**Ranking**

If we want to perform ranking in any categorical column, then we use window function.

1. Use IPL dataset and print the top 5 batsmen of each team

SELECT \* FROM (SELECT BattingTeam, batter, SUM(batsman\_run) AS 'total\_runs',

DENSE\_RANK() OVER(PARTITION BY BattingTeam ORDER BY SUM(batsman\_run) DESC) AS 'rank\_within\_team'

FROM ipl

GROUP BY BattingTeam, batter) t

WHERE t.rank\_within\_team < 6

ORDER BY t.BattingTeam, t.rank\_within\_team;

Q. Virat Kohli scored how much runs after his 50th, 100th, and 200th match

We use cumulative sum here

SELECT \* FROM

(SELECT

CONCAT("Match - ", ROW\_NUMBER() OVER(ORDER BY ID)) AS 'match\_no',

SUM(batsman\_run) AS 'runs\_scored',

SUM(SUM(batsman\_run)) OVER(ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS 'career\_runs'

FROM ipl

WHERE batter = 'V Kohli'

GROUP BY ID) t

WHERE match\_no = 'Match - 50' OR match\_no = 'Match - 100' OR match\_no = 'Match - 200';

Q. They can also ask the question. How many matches did Warner take to score 5000 runs

SELECT \* FROM (SELECT

CONCAT("Match - ", ROW\_NUMBER() OVER(ORDER BY ID)) AS 'match\_no',

SUM(batsman\_run) AS 'runs\_scored',

SUM(SUM(batsman\_run)) OVER(ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS 'career\_runs'

FROM ipl

WHERE batter = 'DA Warner'

GROUP BY ID) t

WHERE career\_runs > 5000

ORDER BY career\_runs

LIMIT 1;

Q. Virat Kohli scored how much runs and average after his 50th, 100th, and 200th match

We use Cumulative average over here

SELECT \* FROM

(SELECT

CONCAT("Match - ", ROW\_NUMBER() OVER(ORDER BY ID)) AS 'match\_no',

SUM(batsman\_run) AS 'runs\_scores',

SUM(SUM(batsman\_run)) OVER w AS 'career\_runs',

AVG(SUM(batsman\_run)) OVER w AS 'career\_avg'

FROM ipl

WHERE batter = 'V Kohli'

GROUP BY ID

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

;

Q. We are going to find the running average of virat kohli, with a margin of 10 matches

SELECT \* FROM

(SELECT

CONCAT("Match - ", ROW\_NUMBER() OVER(ORDER BY ID)) AS 'match\_no',

SUM(batsman\_run) AS 'run\_scores',

SUM(SUM(batsman\_run)) OVER w AS 'career\_runs',

AVG(SUM(batsman\_run)) OVER w AS 'career\_avg',

AVG(SUM(batsman\_run)) OVER(ROWS BETWEEN 9 PRECEDING AND CURRENT ROW) AS 'rolling\_avg'

FROM ipl

WHERE batter = 'V Kohli'

GROUP BY ID

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

;

Q. For a particular restaurant, find its most important food. (Take swiggy)

USE zomato\_case\_study;

SELECT f\_name,

ROUND((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

GROUP BY f\_id) t

JOIN food t3

ON t.f\_id = t3.f\_id

ORDER BY percent\_of\_total DESC;

Q. Find the percentage increase in the views

USE campusx;

SELECT MONTHNAME(date) FROM youtube\_views;

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

ROUND((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), MONTH(date)

ORDER BY YEAR(date), MONTH(date);

* When it is set to quarterly

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

ROUND((SUM(views) - LAG(SUM(views)) OVER(ORDER BY YEAR(date), QUARTER(date))) / LAG(SUM(views)) OVER(ORDER BY YEAR(date), QUARTER(date)) \* 100) AS 'percent\_change'

FROM youtube\_views

GROUP BY YEAR(date), QUARTER(date)

ORDER BY YEAR(date), QUARTER(date);

Q. What if we need to calculate weekly increase

- This shows how we can lag for 7 times, which is a week over here

SELECT \*,

ROUND(((views - LAG(views, 7) OVER(ORDER BY date)) / LAG(views, 7) OVER(ORDER BY date)) \* 100) AS 'weekly\_percent\_change'

FROM youtube\_views;

Percentiles & Quantiles

Q. Find the median marks of all the students

SELECT \*,

PERCENTILE\_DISC(0.5) WITHIN GROUP(ORDER BY marks) OVER() AS 'median\_marks',

FROM marks;

Q. Find branch wise median of student’s marks

SELECT \*,

PERCENTILE\_DISC(0.5) WITHIN GROUP(ORDER BY marks) OVER(PARTITION BY branch) AS 'median\_marks',

FROM marks;

* Removing outliers

INSERT INTO marks VALUE (17, 'mike', 'Work', 0);

UPDATE marks

SET marks = 1

WHERE student\_id = 17;

SELECT \* FROM

(SELECT \*,

PERCENTILE\_CONT(0.25) WITHING GROUP(ORDER BY marks) OVER() AS 'Q1',

PERCENTILE\_CONT(0.75) WITHING GROUP(ORDER BY marks) OVER() AS 'Q3'

FROM marks) t

WHERE t.marks <= t.Q1 - (1.5 \* (t.Q3 - t.Q1));

Q. Sort the marks dataset into categories sharp, medium, poor, based on marks

NTILE divides the dataset into categories

Ex if we have 11 rows. And we have to make a bucket of 3 then first it will create 1st bucket – 3

2nd bucket – 3

3rd bucket – 3

Then we are left with 2, each 1 will be added to 1st and 2nd bucket. Finally we will have

4, 4, 3

SELECT \*,

NTILE(3) OVER(ORDER BY marks DESC) AS 'buckets'

FROM marks;

This will make 3 buckets because we gave 3 in NTILE(3)

If we want to keep student\_id arranged

SELECT \*,

NTILE(3) OVER(ORDER BY marks DESC) AS 'buckets'

FROM marks

ORDER BY student\_id;

Q. Find the Price range of phone and arrange them in 3 buckets – Premium, Budget, Mid range.

- We must use NTILE

SELECT brand\_name, model, price,

NTILE(3) OVER(ORDER BY price)

FROM smartphones;

* We must name them now
* We must use CASE which is if else

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) AS 'bucket'

FROM smartphones) t

;

* If we want to sort them based on brands. Add partition

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(PARTITION BY brand\_name ORDER BY price) AS 'bucket'

FROM smartphones) t

;

* Cumulative Distribution

SELECT \* FROM

(SELECT \*,

ROUND(CUME\_DIST() OVER(ORDER BY marks), 2) AS 'percentile\_score'

FROM marks) t

WHERE percentile\_score > 0.75;

* Till now we have seen how to apply partition by on only 1 column. But what if we have to apply it on 2 columns?

USE flights;

SELECT \* FROM flights;

* If we want to find out the cheapest flight between a pair of city

USE flights;

SELECT \* FROM flights;

SELECT \* FROM

(SELECT Source, Destination, Airline, AVG(price) AS 'avg\_fare',

DENSE\_RANK() OVER(PARTITION BY Source, Destination ORDER BY AVG(price)) AS 'rank'

FROM flights

GROUP BY Source, Destination, Airline) t

WHERE t.rank < 2;