select \* from com;

select avg(ID) from com;

CREATE TABLE Employee (name varchar(10), salary int);

inserting sample data into Employee table

INSERT INTO Employee VALUES (1, 'Rick1', 'manager',1000,1);

INSERT INTO Employee VALUES (2, 'Rick2', 'manager',2000,2);

INSERT INTO Employee VALUES (3, 'Rick3', 'manager',1000,3);

INSERT INTO Employee VALUES (4, 'Rick4', 'manager',9000,2);

INSERT INTO Employee VALUES (5, 'Rick5', 'manager',1000,3);

INSERT INTO Employee VALUES (6, 'Rick6', 'manager',7000,1);

INSERT INTO Employee VALUES (7, 'Rick7', 'manager',1000,3);

INSERT INTO Employee VALUES (8, 'Rick8', 'manager',5000,1);

INSERT INTO Employee VALUES (9, 'Rick9', 'manager',3000,1);

update employee set salary =1500 where emp\_name ='Rick10'

INSERT INTO Employee VALUES (10, 'Rick10', 'manager',500,3);

select \* from employee;

drop table employee;

create table employee (

emp\_id int not null,

emp\_name varchar2(400),

designation varchar2(400),

salary number,

dept\_id number,

primary key (emp\_id)

);

alt

--Sum-MAx-Min-Avg-Count- Are Aggregrate function function !!!

-- Number of rows will decrease in output

select dept\_id , AVG(salary) from employee

group by dept\_id having dept\_id in (1,2);

-- In Analytics function number of rows will not decrease !

-- Performance will be better with analytics functions

--Example rank(),dense\_rank(),row\_number(),lag lead first last

-- analytics function ([arguments]) over (analytics\_clause)

-- avg(salary) over (partition by dept\_id)

--Avg salary per department infront of each row

select dept\_id,emp\_id,emp\_name, salary,avg(salary) over (partition by dept\_id) as avgsalary from employee order by dept\_id;

-- Second Highest salary per dept

select emp\_name ,dept\_id , salary, max(salary) over (partition by dept\_id) as maxSalaryForDept from employee;

-- RANK

Salary

100 100 50 25

1 1 3 4

--Dense rank -- No missing rank

100 100 50 25

1 1 2 3

--Row\_number is a analytic function rownum is phesudo column

-- Row\_Number is physical number

--Count is also analytic function better than using count aggregation function

-- lag is previous value lead is next value.

-- Numbe of employees in each dept

select dept\_id ,count(\*) from employee group by dept\_id order by dept\_id;

--same as

select dept\_id ,count(\*) from employee group by dept\_id order by 1;

--Arrange the salary in descending order for each deprtment

select emp\_name ,dept\_id, salary ,RANK() over (partition by dept\_id order by salary desc) as DEPT\_RANK from employee ;

-- select two top most salary from each dept

select \* from (

select emp\_name ,dept\_id, salary ,RANK() over (partition by dept\_id order by salary desc) as DEPT\_RANK from employee )

where DEPT\_RANK <=2;

select \* from (

select emp\_name ,dept\_id, salary ,RANK() over (partition by dept\_id order by salary desc) as DEPT\_RANK from employee )

where DEPT\_RANK <=5 order by dept\_id;

--- If we do not want to skip rank in between we should use dense rank

select \* from (

select emp\_name ,dept\_id, salary ,DENSE\_RANK() over (partition by dept\_id order by salary desc) as DEPT\_RANK from employee )

where DEPT\_RANK <=5 order by dept\_id;

--selecting both rank and dense rank for each department

select emp\_name,dept\_id ,salary ,

DENSE\_RANK() over (partition by dept\_id order by salary desc) as dense\_rank,

RANK() over (partition by dept\_id order by salary desc) as rank

from employee;

-- Rank and dense rank based on salary

select emp\_name,dept\_id ,salary ,

DENSE\_RANK() over ( order by salary desc) as dense\_rank,

RANK() over ( order by salary desc) as rank

from employee;

---Using RANK, DENSE\_RANk and ROW\_NUMBER together

select emp\_name,dept\_id ,salary ,

DENSE\_RANK() over (partition by dept\_id order by salary desc) as dense\_rank,

RANK() over (partition by dept\_id order by salary desc) as rank,

ROW\_NUMBER() over (partition by dept\_id order by salary desc) as ROW\_NUMBER

from employee;

--using count as analytics function

-- This will take much less time

select dept\_id ,count(\*) over (partition by dept\_id) as count\_per\_dept from employee ;

-- This will take more time so better use count as analytic function instead of aggregation function

select dept\_id, count(\*) from employee group by dept\_id;

--Avoid using winow operation and group by in same query it can be in nested query

--select dept\_id ,count(\*) over (partition by dept\_id) as count\_per\_dept from employee group by dept\_id;

-- Difference Between ROW\_NUMBER analytics function , Pseudoc column ROW\_NUM , Rowid is addrd of row

-- ROWID

-- IS PSEUDO COLUMN , RETURNS THE ADDRESS OF ROW

-- It is used to delete the dublicate records.

