Day 6

Cross Join

```sql
SELECT e.ename, d.dname
FROM emp e CROSS JOIN dept d;
```

```sql
SELECT e.ename, d.dname
FROM emp e, dept d;
```

Inner Join

```sql
SELECT e.ename, d.dname
FROM emp e INNER JOIN dept d
ON e.deptno = d.deptno;
```

```sql
SELECT e.ename, d.dname
FROM emp e , dept d
WHERE e.deptno = d.deptno;
```

LEFT OUTER JOIN

```sql
SELECT e.ename, d.dname
FROM emp e LEFT OUTER JOIN dept d
ON e.deptno = d.deptno;
```

```sql
SELECT e.ename, d.dname
FROM emp e LEFT JOIN dept d
ON e.deptno = d.deptno;
```

RIGHT OUTER JOIN

```sql
SELECT e.ename, d.dname
FROM dept d LEFT JOIN emp e
ON e.deptno = d.deptno;
```

```
SELECT e.ename, d.dname
FROM emp e RIGHT OUTER JOIN dept d
ON e.deptno = d.deptno;
```

```
SELECT e.ename, d.dname
FROM emp e RIGHT JOIN dept d
ON e.deptno = d.deptno;
```

Full Join

- MySQL do not support full join but using set operators we can implement it.
- Set Operators
    1. UNION
        - It supress duplicate element
    2. UNION ALL
        - It doesn't supress duplicate element
- Full join is union of left outer join and right outer join

```
SELECT e.ename, d.dname
FROM emp e LEFT JOIN dept d
ON e.deptno = d.deptno
UNION
SELECT e.ename, d.dname
FROM emp e RIGHT JOIN dept d
ON e.deptno = d.deptno;
```

```
SELECT e.ename, d.dname
FROM emp e LEFT JOIN dept d
ON e.deptno = d.deptno
UNION ALL
SELECT e.ename, d.dname
FROM emp e RIGHT JOIN dept d
ON e.deptno = d.deptno;
```

SELF JOIN

- If we use join on same table themn it is called self join.
- Display name of employee and its manager

```
for( e : emp )
{
    for( m : emp )
    {
        if( e.mgr == m.empno )
            print( e.ename & m.ename )
    }
}
```

```
SELECT e.ename, m.ename
FROM emp e INNER JOIN emp m
ON e.mgr = m.empno;
```

```
SELECT e.ename, m.ename
FROM emp m INNER JOIN emp e
ON e.mgr = m.empno;
```

```
SELECT e.ename, m.ename
FROM emp m , emp e
WHERE e.mgr = m.empno;
```

Table Relations

1. One To One
   - emp(1) <--->(1)address
2. One To Many
   - dept(1) <--->(*)emp
3. Many To one
   - emp(*) <---> (1)dept
4. Many To Many
   - emp() <---->() meeting

Tables

1. dept : deptno, dname

```
CREATE TABLE dept
(
    deptno INT,
    dname VARCHAR(50),
    CONSTRAINT PK_DEPT_DEPTNO
    PRIMARY KEY(deptno)
);
INSERT INTO dept
VALUES
( 10,'DEV'),
( 20,'QA'),
( 30,'ACCOUNT'),
( 40,'OPERATIONS');
```

2. emp : empno, ename, deptno

```
CREATE TABLE emp
(
    empno INT,
    ename VARCHAR(50),
    deptno INT,
    CONSTRAINT PK_EMP_EMPNO PRIMARY KEY(empno),
    CONSTRAINT FK_EMP_DEPT FOREIGN KEY ( deptno ) REFERENCES dept( deptno
)
);

INSERT INTO emp
VALUES
(1, 'Rahul', 10),
(2, 'Amit', 10),
(3, 'Sandeep', 20),
(4, 'Nilesh', 50),
(5, 'Nitin', 50);
```

3. address : loc, pincode, empno

```
CREATE TABLE address
(
    id INT AUTO_INCREMENT,
    loc VARCHAR(50),
    pincode CHAR(6),
    empno INT,
    CONSTRAINT PK_ADDRESS_ID PRIMARY KEY(id), CONSTRAINT FK_ADDRESS_EMP
FOREIGN KEY ( empno ) REFERENCES emp( empno )
);
INSERT INTO address
( loc, pincode, empno )
VALUES
('Karad','11111', 1),
('Kolhapur','22222', 2),
('A.Nagar','33333', 3),
('Junnar','44444', 4),
('Panchgani','55555', 5);
```

4. meeting : meetno, topic, venue

```
CREATE TABLE meeting
(
    meetno INT,
    topic VARCHAR(50),
    venue VARCHAR(50),
    CONSTRAINT PK_MEETING_MEETNO PRIMARY KEY ( meetno )
);
INSERT INTO meeting
VALUES
( 100, 'Scheduling', 'Directors Cabin'),
( 200, 'Annual Meet', 'Board Room'),
( 300,'App Desgin','Co-Directors Cabin');
```

