WINDOWS FUNCTION

```
SELECT *, AVG(class_id) OVER(PARTITION BY enrollment_year)
FROM sql_cx_live.students;

CREATE TABLE students(
    student_id INTEGER PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    branch VARCHAR(255) NOT NULL,
    marks INTEGER NOT NULL
)

INSERT INTO campusx.students VALUES
(1, 'Nitish', 'EEE', 82),
(2, 'Rishab', 'EEE', 91),
(3, 'Anukant', 'EEE', 69),
(4, 'Rupesh', 'EEE', 55),
(5, 'Shubam', 'CSE', 78),
(6, 'Ved', 'CSE', 43),
(7, 'Deepak', 'CSE', 98),
(8, 'Arpan', 'CSE', 95)
```

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

USE campusx;

SELECT *,

AVG(marks) OVER() AS 'Overall_Average',

MIN(marks) OVER(),

MAX(marks) OVER(PARTITION BY branch),

MIN(marks) OVER(PARTITION BY branch)

FROM students

-- Find all the students who have marks higher than the avg marks of their respective branch

SELECT * FROM (SELECT *,

AVG(marks) OVER(PARTITION BY branch) AS 'branch_avg'

FROM campusx.students) t

WHERE t.marks > t.branch_avg

- WINDOWS FUNCTION

-- RANK()

-- if the rank number two or three are same then it will rank like 1 1 3 | 5 5 7 <- here skip 2 and 6

rank the students by their marks

SELECT *,

RANK() OVER(ORDER BY marks DESC)

FROM campusx.students

```
-- rank the students by their marks for each branch

SELECT *,

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

FROM campusx.students


-- DESNC_RANK()

-- if the rank number two or three are same then it will rank like 1 1 2
| 5 5 6  <- here no skip

USE campusx;

SELECT *,

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

FROM students


-- ROW_NUMBER()

USE campusx;

SELECT *,

ROW_NUMBER() OVER(PARTITION BY branch)

FROM students
```

```
-- suppose assign unique roll number

SELECT *,

CONCAT(branch, '-', ROW_NUMBER() OVER(PARTITION BY branch))

FROM marks


-- Find top 2 most paying customers

SELECT * FROM (SELECT *,

SUM(amount) OVER(PARTITION BY user_id) AS 'total_money'

FROM zomato.orders) t1

ORDER BY total_money DESC LIMIT 2


-- Find top 2 most paying customers of each month

SELECT * FROM (SELECT user_id, MONTHNAME(date) AS 'month',
SUM(amount) AS 'total',

RANK() OVER(PARTITION BY MONTHNAME(date) ORDER BY
SUM(amount) DESC) AS 'rank_per_month'

FROM zomato.orders

GROUP BY user_id, MONTHNAME(date)

ORDER BY MONTHNAME(date) DESC) t

WHERE t.rank_per_month < 3;
```

```
-- FIRST_VALUE

-- Find which students have most high marks

SELECT *,

FIRST_VALUE(name) OVER(ORDER BY marks DESC) FROM
campusx.students


-- LAST_VALUE

-- have to change window frame

SELECT *,

LAST_VALUE(marks) OVER(PARTITION BY branch

                  ORDER BY marks DESC

                                      ROWS BETWEEN UNBOUNDED
PRECEDING AND UNBOUNDED FOLLOWING)

FROM campusx.students


-- NTH_VALUE

SELECT *,

NTH_VALUE(name, 2) OVER(PARTITION BY branch

                  ORDER BY marks DESC

                                      ROWS BETWEEN UNBOUNDED
PRECEDING AND UNBOUNDED FOLLOWING)

FROM campusx.students

-- ## for First, last, and nth value  This windows frame have to us
```

```sql
-- Find the branch toppers (name, branch marks)
SELECT name, branch, marks 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 students) t
WHERE t.name = t.topper_name AND t.marks = t.topper_marks
```

```sql
-- LAG() - create a column that show  a specific desided column value side under
SELECT *,
LAG(marks) OVER(ORDER BY student_id)
FROM campusx.students
```

```sql
-- LEAD() create a column that show  a specific decided column value side upper
SELECT *,
LEAD(marks) OVER(ORDER BY student_id)
FROM campusx.students
```

```
-- Find the MoM revenue growth of Zomato

SELECT MONTHNAME(date), SUM(amount),

LEAD(SUM(amount)) OVER() - SUM(amount)

FROM zomato.orders

GROUP BY MONTHNAME(date)


-- Find the top batsman of each ipl team

-- SELECT BattingTeam,batter, SUM(batsman_run) AS 'total_run'FROM
group_sort.ipl

-- GROUP BY BattingTeam, batter

-- ORDER BY total_run DESC


-- RANK() windows function

-- Find the top 5 batsman of each ipl team


SELECT * FROM (SELECT BattingTEAM,batter, SUM(batsman_run),

RANK() OVER(PARTITION BY BattingTeam ORDER BY
SUM(batsman_run) DESC) AS 'rank'

FROM group_sort.ipl

GROUP BY BattingTeam,batter) t

WHERE t.rank < 6
```

-- CUMULATIVE SUM()

