Data base- Database is integrated collection of related information along with the details so that it is available to the several user for the different application.

Row or Record or Tuple

Database Name

Student

Entity

Table Name: Computer Science

Column of Field or Attribute

Roll Number	Name	Address	Fee
1	Rahul	Delhi	10000
2	Raj	Mumbai	5000
3	Rohit	Kolkata	15000

Database

Ex-

Table Name: Users

user_id	password
Sam12	Xyz
Rony23	Zxy
John90	Qwerty
Jemie	iuytr23

Table Name: Page

page_name	likes
Creekyshoes	3000
Etc	10000
Other	50000

Table

Table / Entity - The Table is the structure inside database that contains data, organized in columns and rows.

Student_ID	Name	Address
1	Rahul	Delhi
2	Sonam	Mumbai
3	Sumit	Bihar
4	Nilu	Kolkata

Column / Field / Feature / Attribute - The name of each column in a table is used to interpret its meaning and is called an attribute.

Row / Record / Tuple - Each row in a table represents a record and is called a tuple.

Raw data to Database -

- Rahul, 101, Delhi, rahul@geekyshows.com
- Sona, 102, Kolkata, sona@geekyshows.com
- Sumit, 103, sumit@geekyshows.com, Ranchi
- Sam, sam@geekyshows.com, Noida, 104

Name	Roll	Address	Email_ID
Rahul	101	Delhi	rahul@geekyshows.com
Sona	102	Kolkata	sona@geekyshows.com
Sumit	103	Ranchi	sumit@geekyshows.com
Sam	104	Noida	sam@geekyshows.com

Database Management System (DBMS) - DBMS is the software that allows the access to the data in the database.

Database ← DBMS ← User

Ex- dBase, Microsoft Access, LibreOffice Base, Fox Pro etc.

Relational Database Management System (RDBMS) -

RDBMS avoided the navigation model as in old DBMS and introduced Relational model. The relational model has relationship between tables using primary keys, foreign keys and indexes. Thus the ~~fast~~ fetching and storing of data become faster than the old Navigational model. RDBMS is useful to efficiently manage vast amount of data and is used in large business application.

Ex- SQL Server, Oracle, MySQL, MariaDB, SQLite etc.

Structured Query Language (SQL) - It is commonly used with all relational database for data definition and manipulation.

Features of SQL-

- It is an non procedural language.
- It is an English-like language.
- It can process a single record as well as set of records at a time.
- All SQL statement define what is to be done

rather than how it is to be done.

- SQL has facilities for defining database views, security, transaction etc.

CMD Command-

- mysqld --initialize --console (create root user with temporary password

- mysql -u root -p (Start a client as superuser "root")

- alter user 'root'@'localhost' identified by 'newpassword' (Chang password).

- quit - (to close MySQL)

- ctrl+c - (to close the MySQL Server)

- mysqld --console (Start the MySQL Database Server)

- cd mysql/bin (Shortcut to go the bin folder)

Note- पहले MySQL Server को Start करें फिर user login करके अपने Database को access करें।

help contents - to get help about the RDBMS/DBMS.

Create Database - It is used to create database.

Syntax - CREATE DATABASE database_name;

Ex- CREATE DATABASE my-db;

Guideline for creation of Database -

- Name should start with an alphabet
- Blank space and single quote are not allowed.
- Reserve words of that RDBMS/DBMS can not be used as database name

Note- यहाँ keywords case sensitive नहीं होते हैं।

USE - This is used to tell your RDBMS/DBMS that you want to use this database.

Syntax- USE database_name;

Ex- USE my_db;

Create Table -

Syntax- CREATE TABLE table_name
(
Column_name1 data_type (size),
Column_name2 data_type (size),
Column_name3 data_type (size)
);

Ex- CREATE TABLE my-tab
(
name varchar (30),
roll number (4),
address varchar (100)
);

Name	Roll	Address

(option) constraint)

Syntax- CREATE TABLE table_name
(
Column_name1 data_type (size) [constraints],
Column_name2 data_type (size) [constraints],
Column_name3 data_type (size) [constraints]
);

Note- constraints एक ~~row~~ column के लिये एक से ज्यादा भी हो सकते हैं। constraints लिखना optional है।

Ex- CREATE TABLE my_tab
(
name varchar(30) NOT NULL,
roll number(4) PRIMARY KEY,
address varchar(100)
);

Guideline for creation of Table-

- Table name should start with an alphabet
- In table name, blank spaces and single quotes are not allowed.
- Reserve words of that RDBMS/DBMS cannot be used as table name
- Proper data type and size should be specified.
- Unique column name should be specified.

DESC - This is used to describe your table. DESC only describes structure of table not the information (rows) inside table. DESC is short form of describe.

Syntax- DESC table_name;

Ex- DESC my-tab;

→

Field	Type	Null	Key	Default	Extra
name	varchar(50)	Yes		NULL	
roll	int(4)	Yes		NULL	

SHOW DATABASES- This command is used to view all database name.

Syntax- SHOW DATABASES;

SHOW TABLES- This command is used to view all table of current database.

Syntax- SHOW TABLES;

Ex- SHOW DATABASES;

→

Database
information_schema
my_db
mysql
performance_schema
sys

Ex- SHOW TABLES;

Table_in_my_db
my_tab

Date ___/___/___

Data Types-

- INT or INTEGER- It holds whole number between -32768 and 32767 either it is negative or positive. It can not hold a decimal number. The maximum number of digits may be specified in parenthesis.
  Syntax- Column_name INT(size);
  Ex- roll INT(5);

- DEC or DECIMAL(size, p)- It holds fixed point numbers. Size is the total number of digits and p is the number of digits after the decimal point. The decimal point and the negative sign '-' are not counted in size. If p is 0, value have no decimal point. The maximum number of size for decimal is 65 and for p 30. if p omitted the default is 0. If size is omitted, the default is 10.
  Syntax- column_name DECIMAL(size, p);
  Ex- price DECIMAL(4,2);

- CHAR or CHARACTER- It holds a fixed length string (can contain letters, numbers and special characters). The fixed size is specified in parenthesis. It can store up to 255 characters.
  Syntax- column_name CHAR(size);

Ex- name CHAR(20);

- VARCHAR - It holds a variable length string (can contain letters, numbers and special characters). The maximum size is specified in parenthesis. It can store up to 255 characters.

