create database mysql_practise;

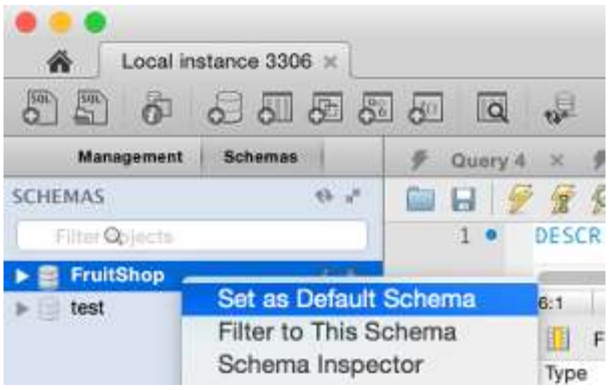
show databases; ➔ display all database

use mysql_practise;

drop table marks; ➔ delete table marks

```
mysql> \! Cls  ➔ clear screen in command line prompt
```

SETTING DEFAULT



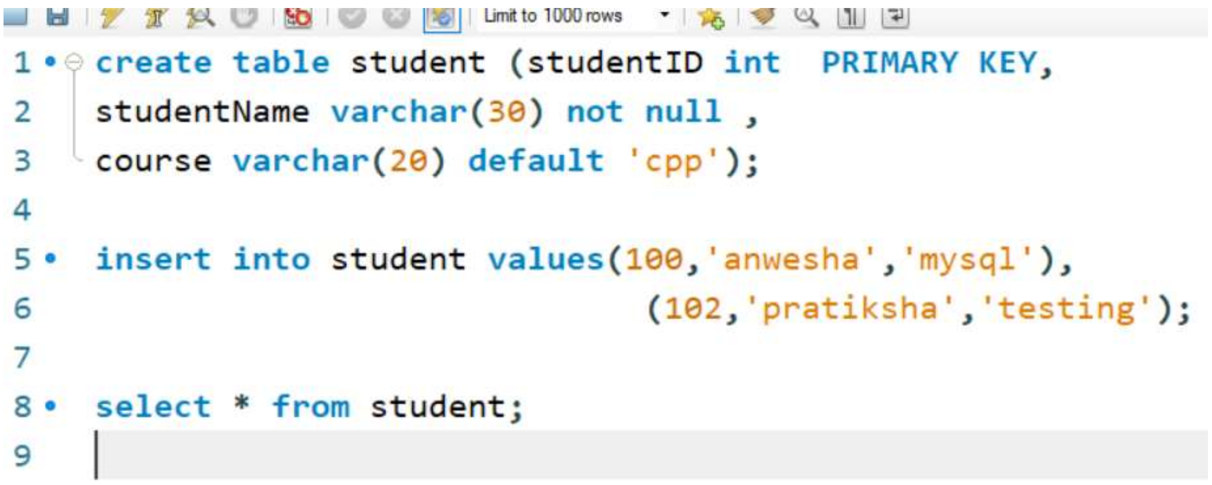
constraints :

MySQL constraints are statements that can be applied at the column level or table level to specify rules for the data that can be entered into a column or data table,

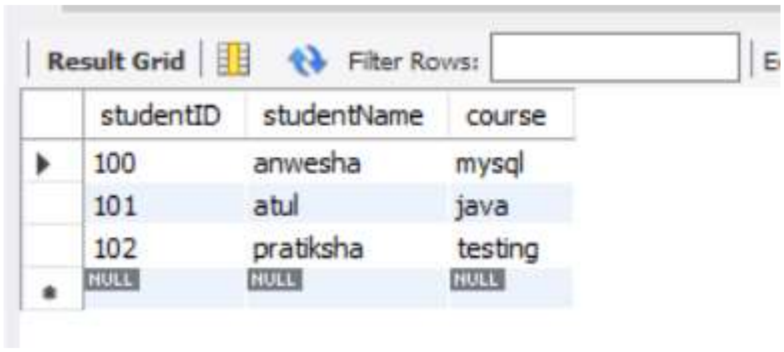
i.e constraints are basically limitations or restrictions on the type of data and hence they ensure reliability

SQL Constraint	**Function**
NOT NULL	It ensures that a column does not accept NULL values.
CHECK	It ensures that a column accepts values within the specified range of values.
UNIQUE	It ensures that a column does not accept duplicate values.
PRIMARY KEY	It uniquely identifies a row in the table. It is a combination of NOT NULL and UNIQUE constraints.
FOREIGN KEY	It is like a primary key constraint only. But it uniquely identifies a row in another table.

DEFAULT	It ensures that the column sets a default value for empty records.



```sql
create table student (studentID int  PRIMARY KEY,
studentName varchar(30) not null ,
course varchar(20) default 'cpp');

insert into student values(100,'anwesha','mysql'),
                                    (102,'pratiksha','testing');

select * from student;

```

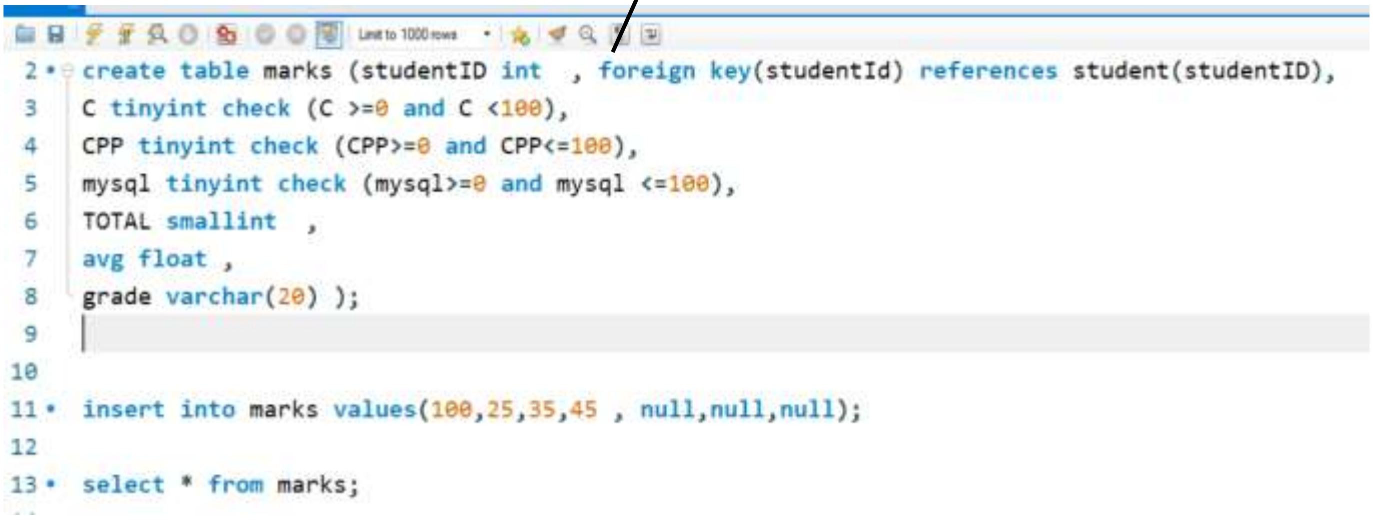


studentID	studentName	course
100	anwesha	mysql
101	atul	java
102	pratiksha	testing
NULL	NULL	NULL

Create marks table

foreign key >> on studentId of marks table

and referecd to student table >> studentD col



```sql
create table marks (studentID int  , foreign key(studentId) references student(studentID),
C tinyint check (C >=0 and C <100),
CPP tinyint check (CPP>=0 and CPP<=100),
mysql tinyint check (mysql>=0 and mysql <=100),
TOTAL smallint  ,
avg float ,
grade varchar(20) );


insert into marks values(100,25,35,45 , null,null,null);

select * from marks;
```

In above table TOTAL , AVG, GRADE colums are empty null because we compute it automaticaly.

If we want to add values for only specific coloms then specifiy that columns name as

Insert into marks (studentid,C,CPP, mysql) values (105,50,50,60)

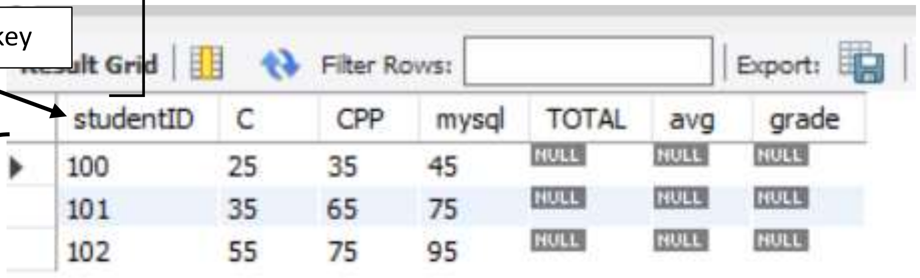
Colom name values

The foreign key is used to link one or more than one table together. It is also known as the **referencing** key

Marks table here foreign key ➔ referned to column **studentID** of student table

Foreign key says allows only values of referencd column

Foreign key

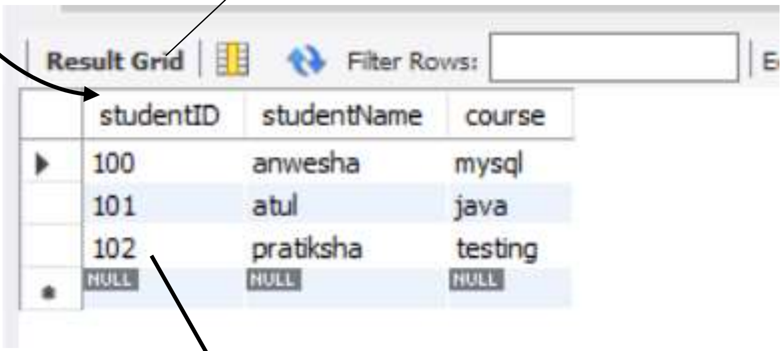


studentID	C	CPP	mysql	TOTAL	avg	grade
100	25	35	45	NULL	NULL	NULL
101	35	65	75	NULL	NULL	NULL
102	55	75	95	NULL	NULL	NULL

2nd table with foreign key

Referced to studentID col of student table

Primary key (does not allow duplicate)



studentID	studentName	course
100	anwesha	mysql
101	atul	java
102	pratiksha	testing
NULL	NULL	NULL

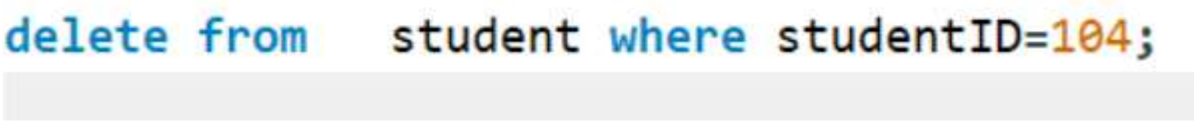
1st table student with primry key

Can not delete because it is referencd for (foreign key) marks table.

In simpley foreign key applied on that column , in which it say only allowed referencd colums data.

Delete

If normal not foreign



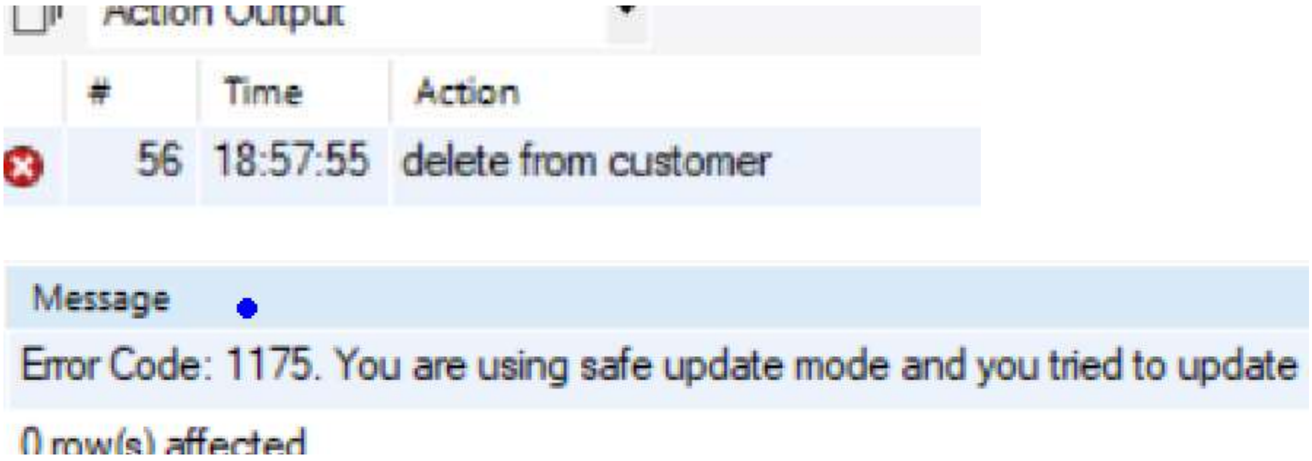
```
delete from   student where studentID=104;
```

Now

<u>Delete form customer</u>;   // this command deletes all record also delete the table structure

But to run this command you need to change setting because by default mysql run in safe mode i.e we change to disable mode of safe

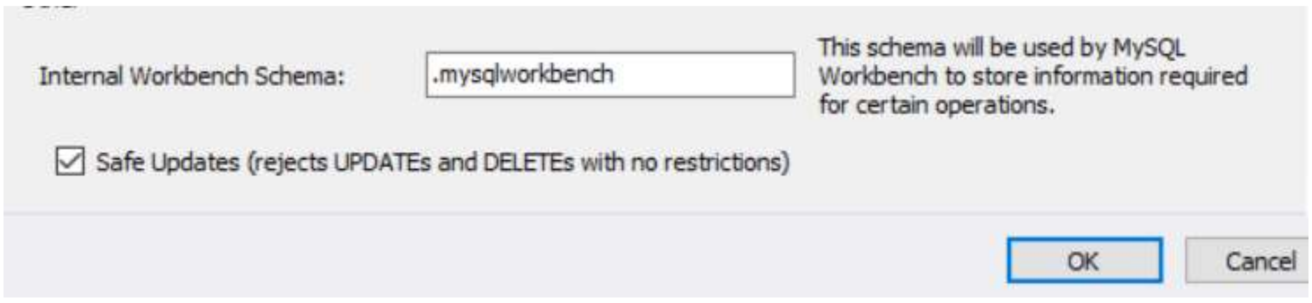
If we try this error message display



```
Action Output
   #    Time       Action
   56   18:57:55   delete from customer

Message
Error Code: 1175. You are using safe update mode and you tried to update
0 row(s) affected
```

To disable safe mode

Edit >> reference >> SQL editor   and reconnect workbench



Internal Workbench Schema: .mysqlworkbench

This schema will be used by MySQL Workbench to store information required for certain operations.

☑ Safe Updates (rejects UPDATEs and DELETEs with no restrictions)

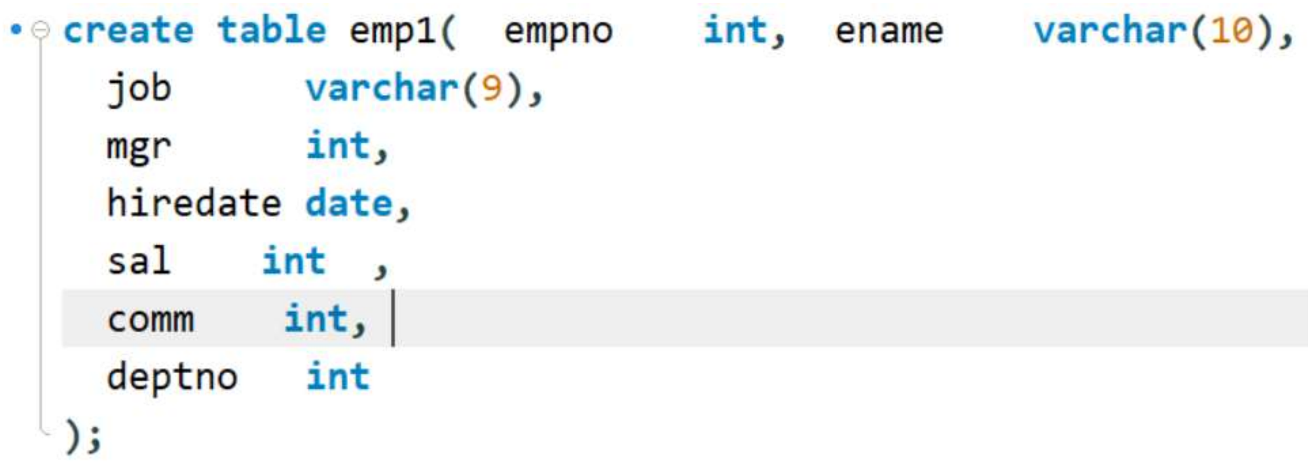
Truncate command

Internally data is stored in pages , size of each page 8kb

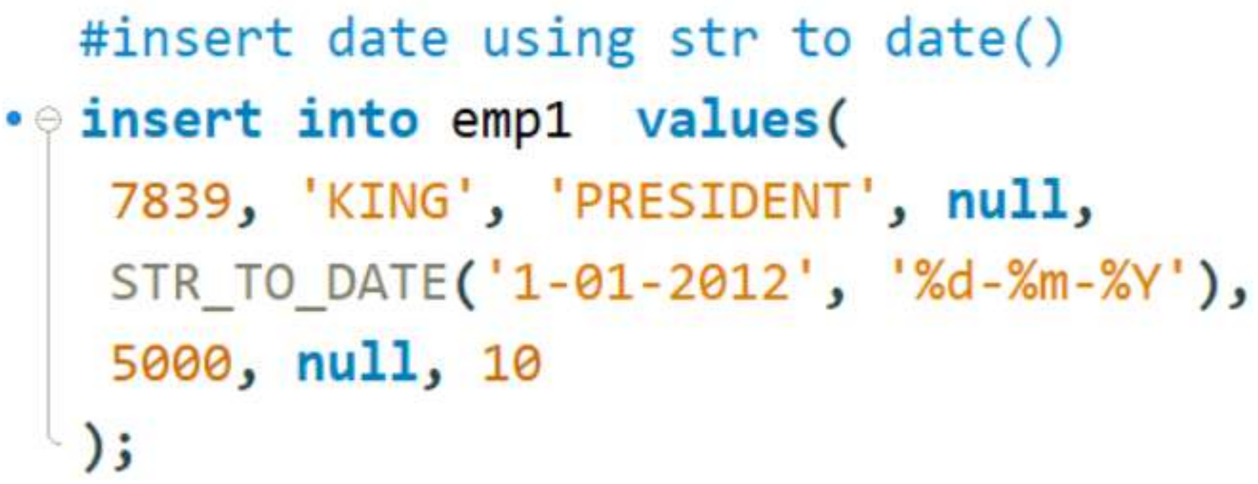
Continious 8 pages is called extents

Truncate command does not support where clause hence used to delete all rows

Emp table >> inserting date in table



```
create table emp1(   empno    int,   ename    varchar(10),
  job      varchar(9),
  mgr      int,
  hiredate date,
  sal    int   ,
  comm    int,
  deptno   int
);
```

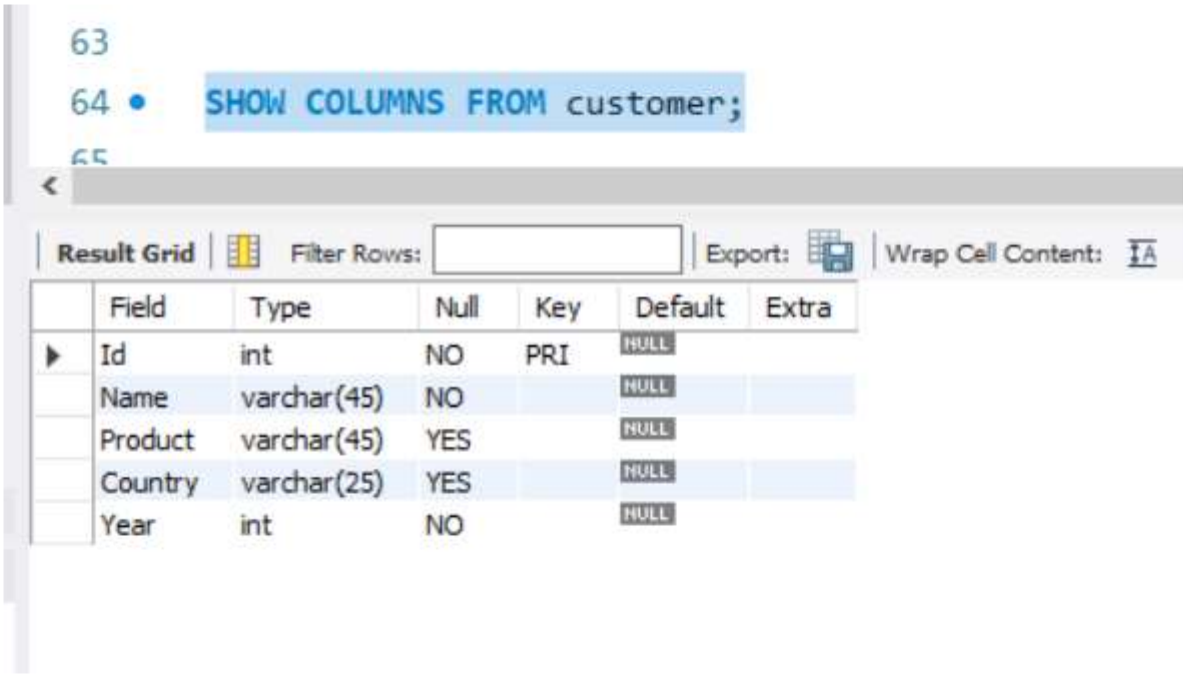


```
#insert date using str to date()
insert into emp1  values(
 7839, 'KING', 'PRESIDENT', null,
 STR_TO_DATE('1-01-2012', '%d-%m-%Y'),
 5000, null, 10
);
```

Alterning Table Structure

- To add additional columns
- delete column
- chage data types

first see structure of table using show command

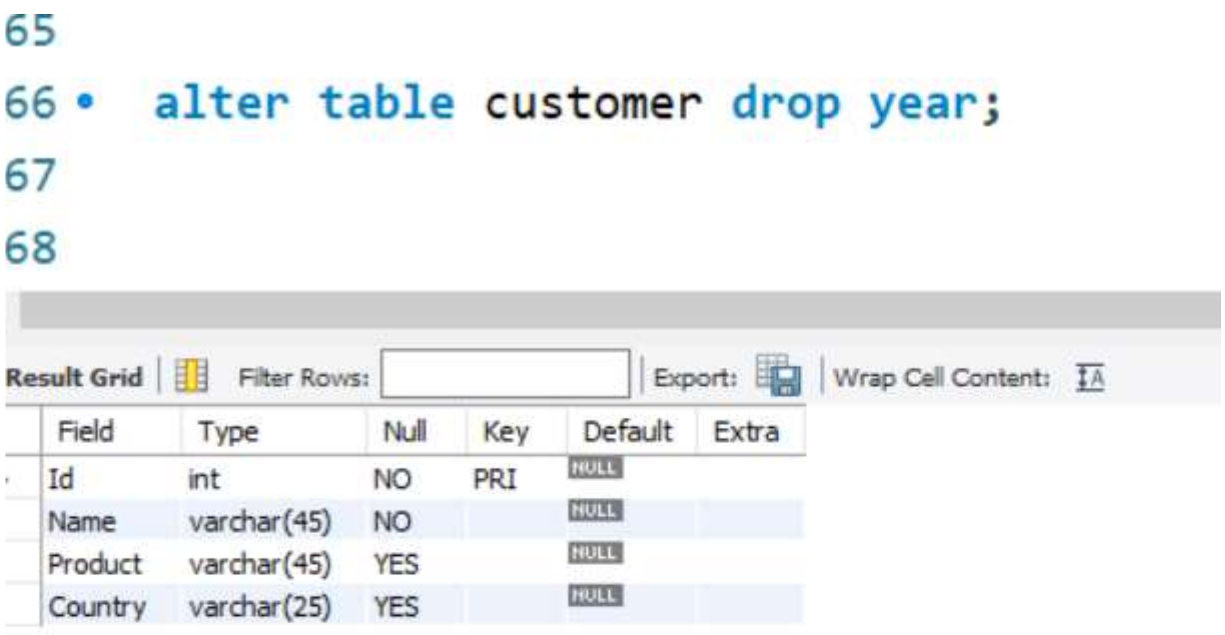


```
SHOW COLUMNS FROM customer;
```

Field	Type	Null	Key	Default	Extra
Id	int	NO	PRI	NULL	
Name	varchar(45)	NO		NULL	
Product	varchar(45)	YES		NULL	
Country	varchar(25)	YES		NULL	
Year	int	NO		NULL	

The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation

Drop

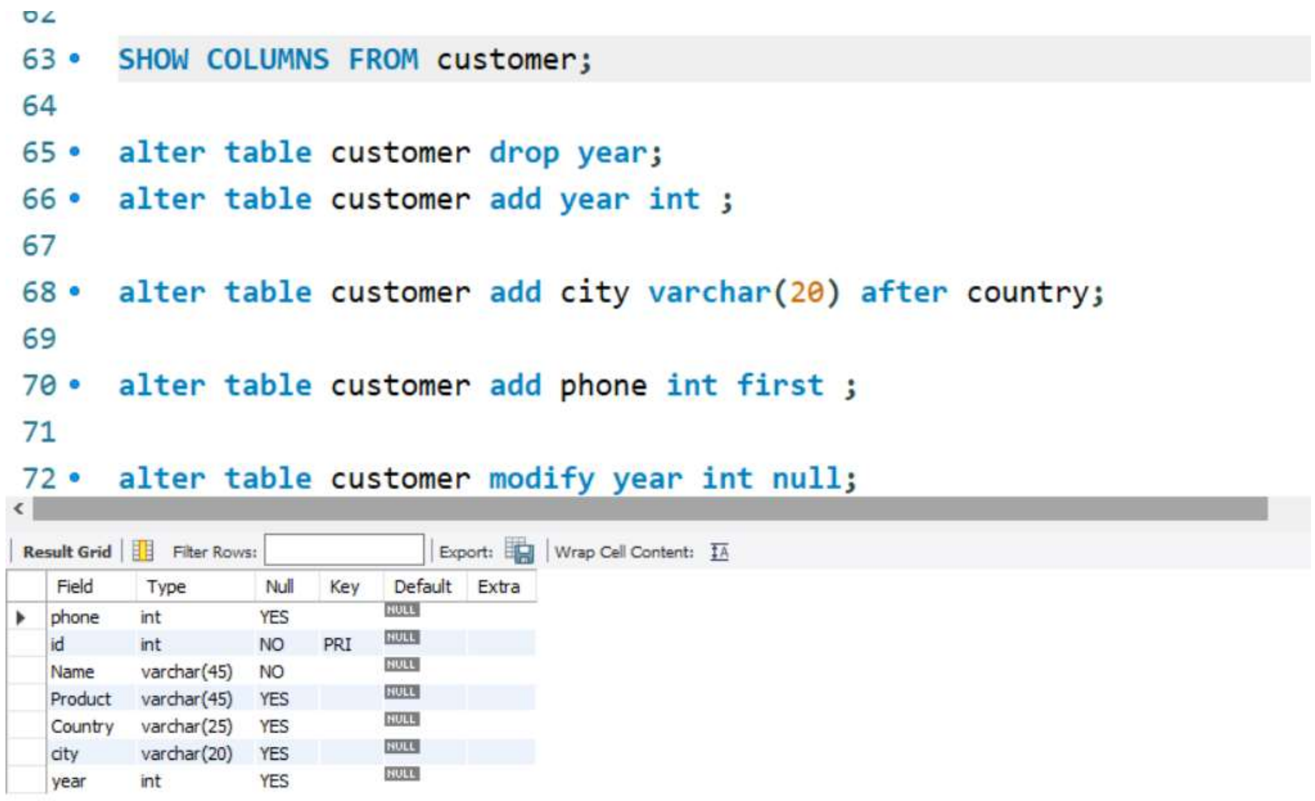


```
alter table customer drop year;
```

