**SQL topics:**

Select, From, Where, Group By, Limit, Join, Order of sql execution, Sub query, Cte

Partition (row number, rank, dense rank)

**Select, From:**

select \* from employees;

select First\_name, last\_name from employees;

select \* from employees where office\_id=1;

select \* from employees where office\_id=1 order by first\_name desc;

select distinct office\_id from employees;

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

**Where:**

select \* from employees where first\_name='north';

select \* from employees where salary>95000;

select \* from employees where job\_title='social worker' and salary>95000;

select \* from employees where last\_name like 'c%';

select \* from employees where job\_title in ('staff scientist', 'social worker');

select \* from employees where not salary>60000;

select \* from employees where reports\_to IS NULL;

select \* from employees where salary between 40000 and 70000;

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

**Group By:**

Average points for each city :

select city, avg(points) as average\_points from customers group by city;

select city, round(avg(points) ,1) as average\_points from customers group by city;

Sum of total points for each city:

select city, sum(points) as total\_points from customers group by city;

Count of total customers based on city:

select count(customer\_id) as total\_count, city from customers group by city order by total\_count desc;

Count of customers based on birth date:

select year(birth\_date) as year, count(customer\_id) as total\_customers from customers

group by year;

Revenue of products based on product\_id:

Select product\_id, sum(sell\_price\*quantity) as revenue from sales group by product\_id;

Find the cities where there are more than two customers:

select count(customer\_id) as count, city from customers group by city having count>2;

select first\_name, avg(points) as average from customers group by first\_name

having average>2000;

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

**Limit:**

select \* from customers limit 5;

select \* from customers limit 5 offset 6;

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

**Join:**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

select employees.employee\_id, employees.job\_title, employees.salary, offices.address

from employees inner join offices on employees.office\_id=offices.office\_id;

Different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables.
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table.
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table.
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table.

**Inner Join:**

select employees.employee\_id, employees.salary, offices.address from employees

inner join offices on employees.office\_id=offices.office\_id;

**Left Join:**

select employees.employee\_id, employees.salary, offices.address from employees

left join offices on employees.office\_id=offices.office\_id;

**Right Join:**

select employees.employee\_id, employees.salary, offices.address

from employees right join offices on employees.office\_id=offices.office\_id;

**Full Join:**

select employees.employee\_id, offices.address from employees full join offices on employees.office\_id=offices.office\_id;

**Self Join:**

Select e.emp\_id, e.emp\_nme, m.emp\_name as manager\_name, e.salary, m.salary as manager\_salary from emp e self join emp m on e.manager\_id=e.emp\_id;

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

**Subqueries:**

A subquery is a select query that is enclosed in another query. The inner select query is usually used to determine the results of the outer select query.

select job\_title from employees where salary=(select max(salary) from employees);

Types of subqueries:

Select, Update, Delete, Insert

select first\_name, last\_name, salary from employees where salary<(select avg(salary) from employees);

**Insert:**

create table products (prod\_id int, item varchar(30), sell\_price float,

product\_type varchar(30));

insert into products

values(101, 'Jewellery', 1800, 'Luxury'),

(102, 'T-Shirt', 100, 'Non-Luxury'),

(103, 'Laptop', 1300, 'Luxury'),

(104, 'Table', 400, 'Non-Luxury');

select \* from products;

create table orders

(order\_id int, product\_sold varchar(30), selling\_price float);

insert into orders

select prod\_id, item, sell\_price

from products where prod\_id in(select prod\_id from products where sell\_price>1000);

**Update:**

update employees set salary=salary\*0.35 where age in (select age from employees\_b where age>=27);

**Delete:**

delete from employees where age in (select age from employees\_b where age<=32);

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

**Partition/ Window functions:**

select e.\*, max(salary) over() as max\_salary from employees e;

select e.\*, max(salary) over(partition by job\_title) as max\_salary from employees e;

**Row Number:**

select e.\*, row\_number() over() as rn from employees e;

select e.\*, row\_number() over(partition by job\_title) as rn from employees e;

select e.\*, row\_number() over(partition by job\_title order by employee\_id) as rn from employees e;

To fetch only two records from each department

select \* from (

select e.\*, row\_number() over(partition by job\_title order by employee\_id)

as rn from employees e) x

where x.rn<3

**Rank:**

select e.\*, rank() over(partition by job\_title order by salary desc) as rnk from employees e;

Fetch top 3 employees in each dept earning the max salary

select \* from(

select e.\*, rank() over(partition by job\_title order by salary desc)

as rnk from employees e) as x

where x.rnk<4;

**Dense Rank:**

select e.\*, rank() over(partition by job\_title order by salary desc)

as rnk, dense\_rank() over(partition by job\_title order by salary desc)

as dense\_rnk from employees e;

**Combination of Rank, Dense Rank and Row Number:**

select e.\*, rank() over(partition by job\_title order by salary desc)

as rnk, dense\_rank() over(partition by job\_title order by salary desc)

as dense\_rnk, row\_number() over(partition by job\_title order by salary desc)

as rn from employees e;

**Lag:**

**Fetching salary of previous employee**

select e.\*, lag(salary) over(partition by job\_title order by employee\_id desc)

as previous\_emp\_sal from employees e;

select e.\*, lag(salary, 2, 0) over(partition by job\_title order by employee\_id desc)

as previous\_emp\_sal from employees e;

**Lead:**

Lag and Lead combined

select e.\*, lag(salary) over(partition by job\_title order by employee\_id desc)

as previous\_emp\_sal,lead(salary) over(partition by job\_title order by employee\_id desc)

as next\_emp\_sal from employees e;

select e.\*, lag(salary) over(partition by job\_title order by employee\_id desc)

as previous\_emp\_sal,

case when e.salary > lag(salary) over(partition by job\_title order by employee\_id desc)

then 'Higher than previous employee'

when e.salary < lag(salary) over(partition by job\_title order by employee\_id desc)

then 'Lower than previous employee'

when e.salary = lag(salary) over(partition by job\_title order by employee\_id desc)

then 'Same as the previous employee'

end sal\_range from employees e;