```
create database employee_1;
use employee_1;
create table employee
( emp_ID int
, emp_NAME varchar(50)
, DEPT_NAME varchar(50)
, SALARY int);

insert into employee values(101, 'Mohan', 'Admin', 4000);
insert into employee values(102, 'Rajkumar', 'HR', 3000);
insert into employee values(103, 'Akbar', 'IT', 4000);
insert into employee values(104, 'Dorvin', 'Finance', 6500);
insert into employee values(105, 'Rohit', 'HR', 3000);
insert into employee values(106, 'Rajesh',  'Finance', 5000);
insert into employee values(107, 'Preet', 'HR', 7000);
insert into employee values(108, 'Maryam', 'Admin', 4000);
insert into employee values(109, 'Sanjay', 'IT', 6500);
insert into employee values(110, 'Vasudha', 'IT', 7000);
insert into employee values(111, 'Melinda', 'IT', 8000);
insert into employee values(112, 'Komal', 'IT', 10000);
insert into employee values(113, 'Gautham', 'Admin', 2000);
insert into employee values(114, 'Manisha', 'HR', 3000);
insert into employee values(115, 'Chandni', 'IT', 4500);
insert into employee values(116, 'Satya', 'Finance', 6500);
insert into employee values(117, 'Adarsh', 'HR', 3500);
insert into employee values(118, 'Tejaswi', 'Finance', 5500);
insert into employee values(119, 'Cory', 'HR', 8000);
insert into employee values(120, 'Monica', 'Admin', 5000);
insert into employee values(121, 'Rosalin', 'IT', 6000);
insert into employee values(122, 'Ibrahim', 'IT', 8000);
insert into employee values(123, 'Vikram', 'IT', 8000);
insert into employee values(124, 'Dheeraj', 'IT', 11000);
COMMIT;
```

```
select * from employee;

-- What is the Max salary of the employee?
select max(salary) as Max_salary from employee;

-- What is the use of the Window Function?
-- Using Aggregate function as Window Function
-- Without window function, SQL will reduce the no of records.
select dept_name, max(salary) from employee
group by dept_name;

-- By using MAX as an window function, SQL will not reduce records but the result will be
shown corresponding to each record.
select e.*,
max(salary) over(partition by dept_name) as max_salary
from employee e;



-- row_number(), rank() and dense_rank()
select e.*,
row_number() over(partition by dept_name) as rn
from employee e;



-- Fetch the first 2 employees from each department to join the company.
select * from (
        select e.*,
        row_number() over(partition by dept_name order by emp_id) as rn
        from employee e) x
where x.rn < 3;



-- Fetch the top 3 employees in each department earning the max salary.
```

```
select * from (
        select e.*,
        rank() over(partition by dept_name order by salary desc) as rnk
        from employee e) x
where x.rnk < 4;

-- Checking the different between rank, dense_rnk and row_number window functions:
select e.*,
rank() over(partition by dept_name order by salary desc) as rnk,
dense_rank() over(partition by dept_name order by salary desc) as dense_rnk,
row_number() over(partition by dept_name order by salary desc) as rn
from employee e;



-- lead and lag

-- fetch a query to display if the salary of an employee is higher, lower or equal to the
previous employee.
select e.*,
lag(salary) over(partition by dept_name order by emp_id) as prev_empl_sal,
case when e.salary > lag(salary) over(partition by dept_name order by emp_id) then 'Higher
than previous employee'
   when e.salary < lag(salary) over(partition by dept_name order by emp_id) then 'Lower
than previous employee'
        when e.salary = lag(salary) over(partition by dept_name order by emp_id) then 'Same
than previous employee' end as sal_range
from employee e;

-- Similarly using lead function to see how it is different from lag.
select e.*,
lag(salary) over(partition by dept_name order by emp_id) as prev_empl_sal,
lead(salary) over(partition by dept_name order by emp_id) as next_empl_sal
from employee e;
```