Field	Type	Null	Key	Default	Extra
Id	int	NO	PRI	NULL	
Name	varchar(45)	NO		NULL	
Product	varchar(45)	YES		NULL	
Country	varchar(25)	YES		NULL	

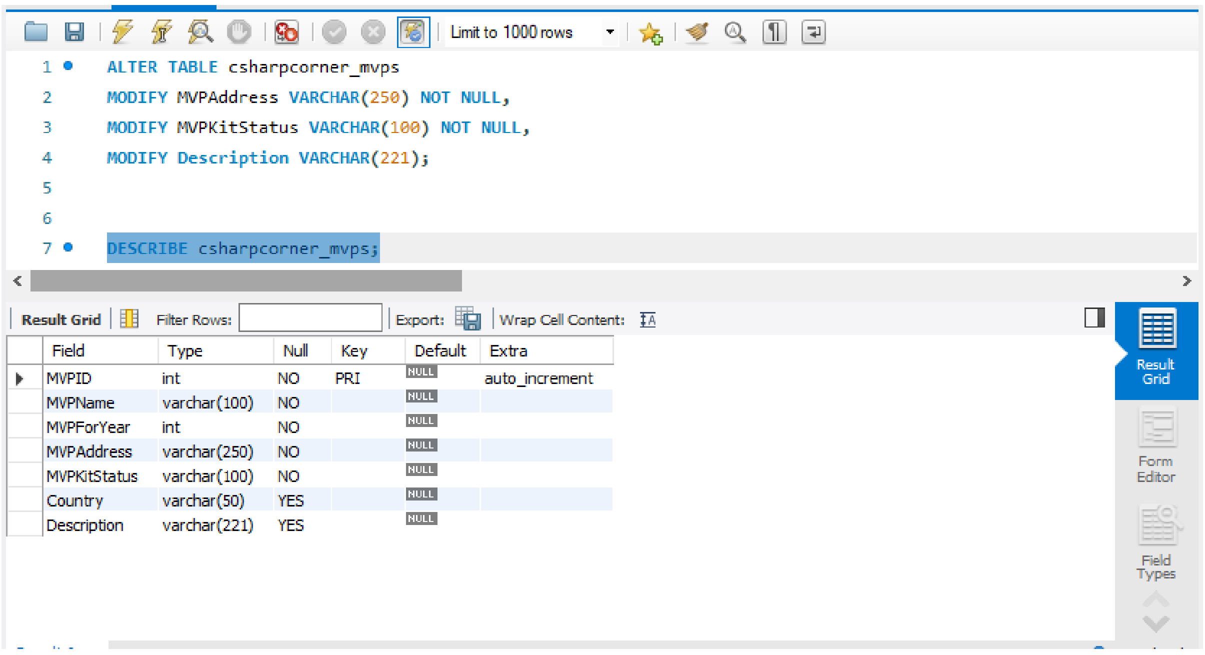
Year column is dropped or deleted

A DROP clause will not work if the column is the only one left in the table.



```
SHOW COLUMNS FROM customer;

alter table customer drop year;
alter table customer add year int ;

alter table customer add city varchar(20) after country;

alter table customer add phone int first ;

alter table customer modify year int null;
```

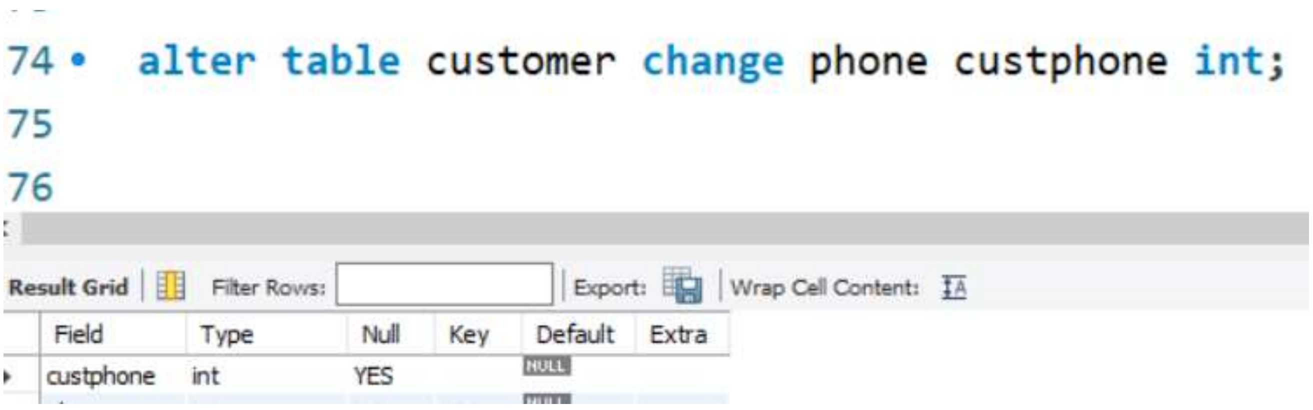
Field	Type	Null	Key	Default	Extra
phone	int	YES		NULL	
id	int	NO	PRI	NULL	
Name	varchar(45)	NO		NULL	
Product	varchar(45)	YES		NULL	
Country	varchar(25)	YES		NULL	
city	varchar(20)	YES		NULL	
year	int	YES		NULL	



```
ALTER TABLE csharpcorner_mvps
MODIFY MVPAddress VARCHAR(250) NOT NULL,
MODIFY MVPKitStatus VARCHAR(100) NOT NULL,
MODIFY Description VARCHAR(221);


DESCRIBE csharpcorner_mvps;
```

Field	Type	Null	Key	Default	Extra
MVPID	int	NO	PRI	NULL	auto_increment
MVPName	varchar(100)	NO		NULL	
MVPForYear	int	NO		NULL	
MVPAddress	varchar(250)	NO		NULL	
MVPKitStatus	varchar(100)	NO		NULL	
Country	varchar(50)	YES		NULL	
Description	varchar(221)	YES		NULL	

Change column name



```
alter table customer change phone custphone int;
```

Field	Type	Null	Key	Default	Extra
custphone	int	YES		NULL	

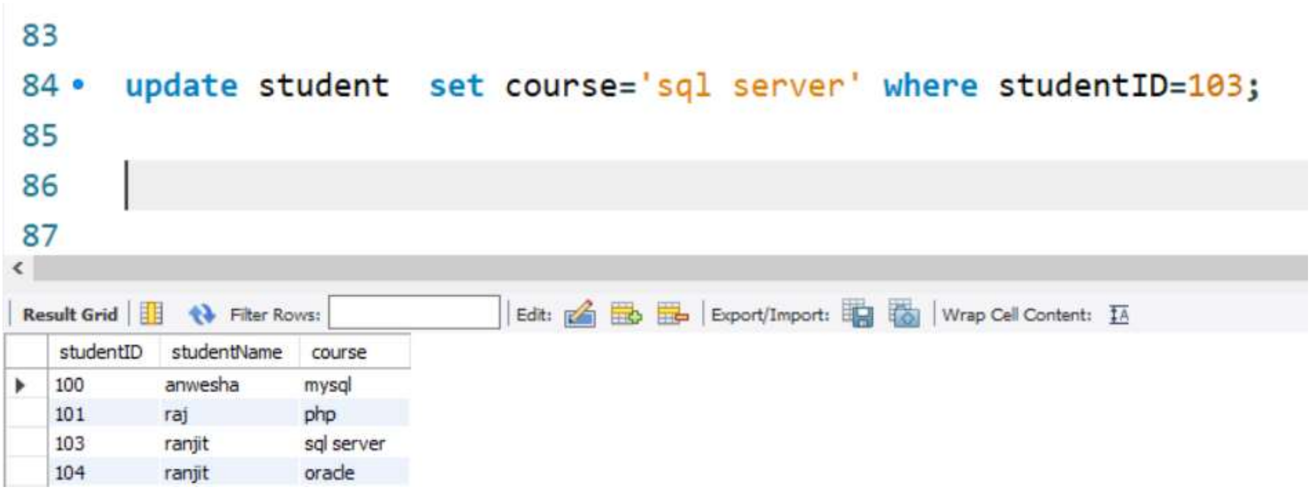
It changes phone >> to custphone

Update :

Note while update use auto safe mode on or off as need

set SQL_SAFE_UPDATES=0; → Set false / OFF

set SQL_SAFE_UPDATES=1; → set True /ON



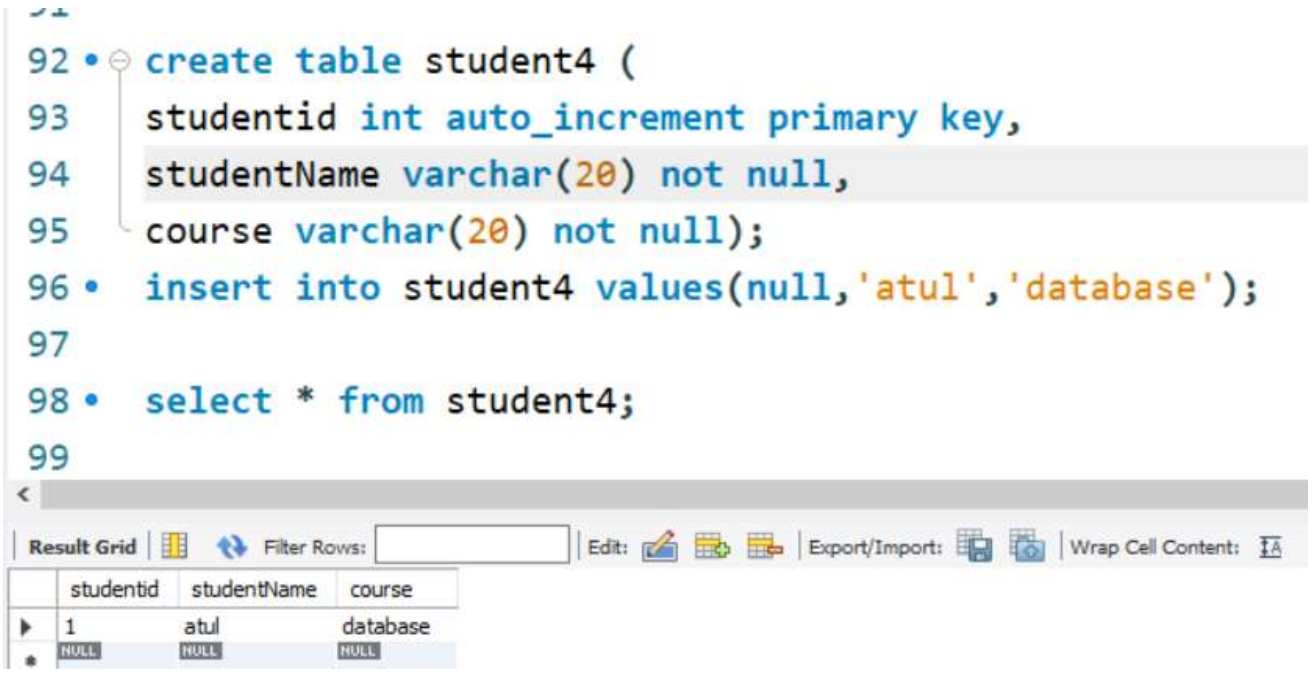
```
update student  set course='sql server' where studentID=103;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

```
  studentID  studentName  course
▶ 100        anwesha      mysql
  101        raj          php
  103        ranjit       sql server
  104        ranjit       oracle
```

Auto increment

Default start 1 increment by 1

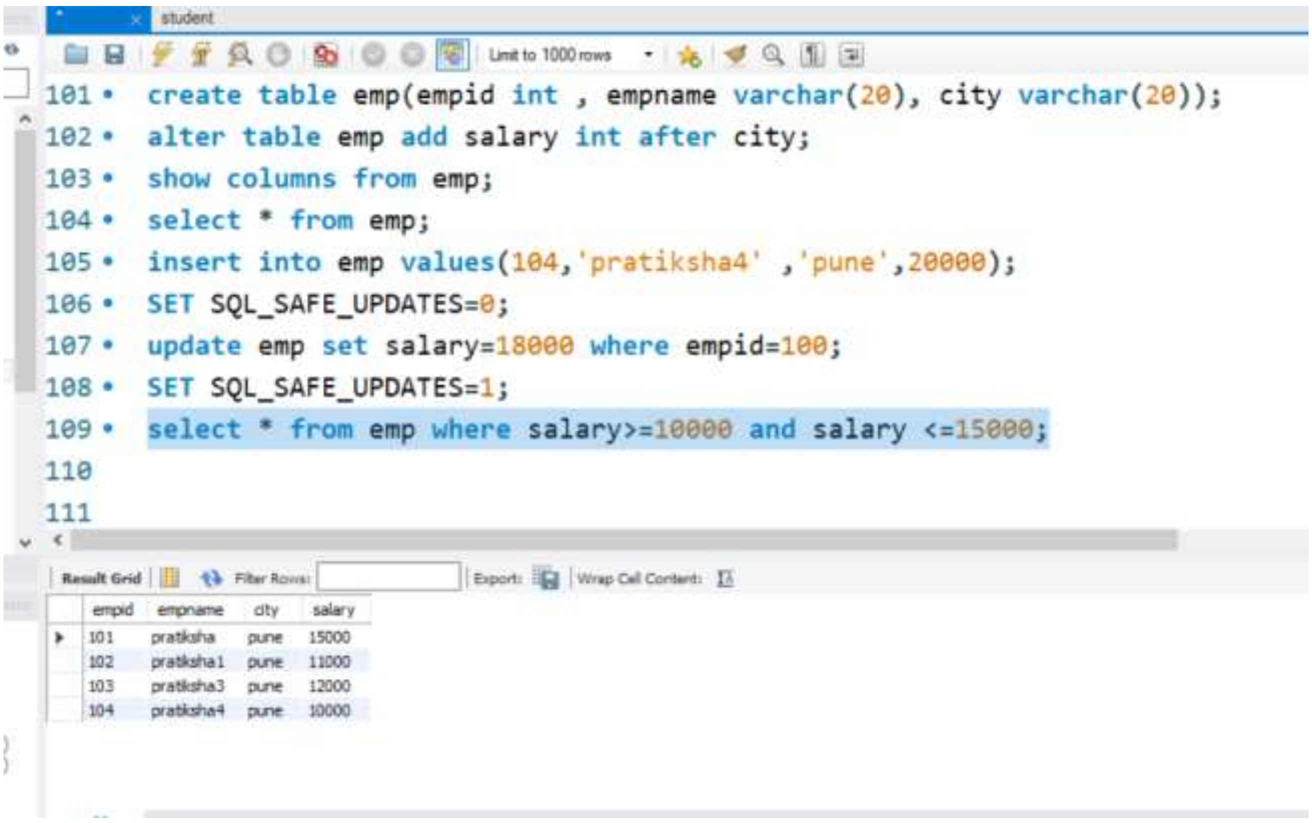


```
create table student4 (
studentid int auto_increment primary key,
studentName varchar(20) not null,
course varchar(20) not null);
insert into student4 values(null,'atul','database');

select * from student4;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

```
  studentid  studentName  course
▶ 1          atul         database
* NULL       NULL         NULL
```

Where Clause



```
create table emp(empid int , empname varchar(20), city varchar(20));
alter table emp add salary int after city;
show columns from emp;
select * from emp;
insert into emp values(104,'pratiksha4' ,'pune',20000);
SET SQL_SAFE_UPDATES=0;
update emp set salary=18000 where empid=100;
SET SQL_SAFE_UPDATES=1;
select * from emp where salary>=10000 and salary <=15000;
```

empid	empname	city	salary
101	pratiksha	pune	15000
102	pratiksha1	pune	11000
103	pratiksha3	pune	12000
104	pratiksha4	pune	10000

In above when we use update command for set salary

Safe mode error will display

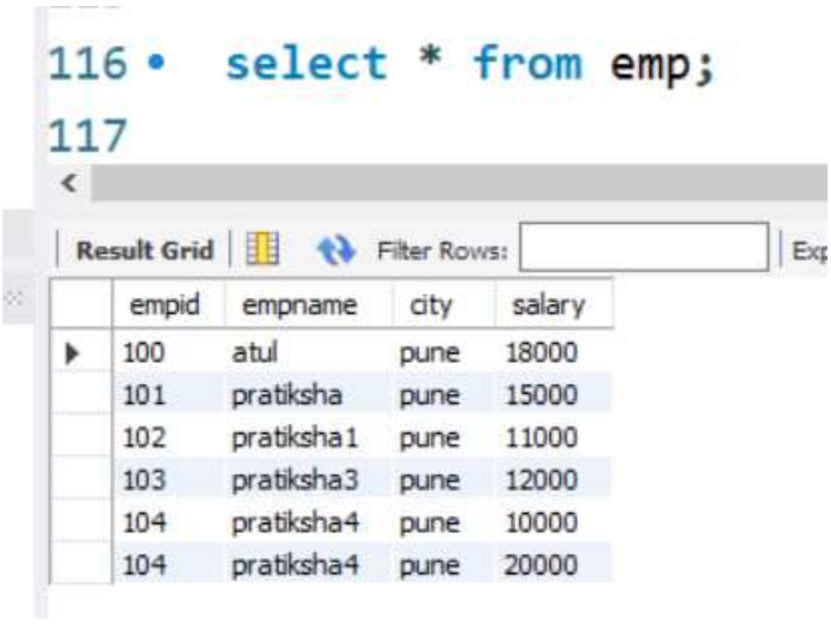
Hence we use SET SQL_SAFE_UPDATES=0 //OFF the mode

SET SQL_SAFE_UPDATES=1 // ON the mode

ORDER BY >> Sort the row

The MySQL ORDER BY clause is used to sort the records in your result set.

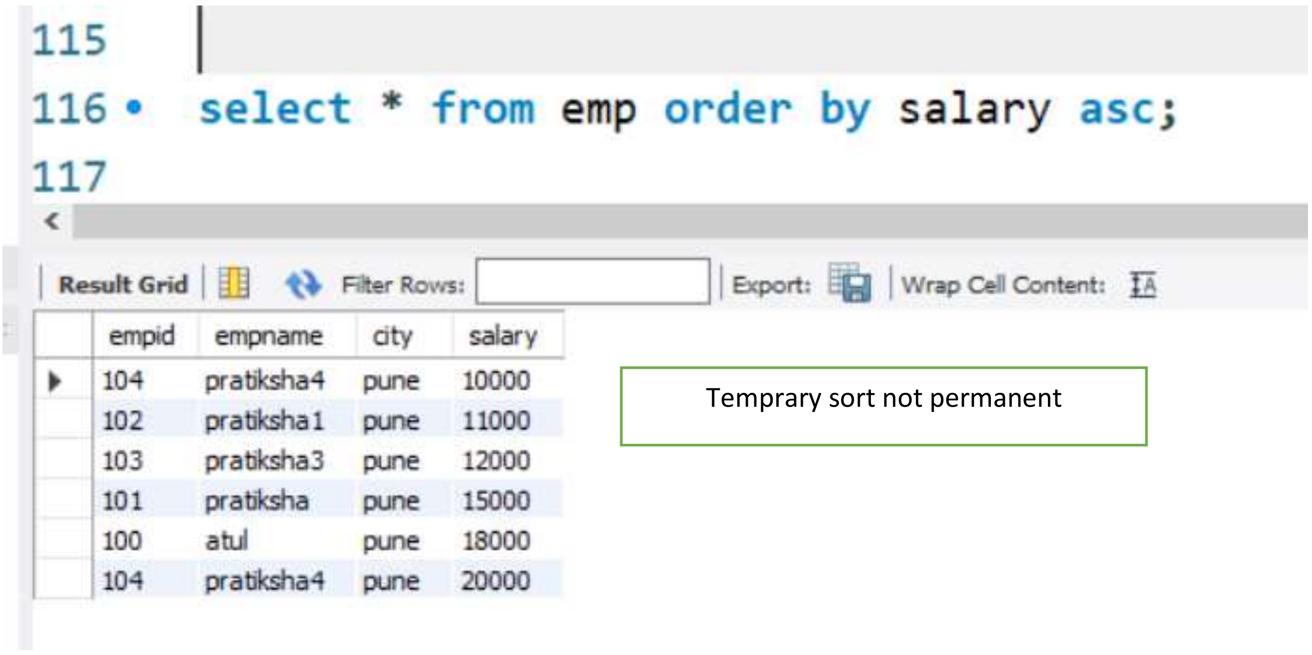
Before sort / order by



```
select * from emp;
```

empid	empname	city	salary
100	atul	pune	18000
101	pratiksha	pune	15000
102	pratiksha1	pune	11000
103	pratiksha3	pune	12000
104	pratiksha4	pune	10000
104	pratiksha4	pune	20000

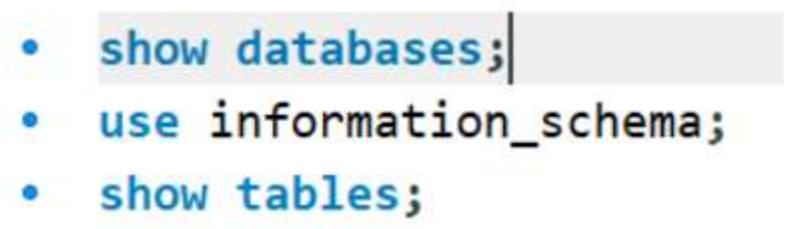
After sort using Order by



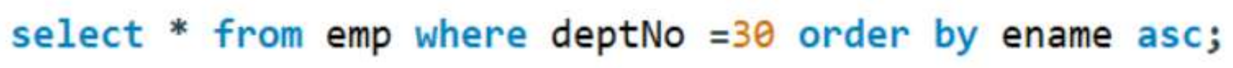
```
select * from emp order by salary asc;
```

empid	empname	city	salary
104	pratiksha4	pune	10000
102	pratiksha1	pune	11000
103	pratiksha3	pune	12000
101	pratiksha	pune	15000
100	atul	pune	18000
104	pratiksha4	pune	20000

Temprary sort not permanent



```
show databases;
use information_schema;
show tables;
```



```
select * from emp where deptNo =30 order by ename asc;
```

Retrive records where deptno is 30 and after it sort in ascending order on

Ename column

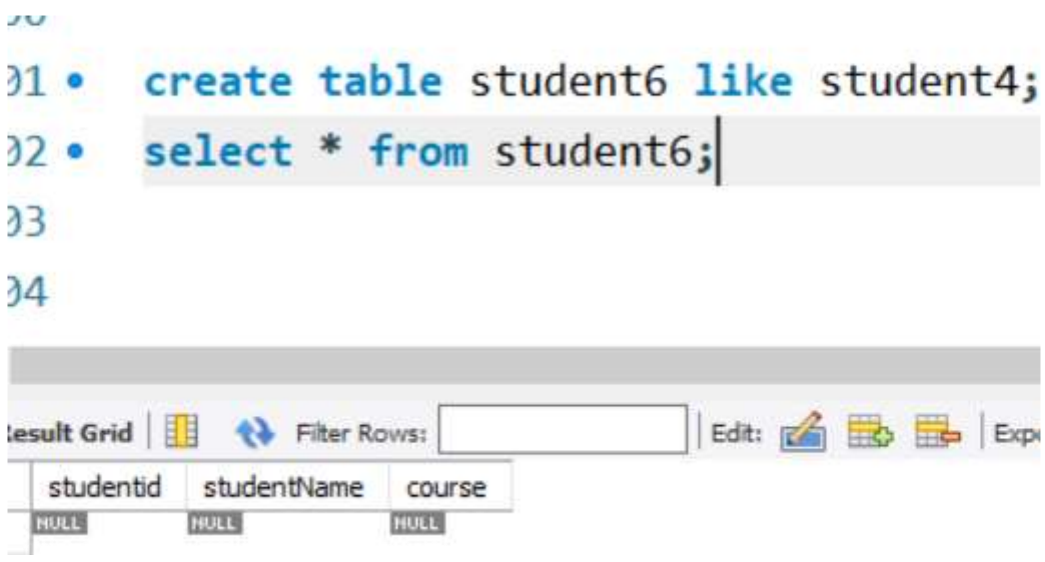
Copy table or duplicate

```
CREATE TABLE new_table    AS SELECT * FROM original_table;
```

Copy ony records . not structre as it is

If we want create structure as it or **empty structure**

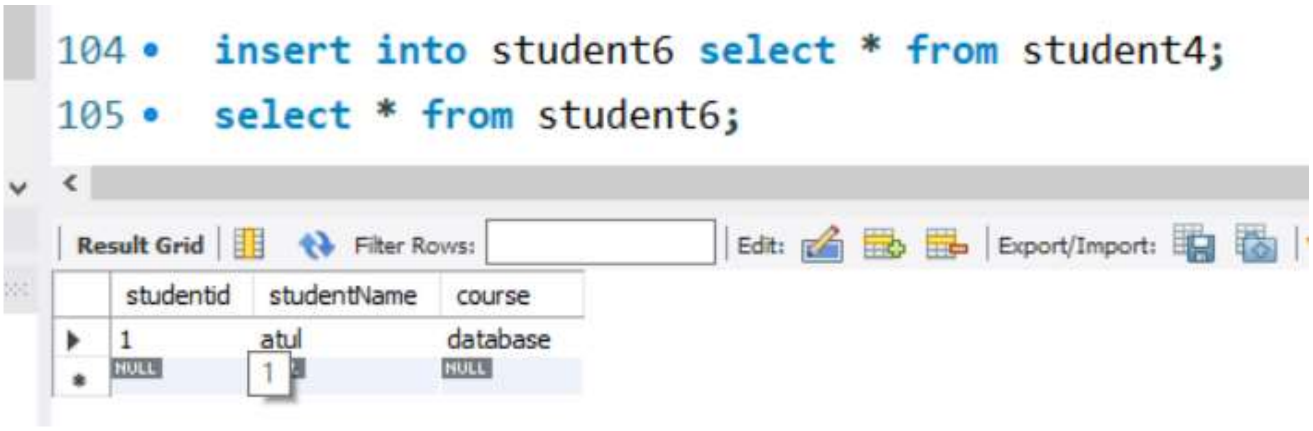
```
CREATE TABLE new_table LIKE original_table;
```



```
create table student6 like student4;
select * from student6;
```

```
studentid  studentName  course
NULL       NULL         NULL
```

Above only structure is create using LIKE to insert data in empty structure form old table as

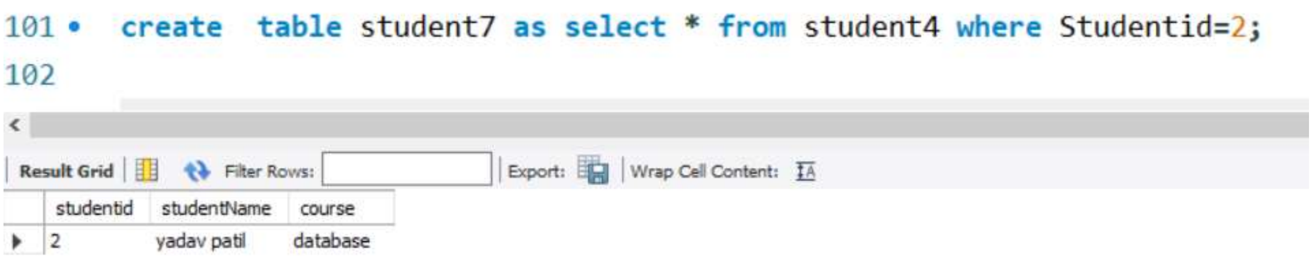


```
insert into student6 select * from student4;
select * from student6;
```

```
studentid  studentName  course
1          atul         database
NULL                    NULL
```

In below fig check condtion where cluase

Create new table with name student7 and insert data form old table as condtion where studentid=7