5. emp_meeting : meetno, empno

```
CREATE TABLE emp_meeting
(
    id INT AUTO_INCREMENT,
    meetno INT,
    empno INT,
    CONSTRAINT PK_EMP_MEETING_ID PRIMARY KEY ( id ),
    CONSTRAINT FK_EM_MEETING FOREIGN KEY ( meetno ) REFERENCES meeting(
meetno ),
    CONSTRAINT FK_EM_EMP FOREIGN KEY ( empno ) REFERENCES emp( empno )
);
INSERT INTO emp_meeting
( meetno, empno )
VALUES
( 100, 3),
( 100, 4),
( 100, 5),
( 200, 1),
( 200, 2),
( 200, 3),
( 200, 4),
( 200, 5),
( 300, 1 ),
( 300, 2 ),
( 300, 5 );
```

- Print meeting topic and emp names attending that meeting

```
SELECT em.meetno, m. topic, em.empno
FROM meeting m INNER JOIN emp_meeting em
ON m.meetno = em.meetno;

SELECT em.meetno, em.empno, e.ename
FROM emp_meeting em INNER JOIn emp e
ON em.empno = e.empno;
```

```
SELECT em.meetno, m. topic, em.empno, e.ename
FROM meeting m INNER JOIN emp_meeting em
ON m.meetno = em.meetno
INNER JOIN emp e
ON em.empno = e.empno;
```

- Print meeting topic and employee names and their addresses

```


SELECT m.meetno, m.topic, em.empno
FROM meeting m INNER JOIN emp_meeting em
ON m.meetno = em.meetno;

SELECT m.meetno, m.topic, em.empno, e.ename
FROM meeting m INNER JOIN emp_meeting em
ON m.meetno = em.meetno
INNER JOIN emp e
ON em.empno = e.empno;

SELECT m.meetno, m.topic, em.empno, e.ename,a.loc
FROM meeting m INNER JOIN emp_meeting em
ON m.meetno = em.meetno
INNER JOIN emp e
ON em.empno = e.empno
INNER JOIN address a
ON e.empno = a.empno;
```

- Print meeting topic and enames and their depts

```
SELECT m.meetno, m.topic, em.empno, e.ename, d.dname
FROM meeting m INNER JOIN emp_meeting em
ON m.meetno = em.meetno
INNER JOIN emp e
ON em.empno = e.empno
INNER JOIN dept d
ON e.deptno = d.deptno;
```

- Print meeting topic and employee names and their depts as well as addresses

```
SELECT m.topic, e.ename, a.loc, d.dname
FROM meeting m INNER JOIN emp_meeting em
ON m.meetno = em.meetno
INNER JOIN emp e
ON em.empno = e.empno
INNER JOIN address a
ON e.empno = a.empno
```

```sql
INNER JOIN dept d
ON e.deptno = d.deptno;
```

- Write a query that lists each order number followed by the name of the customer who made the order.

```sql
SELECT o.onum, c.cnum, c.cname
FROM ORDERS o INNER JOIN CUSTOMERS c
ON o.cnum = c.cnum;
```

- Write a query that gives the names of both the salesperson and the customer for each order along with the order number.

```sql
SELECT c.cname, o.onum, s.sname
FROM CUSTOMERS c INNER JOIN ORDERS o
ON c.cnum = o.cnum
INNER JOIN SALESPEOPLE s
ON o.snum = s.snum;
```

- Write a query that produces all customers serviced by salespeople with a commission above 12%. Output the customer's name, the salesperson's name, and the salesperson's rate of commission.

```sql
SELECT c.cname, s.sname, s.comm
FROM CUSTOMERS c INNER JOIN SALESPEOPLE s
ON c.snum = s.snum
WHERE s.comm > 0.12;
```

- Write a query that calculates the amount of the salesperson's commission on each order by a customer with a rating above 100.

```sql
SELECT c.cname, c.rating, o.onum, s.sname, o.amt * s.comm AS Commision
FROM CUSTOMERS c INNER JOIN ORDERS o
ON c.cnum = o.cnum
INNER JOIN SALESPEOPLE s
ON c.snum = s.snum
WHERE c.rating > 100;
```

- Write a query that produces all pairs of salespeople who are living in the same city.Exclude combinations of salespeople with themselves as well as duplicate rows with the order reversed.

```sql
SELECT s1.sname, s1.city, s2.sname, s2.city
FROM SALESPEOPLE s1 INNER JOIN SALESPEOPLE s2
ON s1.city = s2.city
```

```
 WHERE s1.snum > s2.snum;
--WHERE s1.sname < s2.sname;
```

