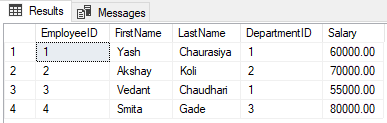
**Practical no 8 :Query based on operators and joins .simple nested query**

**NAME: Tushar Vishal Charavande Roll No:25**

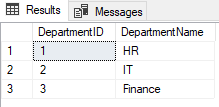
**Div: A Batch no: B1**

select \* from Employees;



Department Table:

select \* from Departments;

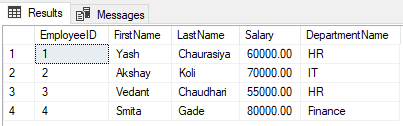


Simple Query with Joins:

SELECT e.EmployeeID,e.FirstName,e.LastName,e.Salary,d.DepartmentName

FROM Employees e JOIN

Departments d ON e.DepartmentID = d.DepartmentID;

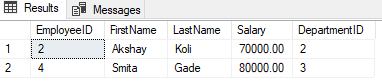


Nested Query with Operators:

SELECT EmployeeID, FirstName, LastName,Salary,DepartmentID

FROM Employees WHERE Salary > (SELECT AVG(Salary) FROM Employees WHERE

DepartmentID = Employees.DepartmentID);



**Practical 7 :- Demonstrate use of operators.**

**NAME: Tushar Vishal Charavande Roll No:25**

**Div: A Batch no: B1**

/\*Arithmetic Operators:

Addition\*/

SELECT 25 + 10 AS AdditionResult;

--Subtraction\*/

SELECT 43 - 5 AS SubtractionResult;

--Multiplication

SELECT 7 \* 7 AS MultiplicationResult;

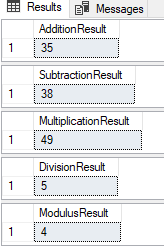
--Division

SELECT 25 / 5 AS DivisionResult;

--Modulus (remainder of division)

SELECT 18 % 7 AS ModulusResult;

**Output:-**



--Comparison Operators:

--Equal to

SELECT \* FROM employee WHERE emp\_salary = 1000;

--Not equal to

SELECT \* FROM employee WHERE dep\_id <> 2;

--Greater than

SELECT \* FROM employee WHERE emp\_salary > 10000;

--Less than

SELECT \* FROM employee WHERE emp\_salary < 10000;

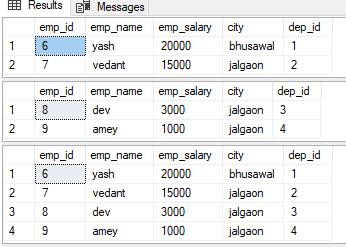
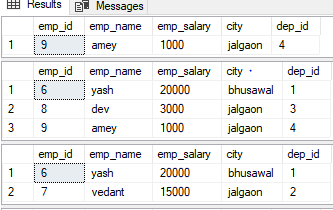
--Greater than or equal to

SELECT \* FROM employee WHERE emp\_salary >= 1000;

--Less than or equal to

SELECT \* FROM employee WHERE emp\_salary <= 10000.00;

**Output:**



--Logical Operators:

--AND

SELECT \* FROM employee WHERE emp\_salary > 10000 and emp\_id = '6';

--OR

SELECT \* FROM employee WHERE emp\_name='dev' OR emp\_salary < 1000;

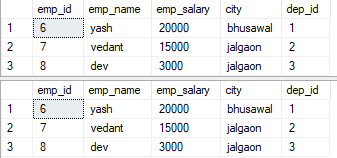
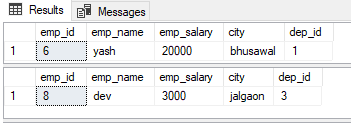
--NOT

SELECT \* FROM employee WHERE NOT emp\_id = '9';

--Range query with BETWEEN

SELECT \* FROM employee WHERE emp\_salary BETWEEN 3000 AND 20000;\

**Output:**



**Practical 10**: Write down SQL by using i. Aggregate functions ii. Date functions iii. String functions.

**NAME: Tushar Vishal Charavande Roll No:25**

**Div: A Batch no: B1**

Aggregate functions:

select count(emp\_salary) as count from e123;

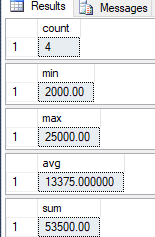
select min(emp\_salary) as min from e123;