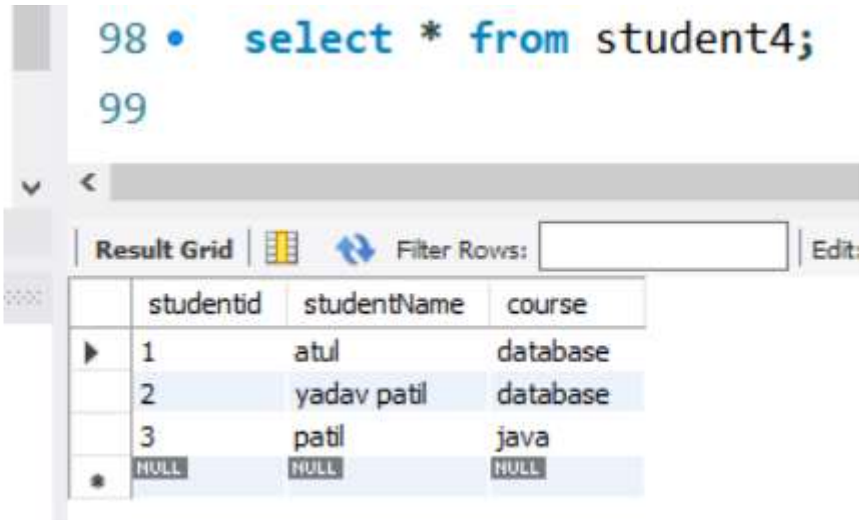
```
create  table student7 as select * from student4 where Studentid=2;
```

```
studentid  studentName  course
2          yadav patil  database
```

Distinct clause >> remote duplicate

DISTINCT clause is used to remove duplicates from the result set (only in output of select)

The DISTINCT clause can only be used with select statement

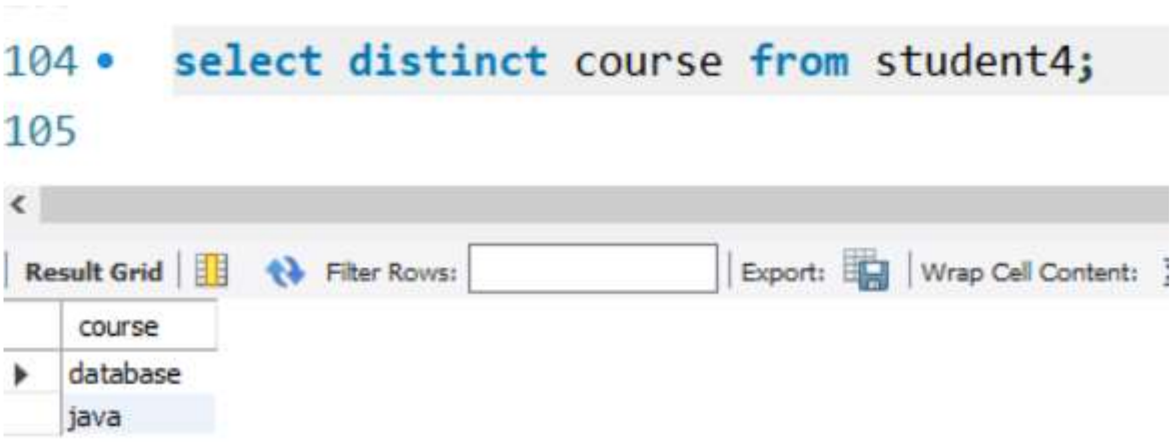


```
select * from student4;
```

```
studentid  studentName  course
1          atul         database
2          yadav patil  database
3          patil        java
NULL       NULL         NULL
```

Now using distinct remove duplicate here course database is duplicated hence one entry of databse is removed as

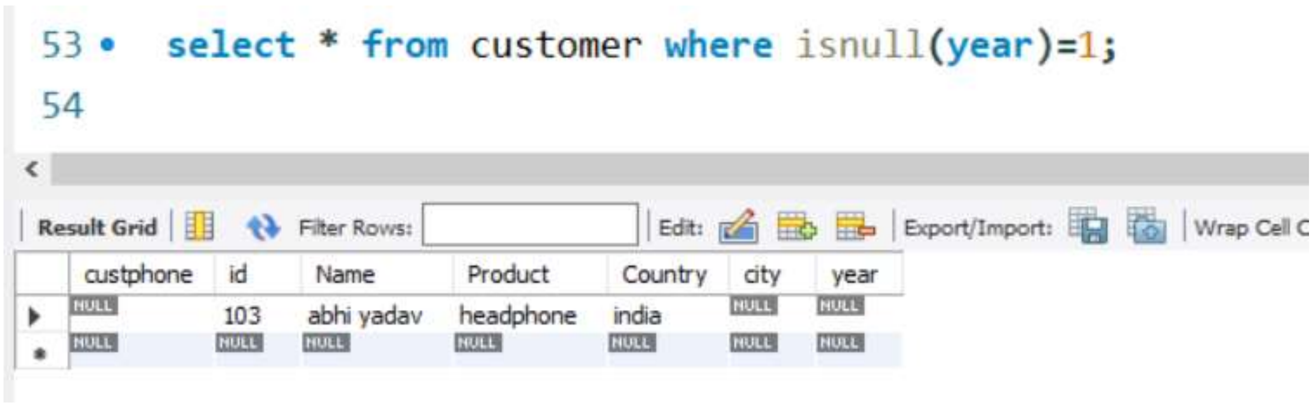
Select >> how to select >> distinct form …



```
select distinct course from student4;
```

```
course
database
java
```

ISNull to check null value , to check colomn returned value is null or not



```
select * from customer where isnull(year)=1;
```

custphone	id	Name	Product	Country	city	year
NULL	103	abhi yadav	headphone	india	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

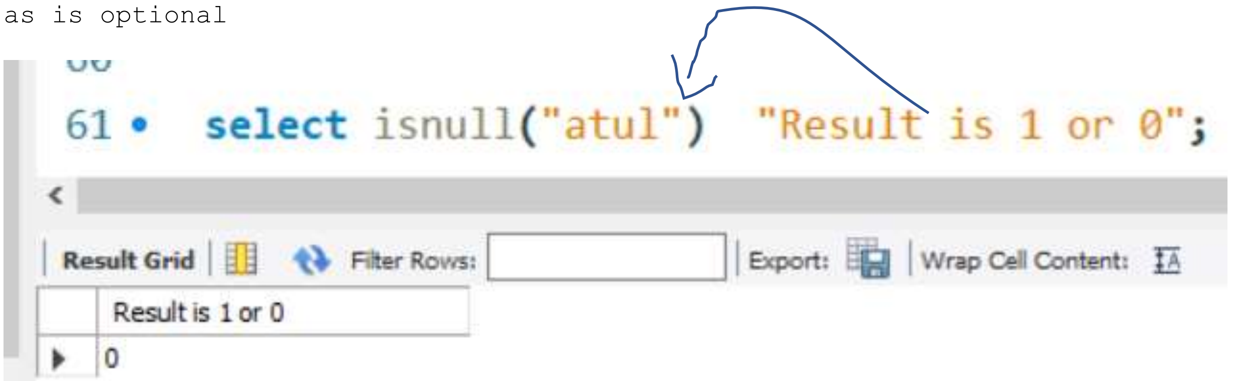
Alias colum name

To provide alias for colomn name in output of select statement

ALIASES can be used to create a <u>temporary name</u> for columns or tables.

```
column_name [ AS ] alias_name
```

```
as is optional
```

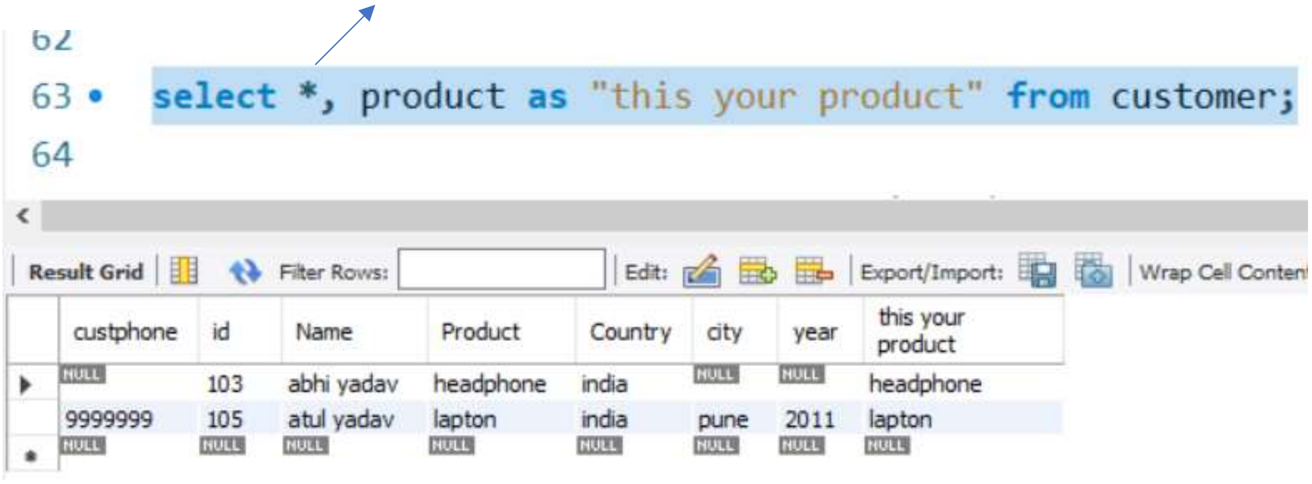


```
select isnull("atul")  "Result is 1 or 0";
```

```
Result is 1 or 0
0
```

`Restult is 1 or 0` `is alias name for result is null`

```
If we want select all records but give alies name to any one

Column as          this select all records
```



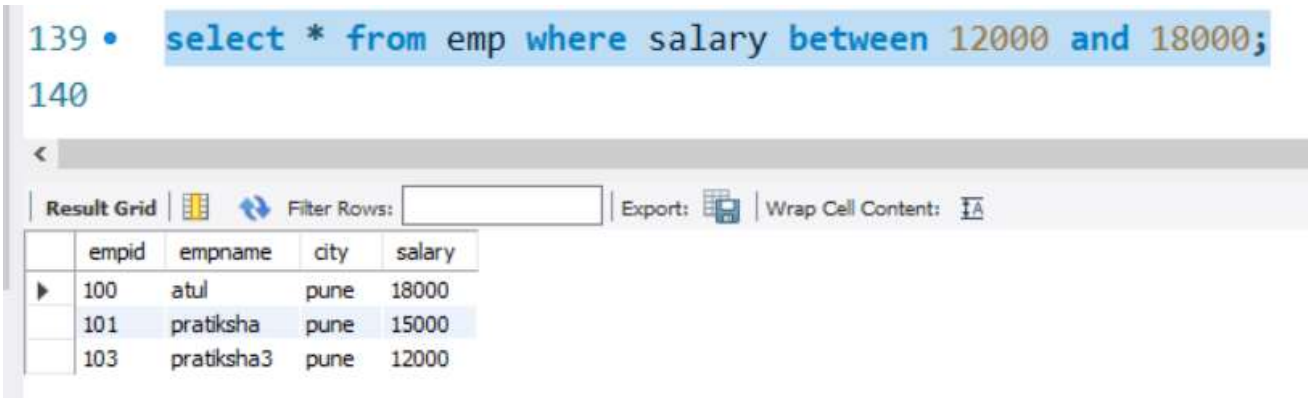
```
select *, product as "this your product" from customer;
```

custphone	id	Name	Product	Country	city	year	this your product
NULL	103	abhi yadav	headphone	india	NULL	NULL	headphone
9999999	105	atul yadav	lapton	india	pune	2011	lapton
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Predicates

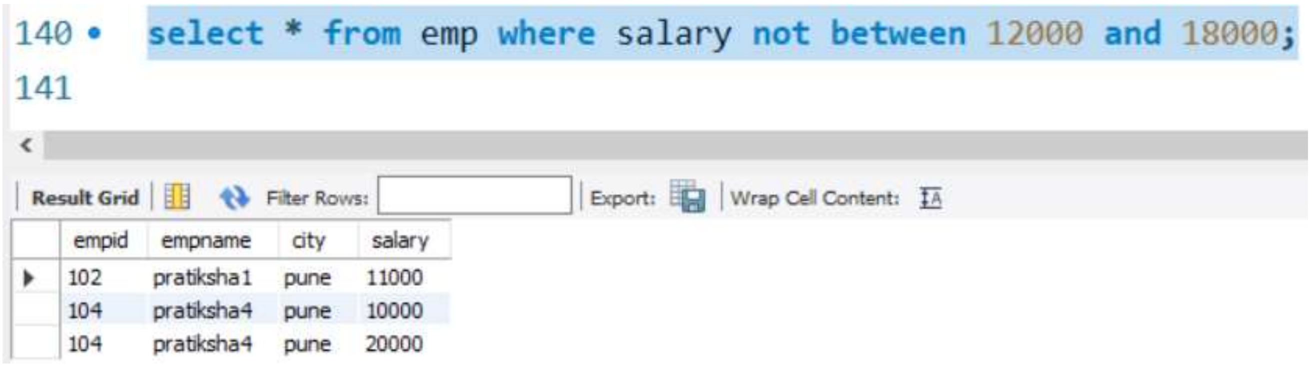
Any operator in where clause is called predicates

1. Between.. and
2. In
3. Like
4. Isnull



```
select * from emp where salary between 12000 and 18000;
```

empid	empname	city	salary
100	atul	pune	18000
101	pratiksha	pune	15000
103	pratiksha3	pune	12000

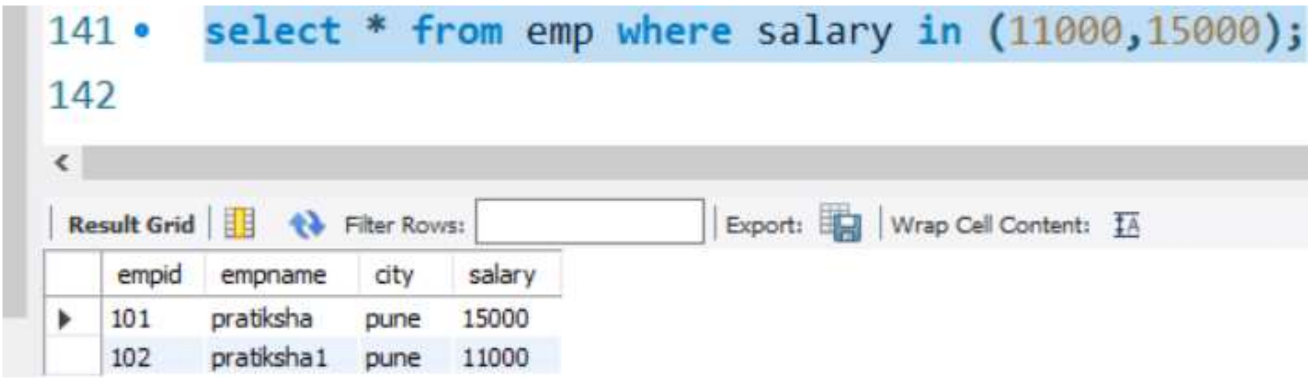


```
select * from emp where salary not between 12000 and 18000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empid	empname	city	salary
102	pratiksha1	pune	11000
104	pratiksha4	pune	10000
104	pratiksha4	pune	20000

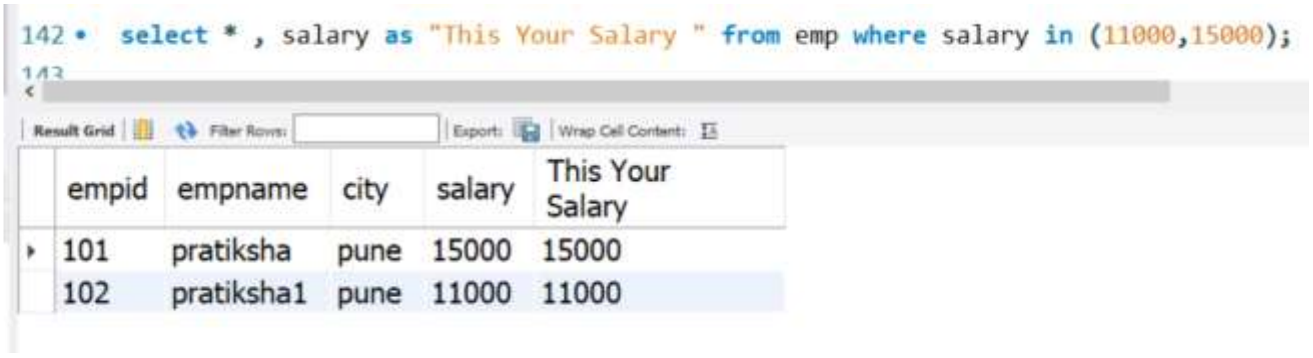
IN check given value present in give value



```
select * from emp where salary in (11000,15000);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empid	empname	city	salary
101	pratiksha	pune	15000
102	pratiksha1	pune	11000

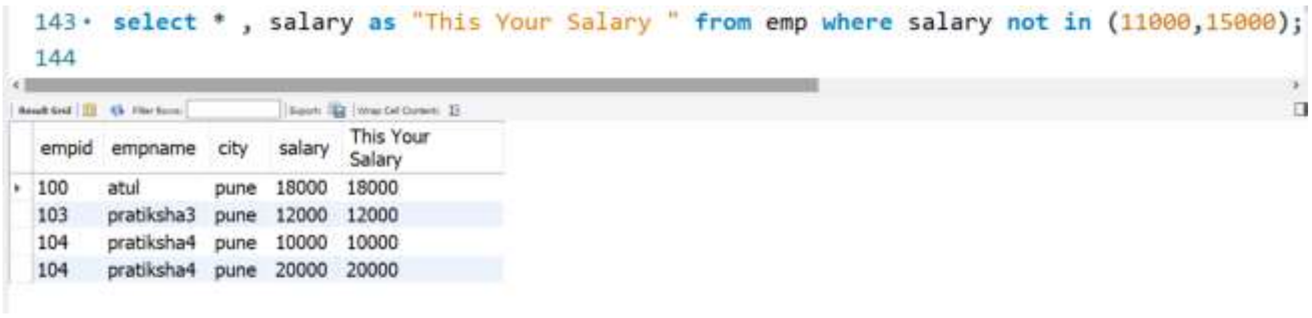


```
select * , salary as "This Your Salary " from emp where salary in (11000,15000);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empid	empname	city	salary	This Your Salary
101	pratiksha	pune	15000	15000
102	pratiksha1	pune	11000	11000

Not IN



```sql
select * , salary as "This Your Salary " from emp where salary not in (11000,15000);
```

empid	empname	city	salary	This Your Salary
100	atul	pune	18000	18000
103	pratiksha3	pune	12000	12000
104	pratiksha4	pune	10000	10000
104	pratiksha4	pune	20000	20000

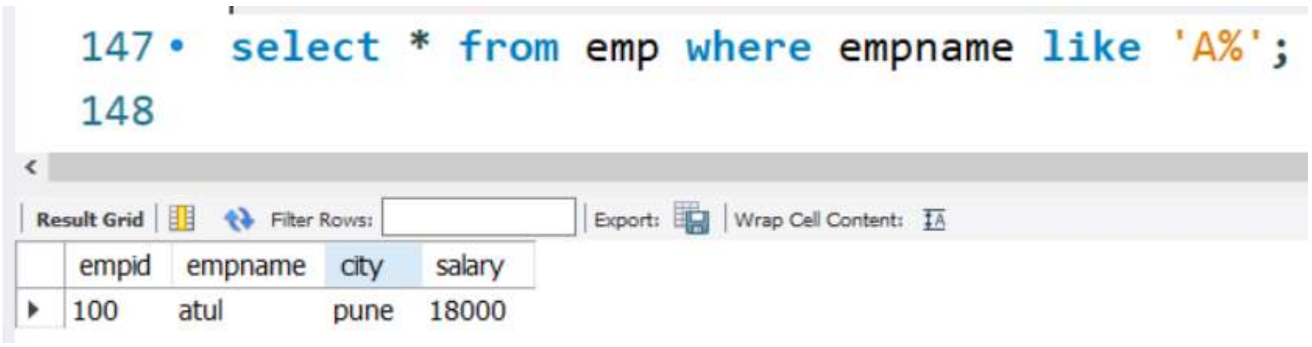
Like

This pattern is used to search particular pattern

‘- ‘ :- used to single character

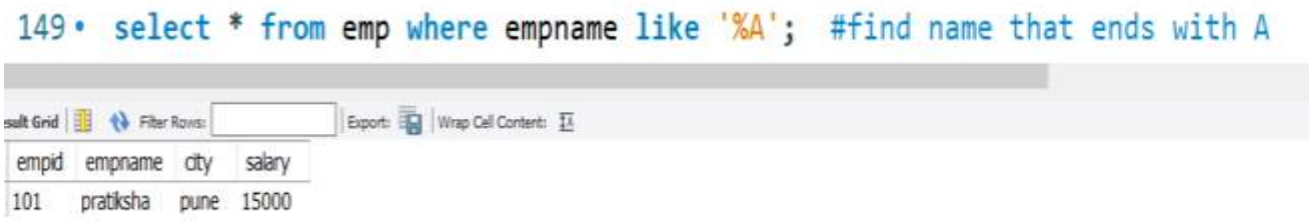
%   :- used for multiple char

Find the employ whose name start with alphabate A



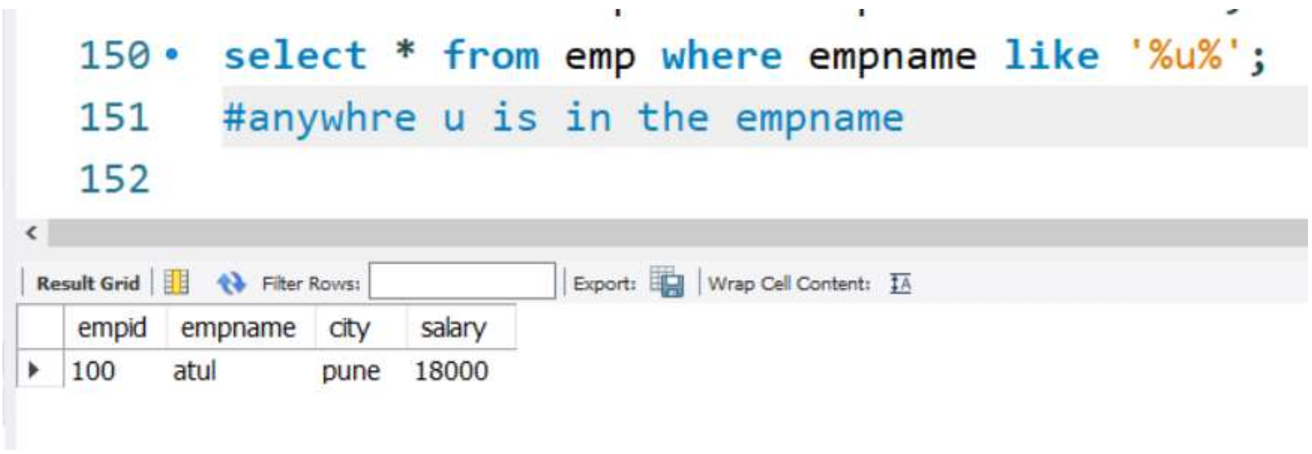
```sql
select * from emp where empname like 'A%';
```

empid	empname	city	salary
100	atul	pune	18000



```sql
select * from emp where empname like '%A';  #find name that ends with A
```

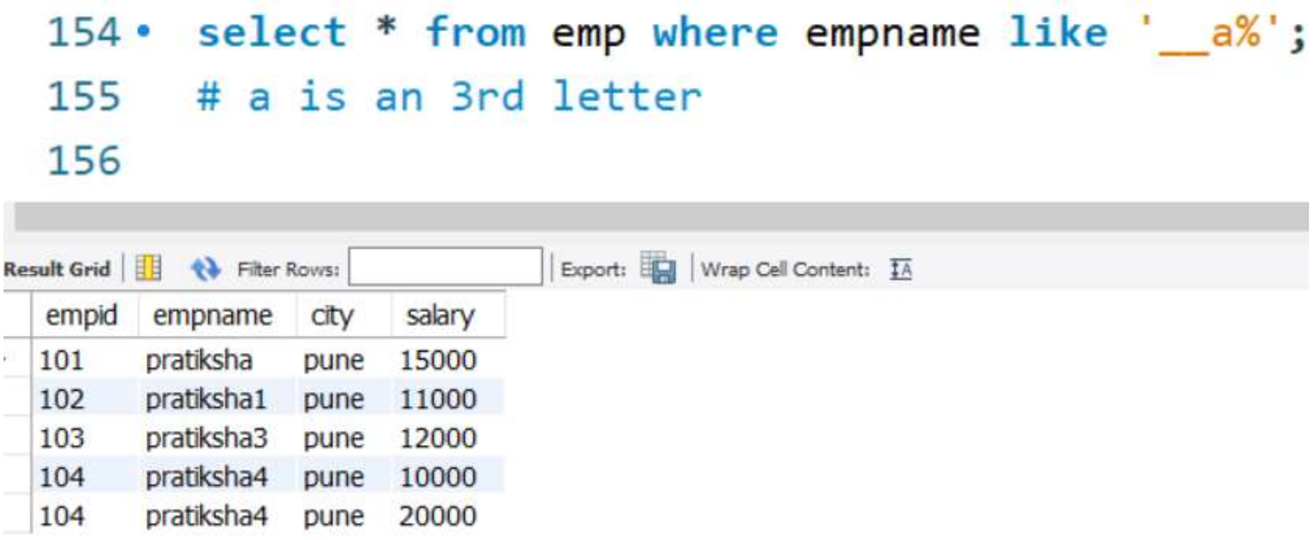
empid	empname	city	salary
101	pratiksha	pune	15000



```
select * from emp where empname like '%u%';
#anywhre u is in the empname
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empid	empname	city	salary
100	atul	pune	18000



```
select * from emp where empname like '__a%';
# a is an 3rd letter
```

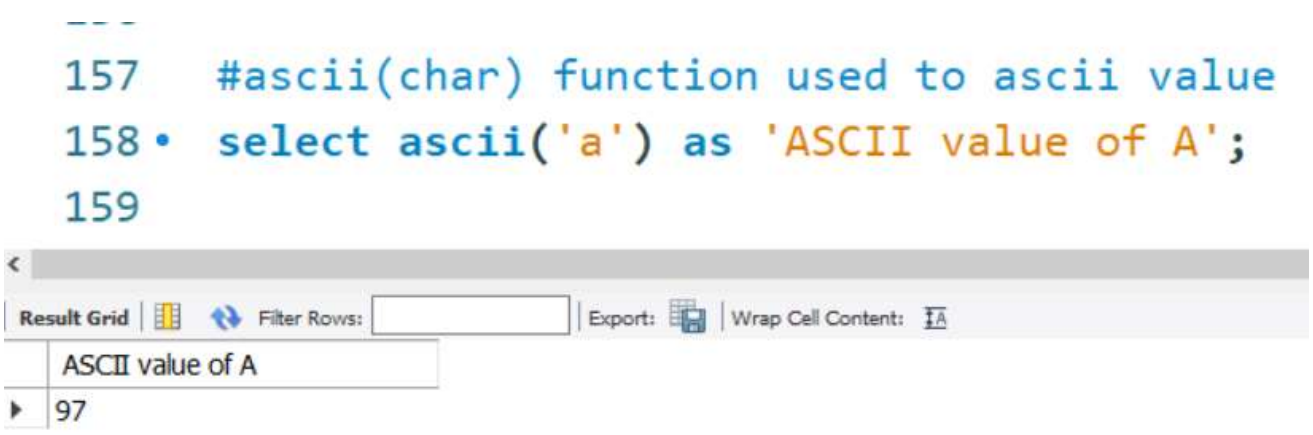
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empid	empname	city	salary
101	pratiksha	pune	15000
102	pratiksha1	pune	11000
103	pratiksha3	pune	12000
104	pratiksha4	pune	10000
104	pratiksha4	pune	20000