Note- If we put a greater value than 255 it will be converted to a TEXT type.

Syntax- column_name VARCHAR(size);

Ex- address VARCHAR(50);

- TEXT - It holds a string with a maximum length of 65535 characters.

Syntax- column_name TEXT;

Ex- address TEXT;

- DATE - It displays Date values in YYYY-mm-dd format.

Syntax- column_name DATE;

Ex- age DATE;

- DATETIME - It displays DATETIME value in YYYY-mm-dd hh:mm:ss format.

Syntax- column_name DATETIME;

Ex- Data_of_join DATETIME;

- TIMESTAMP - It also displays data and time.

Syntax- column_name TIMESTAMP;

Ex- login_dt TIMESTAMP;

Note- It current time & date बताता है।

- TIME - It displays the time value in hh:mm:ss format.

Syntax- column name TIME

Ex- login_t TIME

"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr.A.P.J.Abdul Kalam

Stu_id	Name	Address	DOB	Fee

Stu_id - INT
Name - VARCHAR
Address - TEXT
DOB - DATE
Fee - DEC

→
```
mysql> CREATE TABLE new_tab
    -> (
    -> stu_id INT(5),
    -> name VARCHAR(50),
    -> address TEXT,
    -> dob DATE,
    -> fee DEC(10,2)
    -> );
mysql> DESC new_tab;
```

Field	Type	Null	Key	Default	Extra
stu_id	int(5)	YES		NULL	
name	varchar(50)	YES		NULL	
address	text	YES		NULL	
dob	date	YES		NULL	
fee	decimal(10,2)	YES		NULL	

Data Manipulation Language - help Data Mainpulation

CALL
DELETE
DO
DUAL
HANDLER
IMPORT TABLE
INSERT
INSERT DELAYED
INSERT SELECT
JOIN
LOAD DATA
LOAD XML
REPLACE
SELECT
UNION
UPDATE

Data Definition Language - help Data Definition

AFTER DATABASE
AFTER EVENT
AFTER FUNCTION
AFTER INSTANCE
AFTER LOGFILE GROUP
AFTER PROCEDURE
AFTER SERVER
AFTER TABLE
AFTER TABLESPACE
AFTER VIEW
CONSTRAINT
CREATE DATABSE

CREATE EVENT
CREATE FUNCTION
CREATE INDEX
CREATE LOGFILE GROUP
CREATE PROCEDURE
CREATE SERVER
CREATE SPATIAL REFERENCE SYSTEM
CREATE TABLE
CREATE TABLESPACE
CREATE TRIGGER
CREATE VIEW
DROP DATABASE
DROP EVENT
DROP FUNCTION
DROP INDEX
DROP PROCEDURE
DROP SERVER
DROP SPATIAL REFERENCE SYSTEM
DROP TABLE
DROP TABLESPACE
DROP VIEW
RENAME TABLE
TRUNCATE TABLE

DATA TYPES- help Data Types

AUTO_INCREMENT
BIGINT
BINARY
BIT
BLOB
BLOB DATA TYPE
BOOLEAN

CHAR
CHAR BYTE
DATE
DETETIME
DEC
DECIMAL
DOUBLE
DOUBLE PRECISION
ENUM
FLOAT
INT
INTEGER
LONGBLOB
LONGTEXT
SET DATA TYPE
SMALLINT
TEXT
TIME
TIMESTAMP
TINYBLOB
TINYINT
TINYTEXT
VARBINARY
VARCHAR
~~VAR~~ YEAR DATA TYPE

Data Manipulation -

INSERT INTO - The INSERT INTO statement is used to insert new records / row / tuple in a table.

Syntax - INSERT INTO table_name (column1, column2, ——)
VALUES (value1, 'value2', ———);

Ex - INSERT INTO my_tab (stu_id, name, address, mobil_no)
VALUES (05, 'Anu', 'Delhi', 982112);

my_tab

Stu_id	Name	Address	Mobile no.
05	Anu	Delhi	982112

Rule -

Column and Value order should be same. Any value that goes into a VARCHAR, CHAR, DATE or TEXT column has single quotes ~~around~~ around it. There are no need of quotes for numeric values (INT, DEC)

Need Single Quotes	No Need
CHAR	INT or INTEGER
VARCHAR	DEC or DECIMAL
DATE	
DATETIME	
TIMESTAMP	
TIME	
BLOB	
TEXT	

Note - table में filled सभी चीजों को देखने के लिये SELECT * FROM table_name; का use करते हैं

The work you do is done for your own salva done for your own Swami Vivekananda

```
mysql> CREATE TABLE student
    -> (
    -> stu_id int(4),
    -> name VARCHAR(50),
    -> city VARCHAR(50),
    -> pin INT(20)
    -> );
```

```
mysql> DESC student
```

Field	Type	NULL	Key	Default	Extra
stu_id	int(4)	YES		NULL	
name	varchar(50)	YES		NULL	
city	varchar(50)	YES		NULL	
pin	int(20)	YES		NULL	

```
mysql> INSERT INTO student (stu_id, name, city, pin)
    -> VALUES(01,'Raj','Ranchi',123456);
mysql> SELECT *FROM student
```

stu_id	name	city	pin
1	Raj	Ranchi	123456

INSERT INTO -

1. Without specifing column name -

Syntax - INSERT INTO table_name
VALUES('value1','value2', —);

Ex - INSERT INTO my_tab
VALUES(05,'Anu','Delhi',982112);

my_tab

Stu_id	Name	Address	Mobile_no
05	Anu	Delhi	982112

Rules-

- The value order should be same as column.
- We need to insert record for each column we can not leave any column.