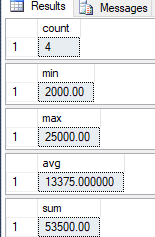
select max(emp\_salary) as max from e123;

select avg(emp\_salary) as avg from e123;

select sum(emp\_salary) as sum from e123;

select\*from e123;



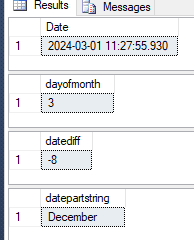


Date functions

select GETDATE() as Date

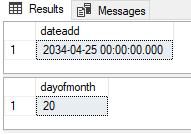
select datediff(year,'2024/02/29','2016/05/7') as datediff

select datename(mm,'2024/12/12') as datepartstring



select day ('2024/03/20') as dayofmonth

select dateadd(year,10,'2024/04/25') as dateadd



**String function:**

select concat('model',' ','price') as concat

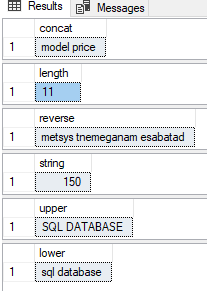
select len('hello world') as length

select reverse('database management system') as reverse

select str(149.8) as string

select upper('sql database') as upper

select LOWER('sql database') as lower

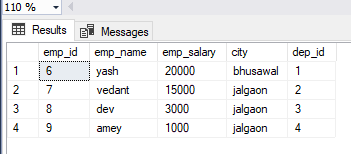


**Practical no 9: write down sql by using WHERE CLAUSE GROUP BY HAVING CLAUSE**

**NAME: Tushar Vishal Charavande Roll No:25**

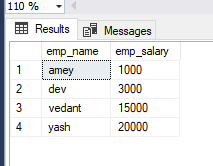
**Div: A Batch no: B1**

select\*from employee;



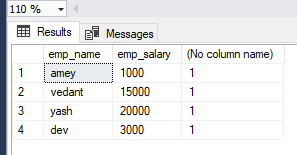
1. WHERE CLAUSE :

select emp\_name,emp\_salary from employee where emp\_salary !=5000 group by emp\_name,emp\_salary;



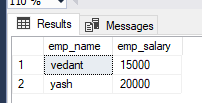
1. GROUP BY:

select emp\_name,emp\_salary,count(\*) from employee group by emp\_name,emp\_salary;



1. HAVING CLAUSE:

select emp\_name,emp\_salary from employee group by emp\_name,emp\_salary having (emp\_salary)>11000;



**PRACTICLE 4:- DEMONSTRATE TO SELECT IN CLAUSE**

**NAME: Tushar Vishal Charavande Roll No:25**

**Div: A Batch no: B1**

**Order by Descending:**

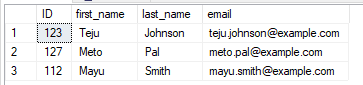
use Collegestudentsdb;

select top 3\* from employees

where (ID>1)

order by first\_name desc

**output:-**



**Order by Ascending:**

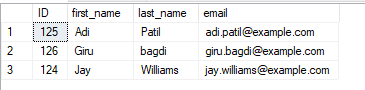
use Collegestudentsdb;

select top 3\* from employees

where (ID>1)

order by first\_name asc

**Output:**



**PRACTICLE 6:- Demonstrate integrity constraints (primary keys,Foreign key,check,Default)**

**NAME: Tushar Vishal Charavande Roll No:25**

**Div: A Batch no: B1**

use Collegestudentsdb;

create table dept1 (deptid int primary key,deptname varchar (50));

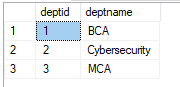
insert into dept1 values (1,'BCA');

insert into dept1 values (2,'Cybersecurity');

insert into dept1 values (3,'MCA');

select \*from dept1;

**output:-**



create table employee2 (id int not null primary key,

name varchar (50) not null,

salary varchar (50) check (salary>0),

address varchar (50) default 'Bhusawal',

deptid int foreign key references dept1(deptid));

insert into employee2 values (1,'Girish',50000,'Jalgaon',1);

insert into employee2 values (2,'Kaveri',30000,'Jalgaon',2);

insert into employee2 values (3,'Gaurav',20000,'Jalgaon',3);

select \*from employee2;

**Output:**