LIKE ‘%[E,I,P]’ → empname where last 3 char is E , I, P

LIKE ‘_ [A-I]%’ → empname where 2nd char is in the range of A to I

Character function



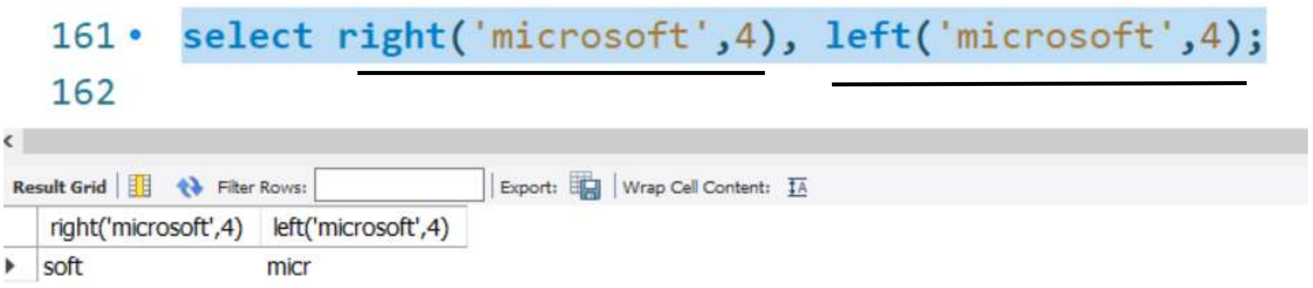
```
#ascii(char) function used to ascii value
select ascii('a') as 'ASCII value of A';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
ASCII value of A
97
```

Select   left('microsoft',4);   extract 4 char from left   o/p micr

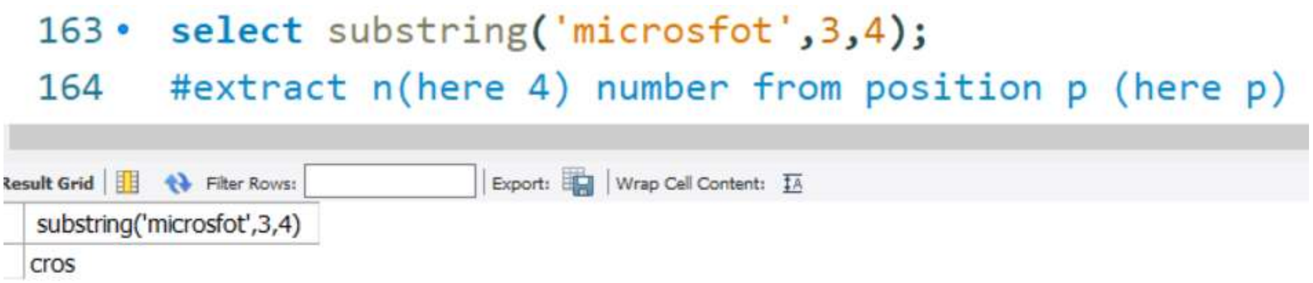
Here one select but two statement left and right are combined



```
select right('microsoft',4), left('microsoft',4);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

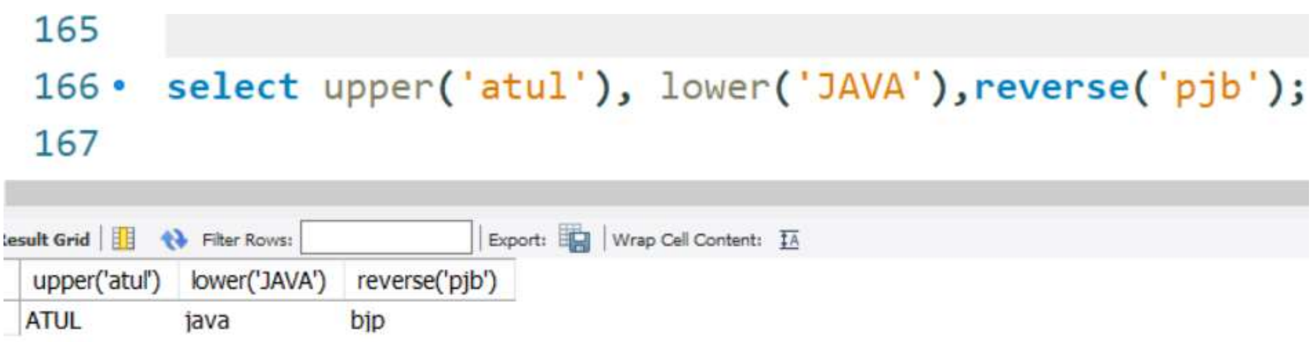
```
right('microsoft',4)   left('microsoft',4)
soft                   micr
```



```
select substring('microsfot',3,4);
#extract n(here 4) number from position p (here p)
```

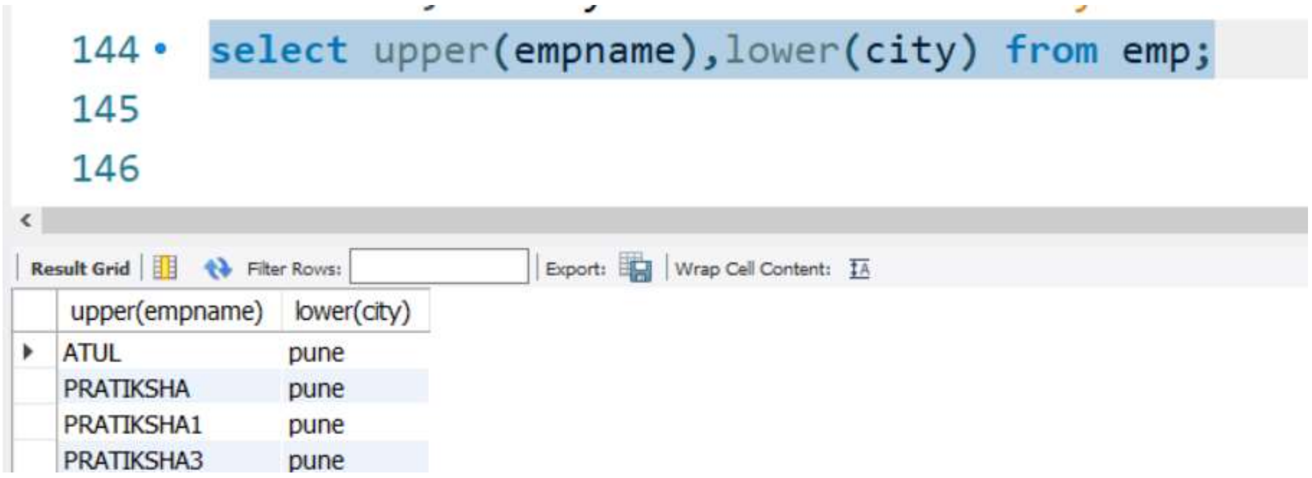
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
substring('microsfot',3,4)
cros
```



```
select upper('atul'), lower('JAVA'),reverse('pjb');
```

upper('atul')	lower('JAVA')	reverse('pjb')
ATUL	java	bjp



```
select upper(empname),lower(city) from emp;
```

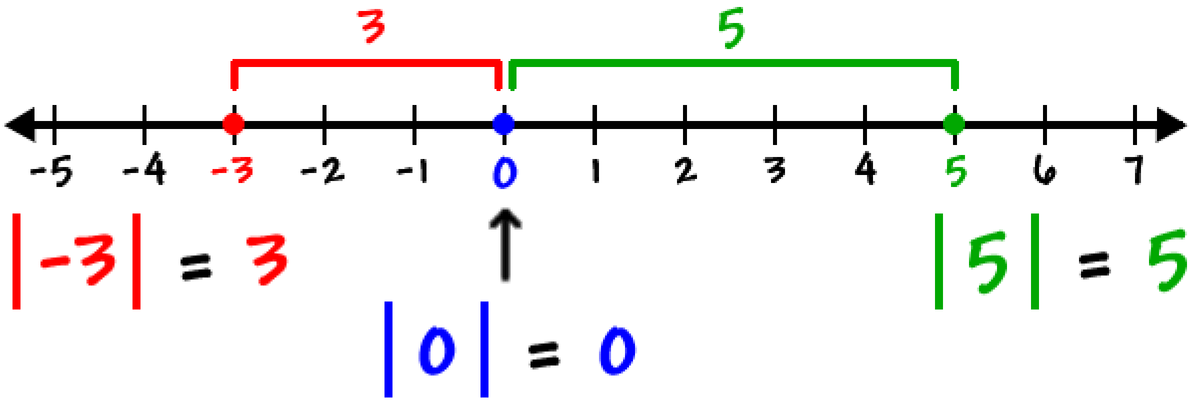
upper(empname)	lower(city)
ATUL	pune
PRATIKSHA	pune
PRATIKSHA1	pune
PRATIKSHA3	pune

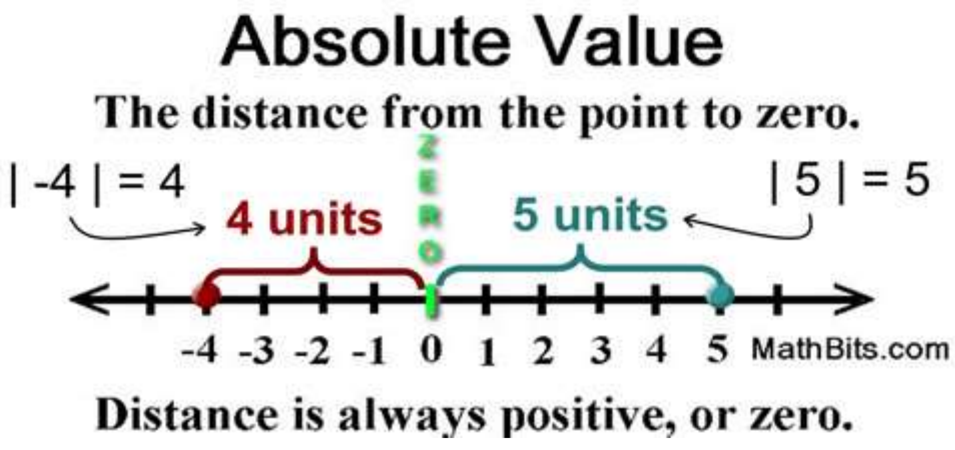
NUMREIC FUNCTION : operate on numeric data type

Abs() :

The absolute value

is it represents **the distance of the number from zero on number line.**

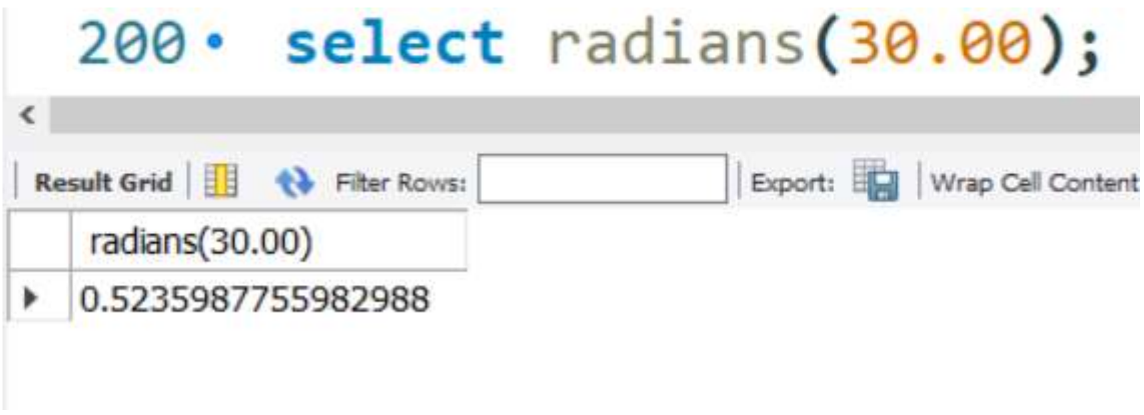




So in mysql abs() for absolute value

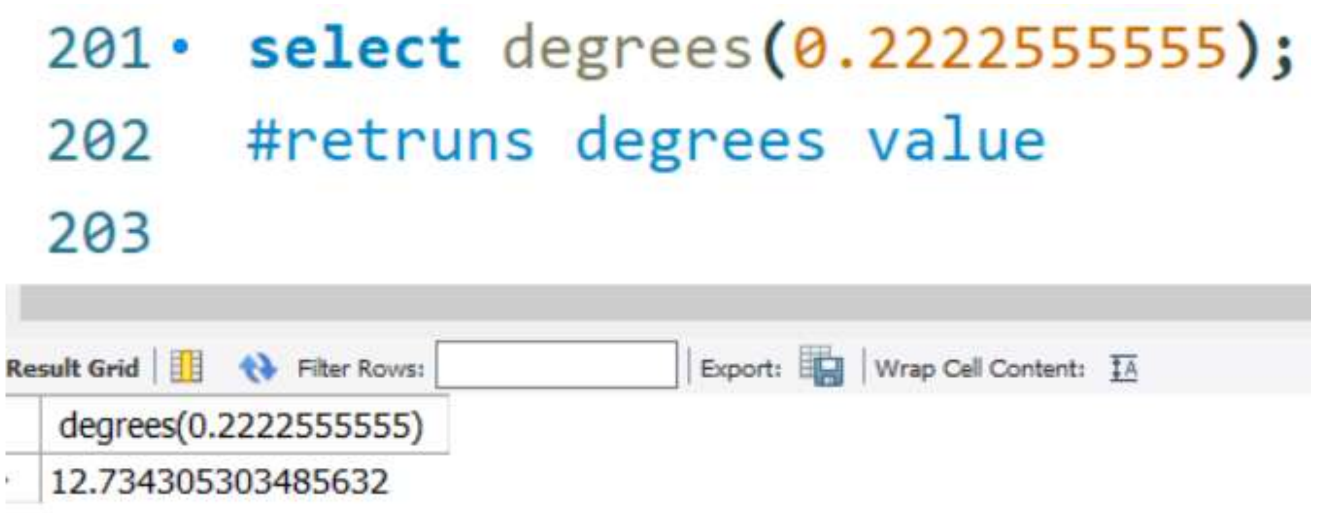
select abs(-4); ➔ 4

select abs(4); ➔ 4



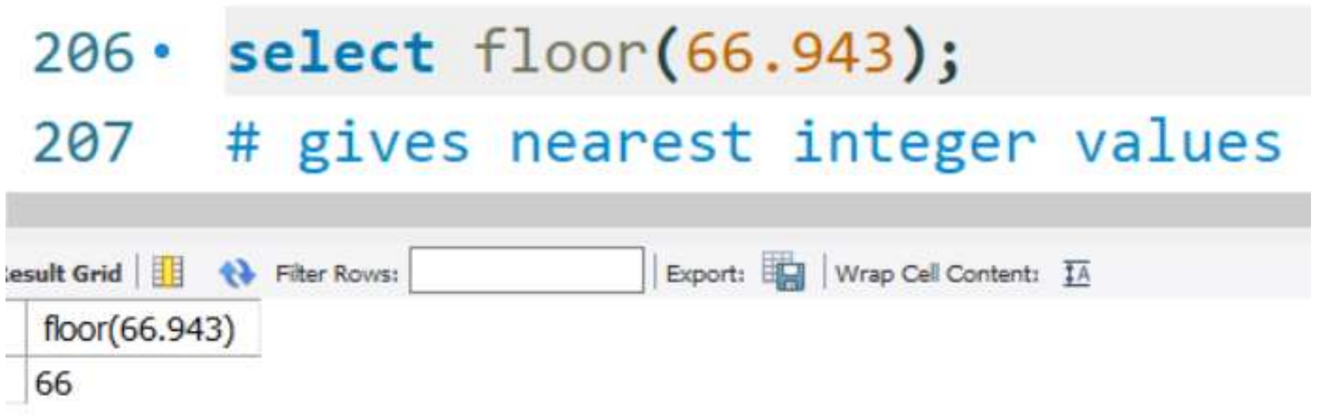
```
select radians(30.00);
```

```
radians(30.00)
0.5235987755982988
```



```
select degrees(0.2222555555);
#retruns degrees value

```

```
degrees(0.2222555555)
12.734305303485632
```



```
select floor(66.943);
# gives nearest integer values
```

```
floor(66.943)
66
```

select round(12.4); ➔ 12

select round(12.5); ➔ 13

select power(5,3); ➔ 125

# 5*5*5=125

select sign(-7); ➔ -1 for minus negative num

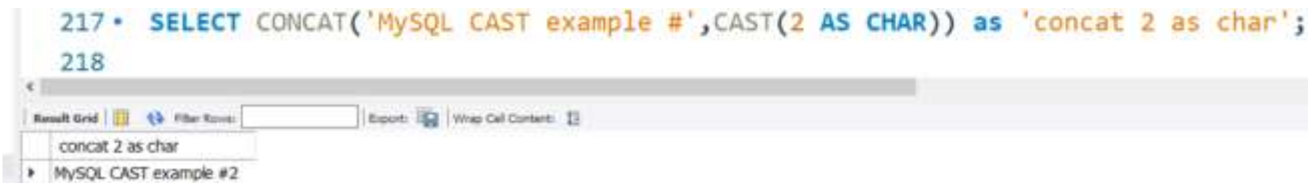
select sign(7); ➔ 1 for plus or postive num

select sign(0); ➔ 0 when num is zero

Converstion Function

Convert one data type to anather

select cast(123 as char);   here 123 treated as char



```
SELECT CONCAT('MySQL CAST example #',CAST(2 AS CHAR)) as 'concat 2 as char';
```

```
concat 2 as char
MySQL CAST example #2
```

Date function

SELECT CURDATE();

SELECT CURDATE() + 1;

SELECT CURDATE();   #current date

SELECT CURDATE() + 1;   #date +1

SELECT CURRENT_TIME();   #time

select CURRENT_TIMESTAMP();   # date and time

SELECT DATE("2017-06-15 09:34:21");   # extract date form date and time

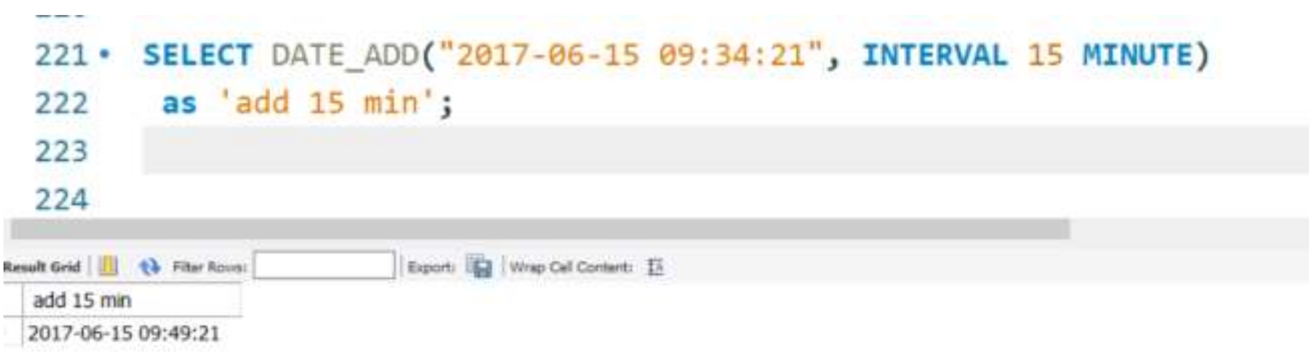
o/p 2017-6-15

SELECT DATE("The date is 2017-06-15");   #extract date == 2017-06-15

SELECT DATEDIFF("2017-06-25", "2017-06-15"); #differnce as day == 10 day

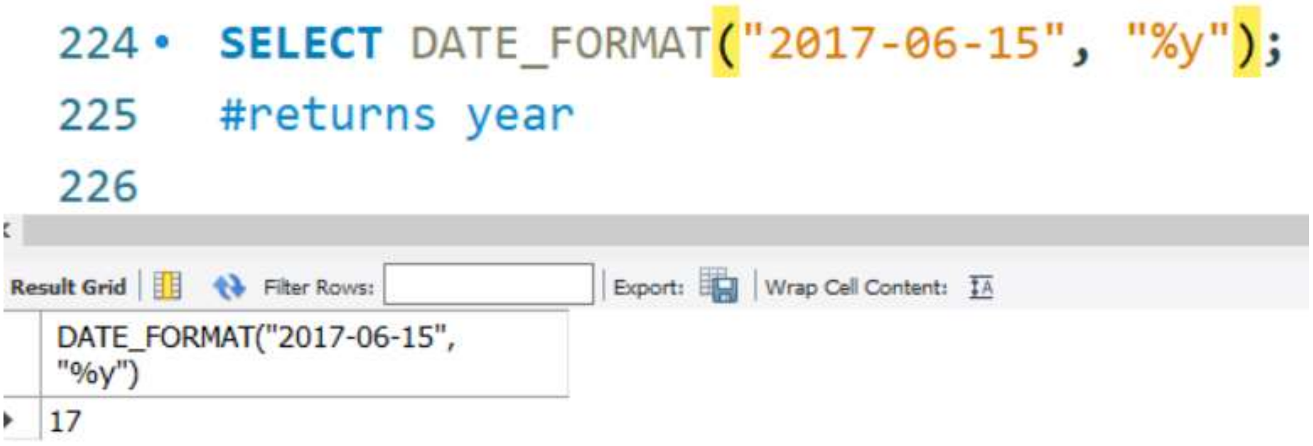
SELECT DATE_ADD("2017-06-15", INTERVAL 10 DAY);   #add 10 days to

Date



```
SELECT DATE_ADD("2017-06-15 09:34:21", INTERVAL 15 MINUTE)
 as 'add 15 min';
```

```
add 15 min
2017-06-15 09:49:21
```



```
SELECT DATE_FORMAT("2017-06-15", "%y");
#returns year
```

```
DATE_FORMAT("2017-06-15", "%y")
17
```

Format	**Description**
%a	Abbreviated weekday name (Sun to Sat)
%b	Abbreviated month name (Jan to Dec)
%c	Numeric month name (0 to 12)
%D	Day of the month as a numeric value, followed by suffix (1st, 2nd, 3rd, ...)
%d	Day of the month as a numeric value (01 to 31)
%e	Day of the month as a numeric value (0 to 31)
%f	Microseconds (000000 to 999999)
%H	Hour (00 to 23)
%h	Hour (00 to 12)
%I	Hour (00 to 12)
%i	Minutes (00 to 59)
%j	Day of the year (001 to 366)
%k	Hour (0 to 23)
%l	Hour (1 to 12)
%M	Month name in full (January to December)
%m	Month name as a numeric value (00 to 12)
%p	AM or PM
%r	Time in 12 hour AM or PM format (hh:mm:ss AM/PM)
%S	Seconds (00 to 59)
%s	Seconds (00 to 59)
%T	Time in 24 hour format (hh:mm:ss)
%U	Week where Sunday is the first day of the week (00 to 53)
%u	Week where Monday is the first day of the week (00 to 53)
%V	Week where Sunday is the first day of the week (01 to 53). Used with %X
%v	Week where Monday is the first day of the week (01 to 53). Used with %x
%W	Weekday name in full (Sunday to Saturday)
%w	Day of the week where Sunday=0 and Saturday=6
%X	Year for the week where Sunday is the first day of the week. Used with %V
%x	Year for the week where Monday is the first day of the week. Used with %v
%Y	Year as a numeric, 4-digit value
%y	Year as a numeric, 2-digit value

Subtract 10 days from a date and return the date:

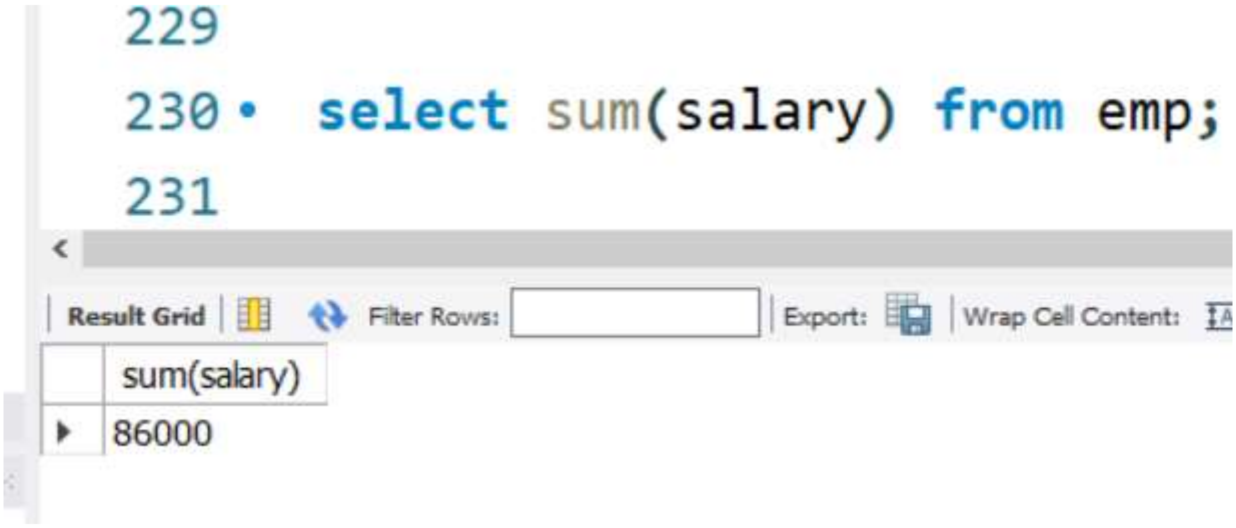
```
SELECT SUBDATE("2017-06-15", INTERVAL 10 DAY);
```

```
SELECT WEEKDAY("2017-06-15");   // 3
```

0 = Monday, 1 = Tuesday, 2 = Wednesday, 3 = Thursday, 4 = Friday, 5 = Saturday, 6 = Sunday.