→ mysql>INSERT INTO student
~~->VALUES * FROM student~~
->VALUES (02,'Sona','Delhi', 652345);
mysql> SELECT *FROM student;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345

INSERT INTO -

2- Changing the order of column-

Syntax- INSERT INTO table_name (column2, column1, —)
VALUES ('value2', value1, ——);

Ex- INSERT INTO my_tab (name, stu_id, mobile_no, address)
VALUES ('Anu', 05, 982112, 'Delhi');

my_tab

Stu_id	Name	Address	Mobil_no
05	Anu	Delhi	982112

→ mysql> INSERT INTO student (name, stu_id, pin, city)
->VALUES ('Ram', 03, 879845, 'Kolkata');
mysql>SELECT *FROM student;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845

"All the work you do is done for your own salvation and not for your own benefit." –Swami Vivekananda

INSERT INTO -

3- Insert Data only in specified columns -

Syntax- INSERT INTO table_name(column1, column2, column—)
VALUES (value1, 'value2', 'value—');

Ex- INSERT INTO my_tab (stu_id, name, address)
VALUES (05, 'Anu', 'Delhi');

my_tab

Stu_id	Name	Address	Mobile_no
05	Anu	Delhi	NULL

→ mysql> INSERT INTO student (stu_id, name, pin)
-> VALUES (4, 'Sonu', 764839);
mysql> SELECT * FROM student

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ran	Kolkata	879845
4	Sonu	NULL	764839

INSERT INTO -

4- Insert multiple record at one time -

Syntax- INSERT INTO table_name
(column1, column2, ——)
VALUES (value1, 'value2' —), (value1, value2' —),
——— ;

Ex- INSERT INTO my_tab
(stu_id, name, address, mobile_no)
VALUES (01, 'Anu', 'Delhi', 982112),
(02, 'Rohan', 'Mumbai', 561425);

my_tab

Stu_id	Name	Address	Mobile_no
01	Anu	Delhi	982112
02	Rohan	Mumbai	561425

→ mysql> INSERT INTO student (stu_id, name, city, pin)
→ VALUES (5, 'Shubham', 'HYD', 542367),
→ (6, 'Amit', 'Kanpur', 564897);
mysql> SELECT * FROM student;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764839
5	Shubham	HYD	542367
6	Amit	Kanpur	564897

SELECT- The SELECT statement is used to select data from a database and retrieve the information.

1- Select all columns from the table-
Syntax- SELECT * FROM table_name;
Ex- SELECT * FROM my_tab;

Name	Roll	Mobile
Rahul	01	8753
John	02	6534

2- Select Particular column from the table-
Syntax- SELECT column_name1, Column_name2, ——
FROM table name

Ex- SELECT name, mobile
FROM my_tab

Name	Mobile
Rahul	8753
John	6534

→ mysql> SELECT * FROM student;

Stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764839
5	Shubham	HYD	542367
6	Amit	Kanpur	564897

→ mysql> SELECT ~~stud~~ stu_id, name FROM student;

stu_id	name
1	Raj
2	Sona
3	Ram
4	Sonu
5	Shubham
6	Amit

Single quote Problem-

INSER INTO my_tab (t_id, c_name, address)
VALUES (142, 'K.K's Company', 'Delhi');

Note- 'K.K's Company' में फंस जायेगा। ; देकर इससे बाहर आ सकते हैं।

"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr.A.P.J.Abdul Kalam

There are two way to solve this problem-

1- Use backslash-
Ex- 'K.K\' & Company'

2- Use two time single quote-
Ex- 'K.K'' & Company'

my_tab

T_id	Cname	Address
142	K.K'& Company	Delhi

→ mysql> INSERT INTO student (stu_id, name, city, pin)
-> VALUES (7, 'K.K'' & Company', 'Lucknow', 234345);
mysql> INSERT INTO student (stu_id, name, city, pin)
-> VALUES (8, 'J.K\' & Company', 'Azamgarh', 785246);
mysql> SELECT *FROM student;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764839
5	Shubham	HYD	542367
6	Amit	Kanpur	564897
7	K.K'& Company	Lucknow	234345
8	J.K'& Company	Azamgarh	785246

SELECT with LIMIT:- SELECT LIMIT is used to specify some number of records to display.

Syntax- SELECT column_name1, column_name2, ...
FROM table_name
LIMIT records_number; or LIMIT starting number, ending number records;

"All the work you do, is done for your own salvation, is done for your own benefit." –Swami Vivekananda

Note- RDBMS/DBMS starting_number 0 से शुरू है।

new_tab

stu_id	Name
5	Anu
6	Sonu
7	Anu
8	Jai
9	Bob
10	Candy

Ex- SELECT stu_id, name
FROM new_tab
LIMIT 3;

stu_id	Name
5	Anu
6	Sonu
7	Anu

→ mysql> SELECT * FROM student
-> LIMIT 5;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764839
5	Shubham	HYD	542367

my_sql> SELECT * FROM student
-> LIMIT 1,3;

~~stu~~ stu_id	name	city	pin
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764839

```
mysql> SELECT stu_id, name FROM student
    -> LIMIT 5;
```

stu_id	name
1	Raj
2	Sona
3	Ram
4	Sonu
5	Shubham

WHERE - WHERE is used to search for a specific data.