SELECT ROWID , emp\_NAME FROM EMPLOYEE;

-- Lets Insert Dublicate

ALTER table employee DROP PRIMARY KEY;

-- Insert Dublicate records

INSERT INTO Employee VALUES (1, 'Rick1', 'manager',1000,1);

select rowid,emp\_id from employee order by emp\_id;

select \* from employee emp1 where emp1.id in (select emp2.id from employee emp2 );

select \* from employee emp order by emp\_id;

--This will give dublicate reccords in table

select \* from employee emp1 where rowid not in (select max(rowid) from employee emp2 where emp1.emp\_id = emp2.emp\_id);

-- Ro delete dublicate records

delete from employee emp1 where rowid not in (select max(rowid) from employee emp2 where emp1.emp\_id = emp2.emp\_id);

--

select rownum, emp\_id, emp\_name from employee;

-- rowid is the address which uniquely identify the row on the disk.

-- can use to delete dublicate rownd from oracle

-- rownum is pseudo column , we can use it to fetch first 10 rows which oracle fetch.

-- Similar to limit in mysql

-- rowid is permanent but rownum is tempory to output of singe query

---http://www.dba-oracle.com/t\_rownum\_rowid\_difference.htm

select \* from employee emp1 where rowid in (select rowid from employee emp2 where emp1.emp\_id = emp2.emp\_id);

select rowid,emp\_id from employee;

select max(rowid) from employee;

select count(\*) from employee group by emp\_id having count(\*) >1;

--------------------------------------------------------------------LEAD /LAG -----------------------------------------------------------------------------------------------------

select \* from employee;

select emp\_id,emp\_name ,salary from employee order by salary;

--LAG window or analytics functions find salary and previous salary

-- In below queries we do not have any partioon thus we are applying on whole table

select emp\_id,emp\_name ,salary, LAG(salary,1,0) over (order by salary) as previousSalary , salary -LAG(salary,1,0) over (order by salary) as salaryDiff from employee order by salary;

select emp\_id,emp\_name ,salary, LEAD(salary,1,0) over (order by salary) as previousSalary , salary -LEAD(salary,1,0) over (order by salary) as salaryNext from employee order by salary;

-- https://www.youtube.com/watch?v=2wowWrlS\_6w&t=492s

----------------------------------------------------------------FIRST\_VALUE and LAST\_VALUE----------------------------------------------------------------------------------------

select emp\_id,emp\_name,salary,dept\_id,

first\_value(salary) over (order by salary) as smallestValueTillHere ,

first\_value(salary) over (order by salary desc) as smallestValueTillHereDesc

from employee;

select emp\_id,emp\_name,salary,dept\_id,

first\_value(salary) over (order by salary) as smallestValueTillHere ,

last\_value (salary) over (order by salary) as lastValueTillHere from employee;

select emp\_id,emp\_name,salary,dept\_id,

last\_value(salary) over (order by salary ) as largestValueTillHere

from employee ;

select emp\_id,emp\_name,salary,dept\_id,

first\_value(salary) over (order by salary) as smallestValueTillHere ,

first\_value(salary) over (order by salary desc) as largestValueTillHere

from employee order by salary ;

--- Smallest Value Till Here For Each Dept

--- But we have to use partition by before order by

select emp\_id,emp\_name,salary,dept\_id,

first\_value(salary) over (partition by dept\_id order by salary ) as smallestValueTillHereForDept ,

last\_value(salary) over (partition by dept\_id order by salary ) as largestValueTillHereForDept

from employee order by salary ;

-- Find 4th Largest salary ---

--rownum is psedudo column

-- This will return the 4 largest salary

select \* from (select \* from employee order by salary desc) where rownum <=4;

-- Fetch all records and assign a row\_number() anamlyitcs fucntion which assign a row\_number value based on order by column in this case its salary.

select e.\*, row\_number() over (order by salary desc) as employeeRowNum from employee e --

-- Fetch the 5rd largest salary using row\_number()

--

select \* from (select e.\*, row\_number() over (order by salary desc) as employeeRowNum from employee e) where employeeRowNum =5; -- This will return only one row even we have duplicates salary

--- Fetch the 5th largest salary using rank() ---- This will return all slaary which were 5th highest and same --

-- 1,2,2,4 -- Rank 3 might be missing in this case.

select \* from (select e.\*,rank()over (order by salary desc) as employeeSalaryRank from Employee e) where employeeSalaryRank =5;

--- We can use Dense rank if we dwould have all ranks in this case

--- Fetch the 5th largest salary using rank() ---- This will return all slaary which were 5th highest and same --

-- 1,2,2,3 -- Rank 3 will not be missing in case of dense rank

select \* from (select e.\*,dense\_rank()over (order by salary desc) as employeeSalaryRank from Employee e) where employeeSalaryRank =5;