Aggrigate function

Sum() : sum of all values in give table



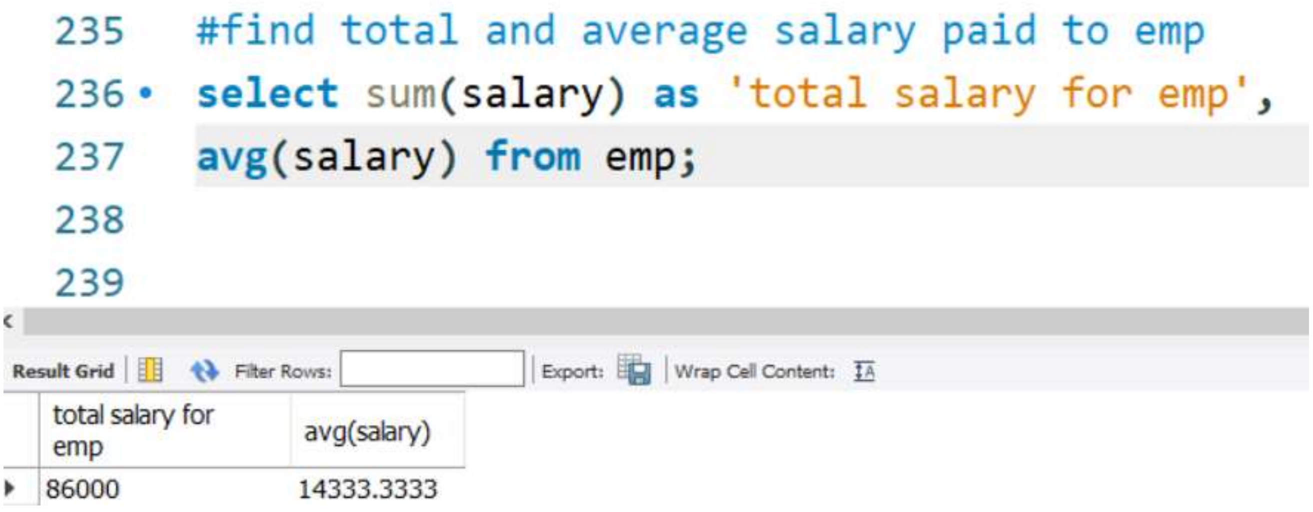
```
select sum(salary) from emp;
```

	sum(salary)
▶	86000

Avg(colomn_name) ;

Avg(salary);

select avg(salary) from emp;



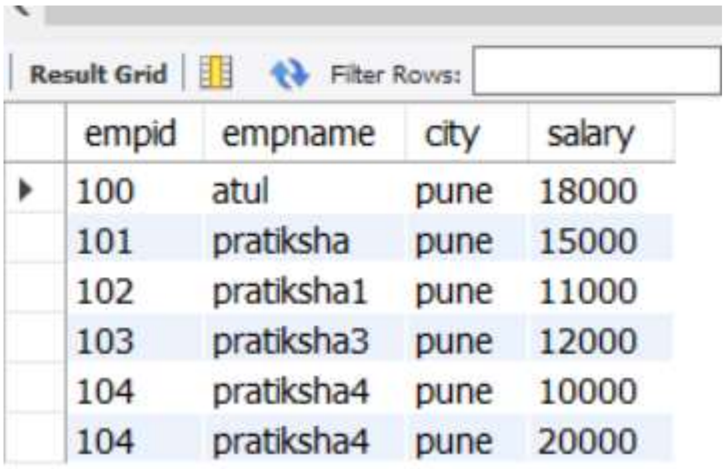
```
#find total and average salary paid to emp
select sum(salary) as 'total salary for emp',
avg(salary) from emp;
```

	total salary for emp	avg(salary)
▶	86000	14333.3333

Max(colom_name) → max value in give column

Min(colom_name) → min value in give column

IMPORTANT



Result Grid | Filter Rows:

	empid	empname	city	salary
▸	100	atul	pune	18000
	101	pratiksha	pune	15000
	102	pratiksha1	pune	11000
	103	pratiksha3	pune	12000
	104	pratiksha4	pune	10000
	104	pratiksha4	pune	20000

Now find employe whose salary is higest

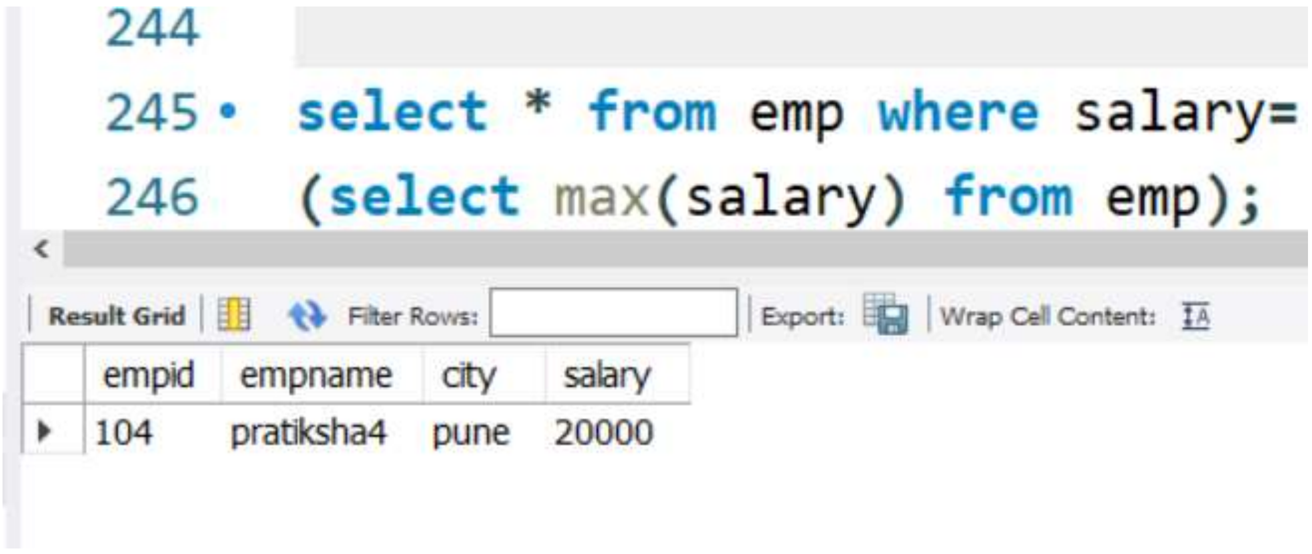
Here 104 pratiksha4 has highest salary

But in large database we don't know whose salary is higest , so we get it using max() fun as

Normal query : select * from emp where salary=<u>20000</u>;

ADV Query :

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



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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	empid	empname	city	salary
▸	104	pratiksha4	pune	20000

Count() : count number of rows in result set

Select count(*) from emp; // count total records

Find total number of employee i.e (count)

whose salary is >= 15000

⇨ select count(*) from emp where salary>=15000;
in above table 3 records hence o/p 3

Group By >> similar things.

the GROUP BY statement is for applying aggregate functions for a group of the result-set with one or more columns.

Simply group of similar things (eg. Same name, same color name etc)

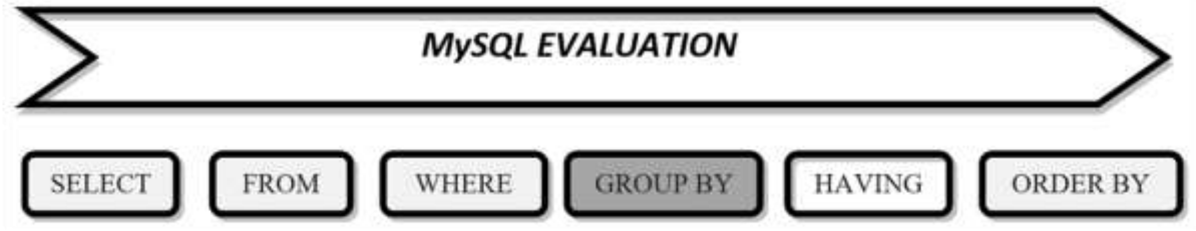
Group by >> fetching information about a >> <u>group of data</u>

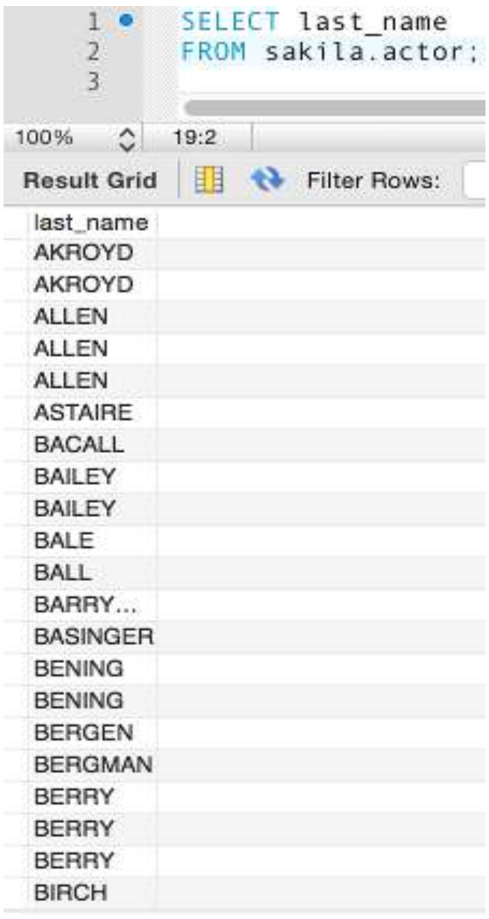
The GROUP BY statement only shows when you have many **similar things.**

The GROUP BY used with >>>> SUM, AVG, COUNT, MIN, and MAX.

used after SELECT, FROM, WHERE

before HAVING, ORDER BY clauses.





```
SELECT last_name
FROM sakila.actor;
```

100%   19:2

Result Grid   Filter Rows:

```
last_name
AKROYD
AKROYD
ALLEN
ALLEN
ALLEN
ASTAIRE
BACALL
BAILEY
BAILEY
BALE
BALL
BARRY...
BASINGER
BENING
BENING
BERGEN
BERGMAN
BERRY
BERRY
BERRY
BIRCH
```

`last_name` column contains a lot of duplicates

hence to group this names

Now, if we add `GROUP BY last_name` to the mix:

SELECT last_name   FROM actor   GROUP BY last_name;

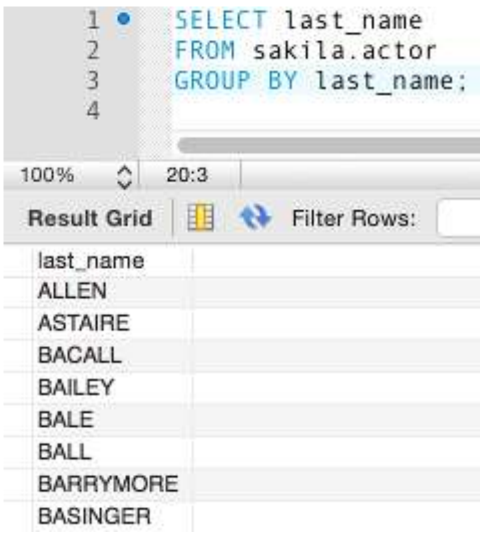
Grouped as their names

We have selected all actors' last names from the table and grouped them by the last name.

If two or more actors share the same last name, it is represented only once in the result set.

For example, if two actors have a last name of "Bailey", that last name is listed once only.

See following image



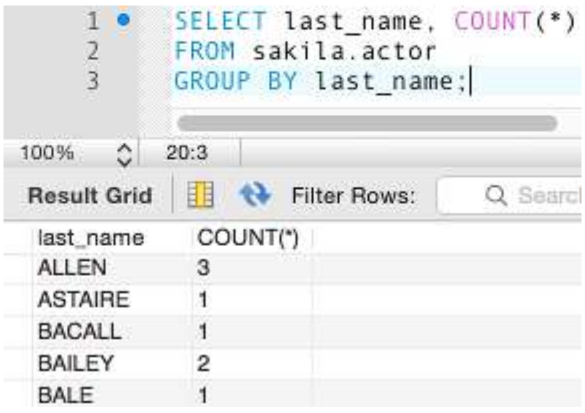
```
SELECT last_name
FROM sakila.actor
GROUP BY last_name;
```

100%   20:3

Result Grid   Filter Rows:

```
last_name
ALLEN
ASTAIRE
BACALL
BAILEY
BALE
BALL
BARRYMORE
BASINGER
```

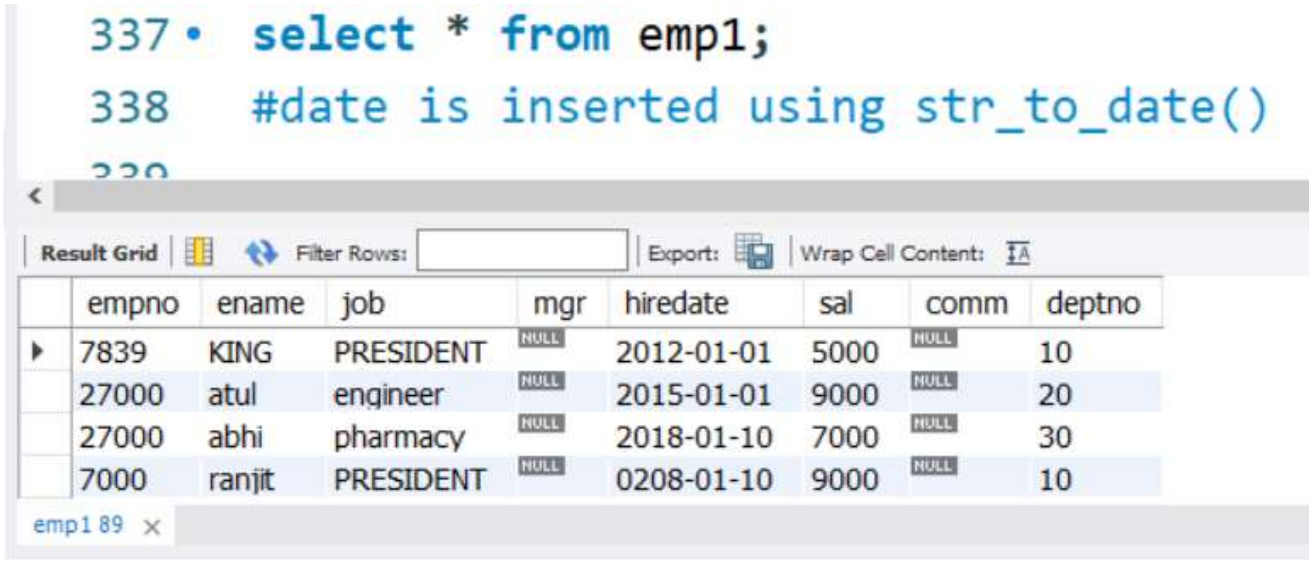
SELECT last_name, COUNT(*) FROM actor GROUP BY last_name;



```
SELECT last_name, COUNT(*)
FROM sakila.actor
GROUP BY last_name;
```

last_name	COUNT(*)
ALLEN	3
ASTAIRE	1
BACALL	1
BAILEY	2
BALE	1

Now see emp1 table

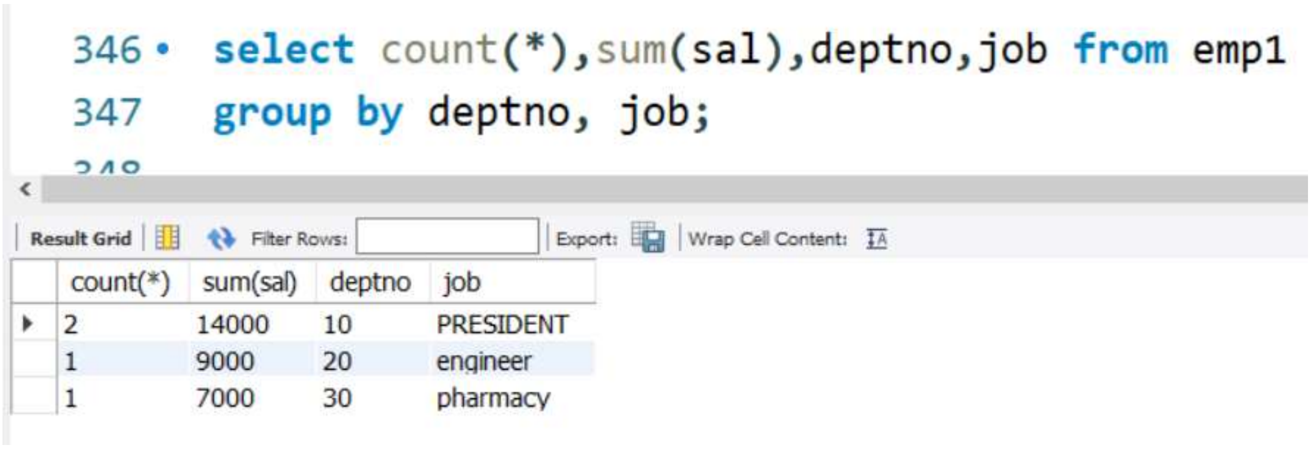


```
337 • select * from emp1;
338   #date is inserted using str_to_date()
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
7839	KING	PRESIDENT	NULL	2012-01-01	5000	NULL	10
27000	atul	engineer	NULL	2015-01-01	9000	NULL	20
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10

Find number of employee & total salary job wise in each dept group by

Select count(*) , sum(sal), job, deptno (hence group) deptno, jb



```
select count(*),sum(sal),deptno,job from emp1
group by deptno, job;
```

count(*)	sum(sal)	deptno	job
2	14000	10	PRESIDENT
1	9000	20	engineer
1	7000	30	pharmacy

In simply , aply group by whose group we want create . in group only one value is taken , after on that group we apply aggrigate function sum , count etc

Having clause

HAVING clause is used in combination with the GROUP BY clause

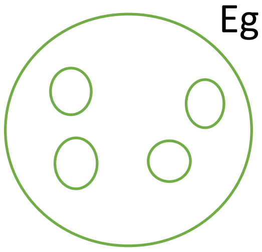
to restrict the groups of returned rows to only those whose the condition is TRUE.

>> grouping data but based on condtion that is having clause

Group by >>> happens only when simliar things are present

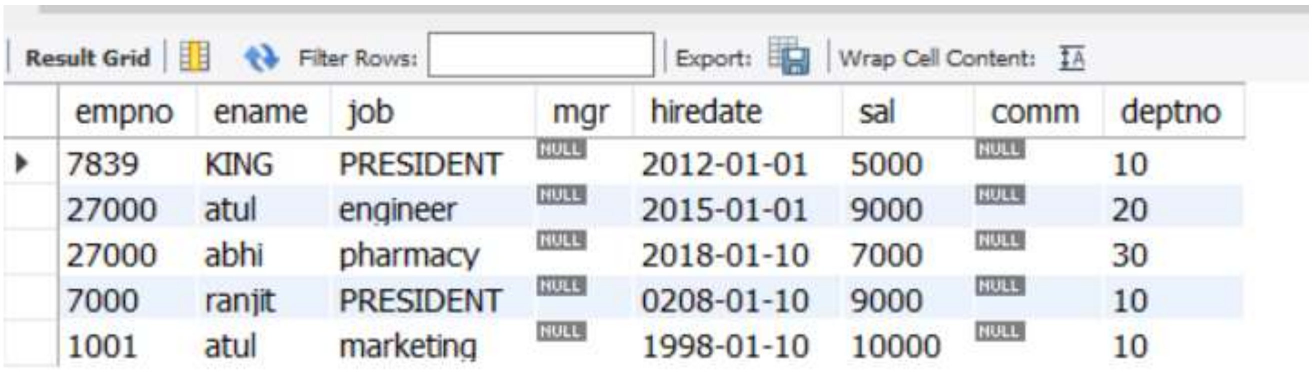
(eg. Two same ename)

Eg. Yellow color dress people>> 1 group



Count(*) in group >>> 4 things

4 things >> 1 group



empno	ename	job	mgr	hiredate	sal	comm	deptno
7839	KING	PRESIDENT	NULL	2012-01-01	5000	NULL	10
27000	atul	engineer	NULL	2015-01-01	9000	NULL	20
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10
1001	atul	marketing	NULL	1998-01-10	10000	NULL	10

1st col 2nd col 3rd col

Ename , count= how may things in grp, total sal of counted emp

Avg of sal counted emp 4th

select ename , count(*), sum(sal), avg(sal) from emp1

group by ename having count(*)>=2;

>> doing group of same name emp

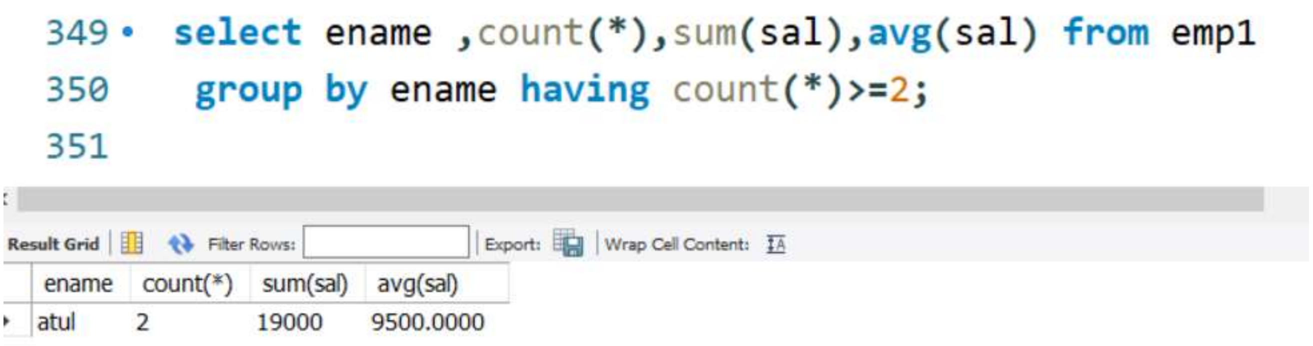
Select ename , count(*), sum(sal), avg(sal) from emp1 group by ename having count>=2

(only when similar things are present . 1 group //here ename with same name are =2 . i.e 1 group with 2 count

➔But when group depends on condition i.e == having clause

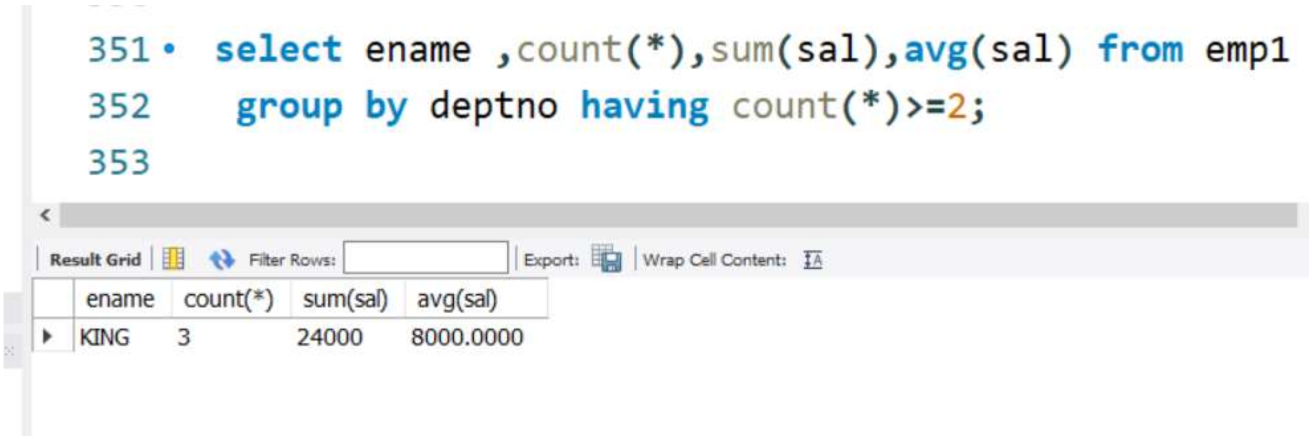
➔ also in simple when we group by using group by then **group done first** .. then on that group operationas are applied eg. Cout max min etc

➔in group by in which order we group in same order we print using select statement



```
select ename ,count(*),sum(sal),avg(sal) from emp1
 group by ename having count(*)>=2;
```

```
ename  count(*)  sum(sal)  avg(sal)
atul   2         19000     9500.0000
```



```
select ename ,count(*),sum(sal),avg(sal) from emp1
 group by deptno having count(*)>=2;
```

```
ename  count(*)  sum(sal)  avg(sal)
KING   3         24000     8000.0000
```

Now one thing observe we made group of deptno but not printed using select command

In above query records are grouped by deptno (same dept numbers are grouped together and make one group)

Count(*)	is counts the records or things in that group
Sum(sal)	is total or sum of salary of employe which only in group
Avg(sal)	is avg of salary of emp only in group == here 3 (count=3)
Group by	group is made from deptno(same dept) but having condtion The group count is >=2

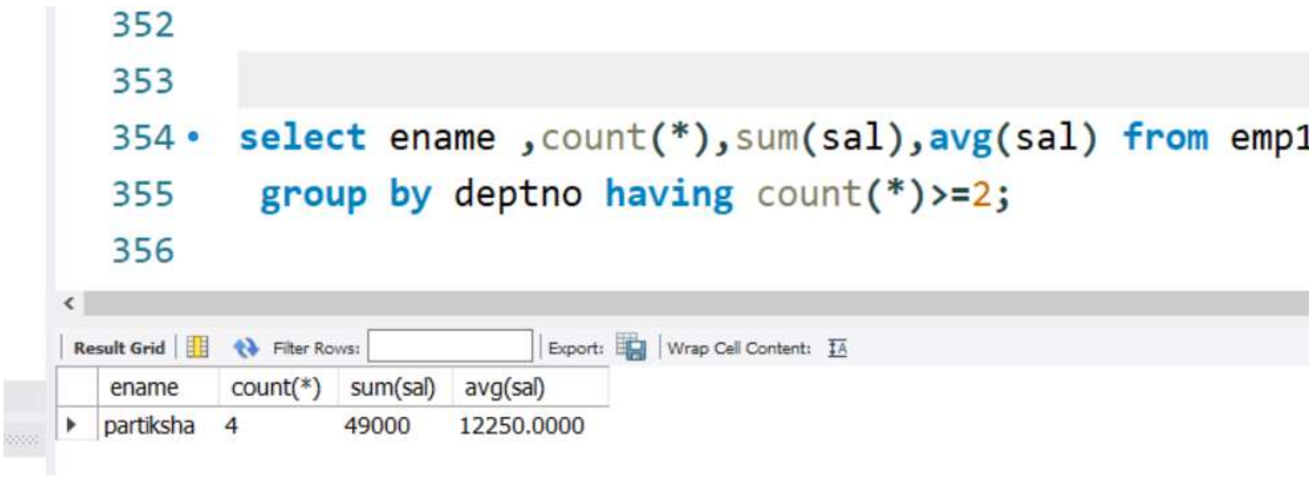
Now one thing observe we made group of deptno but not printed using select command

we select ename here hence 1st emp name displayed among the total things in group

in above image ename= KING is displyed because in the counted group 1st record is with name king employe

Now if we update this king name to >> pratikha then

Pratiskha emp name is display



```
select ename ,count(*),sum(sal),avg(sal) from emp1
 group by deptno having count(*)>=2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

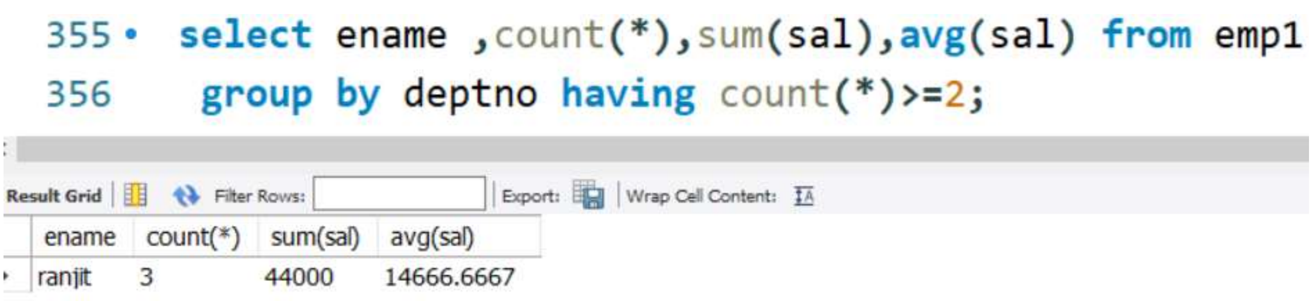
```
ename       count(*)   sum(sal)   avg(sal)
partiksha   4          49000      12250.0000
```

If we delete this record as

Delete from emp1 where empno= 7839;

Now after grouping , in that group 1st emp is ranjit (we group by deptno)

Hence 1st empname is display ranjit >> because we use empname in select

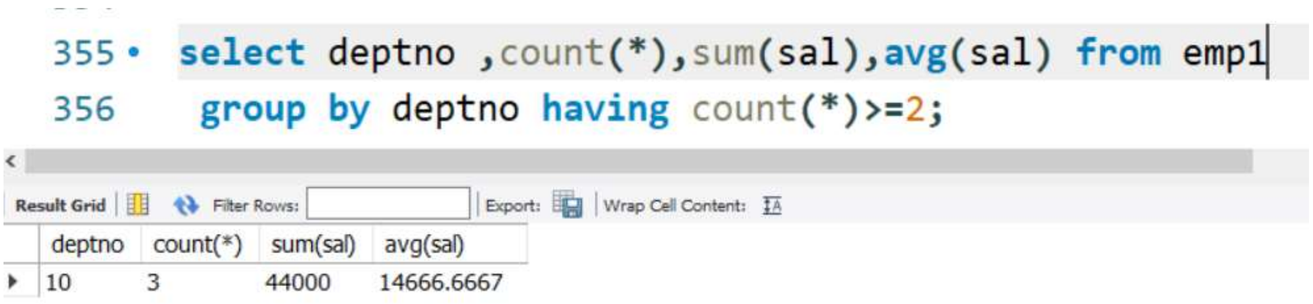


```
select ename ,count(*),sum(sal),avg(sal) from emp1
 group by deptno having count(*)>=2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ename	count(*)	sum(sal)	avg(sal)
ranjit	3	44000	14666.6667

Now if we use dept no in select statement then deptno is displayed