~~Syntax~~-

1- Specific data from all column-

Syntax-
```
SELECT *
FROM table_name
WHERE column_name operator 'value'
```

Ex- new_tab

stu_id	Name
5	Anu
6	Sonu
7	Anu

(i)
```
SELECT *
FROM new_tab
WHERE name='Anu';
```

stu_id	Name
5	Anu
7	Anu

(ii)
```
SELECT *
FROM new_tab
WHERE stu_id=5;
```

stu_id	Name
5	Anu

2- Specific data from specific column-

Syntax-
```
SELECT column_name
FROM table_name
WHERE column_name operator 'value';
```

Ex-

new_tab

stu_id	Name
5	Anu
6	Sonu
7	Anu

(i)
```
SELECT name
FROM new_tab
WHERE name='Anu';
```

Name
Anu
Anu

(ii)
```
SELECT name
FROM new_tab
WHERE stu_id=5;
```

Name
Anu

Date ___/___/___

Note- Value can be text or numeric. If it is text then we have to put single quote.

Operator	Description
=	Equal
<> or !=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible value for a column.

= (Equal)

→ mysql> SELECT * FROM student;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764839
5	Shubham	HYD	542367
6	Amit	Kanpur	564897
7	K.K's Company	Lucknow	234345
8	J.K's Company	Azamgarh	782246
9	Anu	Delhi	652345

mysql> SELECT * FROM student
-> WHERE city = 'Delhi';

stu_id	name	city	pin
2	Sona	Delhi	652345
9	Anu	Delhi	652345

```
mysql> SELECT stu_id, name. FROM student
    -> WHERE city='Delhi';
```

stu_id	name
2	Sona
9	Anu

<> or != (NOT equal)-

```
-> mysql> SELECT * FROM student
    -> WHERE city != 'Delhi';
```

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Ram	Kolkata	879845
3	Shubham	HYD	542367
5	Amit	Kanpur	564897
7	K.K's Company	Lucknow	234345
8	J.K's company	Azamgarh	785246

> (Greater than) -

```
-> mysql> SELECT * FROM student
    -> WHERE stu_id>4;
```

stu_id	name	city	pin
5	Shubham	HYD	542367
6	Amit	Kanpur	564897
7	K.K's Company	Lucknow	234345
8	J.K's Company	Azamgarh	785246
9	Anu	Delhi	652345

"No religion has mandated killing others as a requirement for its sustenance or propagation." - Dr. A.P.J. Abdul Kalam

< (Less than)-

→ myaql> SELECT * FROM student
-> WHERE stu_id < 4;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845

myaql> SELECT * FROM student
-> WHERE name < 'c' ;

~~student~~

stu_id	name	city	pin
6	Amit	Kanpur	564897
9	Anu	Delhi	652345

Note - कुछ RDBMS name > 'c' में c से छोटा से लेकर c के equal तक भी ले लेते हैं। ये alphabetical order के अनुसार काम करता है।

>= (Greater than or equal)-

→ myaql> SELECT * FROM student
-> WHERE stu_id >= 8;

stu_id	name	city	pin
8	J.K's Company	Azamgarh	785246
9	Anu	Delhi	652345

<= (~~Greater~~ Less than or equal)-

→ myaql> SELECT * FROM student
-> WHERE stu_id <= 2;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345

"All the work you do, is done for your own salvation, not for your own benefits." —Swami Vivekananda

NULL Value - NULL values represent missing unknown data.

Note - NULL ≠ 0

1- IS NULL - This is used to select only the ~~not~~ records with NULL value in the column.

Syntax - SELECT column_name FROM table_name
WHERE column_name IS NULL;

Ex - SELECT emp_id, emp_name, city
FROM emp
WHERE city IS NULL;

→ mysql> SELECT * FROM student
-> WHERE city IS NULL;

stu_id	name	city	pin
4	Sonu	NULL	764879

2- IS NOT NULL - This is used to select only the records with no NULL values in the column.

Syntax - SELECT column_name FROM table_name
WHERE column_name IS NOT NULL;

Ex - SELECT emp_id, emp_name, city
FROM emp
WHERE ~~emp~~ city IS NOT NULL;

→ mysql> SELECT * FROM student
-> WHERE city IS NOT NULL;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Shubham	HYD	542367
6	Amit	Kanpur	564897
7	K.K's Company	Lucknow	234345
8	J.K's Company	Azamgarh	785246
9	Anu	Delhi	652345

AND - The AND operator displays a record if both the first condition AND the second condition are true.

Syntax- SELECT * FROM table_name
WHERE column_name = "Value"
AND column_name = 'value'

Ex- new_tab

stu_id	Name
5	Anu
6	Sonu
7	Anu

SELECT * FROM new_tab
WHERE name = 'Anu'
AND stu_id = 5;

Stu_id	Name
5	Anu

→ mysql> SELECT * FROM student
-> WHERE name = 'sona'
-> AND stu_id = 2;

stu_id	name	city	pin
2	Sona	Delhi	652345

mysql> SELECT * FROM student
-> WHERE name = 'sona'
-> AND stu_id = 1;
Empty set <0.00 sec>

OR - The OR operator displays a record if either the first condition OR the second condition is true.