- Write a query that produces the names and cities of all customers with the same rating as Hoffman.

```
SELECT c1.cname, c1.city, c1.rating
FROM CUSTOMERS c1 INNER JOIN CUSTOMERS c2
ON c1.rating = c2.rati  ng
WHERE c2.cname = 'Hoffman' AND c1.cname != 'Hoffman';
```

Sub Query

- If we write select query inside select query then it is called sub query.
- If we sub query returns single row then it is called single row sub query.
- Find all emps whose sal is more than avg sal of emps.

```
SELECT * FROM emp;
SELECT AVG(sal) FROM emp;

-- 2073.214286
SELECT * FROM emp WHERE sal > 2073.214286;

SET @AVG_SAL = 2073.214286;
SELECT * FROM emp WHERE sal > @AVG_SAL;

SET @AVG_SAL = ( SELECT AVG(sal) FROM emp );
SELECT * FROM emp WHERE sal > @AVG_SAL;

SELECT * FROM emp WHERE sal > ( SELECT AVG(sal) FROM emp );
```

- Find emp with 3rd highest sal

```
SELECT sal FROM emp ORDER BY sal DESC;
SELECT DISTINCT sal FROM emp ORDER BY sal DESC;
SELECT DISTINCT sal FROM emp ORDER BY sal DESC LIMIT 2,1;

SET @SAL = 2975.00;
SELECT * FROM emp WHERE sal = @SAL;

SET @SAL = ( SELECT DISTINCT sal FROM emp ORDER BY sal DESC LIMIT 2,1 );
SELECT * FROM emp WHERE sal = @SAL;

SELECT * FROM emp WHERE sal = ( SELECT DISTINCT sal FROM emp ORDER BY sal
DESC LIMIT 2,1 );
```

- Display emps whose sal is more than sal of all salesman

```
SELECT * FROM emp;
SELECT sal FROM emp WHERE job='salesman';
SELECT MAX(sal) FROM emp WHERE job='salesman';

SET @SAL = 1600.00;
SELECT * FROM emp WHERE sal > @SAL;

SET @SAL = ( SELECT MAX(sal) FROM emp WHERE job='salesman' );
SELECT * FROM emp WHERE sal > @SAL;

SELECT * FROM emp WHERE sal > ( SELECT MAX(sal) FROM emp WHERE
job='salesman' );
```

- If sub query returns multipe rows then such query is called multi row sub query.

```
SELECT * FROM emp WHERE sal > ALL ( SELECT sal FROM emp WHERE
job='salesman' );
```

- IF we want to process multi row sub query then we should use ALL, ANY and IN operator.
- ALL : equivalent to logical AND
- ANY : equivalent to logocal OR
- IN : To check range of values
- Display emps whose sal is more than sal of any salesman;

```
SELECT * FROM emp;
SELECT sal FROM emp WHERE job ='salesman';
SELECT MIN(sal) FROM emp WHERE job ='salesman';

SET @SAL = ( SELECT MIN(sal) FROM emp WHERE job ='salesman' );
SELECT * FROM emp WHERE sal > @SAL;


SELECT * FROM emp WHERE job != 'salesman' AND sal > ( SELECT MIN(sal) FROM
emp WHERE job ='salesman' ) ;

SELECT * FROM emp WHERE job != 'salesman' AND sal >  ANY( SELECT sal FROM
emp WHERE job ='salesman' ) ;
```

- Display depts in which emps are there

```
SELECT * FROM dept;
SELECT deptno from emp;
SELECT distinct deptno from emp;

SELECT * FROM dept
```

```
WHERE deptno = ANY ( SELECT distinct deptno from emp );

SELECT * FROM dept
WHERE deptno IN ( SELECT distinct deptno from emp );
```

Co-Related Sub Query

```
SELECT * FROM dept
WHERE deptno IN ( SELECT deptno from emp );


SELECT * FROM dept d
WHERE deptno IN ( SELECT deptno from emp e WHERE e.deptno = d.deptno );
```

- If sub query depends of outer query then it is called co related sub query.
- Display depts in which emps exists

```
SELECT * FROM dept d
WHERE exists ( SELECT deptno from emp e WHERE e.deptno = d.deptno );
```

- Display depts in which emps doesn't exists

```
SELECT * FROM dept d
WHERE NOT EXISTS ( SELECT deptno from emp e WHERE e.deptno = d.deptno );
```

- Delete emps from emp table whose sal is less than avg(sal) of emp.

```
SELECT AVG(sal) from emp;

DELETE FROM emp
WHERE sal < ( SELECT AVG(sal) from emp );
-- Not OK
```

- We can not perform DML operations on table which is used in sub query.

```
DELETE FROM dept d
WHERE NOT EXISTS ( SELECT deptno from emp e WHERE e.deptno = d.deptno );
-- OK
```