```
select deptno ,count(*),sum(sal),avg(sal) from emp1
 group by deptno having count(*)>=2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

deptno	count(*)	sum(sal)	avg(sal)
10	3	44000	14666.6667

It is recommened and meaningfull use column in select statement by

which we are grouping i.e here group by deptno and also display that deptno grup

group by and select column is should same

we select or print only that which is we grouped

in above fig we group by = deptno hence we select deptno

that means after grouped we print that using select statement hence

it is same

Rollup >> is used to subtoal

1 Group = addition of all values 1st in group

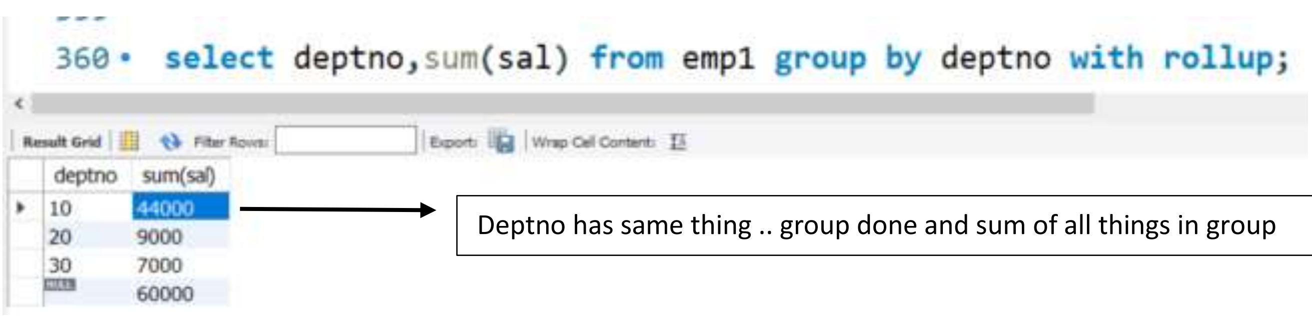
+

2 Group = addition of all values in 2nd group

= addtion on both group >>> is subtoal

Rollup :

Rollup is used to generate subtoal as well grand total



```
select deptno,sum(sal) from emp1 group by deptno with rollup;
```

deptno	sum(sal)
10	44000
20	9000
30	7000
NULL	60000

Deptno has same thing .. group done and sum of all things in group

Here is grouped (similar items)= 1 group and their addiiton

and non similar and their values

final total of all grouped + non grouped

Now

select deptno, job, count(*), sum(sal) from emp1

group by deptno, job with rollup;

1st Done group

Here two groups 1st = deptno 2nd= job ,, job is put in the 1st group

Then count = all things in group

Then sum =sal of all things in group

Rollup is show subtoal (additon in group) and overall addtion all group(grand)

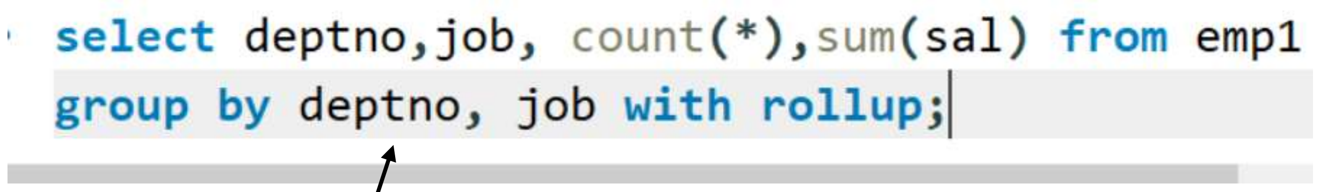
In Rollup

Group by = is responsible for making addition of things in their group

Rollup = is responsible for showing addition value (subtoal for same group)

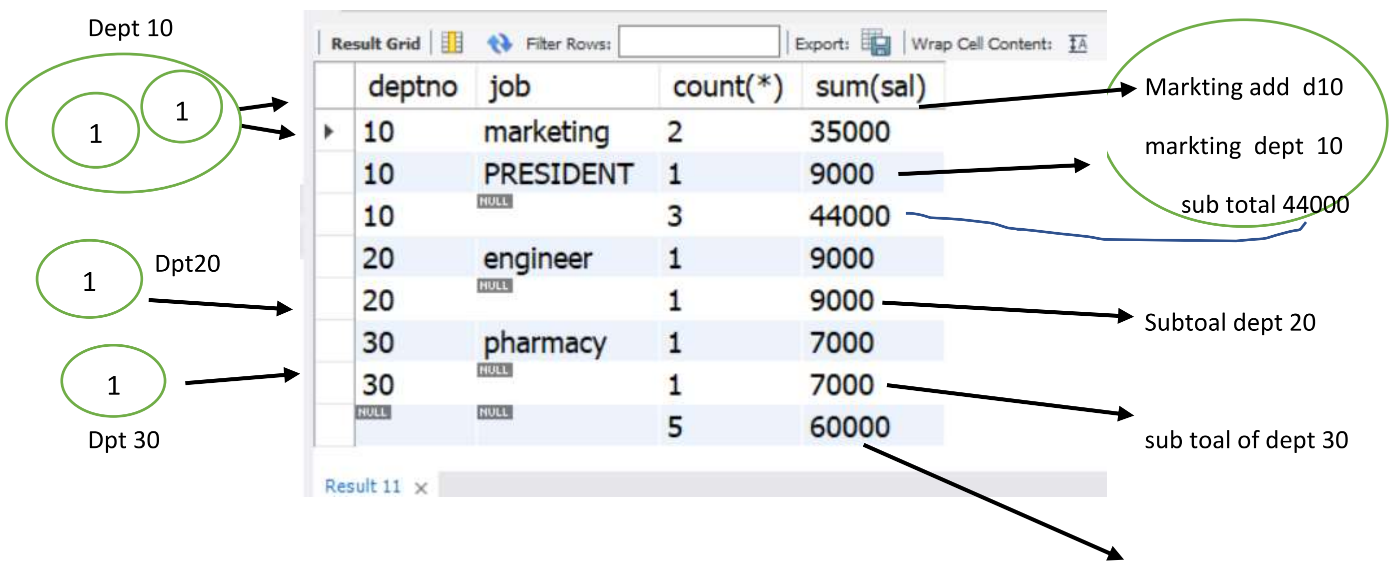
And make addtion of all groups (grand total)

Because rollup is used for subtoal and grand total



```
select deptno,job, count(*),sum(sal) from emp1
group by deptno, job with rollup;
```

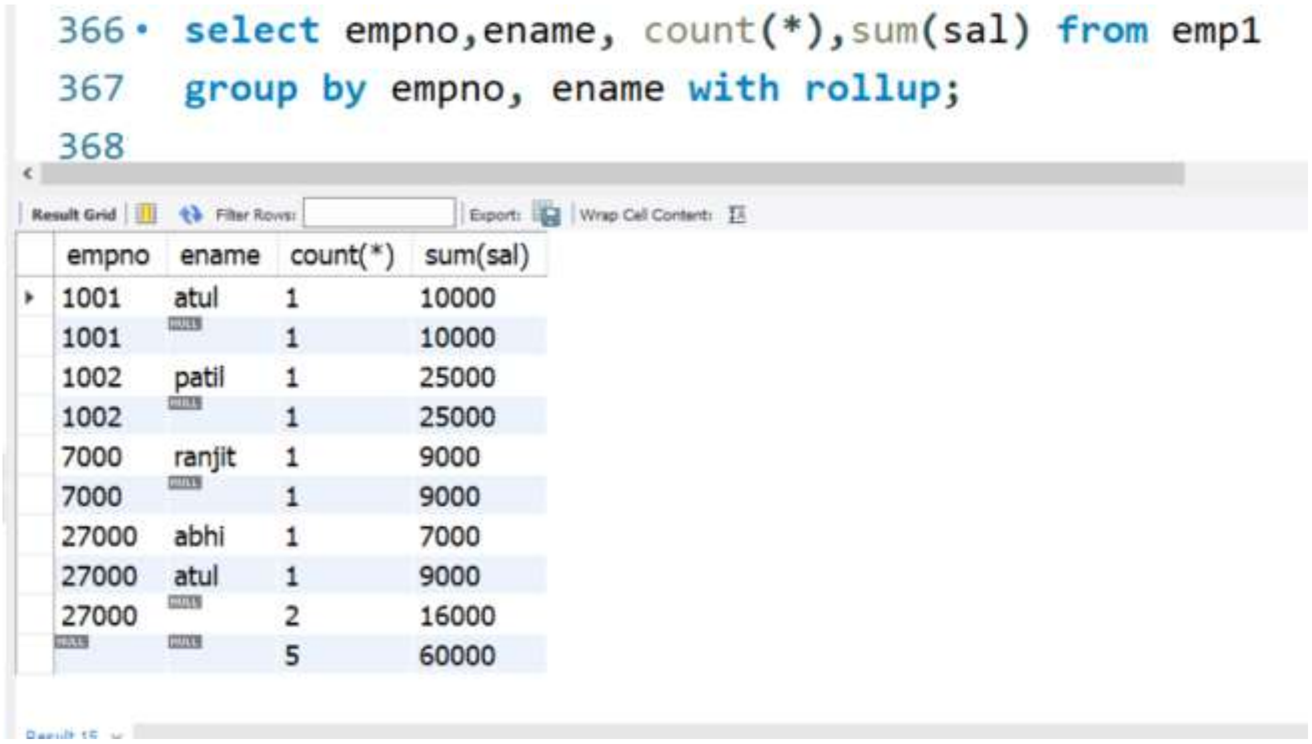
1 group done= similar things



deptno	job	count(*)	sum(sal)
10	marketing	2	35000
10	PRESIDENT	1	9000
10	NULL	3	44000
20	engineer	1	9000
20	NULL	1	9000
30	pharmacy	1	7000
30	NULL	1	7000
NULL	NULL	5	60000

Grand total

All group



```
select empno,ename, count(*),sum(sal) from emp1
group by empno, ename with rollup;
```

empno	ename	count(*)	sum(sal)
1001	atul	1	10000
1001	NULL	1	10000
1002	patil	1	25000
1002	NULL	1	25000
7000	ranjit	1	9000
7000	NULL	1	9000
27000	abhi	1	7000
27000	atul	1	9000
27000	NULL	2	16000
NULL	NULL	5	60000

In above query,

1. Group are done by empno and ename (similar are grouped)

2. Ename are grouped and added it to correspond group of empno

3. The counts how may things in group display in count(*) col

4. Rollup calculates additions of things (subtoal) and show it

5. Also rollup calucates additions of group and show grand total

PARTITIONING IN MYSQL

Partitioning >>

is used to divide the result set in partition and perform operation on it

PARTITION BY gives >> aggregated columns with each record in the specified table.

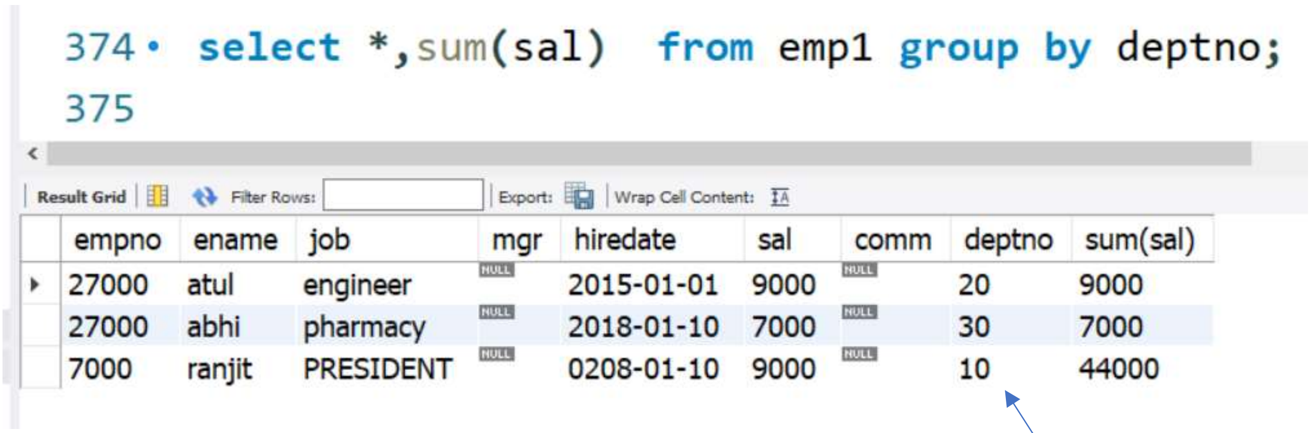
It is always used >> inside OVER() clause

Partition by >> show all rows

No. of records will not be reduced

Group by >> show only one i.e 1st row form group

Reduces the no. of records

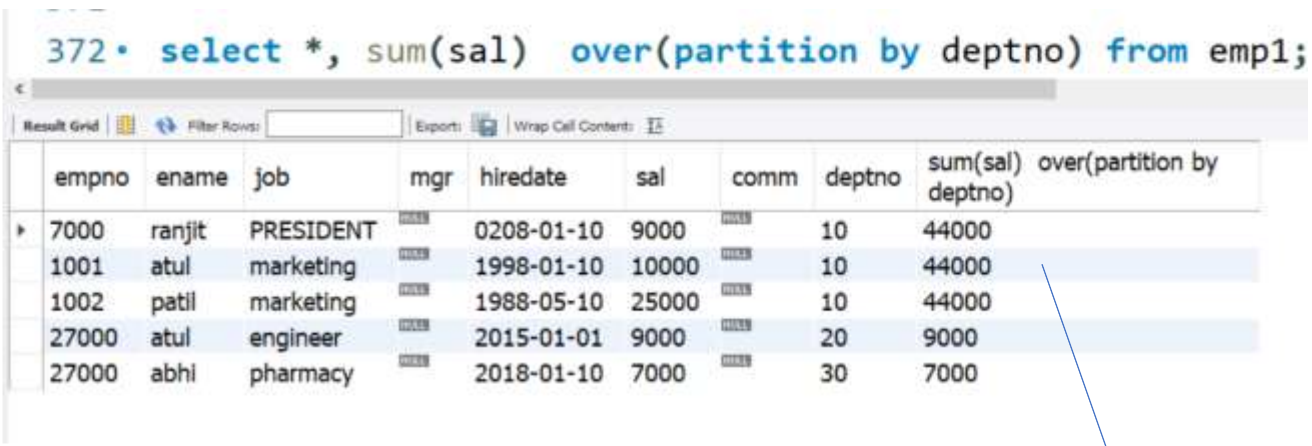


```
select *,sum(sal)  from emp1 group by deptno;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno	sum(sal)
27000	atul	engineer	NULL	2015-01-01	9000	NULL	20	9000
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30	7000
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10	44000

Grouped by dept no using group by

Hence only one 1st row form group



```sql
select *, sum(sal)  over(partition by deptno) from emp1;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno	sum(sal) over(partition by deptno)
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10	44000
1001	atul	marketing	NULL	1998-01-10	10000	NULL	10	44000
1002	patil	marketing	NULL	1988-05-10	25000	NULL	10	44000
27000	atul	engineer	NULL	2015-01-01	9000	NULL	20	9000
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30	7000

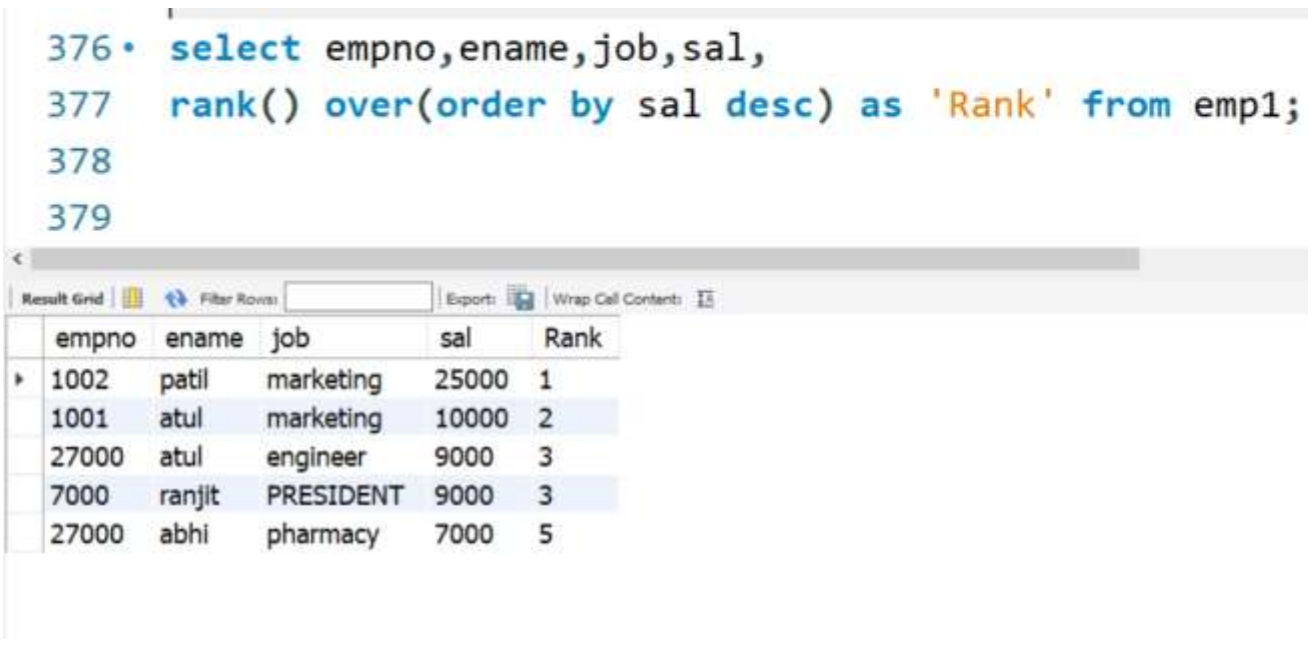
Partition by all row

Rank Function:

The RANK() function in MySQL will display the rank of a row

Rank() function is used with over clause   over (……partition/order by)

Over () >> previous function is applied on over as



```sql
select empno,ename,job,sal,
rank() over(order by sal desc) as 'Rank' from emp1;
```

empno	ename	job	sal	Rank
1002	patil	marketing	25000	1
1001	atul	marketing	10000	2
27000	atul	engineer	9000	3
7000	ranjit	PRESIDENT	9000	3
27000	abhi	pharmacy	7000	5

select empno ,ename, job, sal,

rank() $\boxed{\text{over(order by sal desc)}}$ as 'Rank'   from emp1;

in above query rank function is applied on over , how over is ordered by in descending order on sal column

on that desc order rank is given

>> main use of over , the function used before over clause eg. Rank(), Sum() is applied on over(...)

Over(...) >> makes partition as in argument

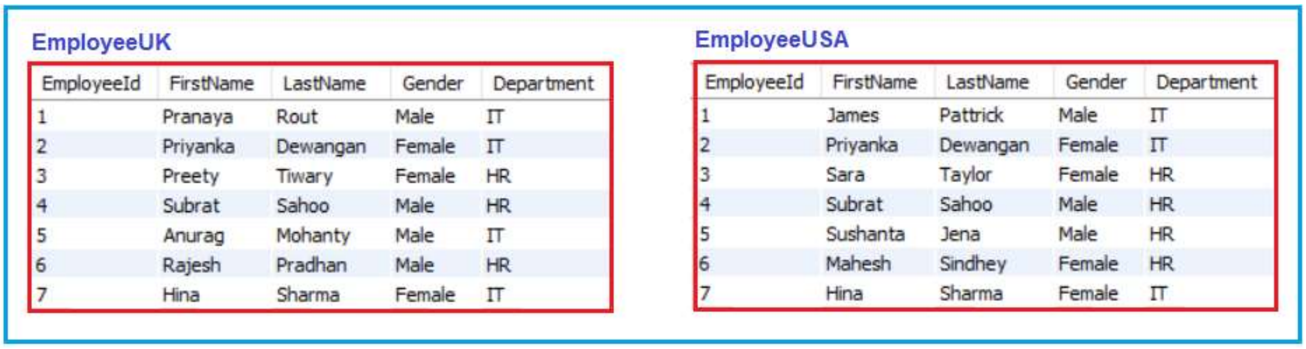
Set Operaton on mysql

Set = group of elemnts

Sets in mathematics, are simply a collection of distinct objects

Union: combine the two table or column of same table or different table

Combine element form set A and set B, but not duplicate i.e. disctints



EmployeeUK

EmployeeId	FirstName	LastName	Gender	Department
1	Pranaya	Rout	Male	IT
2	Priyanka	Dewangan	Female	IT
3	Preety	Tiwary	Female	HR
4	Subrat	Sahoo	Male	HR
5	Anurag	Mohanty	Male	IT
6	Rajesh	Pradhan	Male	HR
7	Hina	Sharma	Female	IT

EmployeeUSA

EmployeeId	FirstName	LastName	Gender	Department
1	James	Pattrick	Male	IT
2	Priyanka	Dewangan	Female	IT
3	Sara	Taylor	Female	HR
4	Subrat	Sahoo	Male	HR
5	Sushanta	Jena	Male	HR
6	Mahesh	Sindhey	Female	HR
7	Hina	Sharma	Female	IT

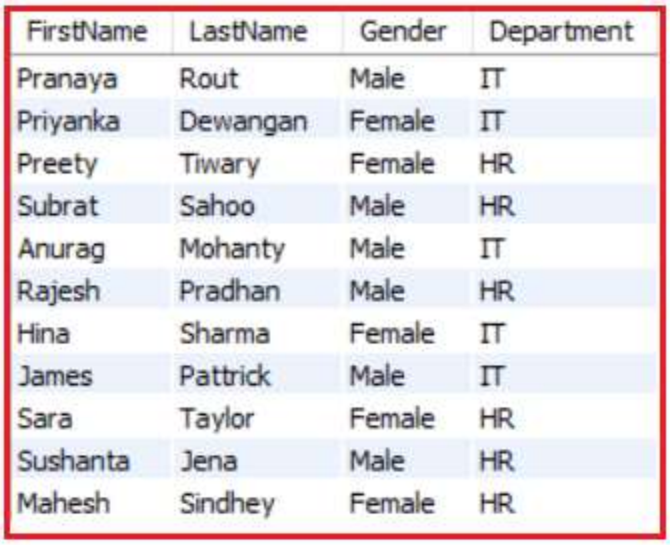
The UNION operator is used to combine the result set of two or more tables SELECT statements into a single result set by removing the duplicate records. Above fig two different tables

Now

```
SELECT FirstName, LastName, Gender, Department FROM EmployeeUK
UNION

SELECT FirstName, LastName, Gender, Department FROM
EmployeeUSA;
```

Here union means combines two tables data but not duplicate

Hence result is



FirstName	LastName	Gender	Department
Pranaya	Rout	Male	IT
Priyanka	Dewangan	Female	IT
Preety	Tiwary	Female	HR
Subrat	Sahoo	Male	HR
Anurag	Mohanty	Male	IT
Rajesh	Pradhan	Male	HR
Hina	Sharma	Female	IT
James	Pattrick	Male	IT
Sara	Taylor	Female	HR
Sushanta	Jena	Male	HR
Mahesh	Sindhey	Female	HR

In this firstName contains form both table empUk and empUSA

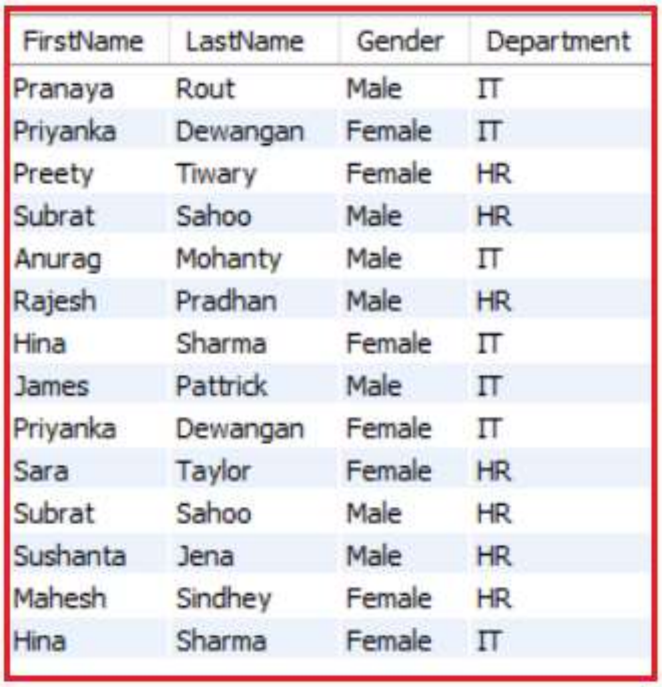
1 st 8 from 1st table

Last 3 from 2nd table

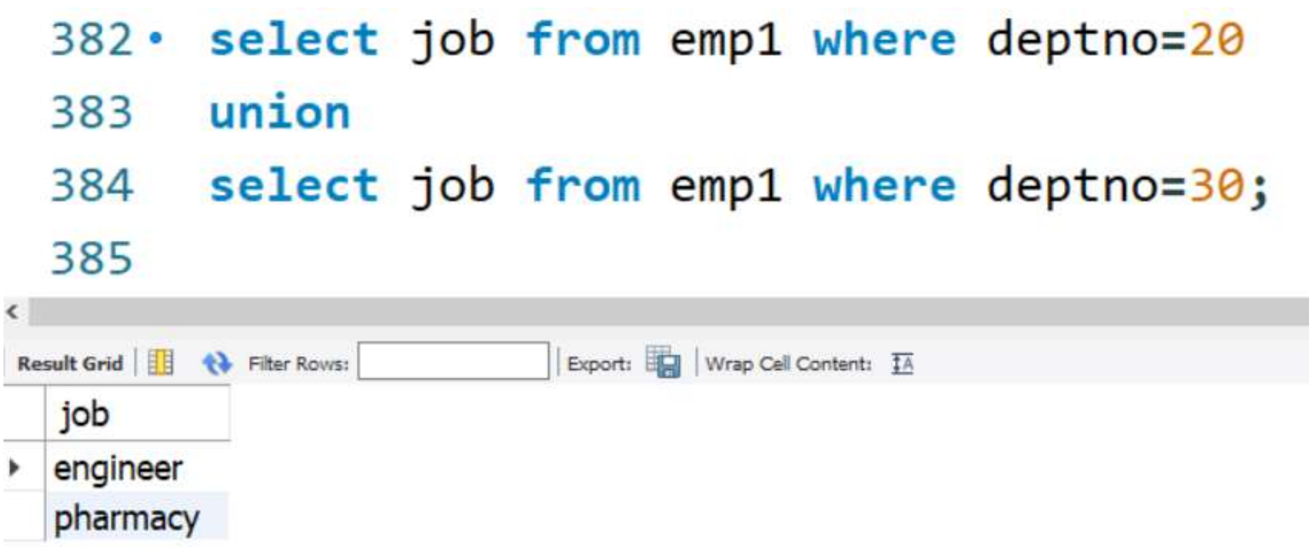
Others are same in both hence taken once

The UNION ALL >> including duplicate value

```
SELECT FirstName, LastName, Gender, Department FROM EmployeeUK
UNION ALL

SELECT FirstName, LastName, Gender, Department FROM
EmployeeUSA;
```



FirstName	LastName	Gender	Department
Pranaya	Rout	Male	IT
Priyanka	Dewangan	Female	IT
Preety	Tiwary	Female	HR
Subrat	Sahoo	Male	HR
Anurag	Mohanty	Male	IT
Rajesh	Pradhan	Male	HR
Hina	Sharma	Female	IT
James	Pattrick	Male	IT
Priyanka	Dewangan	Female	IT
Sara	Taylor	Female	HR
Subrat	Sahoo	Male	HR
Sushanta	Jena	Male	HR
Mahesh	Sindhey	Female	HR
Hina	Sharma	Female	IT



```sql
select job from emp1 where deptno=20
union
select job from emp1 where deptno=30;
```

```
job
engineer
pharmacy
```

Intersect Operator

common in both the result set.

```
SELECT column_lists FROM table_name WHERE condition

INTERSECT ✗

SELECT column_lists FROM table_name WHERE condition;
```

BUT INTERSECT NOT IN mysql

Using IN Operator to achieve INTERSECT functionality:

```
SELECT * FROM EmployeeUK WHERE FirstName IN

(SELECT FirstName FROM EmployeeUSA);
```

```
SELECT column1 [, column2 ] FROM table1 [, table2 ]
[WHERE condition]

EXCEPT ✗

SELECT column1 [, column2 ] FROM table1 [, table2 ]
[WHERE condition]
```