Syntax - SELECT * FROM table_name
WHERE column_name = 'value'
OR column_name = 'value';

Ex - new_tab

stu_id	Name
5	Anu
6	Sonu
7	Anu

SELECT * FROM new_tab
WHERE name = 'Anu'
OR stu_id = 5;

stu_id	Name
5	Anu
7	Anu

Date ___/___/_____

→ mysql> SELECT * FROM ~~order~~ student
-> WHERE city = 'Delhi'
-> OR name = 'Raj'

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
9	Anu	Delhi	652345

Combination of AND & OR-

Syntax- SELECT * FROM table_name
WHERE column_name = 'value'
AND (column_name = 'value' OR column_name1 = 'value');

Ex- new_tab

stu_id	Name	Address
5	Anu	Delhi
6	Sonu	Mumbai
7	Anu	Delhi

SELECT * FROM new_tab
WHERE name = 'Anu'
AND (stu_id = 5 OR address = 'Kolkata');

stu_id	Name	Address
5	Anu	Delhi
7	Anu	Kolkata

Note- इसी तरह AND और OR के अलग-अलग Combination को use कर सकते हैं।

→ mysql> SELECT * FROM student
-> WHERE city = 'Delhi'
-> AND (stu_id = 2 OR name = 'Raj');

stu_id	name	city	pin
2	Sona	Delhi	652345

```
mysql> SELECT * FROM student
    -> WHERE city = 'Delhi'
    -> OR (stu_id = 1 AND name = 'Raj');
```

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
9	Anu	Delhi	652345

IN- The IN operator allows you to specify to specify multiple values a WHERE clause.

Syntax- SELECT * FROM table_name
WHERE column_name IN ('value1', value2', ...);

Ex- SELECT * FROM new_tab
WHERE name IN('Anu', 'Sonu');

```
→ mysql> SELECT * FROM student
    -> WHERE city IN ('Delhi', 'Ranchi');
```

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
9	Anu	Delhi	652345

```
mysql> SELECT stu_id, name FROM student
    -> WHERE city IN ('Delhi', 'Ranchi');
```

stu_id	name
1	Raj
2	Sona
9	Anu

NOT IN-

Syntax- SELECT *
FROM table_name
WHERE column_name NOT IN ('value', 'value2', ...)

Ex- new_tab

stu_id	Name
5	Anu
6	Sonu
7	Anu
8	Jai
9	Bob
10	Candy

```
SELECT *
FROM new_tab
WHERE name NOT IN ('Anu', 'Sonu');
```

stu_id	Name
8	Jai
9	Bob
10	Candy

→ mysql > SELECT * FROM student
→ WHERE city NOT IN ('Delhi', 'Ranchi');

stu_id	name	city	pin
3	Ram	Kolkata	879845
5	Shubham	HYD	547367
6	Amit	Kanpur	564897
7	K.K's Company	Lucknow	234345
8	J.K's Company	Azamgarh	785246

BETWEEN - The BETWEEN operator selects values within a range. The values can be numbers, text, or dates.

1- Between Number -

Syntax- SELECT * FROM table_name
WHERE column_name BETWEEN value1 AND value2;

Ex- new_tab

student_id	Name
5	Anu
6	Sonu
7	Anu
8	Jai

SELECT * FROM newtab
WHERE stu_id BETWEEN 6 AND 8;

student_id	Name
6	Sonu
7	Anu
8	Jai

Note- कुछ RDBMS 6 और 8 को include नहीं करते हैं।

mysql> SELECT * FROM student
-> WHERE stu_id BETWEEN 3 AND 6;

stu_id	name	city	pin
3	Ram	Kolkata	879845
4	Sonu	NULL	764839
5	Shubham	HYD	542367
6	Amit	Kanpur	564897

```
mysql> SELECT stu_id, city FROM student
    -> WHERE stu_id BETWEEN 2 AND 4;
```

stu_id	city
2	Delhi
3	Kolkata
4	NULL

2- Between Text -

Syntax- SELECT * FROM table_name
WHERE column_name
BETWEEN 'value1' AND 'value2';

Ex- SELECT * FROM new_tab
WHERE name BETWEEN 'B' AND 'J';

Note- include B but not J.

→
```
mysql> SELECT * FROM student
    -> WHERE name BETWEEN 'A' AND 'H';
```

stu_id	name	city	pin
6	Amit	Kanpur	564897
9	Anu	Delhi	652345

3- BETWEEN Date -

Syntax- SELECT * FROM table_name
WHERE column_name
BETWEEN 'YYYY/mm/dd' AND 'YYYY/mm/dd';

Ex- new_tab

DOB	Name
2000-01-04	Anu
2001-02-12	Sonu
2000-12-25	Anu
2003-04-22	Jai
2004-05-23	Bob
2001-02-14	Candy

"All the work you do, is done for your own salvation; is done for your own benefit." —Swami Vivekananda

SELECT * FROM new_tab
WHERE date
BETWEEN '2000-01-04' AND '2001-02-14';

DOB	Name
2000-01-04	Anu
2001-02-12	Sonu
2000-12-25	Anu
2001-02-14	Candy

Note- including both 2000/01/04 And 2001/02/14 dates

→ mysql> SELECT * FROM stu_tab;

stu_id	name	dob
1	Raj	2000-01-04
2	Sonu	2001-02-12
3	Anu	2000-12-25
4	Jai	2001-02-14
5	Sona	2002-02-04

mysql>SELECT * FROM stu_tab
->WHERE dob BETWEEN '2000-12-25' AND '2001-12-26';

stu_id	name	dob
2	Sonu	2001-02-12
3	Anu	2000-12-25
4	Jai	2001-02-14

~~NOTBT~~

NOTBETWEEN- To display the data which is not in the range.

