Joins :
  cross join
  inner join

with the help of alias support:

>select e.eid, e.ename, e.salary, d.dname from dept d INNER JOIN emp e ON d.did = e.deptid AND d.dname = 'HR';

another syntaxt of INNER JOIN (without INNER JOIN command):

>select e.eid, e.ename,e.salary,d.dname from dept d,emp e where e.deptid = d.did AND d.dname = 'HR';

Outer Join:
==========

a. left outer join:
b. right outer join
c. full outer join

a. left outer join:

--to get the unmatched records from the left table use the left outer join (it will show the full details of left table and null value for right table of any unmatched records)

>select e.eid, e.ename, e.salary, d.dname from dept d LEFT OUTER JOIN emp e ON d.did = e.deptid;

eid	ename	salary	dname
1000	ravi	80000	Sales
1001	amit	82000	Sales
1003	dinesh	78000	Sales
1007	ravi	50000	Sales
1006	rakesh	82000	Marketing
1002	mukesh	84000	Accounts
1004	manoj	72000	Accounts
1005	chandan	62000	HR
NULL	NULL	NULL	IT

--here since IT dept we don't have any emp then all the dept details will be printed and unmatched emp from that dept will have null value.

Right Outer Join:

--to get the unmatched records from the right table use the right outer join (it shows all the details from the right table and null values for the left table)

>select e.eid, e.ename, e.salary, d.dname from dept d RIGHT OUTER JOIN emp e ON d.did = e.deptid;

```
| eid  | ename   | salary | dname     |
+------+---------+--------+-----------+
| 1000 | ravi    |  80000 | Sales     |
| 1001 | amit    |  82000 | Sales     |
| 1002 | mukesh  |  84000 | Accounts  |
| 1003 | dinesh  |  78000 | Sales     |
| 1004 | manoj   |  72000 | Accounts  |
| 1005 | chandan |  62000 | HR        |
| 1006 | rakesh  |  82000 | Marketing |
| 1007 | ravi    |  50000 | Sales     |
| 1008 | praveen |  50000 | NULL      |
| 1009 | mohit   |  55000 | NULL      |
+------+---------+--------+-----------+
```

Full outer join:

--it displays the null values from both side for all the unmatched records.

Note: Mysql does not support the full outer join.

--in order to use full outer join in mysql, we need to use union of Left outer join and right outer join

```
>select e.eid, e.ename, e.salary, d.dname from dept d LEFT OUTER JOIN emp e ON d.did = e.deptid
union
select e.eid, e.ename, e.salary, d.dname from dept d RIGHT OUTER JOIN emp e ON d.did = e.deptid;
```

Self Join:
=======

--here we use joining a table to itself.

--here joining condition column must belongs to the same data type.

Note: - if we want ot compare two tables same columns values then we use
use INNER JOIN where as if we want to compare 2 diff column value within a single
table then we should use self join.

--whenever a table contains hirarical data then only we allowed to use self join.

emp ------> manager
student ----> monitor

--when we use the self join, we must take the support of alias.

```
create table employee
(
eid int primary key,
ename varchar(12),
salary int,
mgrid int
);
```

```
mysql> insert into employee values(10, 'ram', 80000, null); // Ram does not have any manager

mysql> insert into employee values(12, 'ravi', 82000, 10); // ravi manager is Ram

mysql> insert into employee values(13, 'amit', 80000, 10); //amit manager is Ram

mysql> insert into employee values(14, 'sunil', 82000, 12); //sunil manager is ravi
```

```
Q/- display the employee name and their manager name?

> select e1.ename EMPLOYEE, e2.ename MANAGER from employee e1, employee e2 where e1.mgrid = e2.eid;


Getting the data from 3 tables:
========================

create table course
(
cid int primary key,
cname varchar(12),
fee int
);

create table teacher
(
tid int primary key,
tname varchar(12),
taddress varchar(12),
course_id int unique,
foreign key (course_id) references course(cid)
);

create table student
(
roll int primary key,
sname varchar(12),
saddress varchar(12),
course_id int,
foreign key (course_id) references course(cid)
);

inserting records to the course table:
-----------------------------------------------

mysql> insert into course values(1, 'Java', 8000);

mysql> insert into course values(2, 'Spring', 8200);

mysql> insert into course values(3, 'SQL', 4200);

mysql> insert into course values(4, 'SpringBoot', 9000);

inserting records to the teacher table:
-----------------------------------------------

insert into teacher values (10,'ratan','patna',1);
insert into teacher values (12,'praveen','delhi',1);
 insert into teacher values (13,'aanand','delhi',3);

inserting records to the student table:
-----------------------------------------------

mysql> insert into student values(1000,'amit','delhi',1);
mysql> insert into student values(1001,'ravi','delhi',1);
mysql> insert into student values(1002,'sunil','delhi',2);
mysql> insert into student values(1003,'suresh','delhi',3);


Q/ get the student details who is taught by ratan?


>select s.roll, s.sname, s.saddress, t.tname, c.cname
from
```

```
course c INNER JOIN teacher t INNER JOIN student s
ON
c.cid = t.course_id AND c.cid = s.course_id AND t.tname ='ratan';


Subqueries:
==========

--a query inside another query is called a subquery or nested query.

--subqueries are use to retrieve the data from single table or multiple tables based
on more than one step process.

--here outer query is called as parent query and inner query is called as child query.

--child query will execute first then only parent query will be executed.

child query :  it provides the value/data to the parent query.
parent query : it receives the value/data from the child query.


rules:

1. In child query we can not use order by clause but inside the parent query we can use order by.

2. group by clause we can use both inside the parent and child query as well.


Subqueries we can categories into following categories:
------------------------------------------------------------------

1. single row and single col SQ: scalar value SQ.

2. multiple row and single column SQ

3. multiple column SQ.


1. single row and single col SQ: scalar value SQ:
-----------------------------------------------------

--here child query will return only a single value to the parent query.

--here mostly same col name which is there inside the 'where' clause of the parent query , will be
there inside the 'select' clause of the child query.

Q/- WAQ to display emp details who is working in HR dept.

using join:
>select e.eid, e.ename, e.salary, e.deptid from dept d INNER JOIN emp e ON d.did = e.deptid AND
d.dname = 'HR';

using SQ:

> select * from emp where deptid = (select did from dept where dname = 'HR');

Q/- WAQ to display emp details who is working with 'amit';

> select * from emp where deptid = (select deptid from emp where ename ='amit');

Q/- WAQ to display emp details who is getting more salary than avg of emp table.

> select * from emp where salary >  (select avg(salary) from emp);

Q/- WAQ to display second highest salaried employee;
```