Except operator not in mysql

```
SELECT * FROM EmployeeUK WHERE FirstName NOT IN

(SELECT FirstName FROM EmployeeUSA);
```

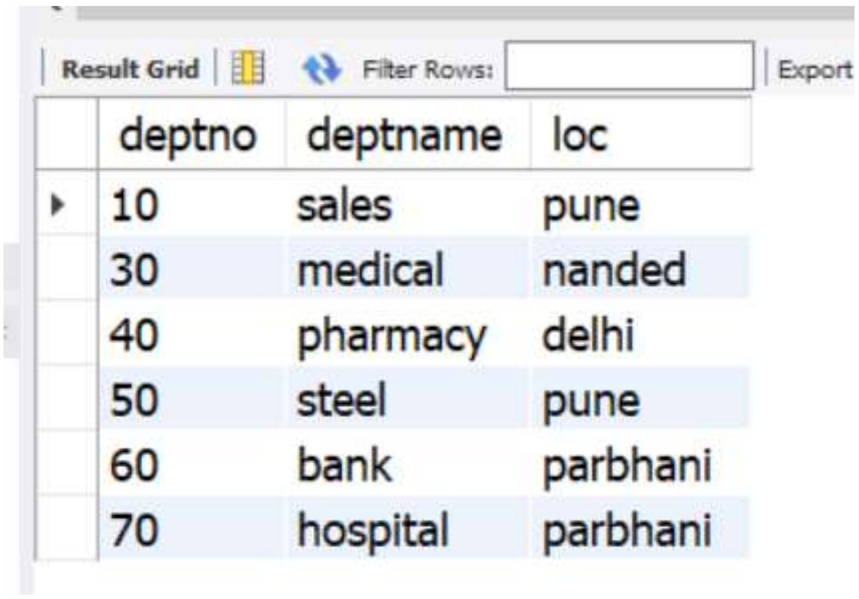
Join in mysql

Combining data from more than one table

It is used to retirve data form multiple table

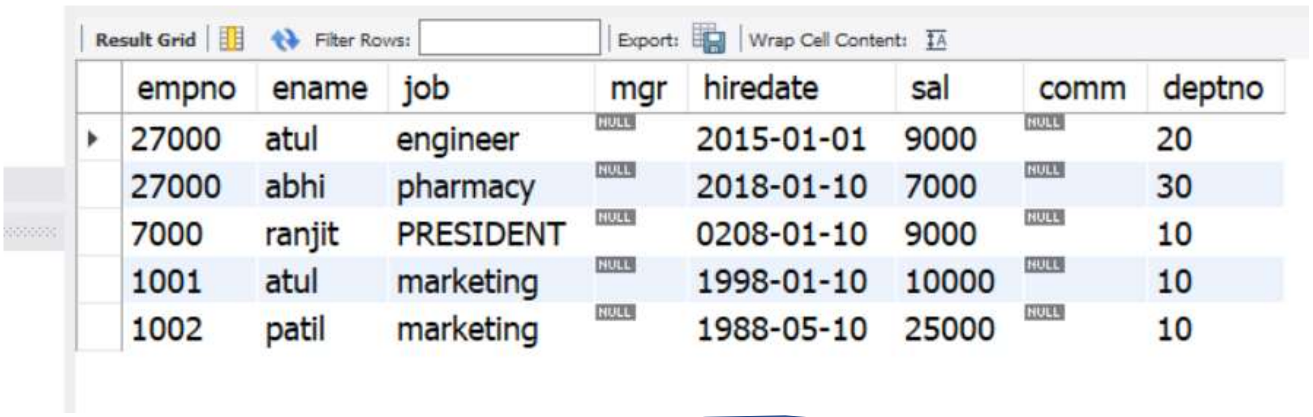
Fetch records from multiple table

Dept table



deptno	deptname	loc
10	sales	pune
30	medical	nanded
40	pharmacy	delhi
50	steel	pune
60	bank	parbhani
70	hospital	parbhani

Emp table

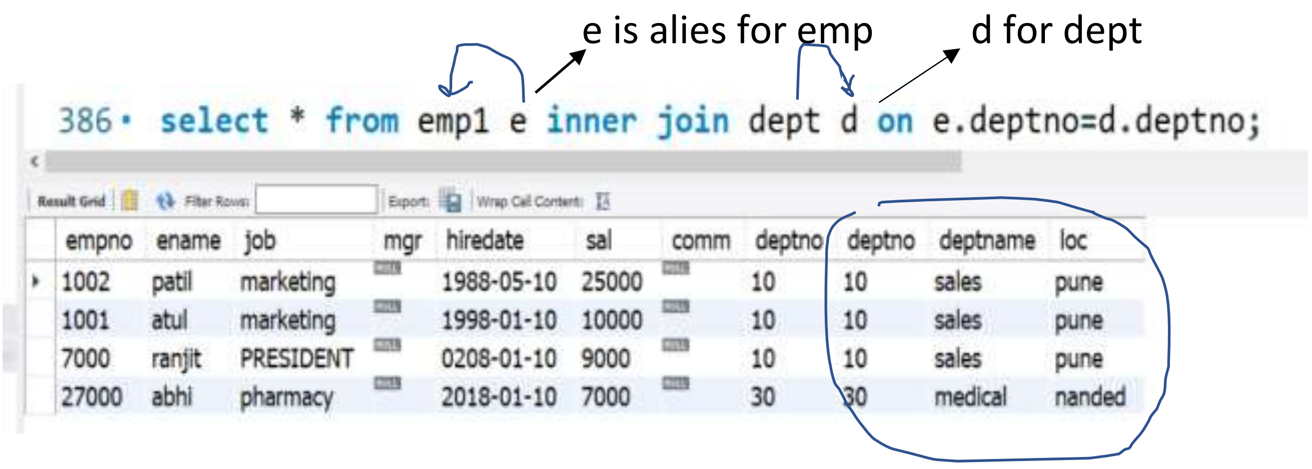


empno	ename	job	mgr	hiredate	sal	comm	deptno
27000	atul	engineer	NULL	2015-01-01	9000	NULL	20
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10
1001	atul	marketing	NULL	1998-01-10	10000	NULL	10
1002	patil	marketing	NULL	1988-05-10	25000	NULL	10

Now …display the ditails of employee (emp table) also details of

Department (dept table)

here we need to join record or columns of two table using join



```
386 •  select * from emp1 e inner join dept d on e.deptno=d.deptno;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno	deptno	deptname	loc
1002	patil	marketing	NULL	1988-05-10	25000	NULL	10	10	sales	pune
1001	atul	marketing	NULL	1998-01-10	10000	NULL	10	10	sales	pune
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10	10	sales	pune
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30	30	medical	nanded

1st table emp　　　　**2nd** table　**Dept**

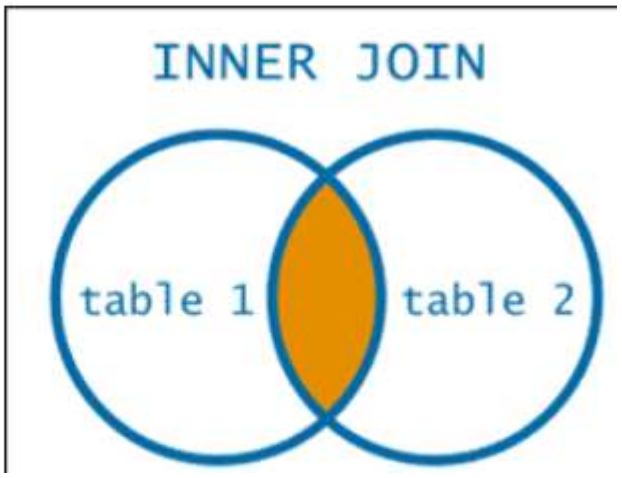
While joing dept number form 1st table , Is checked in 2nd table if matches

Then only join because only intersect is joined

Here

Join or inner is performed on columns detpno of two table

Join are joins same value in both table



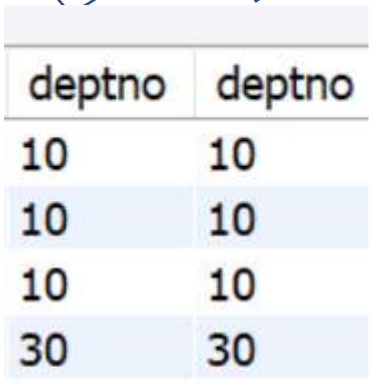
Both matched

1st table row

(all row / select stmt)

2nd table row

all row/selected



deptno	deptno
10	10
10	10
10	10
30	30

While joing dept number form 1st table , Is checked in 2nd table if matches

Then only join because only intersect (same row) is joined

Select ename ,deptname from emp1 e join dept d on e.deptno=d.deptno;

Ename from Emp table + deptname from dept table hence say join

In above query we selecting ename form 1 table and deptname from

1 table . i.e. two different table hence say to join but how join

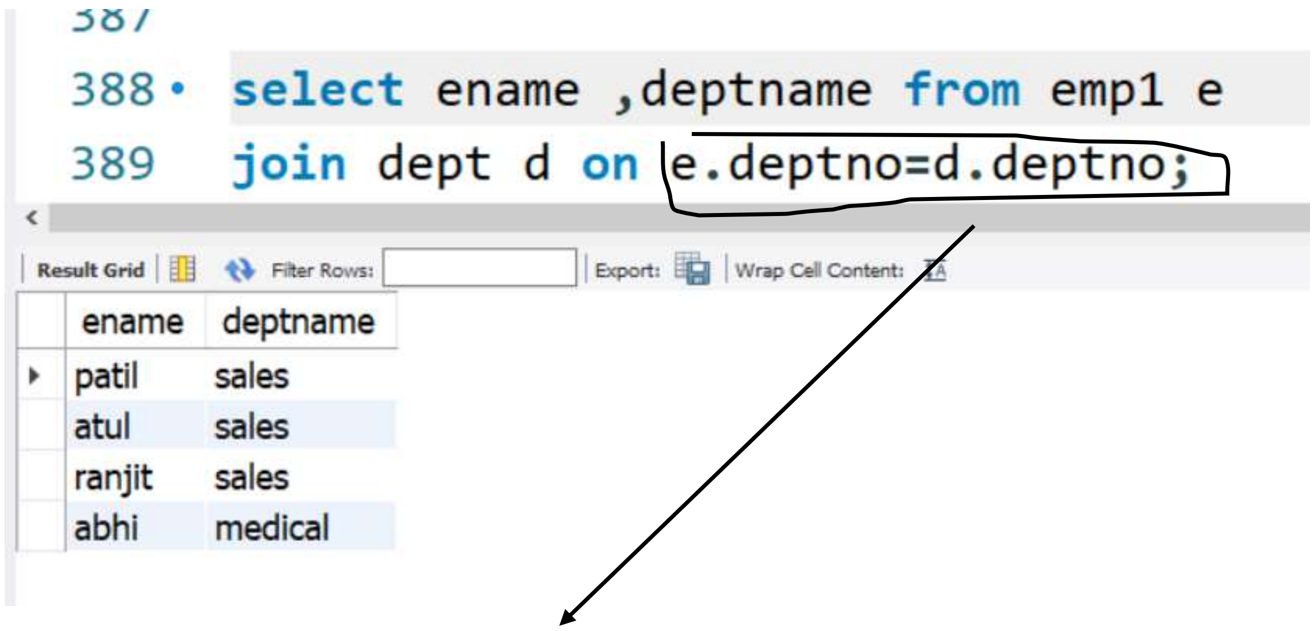
Combine common rows form e.deptno=d.deptno

e.Deptno = from emp table and

d.deptno= from dept table

join = is intersect join only common row in specified colomn name

as join is retrive data from two different table in single select statement



```
select ename ,deptname from emp1 e
join dept d on e.deptno=d.deptno;
```

ename	deptname
patil	sales
atul	sales
ranjit	sales
abhi	medical

As join is performed on deptno of both table , and join only matched rows. After join show related row form both table but not all colmon only ename and deptname as we select in statement

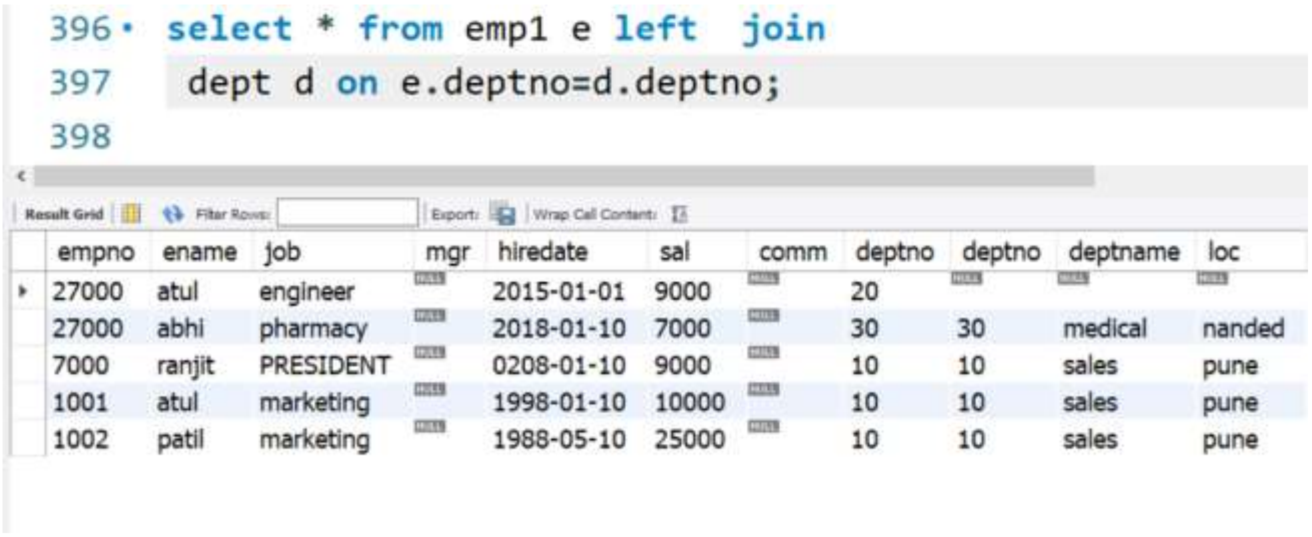
Left JOIN

:- Get all records from the table1 (**LEFT** table1) and the matched records from the
table2 (**RIGHT** table2). If no match the result is NULL from the table2.

:- if No matched row found then null displayed.

now observe in inner join only matched rows are join and after correspondeace row are dispalyed .

but if we want display all records form 1st table .. and then perform join them with all (but inner join only matched show), here all and after show their corrspondace value or row , if now row then show null



```sql
select * from emp1 e left  join
 dept d on e.deptno=d.deptno;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno	deptno	deptname	loc
27000	atul	engineer	NULL	2015-01-01	9000	NULL	20	NULL	NULL	NULL
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30	30	medical	nanded
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10	10	sales	pune
1001	atul	marketing	NULL	1998-01-10	10000	NULL	10	10	sales	pune
1002	patil	marketing	NULL	1988-05-10	25000	NULL	10	10	sales	pune

e.deptno=d.deptno >> are mathced , if match show their corr value

if not match then show null… here dpetno 20 not matched (as it is not both table) hence show null

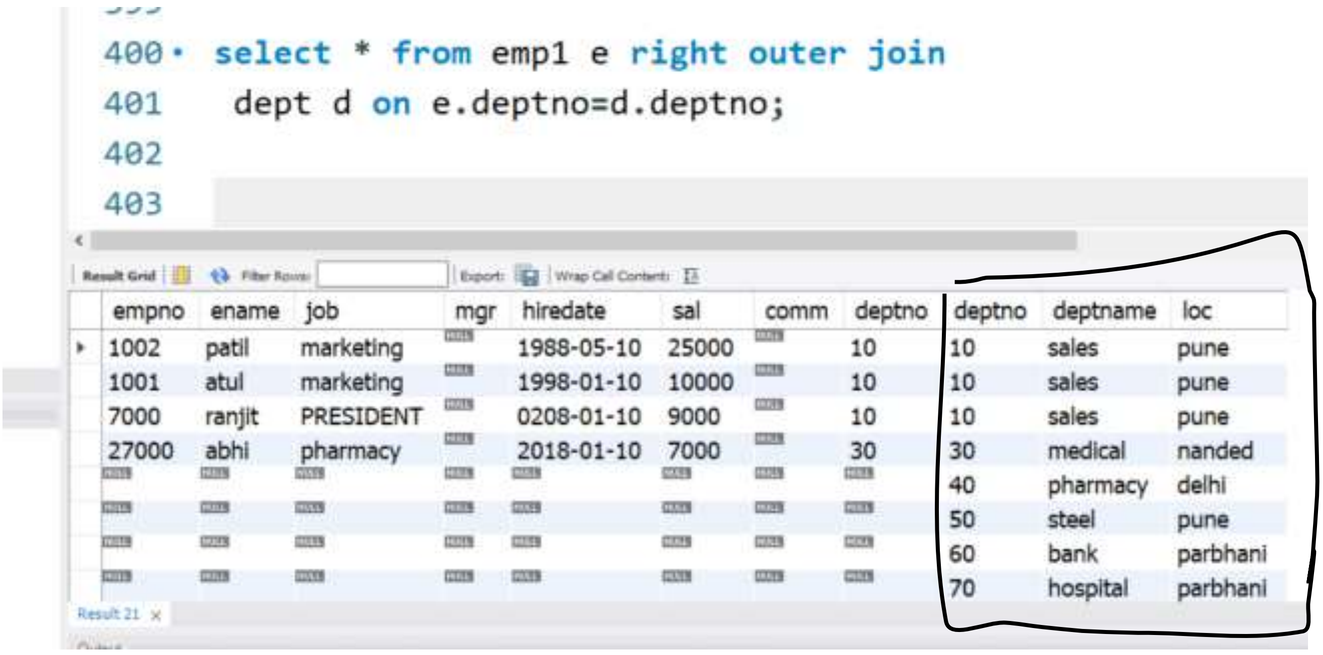
in left join >>> 1st display all row form left table i.e 1st table

then match with 2nd table , if match show corr value. No match show null

Right Outer Join

Get all records from the table2 (**RIGHT** table1) and the matched records from

the table1 (**LEFT** table2). If no match the result is NULL from the table1.



```sql
select * from emp1 e right outer join
 dept d on e.deptno=d.deptno;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno	deptno	deptname	loc
1002	patil	marketing	NULL	1988-05-10	25000	NULL	10	10	sales	pune
1001	atul	marketing	NULL	1998-01-10	10000	NULL	10	10	sales	pune
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10	10	sales	pune
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30	30	medical	nanded
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	40	pharmacy	delhi
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	50	steel	pune
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	60	bank	parbhani
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	70	hospital	parbhani

Right join >> 2nd table all rows are displayed and match to 1st one if

matches then show correspond value

If no matches show null value

FULL JOIN

All records from both table

MySQL does not support **`FULL JOIN not in mysql`**

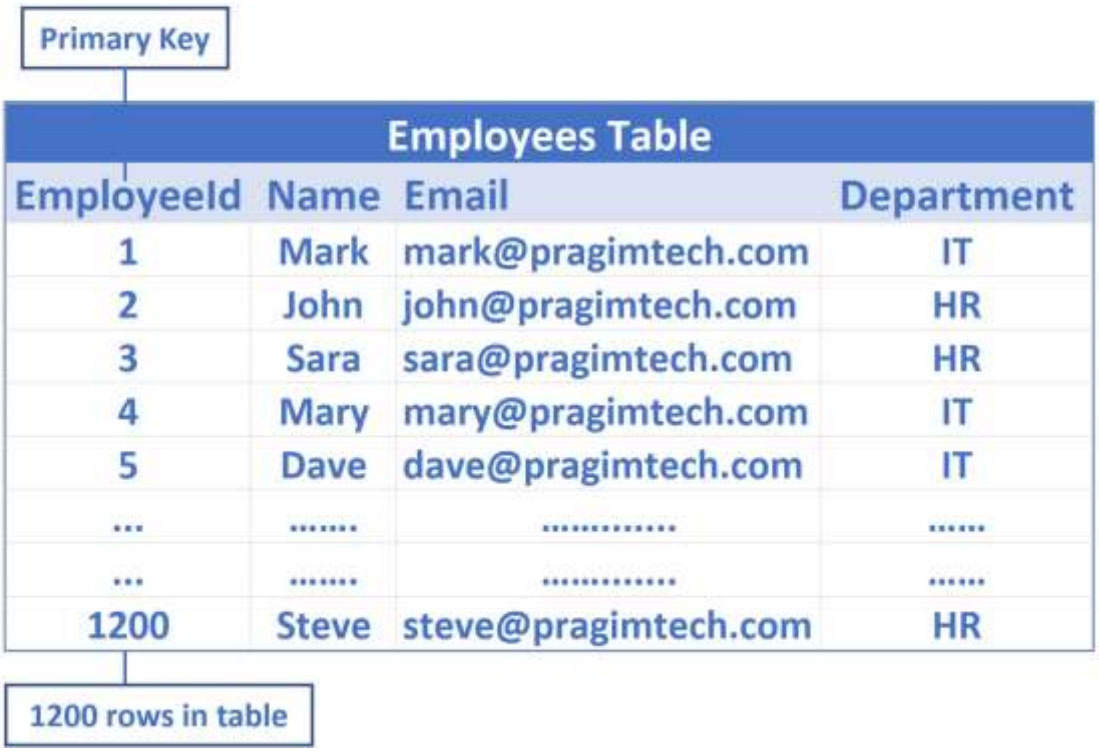
➔use union all

Union all applied on only same column in both table

INDEX in Mysql

It enables you to improve the faster retrieval of records on a database table

In index colmns are sorted



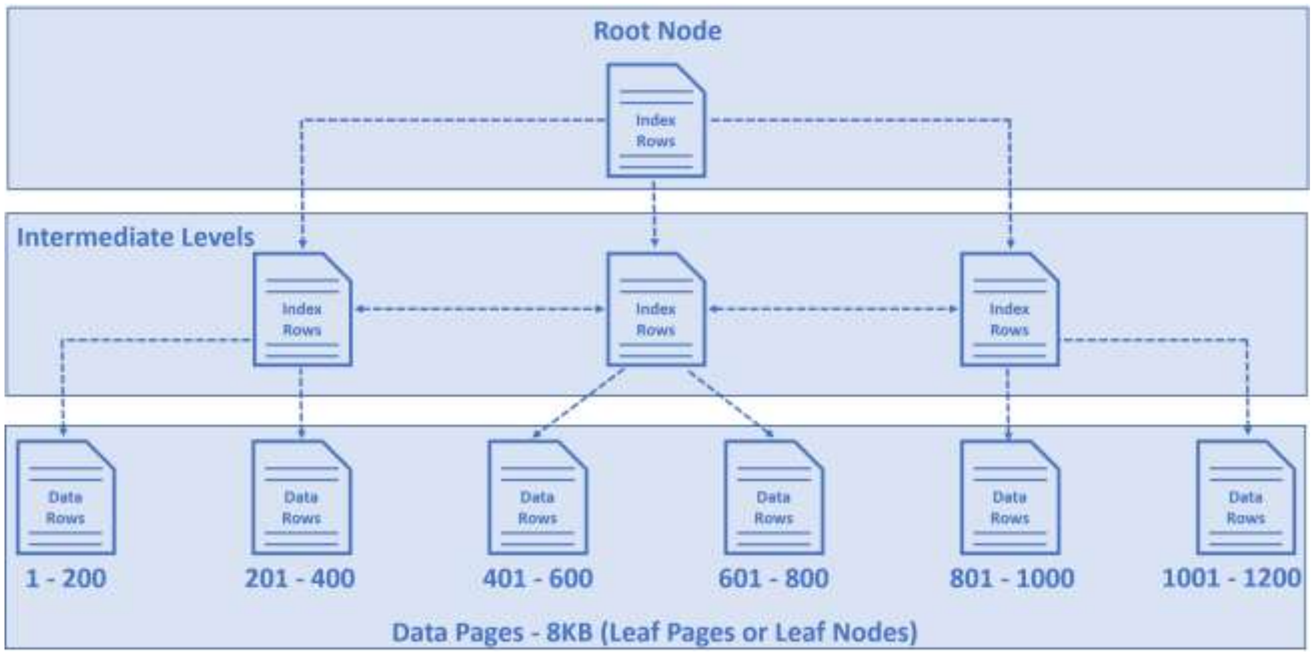
Primary Key
1200 rows in table

Employees Table

EmployeeId	Name	Email	Department
1	Mark	mark@pragimtech.com	IT
2	John	john@pragimtech.com	HR
3	Sara	sara@pragimtech.com	HR
4	Mary	mary@pragimtech.com	IT
5	Dave	dave@pragimtech.com	IT
...	....	....	......
...	....	....	......
1200	Steve	steve@pragimtech.com	HR

`EmployeeId` is the primary key, so by default a clusterd index on the EmployeeId column is created.

This means employee data is sorted by EmployeeId column and physically stored in a series of data pages in a tree like structure that looks like the following.

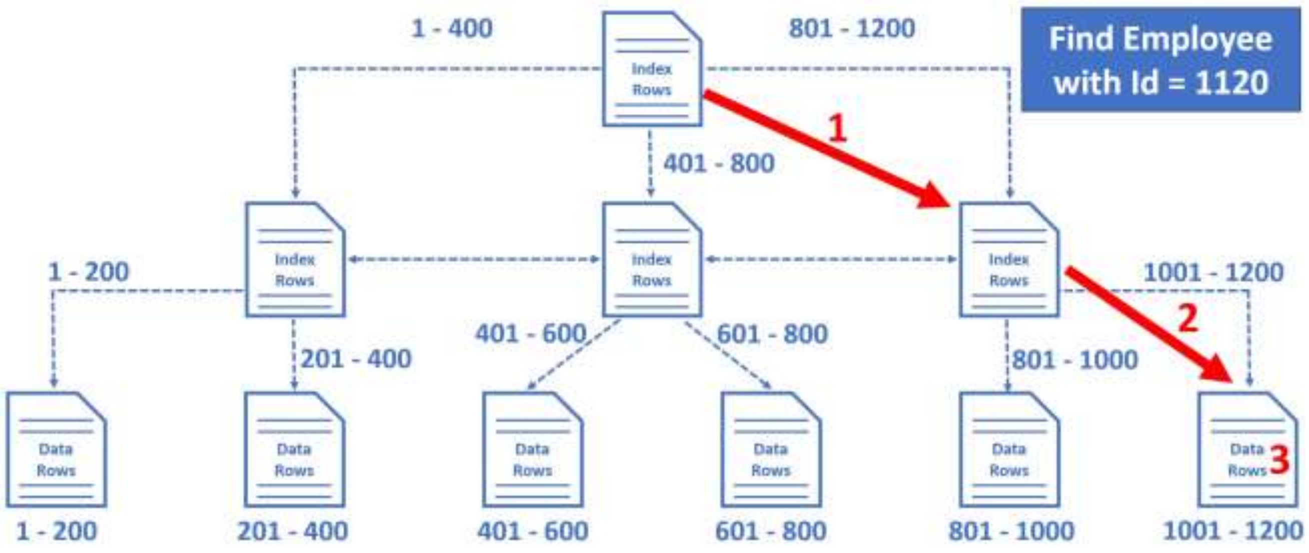


The nodes at the bottom of the tree are called data pages or leaf nodes and contain the actual data rows, in our case employee rows

These employee rows are sorted by `EmployeeId` column, because EmployeeId is the primary key and by default a clusterd index on this column is created.

For our example, let's say in `Employees` table we have 1200 rows and let's assume in each data page we have 200 rows.

The node at the top of the tree is called Root Node.

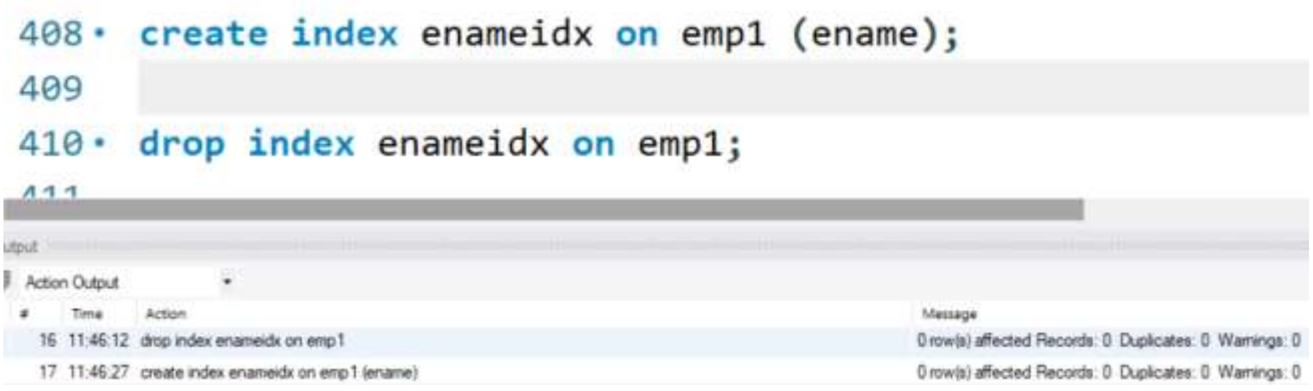


Notice in just 3 operations we find record

Cluster index >> default index >> primary key

What if we serach by Employee name?   At the moment, there is no index on the Name column

Create using create index command



```
create index enameidx on emp1 (ename);

drop index enameidx on emp1;
```

```
utput
Action Output
#   Time      Action                                    Message
16  11:46:12  drop index enameidx on emp1               0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
17  11:46:27  create index enameidx on emp1 (ename)    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

View in mysql

View is like window throug which we can access records.