-- Find the total runs of virat kohli in his 50th match, 100 and 200 match

```
SELECT * FROM (SELECT CONCAT("Match-",CAST(ROW_NUMBER() OVER(ORDER BY ID) AS CHAR)) AS 'match_no',

SUM(batsman_run) AS 'runs_scored',

SUM(SUM(batsman_run)) OVER(ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS 'carrier_runs'

## AVG(SUM(batsman_run)) OVER(*copy upper ROWS BETW......) <- for CUM AVG SUM()  AS 'carrier average'

## AVG(SUM(batsman_run)) OVER(ROWS BETWEEN 9 PRECEDING AND CURRENT ROW) <- for running AVG()

FROM group_sort.ipl

WHERE batter = 'V Kohli'

GROUP BY ID) t

WHERE t.match_no = 'Match-5' OR t.match_no = 'Match-10' OR t.match_no = 'Match-13'

-- Instead of use this OVER() under ROWS BETWN... for every window can use

# WINDOWS w AS (ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) <- just consider jumping range data
```

```
-- Find the total runs of virat kohli in his 50th match, 100 and 200
match

USE group_sort;

SELECT * FROM (SELECT CONCAT('Match-',CAST(ROW_NUMBER()
OVER(ORDER BY ID)AS CHAR)) AS 'match_no',

SUM(batsman_run) AS 'total_run',

SUM(SUM(batsman_run)) OVER w AS 'cum_sum',

AVG(SUM(batsman_run)) OVER w AS 'x',

AVG(SUM(batsman_run)) OVER(ROWS BETWEEN 9 PRECEDING AND
CURRENT ROW)

FROM ipl

WHERE batter = 'V Kohli'

GROUP BY ID

WINDOW w AS(ROWS BETWEEN UNBOUNDED PRECEDING AND
CURRENT ROW)) t


-- SELECT * FROM group_sort.ipl
-- PERCENT OF TOTAL

-- Find the most selled item for each rasturant

USE zomato;

SELECT f_name,

(total_value/SUM(total_value) OVER()) * 100 AS 'percent_of_total'

FROM (SELECT f_id, SUM(amount) AS 'total_value' FROM orders t1

JOIN order_details t2

ON t1.order_id = t2.order_id

WHERE r_id = 1 # 2, 3, 4 for each resturant

GROUP BY f_id) t

JOIN food t3

ON t.f_id = t3.f_id

ORDER BY percent_of_total DESC


-- PERCENT OF TOTAL

-- Find the most selled item for each rasturant

USE zomato;

SELECT * FROM(SELECT t4.r_name, t3.f_name, COUNT(*),

RANK() OVER(PARTITION BY r_name ORDER BY COUNT(*) DESC) AS
'rank'

FROM  orders t1

JOIN  order_details t2 ON  t1.order_id = t2.order_id
JOIN food t3 ON t2.f_id = t3.f_id

JOIN restaurants t4 ON t1.r_id = t4.r_id

GROUP BY t4.r_name, t3.f_name

ORDER BY t4.r_name) t

WHERE t.rank < 2
```

```
-- PERCENT CHANGE

USE campusx;

SELECT YEAR(date),MONTHNAME(date),SUM(views) AS 'views',

((SUM(views)- LAG(SUM(views)) OVER(ORDER BY YEAR(date),
MONTH(date)))/

LAG(SUM(views)) OVER(ORDER BY YEAR(date), MONTH(date)))*100
AS 'percent_change'

FROM youtube_views

GROUP BY YEAR(date), MONTHNAME(date)

ORDER BY YEAR(date), MONTH(date)

--** LAG(view, 7) OVER(ORDER BY date) <- you can alos change LAG
LEAD bottom upper range
```

```
-- PERCENTILE AND QUARTILE


-- Find the median marksof alll the studets

USE campusx;

SELECT *,

PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY marks) OVER() AS
'median_marks'

FROM students


-- Find branch wise median

USE campusx;

SELECT *,

PERCENTILE_DISC(0.5) WITHIN GROUP(PARTITION BY branch ORDER
BY marks) OVER() AS 'median_marks',

PERCENTILE_CONT(0.5) WITHIN GROUP(PARTITION BY branch ORDER
BY marks) OVER() AS 'median_marks_cont'

FROM students
```

```
-- OUTLIER REMOVING
SELECT * FROM (SELECT *,
PERCENTILE_CONT(0.25) WITHIN GROUP(ORDER BY marks) OVER() AS
'Q1',
PERCENTILE_CONT(0.75) WITHIN GROUP(ORDER BY marks) OVER() AS
'Q3'
FROM campusx.students) t
WHERE t.marks < (t.Q3 +(1.5*(t.Q3-t.Q1))
ORDER BY t.student_id


-- SEGMENTATION  : NTILE() <- bucket use
SELECT *,
NTILE(3) OVER(ORDER BY marks DESC) AS 'buckets'
FROM campusx.students;
SELECT brand_name, model, price,
CASE
   WHEN bucket = 1 THEN 'budget'
   WHEN bucket = 2 THEN 'mid_range'
   WHEN bucket = 3 THEN 'premium'
END AS 'phone_type'
FROM (SELECT brand_name, model, price,
NTILE(3) OVER(ORDER BY price)
FROM campusx.smartphones_cleaned_v6) t
```

```
-- CUMULATIVE DISTRIBUTION : like percentile

SELECT *,

CUME_DIST() OVER(ORDER BY marks) AS 'percentile_score'

FROM  campusx.students


-- PARTITION BY ON MULTIPLE COLUMN

SELECT source,destination,airline,AVG(price) AS 'avg_fare',

DENSE_RANK() OVER(PARTITION BY source,destition ORDER BY
AVG(price))

FROM flights

GROUP BY source, destination, airline


-- PERCENT OF TOTAL

-- Find the most selled item for each rasturant

USE zomato;

SELECT * FROM(SELECT t4.r_name, t3.f_name, COUNT(*),

RANK() OVER(PARTITION BY r_name ORDER BY COUNT(*) DESC) AS
'rank'

FROM  orders t1

JOIN  order_details t2 ON  t1.order_id = t2.order_id

JOIN food t3 ON t2.f_id = t3.f_id

JOIN restaurants t4 ON t1.r_id = t4.r_id
GROUP BY t4.r_name, t3.f_name

ORDER BY t4.r_name) t

WHERE t.rank < 2
```