Date.....

SQL JOINTS

→ join is used to combine rows from two or more tables, based on related columns between them.

Eg →

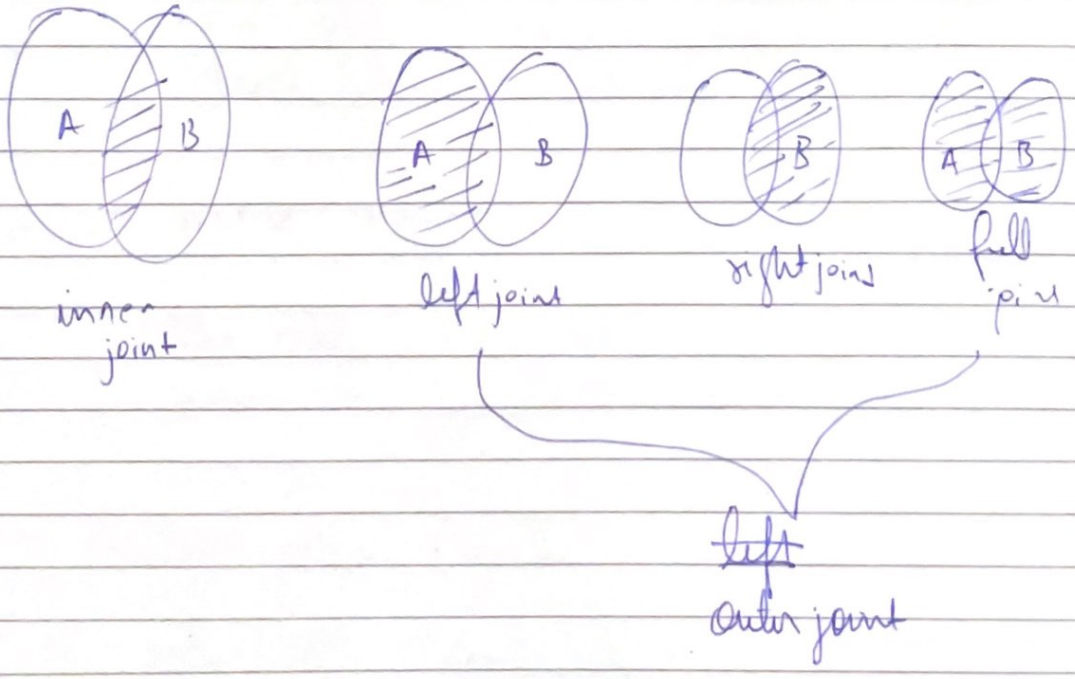
Employee

id	name
101	
102	

salary.

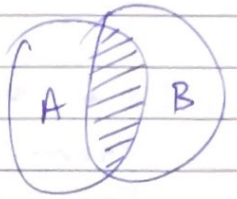
id	salary
102	
103	

venn diagrams → types of joint →



Date.....

inner joint → Return records that ~~having~~ have matching values in both tables.



Syntax.

```
Select column(s)
from table A
inner joint table B.
on table A. col_name = table B col. name;
```

inner joint Eg →

student.

Student id	name
101	adam.
102	bob
103	Casey

course.

Student_id	course
102	English
105	math
103	Sci
107	CS.

Result.

student id	name	course
102	bob	english
103	casey	Science.

```
Select *
from student
inner joint course
on
student.student_id =
Course.student_id.
```

Date.....

Alias → means alternate name.

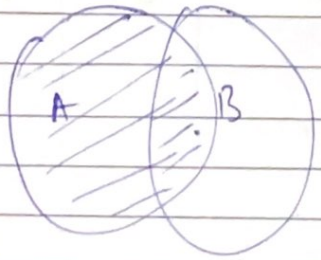
```
Select *
from student as s
inner join course as c
on student.id = course.id;
```

or

```
on s.id = c.id;
```

LEFT JOIN ...

→ Returns all record from the left table, and the matched records from the right table.



syntax.

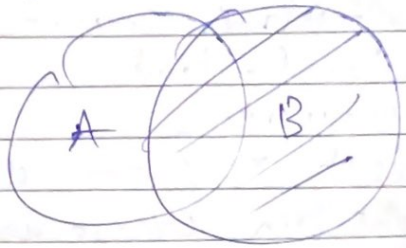
```
select column(s)
from ~~column~~ table A
left join table B
on table A.col_name = table B.
                      colname;
```

Date.....

Result.

101	Adam	null
102	bob	english
103	casey	science.

Right join.



Syntax.

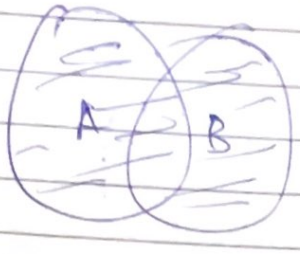
```
select columns(s)
from table B.
Right join table A
on table B. id = table A. id ;
```

~~left join~~ full join.

No Option of full join in mysql, that's why we use Union or indirect method.

left join
UNION
Right join.

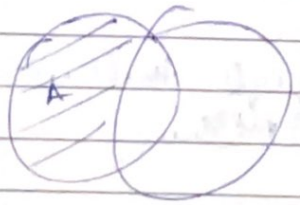
Date.....



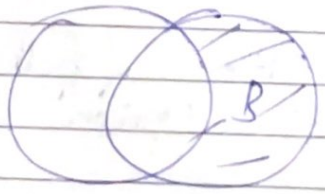
left join
Union
Right join.

```
Select * from student as a
left join course as b
on a.id = b.id
  UNION
select * from student as a
right join course as b
  On a.id = b.id;
```

Extra join



left exclusive join



right exclusive join.