Logical data only no physically

Some we perform same query again again , we do save this file .sql or .txt file and

Again run this

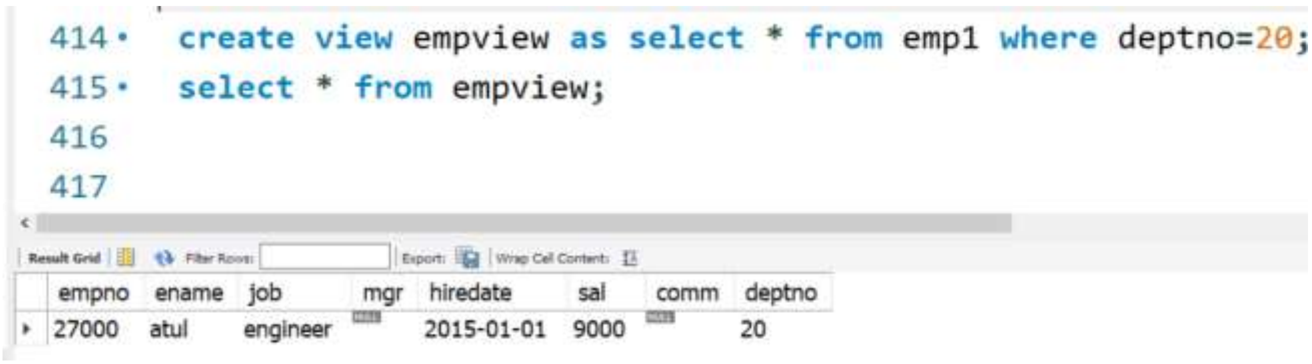
But we can store this on server this called database view or simple view

:- access only specific row of table

:- insert, update, delete >> only on specific row

➔ following example create a view with name …… that provides access to only

Details of emp who work in deptno 20 only



```
create view empview as select * from emp1 where deptno=20;
select * from empview;
```

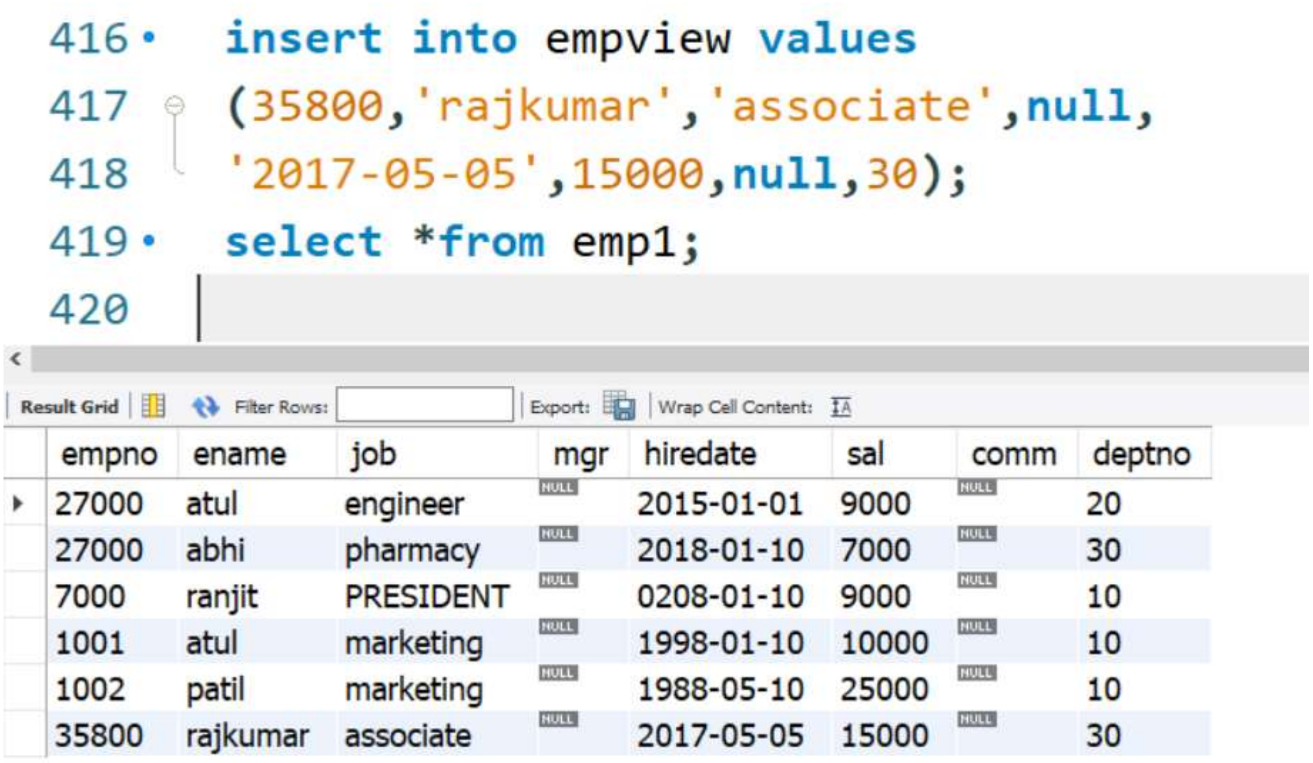
empno	ename	job	mgr	hiredate	sal	comm	deptno
27000	atul	engineer	NULL	2015-01-01	9000	NULL	20

When we use select with empview only dept 20 row is display

:- access only specific row of table

Now once view is created we use it like normal table we perform insert update etc on view

Any Operation on view >>> atuomatically perform on table also drop view empview ;



```
insert into empview values
(35800,'rajkumar','associate',null,
'2017-05-05',15000,null,30);
select *from emp1;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
27000	atul	engineer	NULL	2015-01-01	9000	NULL	20
27000	abhi	pharmacy	NULL	2018-01-10	7000	NULL	30
7000	ranjit	PRESIDENT	NULL	0208-01-10	9000	NULL	10
1001	atul	marketing	NULL	1998-01-10	10000	NULL	10
1002	patil	marketing	NULL	1988-05-10	25000	NULL	10
35800	rajkumar	associate	NULL	2017-05-05	15000	NULL	30

Here no need to use to_str_date() >> because data type of column is date

Now observe while creating view we use create view as select statement

<u>Hence while only selecting data (select command) from view the conditon is check</u>

Hence only dpetno 20 data is display

But while inserting not check , so we can insert directly

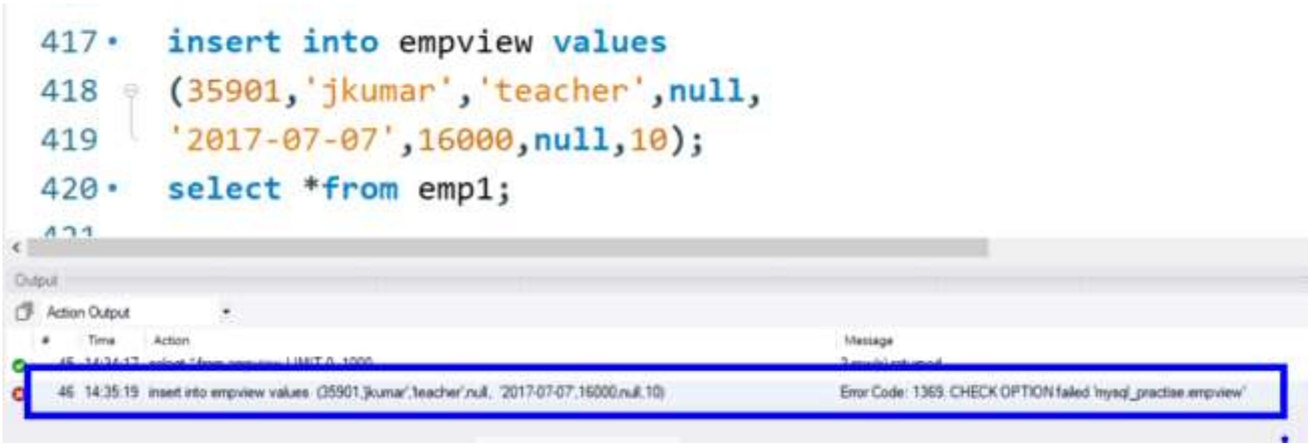
Select view data >> condition is check

Insert view >> by default not check

We mudy add check option

```
create view empview as select * from emp1
where deptno=20 with check option;
```

Now if try to insert row in view error message will display



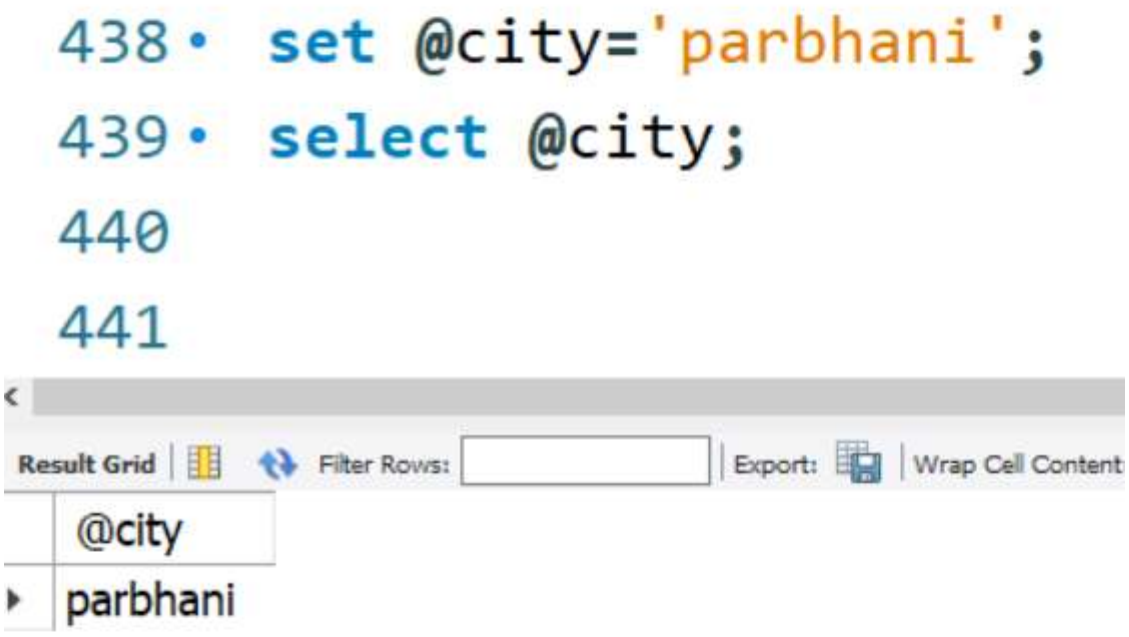
```
insert into empview values
(35901,'jkumar','teacher',null,
'2017-07-07',16000,null,10);
select *from emp1;
```

```
Output
Action Output
#   Time      Action                                                                    Message
45  14:24:17  select * from empview LIMIT 0, 1000                                      2 row(s) returned
46  14:35:19  insert into empview values (35901,'jkumar','teacher',null, '2017-07-07',16000,null,10)   Error Code: 1369. CHECK OPTION failed 'mysql_practise.empview'
```

Error check option failed

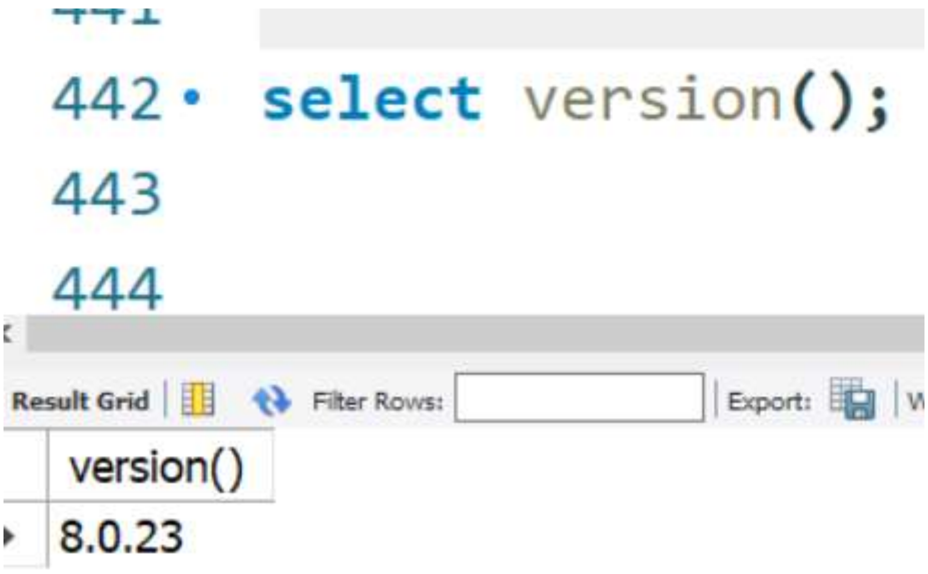
TSQL

We can declare a variable in MySQL with the help of SET command. Before declaring a variable we need to prefix the symbol ‘@’



```
set @city='parbhani';
select @city;
```

```
@city
parbhani
```



```
select version();
```

```
version()
8.0.23
```

Varibles in Mysql

there are two types of variable

1. Ordinary SQL variable >> with @
2. Local varible   >> without @

Ordinary sql variable >> (maintain data until termination of connection) prefixed with @ , these varibale can be used in stored procedures as they are ordinary sql varible.

Local Varible >> these are the varible without @

:- generally used in store procedure , function

:- they must delcared with DECLARE keyword before used

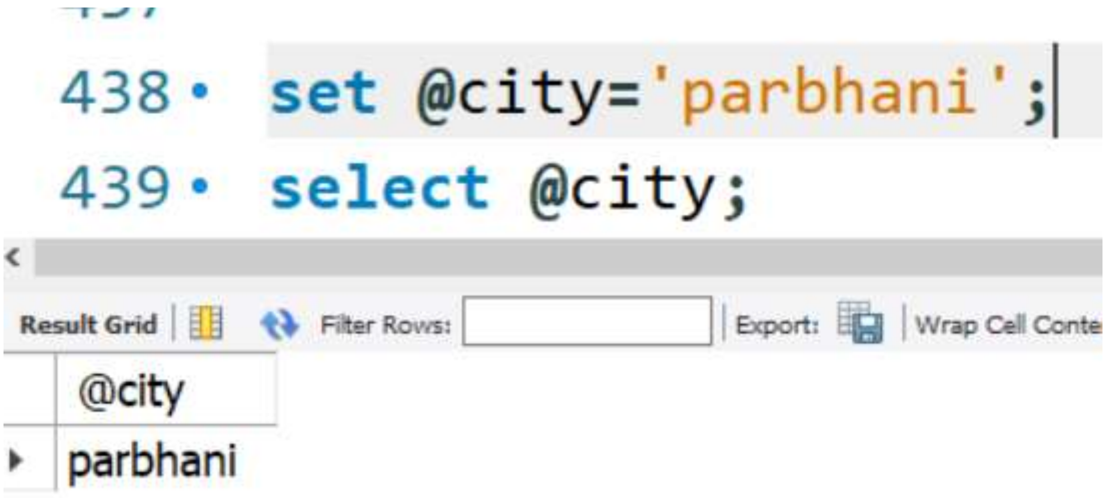
:- the data is lost as soon as function or procedure termniated

:- declared in BEGIN..END and 1st line decare before other command In the begin..end

:- syntax

DECLARE var1,var2,… datatype [DEFAULT value]

Ordinary variable without delcare keyword, using @



```
set @city='parbhani';
select @city;
```

```
@city
parbhani
```

Note that , when you use begin..end, function, procedure then there is block of statement but if we put ; at every statement then sql can determine whole block , it just consider all statement are separate or indiviudals

hence we must change default delimiter i.e ; (semi colon)

use command in sql command line prompt

```
mysql> delimiter //
```

now default delimiter is changed to //

you can again back to normal as

```
mysql>delimiter ;
```

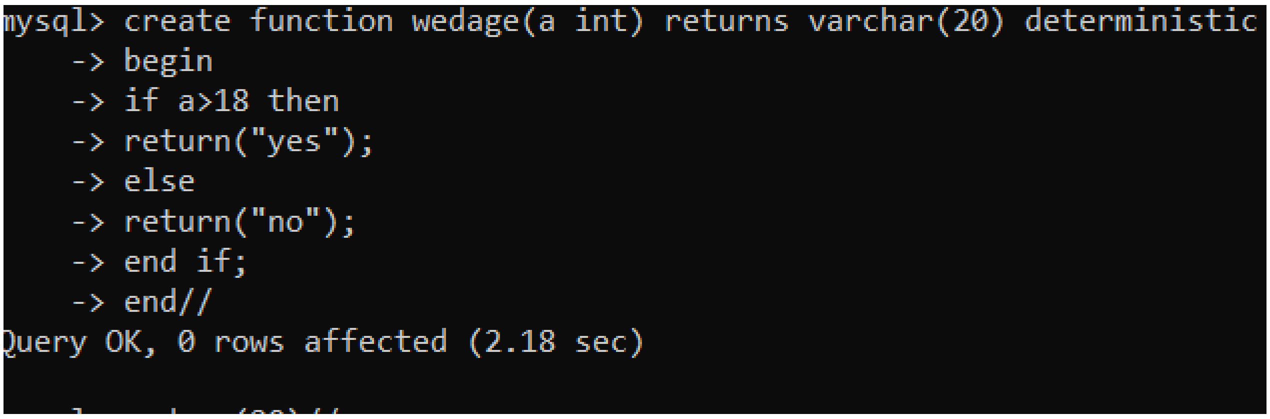
store procedure in mysql

stored procedure > > collection on pre compiled sql statement

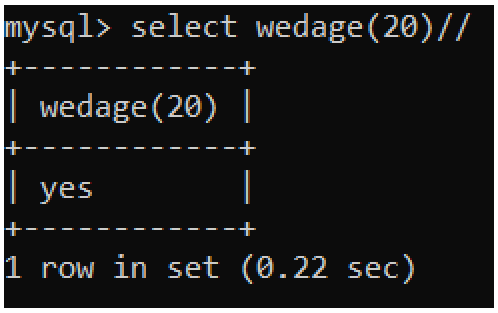
```
mysql> delimiter //
mysql> create procedure p2()
    -> begin
    -> select count(*) as 'All row' from emp1;
    -> end//
Query OK, 0 rows affected (1.08 sec)
mysql> call p2()//
```

```
+---------+
| All row |
+---------+
|      8 |
+---------+
1 row in set (0.23 sec)
Query OK, 0 rows affected (0.23 sec)
mysql>
```

function is smiliar like procedure but it return value



```
mysql> create function wedage(a int) returns varchar(20) deterministic
    -> begin
    -> if a>18 then
    -> return("yes");
    -> else
    -> return("no");
    -> end if;
    -> end//
Query OK, 0 rows affected (2.18 sec)
```



```
mysql> select wedage(20)//
+------------+
| wedage(20) |
+------------+
| yes        |
+------------+
1 row in set (0.22 sec)
```

MySQL Transaction

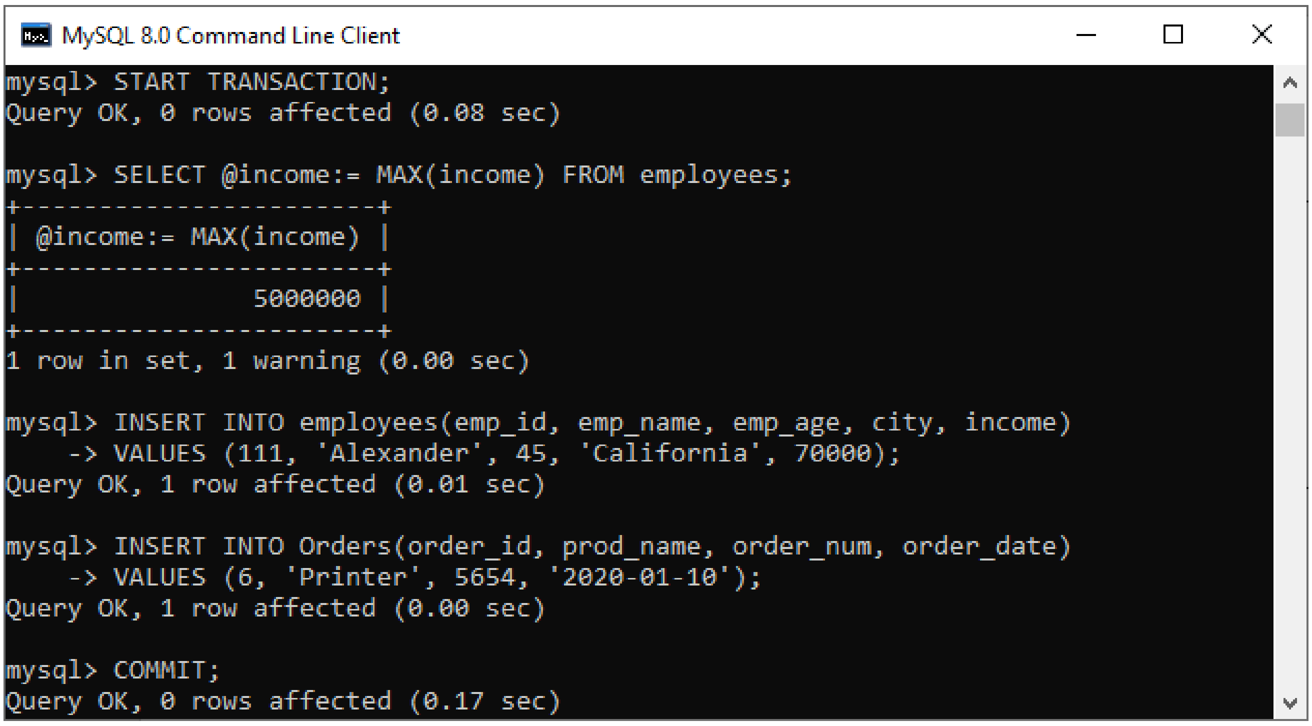
Either all modification is successful when the transaction is committed.

Or, all modifications are undone when the transaction is rollback.

- SET autocommit = OFF:
- SET autocommit = ON:

See in below as we started transaction as START TRANSACTION … After that we commint chages

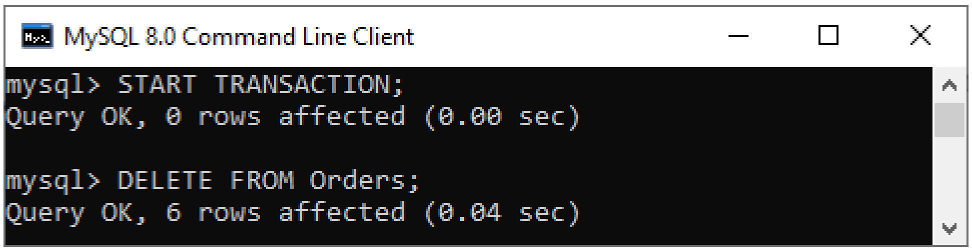
If we not commit changes are not made in database or table



MySQL 8.0 Command Line Client

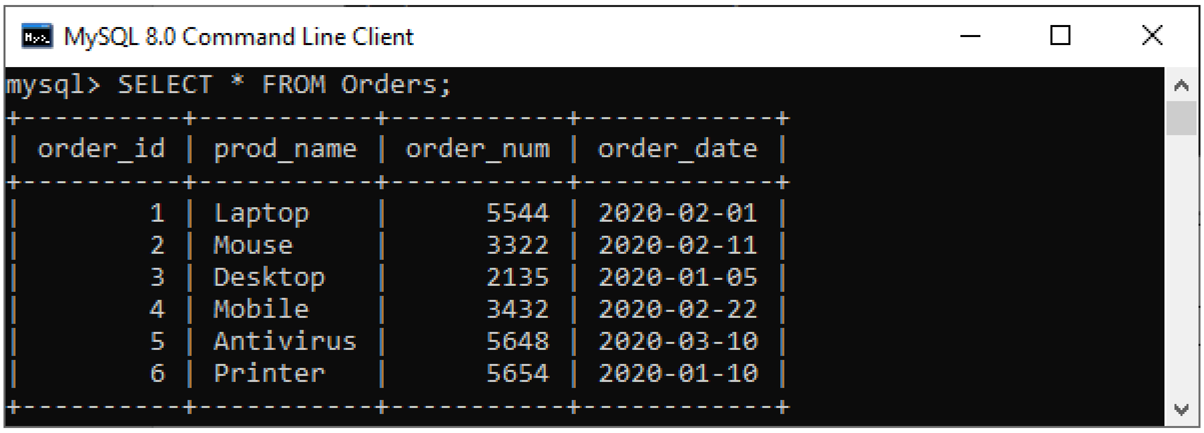
```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.08 sec)

mysql> SELECT @income:= MAX(income) FROM employees;
+-----------------------+
| @income:= MAX(income) |
+-----------------------+
|               5000000 |
+-----------------------+
1 row in set, 1 warning (0.00 sec)

mysql> INSERT INTO employees(emp_id, emp_name, emp_age, city, income)
    -> VALUES (111, 'Alexander', 45, 'California', 70000);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Orders(order_id, prod_name, order_num, order_date)
    -> VALUES (6, 'Printer', 5654, '2020-01-10');
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.17 sec)
```

Now


MySQL 8.0 Command Line Client

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM Orders;
Query OK, 6 rows affected (0.04 sec)
```

Start transaction , then execute statement


MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM Orders;
+----------+-----------+-----------+------------+
| order_id | prod_name | order_num | order_date |
+----------+-----------+-----------+------------+
|        1 | Laptop    |      5544 | 2020-02-01 |
|        2 | Mouse     |      3322 | 2020-02-11 |
|        3 | Desktop   |      2135 | 2020-01-05 |
|        4 | Mobile    |      3432 | 2020-02-22 |
|        5 | Antivirus |      5648 | 2020-03-10 |
|        6 | Printer   |      5654 | 2020-01-10 |
+----------+-----------+-----------+------------+
```

Here record will display because we set autocommit off and then start transaction but

Not commited chages

Therefore if we want to make changes permanent, use the COMMIT statement. Otherwise,

execute the   ROLLBACK statement to roll back the changes in the first session.

Triggers

Triggers are stored sub programs that are automatically execucated based on specific event

Function , procedure >> called explicitaly

Triggers   >> called automatically

Based on event Triggers are classified on

DDL triggers … create alter etc

DML trigges .. insert, update etc