1- NOT BETWEEN Number-

Syntax- SELECT * FROM table_name
WHERE column_name
NOT ~~NOT~~ BETWEEN value1 AND value2;

Ex- SELECT * FROM Products
WHERE stu_id
NOT BETWEEN 3 AND 7;

stu_id	Name
5	Anu
6	Sonu
7	Anu
8	Jai

stu_id	Name
8	Jai

Note- कुछ RDBMS में 3 and 7 भी include करता है और कुछ में नहीं।

→ mysql> SELECT * FROM student
-> WHERE stu_id NOT BETWEEN 4 AND 8;

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
9	Anu	Delhi	652345

2- NOT BETWEEN Txt -

Syntax- SELECT * FROM table_name
WHERE column_name
NOT BETWEEN 'value1' AND 'value2';

Ex- new_tab

stu_id	Name
5	Anu
6	Sonu
7	Anu
8	Jai
9	Bob
10	Candy

SELECT * FROM new_tab
WHERE name
NOT BETWEEN 'B' AND 'J'

stu_id	Name
5	Anu
6	Sonu
7	Anu
8	Jai

Note- include B but not include J)

→ mysql > SELECT * FROM student
→ WHERE name NOT BETWEEN 'A' AND 'J';

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764839
5	Shubham	HYD	542367
7	Kiki & Company	Lucknow	234345
8	J.K & company	Aligarh	785246

"No religion has mandated kindness as a cornerstone of its teachings" - A.P.J. Abdul Kalam

Date ___/___/_____

3- NOT BETWEEN Date-

Syntax- SELECT * FROM table_name
WHERE column_name
NOT BETWEEN 'yyyy-mm-dd' AND 'yyyy-mm-dd';

Ex- SELECT * FROM newtab
WHERE date
NOT BETWEEN '2000-01-01' AND '2001-12-31';

Note. including both 2000-01-01 And 2001-12-31

→ mysql> SELECT * FROM stu_tab
→ WHERE dob
→ NOT BETWEEN '2000-12-25' AND '2001-12-26';

stu_id	name	dob
1	Raj	2000-01-04
5	Sona	2002-02-04

BETWEEN with NOT IN-

Syntax- SELECT * FROM table_name
WHERE (column_name BETWEEN value1
AND value2)
AND NOT column_name IN (value1, value2);

Ex- SELECT * FROM emp
WHERE (salary BETWEEN 25000 AND 50000)
AND NOT dept IN ('IT', 'HR');

→ mysql> SELECT * FROM student
→ WHERE (stu_id BETWEEN 2 AND 8)
→ AND NOT city IN ('Delhi', 'HYD');

stu_id	name	City	pin
3	Ram	Kolkata	879845
6	Amit	Kanpur	564897
7	K.K's Company	Lucknow	234345
8	S.K's Company	Azamgarh	785246

"All the work you do, is done for your own salvation, is done for your own benefit." – Swami Vivekananda

LIKE - The LIKE operator is used to search for a specified pattern in a column.

Syntax- SELECT * FROM table_name
WHERE column_name LIKE 'pattern';

Ex- SELECT * FROM new_tab
WHERE name LIKE '%nu';

Wild cards - Wildcards are used to search for data within a table. These characters are used with the LIKE operator.

Wildcards	Description
%	Zero or more character
_	One single character
[charlist]	Sets and range of characters to match
[^charlist] or [!charlist]	Matches only a character NOT specified within the brackets

Note- [charlist] and [^charlist] तब MySQL RDBMS में work नहीं करता

1- % - Zero or more character

- 'Geek%' - All string with geek. Ex- Geekyshows.
- '%shows' - All ending with shows. Ex- Geekyshows.
- '%sh%' - All containing with sh. Ex- Geekyshows.

→ mysql > SELECT * FROM student
-> WHERE name LIKE 'so%';

stu_id	name	city	pin
2	Sona	Delhi	652345
4	Sonu	NULL	764839

mysql> SELECT * FROM student
-> WHERE name LIKE '%ny';

stu_id	name	city	pin
7	K.K's Company	Lucknow	234345
8	J.K's Company	Azangarh	785246

mysql> SELECT * FROM student
-> WHERE city LIKE '%an%';

stu_id	name	city	pin
1	Raj	Ranchi	123456
6	Amit	Kanpur	564897

2- _ - One single character

- 'show_' - Starting with show then any character. Ex- shows
- '_ek' - any character then ek. Ex- geek.
- 'G_e_k - G then any character, then e then any character, then k. Ex- Geek.

→ mysql> SELECT * FROM student
-> WHERE name LIKE '_nu';

stu_id	name	city	pin
9	Anu	Delhi	652345

mysql> SELECT * FROM student
-> WHERE name LIKE 'Sh_bh_m';

stu_id	name	city	pin
5	Shubham	HYD	542367

```
mysql> SELECT * FROM student
    -> WHERE name LIKE 'Ra_';
```

stu_id	name	city	pin
1	Raj	Ranchi	123456
3	Ram	Kolkata	879845

NOT LIKE - यह LIKE का opposite है इसके साथ भी हम सभी wildcards का use कर सकते हैं।

Syntax- SELECT * FROM table_name
WHERE column_name NOT LIKE 'pattern';

Ex- SELECT * FROM new_tab
WHERE name NOT LIKE '%nu';

→
```
mysql> SELECT * FROM student
    -> WHERE name NOT LIKE '%nu';
```

stu_id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	~~HYD~~ Kolkata	879845
~~5~~	Shubham	~~Kanpur~~ HYD	542367
6	Amit	Kanpur	564897
7	K.K's Company	Lucknow	234345
8	J.K's Company	Azamgarh	785246

ORDER BY- This used to short the records.

- ASC- It sorts in ascending order (by default).
- DESC- It sorts in decending order.