first highest salaries employee:

```
>select * from emp where salary = (select max(salary) from emp);
```

second highest salaries employee:

```
>select * from emp where salary = (select max(salary) from emp where salary < (select max(salary)
from emp));
```

Q/- WAQ to display the details of employees who is working under ram;

```
> select * from employee where mgrid = (select eid from employee where ename = 'ram');
```

2. multiple row and single col SQ:
==========

--in this, child query will return multiple rows and a single column to the parent query.

--In this case inside the parent query we should use one of the following operators:

IN
ANY
ALL

ex:

```
 select * from emp where salary IN (select salary from emp where eid > 1002);
                                (10000, 80000, 70000)

 select * from emp where salary = ANY(select salary from emp where eid > 1002);
                                (10000, 80000, 70000)

 select * from emp where salary > ANY(select salary from emp where eid > 1002);
                                (10000, 80000, 70000)

 select * from emp where salary < ANY (select salary from emp where eid > 1002);
                                (10000, 80000, 70000)



select * from emp where salary < ALL (select salary from emp where eid > 1002);
                                (10000, 80000, 70000)
```

= any()

IN: it checks equal to any number in the list (using OR)

ANY: it compares any value in the list

ALL : it compares all the values in the list

salary > ANY (----) : here it checks salary should be greater than any of the 4 values in the list

salary > ALL (----) : here it checks salary should be greater than all of the 4 values in the list

ex:

any:

< any() : less than any : less than maximum.

>any (): greater than any : greater than minimum

= any() equal to any : it is equal to IN operator.

all:

all(10, 20, 30 ,40)

< all() : less than all : less than minimum.

>all(): greater than all : greater than maximum

= all(): equal to all : It is meaningless (because one value can not be equal to multiple values)

Q/- WAQ to display the emp who is getting max salary in each dept ?

> select * from emp where salary IN (select max(salary) from emp group by deptid);

3. multiple col SQ:
==========

--if we try to compare muliple column values of the child query with the multiple column of the parent query then we use this type of query.

syn:

select * from tb_name where (col1, col2, ...) IN (select col1, col2,... from table where condition)

ex:

Q/- WAQ to display the emp details whose salary and deptid is matched with the salary and deptid of ravi.

>select * from emp where (salary, deptid) IN (select salary, deptid from emp where ename = 'ravi');

SQ in DML: (insert, update, delete):
==========

SQ in insert:

```
mysql> create table x1(id int, ename varchar(12));
mysql> insert into x1 (select eid, ename from emp);
```

>insert into x1 values(501, (select ename from emp where eid=1001));

SQ in update:
---

here SQ is allowed inside where clause or Set clause also

> update x1 set ename = (select ename from emp where eid = 1001) where id = (select eid from emp where ename ='chandan');

SQ in delete:

```
delete from x1 where id = (select eid from emp where ename ='chandan');


100:

Autoincrement in mysql
==================

--mysql support aut_increment where as oracle db uses squence comcept to auto generate the ID field.

create table student1
(
roll int primary key auto_increment,
name varchar(12),
marks int
);

>alter table student1 auto_increment =1000;


Limit:
========

>select * from emp LIMIT 4;

--getting records from 3 -6th row

>select * from emp LIMIT 4 OFFSET 2;
```