Memo

Date

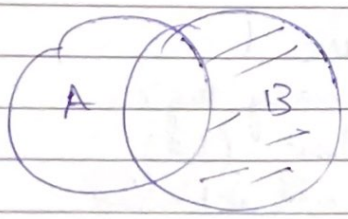
Date Date.....

→

Left exclusive join.

```
select *
from student as s
left join course as c
on s.studentid = c.student_id.
where b.student-id is NULL;
```

Right exclusive join




```
select *
from Student as a
right join course as b
on a.student_id = b.student_id
where a.studentid is NULL;
```

Date.....

self join

Regular join but the table is join within itself

syntax:

```
Select column(s)
from table as a
join table as b.
on a.col-name = b.col-name;
```

⇒ create table employee (
id int primary key,
name var char (50),
manager-id int
);

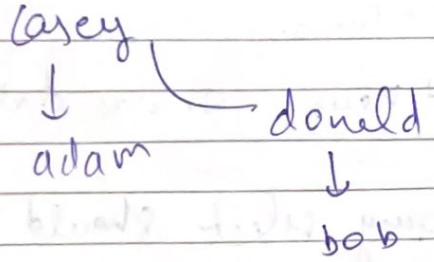
insert into employee (id, name, manager_id)
values
(101, "adam", 103),
(102, "bob", 104),
(103, "casey" NULL),
(104, "donald", 103);

id	name	manager_id
101	adam	103
102	bob	104
103	casey	null
104	donald	103

Teacher's Sign .....

Self Join Date.....

```
select *
from employee as a
    join employee as b
        on a.id = b.manager_id;
```

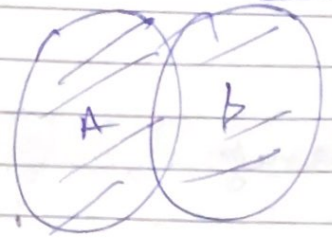


or.

```
Select a.name as manager name, b.name
    from employee as a
    join employee as b
    on a.id = b.manager_id;
```

Date.....

UNION



Union will combine both, removes the duplicates.

to use it →

→ every select should have same no of column.

→ column must have similar data types

→ columns in every select should be in same order.

Syntax:

```
select column(s) from table A
UNION
select column(s) from table B.
```

→ Select name from employee
UNION
select name from employee;

Date.....

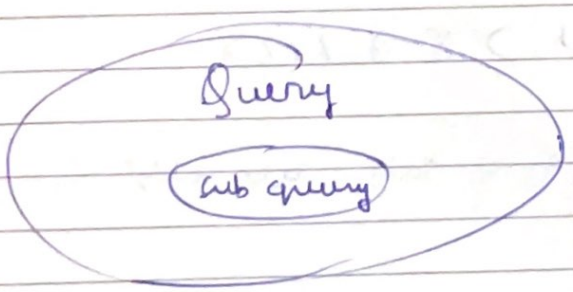
UNION ALL → gives us all the duplicates.

```
Select name from employee
UNION ALL
Select name from employee;
```

SQL SUB QUERIES.

Sub query
Nested query
or inner query → is a query within another sql query.

It involves 2 statement.



we write sub query in
→ select
from
where

Syntax:
```
Select subquery column(s)
from table name
where col. name operator
(sub query);
```


Teacher's Sign .....

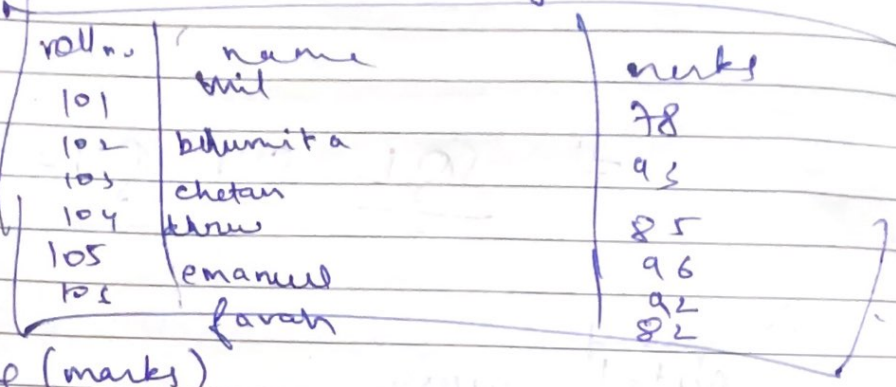
Date.....

SQL sub Query Eg →

Q.) get names of all student who scored more than class avg

Step 1 → calc. Avg

Step 2 → compare marks > avg.



roll no.	name	marks
101	mit	78
102	bhumika	95
103	chetan	
104	dhruv	85
105	emanuel	96
106	farah	92
		82

```
select avg (marks)
  from student;

  select name, marks
    from student
      where marks > 87.6667;
```

instead of writing this, we use sub query.

```
select name, marks
  from student
  where marks > (select Avg (marks) from stu);
```

Spiral

Teacher's Sign .....

Date.....

Q.) find names of all student who have even roll numbers.

```
Select name, roll_no
from sub
where roll_no in (select rollno
                  from sub
                  where roll no %2 = 0);
```

SQL sub query

Example with from.

Q.) find max marks from student of delhi

SQL subquery by select

Date.....

MY SQL Views

views are virtual table

table	view
↓	↓
real	temperory tables.
↓	
real data	

create view view1 as
select rollno, name from student;

select * from view1;

or

select * from view1
where marks > 90;

DROP view view1;