Date ___/___/_____

1- Sort in descending order-

Syntax- SELECT * FROM table_name
ORDER BY column_name DESC;

Ex- SELECT * FROM emp
ORDER BY emp_name DESC;

→ mysql> SELECT * FROM student
-> ORDER BY stu_id DESC;

stu_id	name	city	pin
9	Anu	Delhi	652345
8	J.K.'s Company	Azamgarh	785246
7	K.K.'s Company	Lucknow	234345
6	Amit	Kanpur	564897
5	Shubham	HYD	542367
4	Sonu	NULL	764899
3	Ram	Kolkata	879845
2	Sona	Delhi	652345
1	Raj	Ranchi	123456

2- Sort in ascending order-

Syntax- SELECT * FROM table_name
ORDER BY column_name ASC;

Note- यहाँ ASC लिखना optional है।

Ex- (i) SELECT * FROM emp
ORDER By emp_name;

(ii) SELECT * FROM emp
ORDER By emp_name ASC;

→ mysql> SELECT * FROM student
-> ORDER By stu_id;

stu-id	name	city	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764839
5	Shubham	HYD	542367
6	Amit	Kanpur	564897
7	K.K's Company	Lucknow	234345
8	J.K's Company	Azamgarh	785246
9	Anu	Delhi	652345

NOT NULL - By default, a table's column can hold NULL values. The NOT NULL constraint enforces a field to always contain a value. This means that you ~~can~~ cannot insert a new record, or update a record without adding a value to this field. Syntax- column_name NOT NULL

Ex- CREATE TABLE student
(
Name VARCHAR(30),
Roll INTEGER(5),
Mobile_no INTEGER(10) NOT NULL
);

Name	Roll	Mobile_no

→ mysql> CREATE TABLE n_tab
-> (
-> id INT(5) NOT NULL,
-> name VARCHAR(30),
-> city VARCHAR(50),
->);

mysql> DESC n_tab;

Field	Type	Null	Key	Default	Extra
id	int (5)	NO		NULL	
name	varchar(30)	YES		NULL	
city	varchar(50)	YES		NULL	

```
mysql> INSERT INTO n_tab (id, name, city)
   -> VALUES (1, 'Sumit', 'Patana');
mysql> SELECT * FROM n_tab;
```

id	name	city
1	Sumit	Patna

```
mysql> INSERT INTO n_tab (name, city)
   -> VALUES ('Ramesh', 'Ranchi');
ERROR 1364 (HY000): Field 'id' doesn't have a
default value
```

UNIQUE KEY - The UNIQUE constraint uniquely identifies each record in a database table. There can be many UNIQUE constraints per table. A UNIQUE key column can contain NULL values.

Syntax - columnname datatype(size) UNIQUE KEY

Ex - CREATE TABLE student
(
Name varchar(30),
Roll integer (5),
Mobile_no integer(10) UNIQUE KEY
);

Name	Roll	Mobile_no

→ mysql> CREATE u_tab
-> (
-> stu_id INT(5) UNIQUE KEY,
-> name VARCHAR(30),
-> roll INT(5) UNIQUE KEY,
-> city VARCHAR(40)
->);

mysql> DESC u_tab;

Field	Type	Null	Key	Default	Extra
stu_id	int(5)	YES	UNI	NULL	
name	varchar(30)	YES			
roll	int(5)	YES	UNI	NULL	
city	varchar(40)	YES			

~~mysql> SELECT * FROM u_tab;~~

~~stu_id~~	~~name~~	~~roll~~	~~city~~
~~1~~	~~Ram~~	~~101~~	~~Delhi~~

mysql> INSERT INTO u_tab (stu_id, name, roll, city)
-> VALUES (1, 'Ram', 101, 'Delhi');

mysql> SELECT * FROM u_tab;

stu_id	name	roll	city
1	Ram	101	Delhi

mysql> INSERT INTO u_tab (name, roll, city)
-> VALUES ('Sam', 102, 'Delhi');

mysql> SELECT * FROM u_tab;

stuid	name	roll	city
1	Ram	101	Delhi
NULL	Sam	102	Delhi

Date ___/___/_____

```
mysql>INSERT INTO u_tab(stu_id, name, roll, city)
    -> VALUES (1, 'Raj', 103, 'Ranchi');
ERROR 1062 (23000): Duplicate entry '1' for key
'stu_id'
mysql>INSERT INTO u_tab(name, roll, city)
    -> VALUES ('Rani', 103, 'Jaipur');
mysql>SELECT *FROM u_tab;
```

stu_id	name	roll	city
1	Ram	101	Delhi
NULL	Sam	102	Delhi
NULL	Rani	103	Jaipur

PRIMARY KEY – The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain UNIQUE values. A primary key column cannot contain NULL values. Most tables should have a primary key, and each table can have only ONE primary key.

Syntax – column_name datatype(size) NOT NULL PRIMARY KEY

or column_name datatype(size) NOT NULL
PRIMARY KEY(column_name)

Ex-
```
CREATE TABLE student
(
    Name VARCHAR(30)
    Roll INTEGER(5) NOT NULL PRIMARY KEY,
    Mobile_no INTEGER(10)
);
```
OR

```
CREATE TABLE student
(
    Name        VARCHAR(30)
    Roll        INTEGER(5) NOT NULL,
    Mobile_no   INTEGER(10),
    PRIMARY KEY(Roll)
);
```

Name	Roll	Mobile_no

```
→ mysql>CREATE TABLE p_tab
    ->(
    -> id INT(5) NOT NULL PRIMARY KEY,
    -> name VARCHAR(40),
    -> roll INT(5) UNIQUE KEY,
    -> city VARCHAR(50)
    -> );
mysql>DESC p_tab;
```

