MAX(marks) OVER(),

MIN(marks) OVER(PARTITION BY branch),

MAX(marks) OVER(PARTITION BY branch)

FROM marks

ORDER BY student\_id;

Aggregate Function with OVER()

* Find all the students who have marks higher than the average marks of their respective branch

SELECT \* FROM (SELECT \*,

AVG(marks) OVER(PARTITION BY branch) AS 'branch\_avg'

FROM marks) t

WHERE t.marks > t.branch\_avg;

**RANK/DENSE\_RANK/ROW\_NUMBER**

**RANK**

SELECT \*,

RANK() OVER(ORDER BY marks DESC)

FROM marks;

Above code is for **ranking** on the complete marks data

SELECT \*,

RANK() OVER(PARTITION BY branch ORDER BY marks DESC)

FROM marks;

The above code was for just rank. Below one includes **DENSE RANK**

SELECT \*,

RANK() OVER(PARTITION BY branch ORDER BY marks DESC),

DENSE\_RANK() OVER(PARTITION BY branch ORDER BY marks DESC)

FROM marks;

**ROW\_NUMBER**

SELECT \*,

ROW\_NUMBER() OVER()

FROM marks;

----------------------

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY branch)

FROM marks;

**CONCAT**

SELECT \*,

CONCAT(branch, '-', ROW\_NUMBER() OVER(PARTITION BY branch))

FROM marks;

1. Find top 2 most paying customers of each month

First, we need to find out how much each customer has paid for each month

SELECT \* FROM (SELECT MONTHNAME(date) AS 'month\_name', user\_id, SUM(amount) AS 'total',

RANK() OVER(PARTITION BY MONTHNAME(date) ORDER BY SUM(amount) DESC) AS 'month\_rank'

FROM session\_35.orders

GROUP BY MONTHNAME(date), user\_id

ORDER BY MONTHNAME(date)) t

WHERE t.month\_rank < 3

ORDER BY month\_name DESC, month\_rank ASC;

1. Create roll no from branch and marks

**FIRST\_VALUE/LAST VALUE/NTH\_VALUE**

Using marks dataset

Q. Find out the name of the boy who has the highest marks

SELECT \*,

FIRST\_VALUE(name) OVER(ORDER BY marks DESC)

FROM marks;

* Who has the least marks in a particular branch

SELECT \*,

LAST\_VALUE(name) OVER(PARTITION BY branch ORDER BY marks DESC

ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)

FROM marks;

Q. Find the 2nd topper of each branch

SELECT \*,

NTH\_VALUE(name, 2) OVER(PARTITION BY branch

ORDER BY marks DESC

ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)

FROM marks

If we give 5 which is not there, it will give NULL values

1. Find the branch toppers

SELECT name, marks, branch FROM (SELECT \*,

FIRST\_VALUE(name) OVER(PARTITION BY branch ORDER BY marks DESC) AS 'topper\_name',

FIRST\_VALUE(marks) OVER(PARTITION BY branch ORDER BY marks DESC) AS 'topper\_marks'

FROM marks) t

WHERE t.name = t.topper\_name AND t.marks = t.topper\_marks

SELECT name, marks, branch FROM (SELECT \*,

FIRST\_VALUE(name) OVER w AS 'topper\_name',

FIRST\_VALUE(marks) OVER w AS 'topper\_marks'

FROM marks

WINDOW w AS (PARTITION BY branch ORDER BY marks DESC

ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)) t

WHERE t.name = t.topper\_name AND t.marks = t.topper\_marks;

SELECT name, marks, branch FROM (SELECT \*,

LAST\_VALUE(name) OVER w AS 'last\_marks\_name',

LAST\_VALUE(marks) OVER w AS 'least\_marks'

FROM marks

WINDOW w AS (PARTITION BY branch ORDER BY marks DESC

ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)) t

WHERE t.name = t.last\_marks\_name AND t.marks = t.least\_marks;

FRAME Clause

1. Find the last guy of each branch
2. Alternate way of writing Window functions
3. Find the 2nd last guy of each branch, 5th topper of each branch

**LEAD & LAG**

SELECT \*,

LAG(marks) OVER(PARTITION BY branch ORDER BY student\_id),

LEAD(marks) OVER(PARTITION BY branch ORDER BY student\_id)

FROM marks;

A screenshot of a data

Description automatically generated

* Find the MoM revenue growth of Zomato

USE session\_35;

SELECT MONTHNAME(date), SUM(amount)

FROM orders

GROUP BY MONTHNAME(date)

ORDER BY MONTHNAME(date) DESC;

This right now results in this

A screenshot of a computer

Description automatically generated

What we need to do is subtracts June’s sum with previous and divide it with that only to get growth %

SELECT MONTHNAME(date), SUM(amount),

LAG(SUM(amount)) OVER(ORDER BY MONTHNAME(date) DESC)

FROM orders

GROUP BY MONTHNAME(date)

ORDER BY MONTHNAME(date) DESC;

A screenshot of a computer

Description automatically generated

SELECT MONTHNAME(date), SUM(amount),

((SUM(amount) - LAG(SUM(amount)) OVER(ORDER BY MONTHNAME(date) DESC)) / LAG(SUM(amount)) OVER(ORDER BY MONTHNAME(date) DESC)) \* 100

FROM orders

GROUP BY MONTHNAME(date)

ORDER BY MONTHNAME(date) DESC;

A screenshot of a computer

Description automatically generated