Field	Type	NULL	Key	Default	Extra
id	int(5)	NO	PRI	NULL	
name	varchar(40)	YES		NULL	
roll	int(5)	YES	UNI	NULL	
city	varchar(50)	YES		NULL	

```
mysql> INSERT INTO p_tab (id, name, roll, city)
    -> VALUES (1, 'Rahul', 101, 'Delhi');
mysql> SELECT * FROM p_tab;
```

id	name	roll	city
1	Rahul	101	Delhi

```
mysql> INSERT INTO p_tab (id, name, roll, city)
    -> VALUES (2, 'Rani', 101, 'Kolkata');
ERROR 1062 (23000): Duplicate entry '101' for key
'roll'.
mysql> INSER INTO p_tab (id, name, roll, city)
    -> VALUES (1, 'Rani', 102, 'Kolkata');
ERROR 1062 (23000): Duplicate entry '1' for key
'PRIMARY'
mysql> INSERT INTO p_tab (name, roll, city)
    -> VALUES ('Rani', 102, 'Kolkata');
ERROR 1364 (HY000): Field 'id' doesn't have a
default value.
mysql> SELECT * FROM p_tab;
```

id	name	roll	city
1	Rahul	101	Delhi

AUTO INCREMENT - Auto increment is used to generate an unique number, when a new record is inserted into a table.

Note- ये NULL value Pass करने पर भी auto increment का देगा और एक table में एक ही auto increment हो सकता है। ये by 1 से auto increment करता है। इसका use PRIMARY KEY के साथ करते हैं।

Syntax- column_name datatype AUTO_INCREMENT
or column_name datatype(size) AUTO_INCREMENT

Ex- CREATE TABLE ~~table_name~~ emp
(
~~Column_name~~ Emp_id int NOT NULL AUTO_INCREMENT
Emp_name VARCHAR (50) NOT NULL,
City VARCHAR (50),
PRIMARY KEY (Emp_id)
);

- INSERT INTO emp (Emp_name, Emp_name, city)
  VALUES (NULL, 'Shubham', 'Delhi'), (NULL, 'Ankit', 'Mumbai');
  Note- NULL को ignore करके value 1 से start करेगा।

- INSERT INTO emp (emp_name, city)
  VALUES ('Shubham', 'Delhi'), ('Ankit', 'Mumbai');
  Note- हमने जो column auto increment है उसे skip कर सकते हैं

- INSERT INTO emp (Empid, Emp_name, city)
  VALUES (NULL, 'Shubham', 'Delhi'), (1, 'Ankit', 'Mumbai');
  Note- कुछ RDBMS (1, 'Ankit', 'Mumbai') को add करना skip कर देते हैं और कुछ Error देते हैं, क्योंकि NULL के जगह 1 पहले ही add कर दिया गया है।

→
```
mysql>CREATE TABLE ap_tab
    -> (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> name VARCHAR(50),
    -> city VARCHAR(60),
    -> PRIMARY KEY(id)
    -> );
mysql> DESC ap_tab;
```

Filed	Type	NULL	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(50)	YES		NULL	
city	varchar(60)	YES		NULL	

```
mysql> INSERT INTO ap_tab (name, city)
    -> VALUES ('Raj', 'Delhi');
mysql> SELECT * FROM ap_tab;
```

id	name	city
1	Raj	Delhi

```
mysql> INSERT INTO ap_tab (name, city)
    -> VALUES ('Rani', 'Kolkata');
mysql> SELECT * FROM ap_tab;
```

id	name	city
1	Raj	Delhi
2	Rani	Kolkata

```
mysql> INSERT INTO ap_tab (id, name, city)
    -> VALUES (NULL, 'Rahul', 'hdy');
mysql> SELECT * FROM ap_tab;
```

id	name	city
1	Raj	Delhi
2	Rani	Kolkata
3	Rahul	hdy

```
mysql> INSERT INTO ap_tab (id, name, city)
    -> VALUES (NULL, 'Raja', 'Ranchi'),
    -> (4, 'Sam', 'Patna');
ERROR 1062 (23000): Duplicate entry '4' for key
'PRIMARY'
mysql> SELECT * FROM ap_tab;
```

id	name	city
1	Raj	Delhi
2	Rani	Kolkata
3	Rahul	hdy

```
mysql> INSERT INTO ap_tab (id, name, city)
    -> VALUES (34, 'Raju', 'Patana');
```

```
mysql> SELECT * FROM ap_tab;
```

id	name	city
1	Raj	Delhi
2	Rani	Kolkata
3	Rahul	hdy
34	Raju	Patana

```
mysql> INSERT INTO ap_tab (name, city)
    -> VALUES ('Ali', 'Ranchi');
mysql> SELECT * FROM ap_tab;
```

id	name	city
1	Raj	Delhi
2	Rani	Kolkata
3	Rhahul	hdy
34	Raju	Patma
35	Ali	Ranchi

AUTO_INCREMENT start with a particular number-

Syntax- ALTER TABLE table_name AUTO_INCREMENT =number_value;

```
→ mysql> CREATE TABLE k_tab
    -> (
    -> id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR (50);
    -> );
mysql> DESC k_tab;
```

Field	Type	Null	Key	Default	Extra
id	int(10)	NO	PRI	NULL	auto_increment
name	varchar(50)	YES		NULL	

```
mysql> ALTER TABLE k_tab AUTO_INCREMENT=10;
~~mysql> VALUES INTO k_tab (name)~~
  ~~-> ;~~
mysql> INSERT INTO k_tab (name)
   -> VALUES ('Ram'),('Ali');
mysql> SELECT * FROM k_tab;
```

id	name
10	Ram
11	Ali

AUTO_INCREMENT without specifying column name insert a new record-

```
-> mysql> INSERT INTO k_tab
   -> VALUES ('Raj');
ERROR 1136 (21S04): Column count doesn't match value count at row 1
mysql> INSER INTO k_tab
   -> VALUES ('','Raj');
ERROR 1366 (HY000): Incorrect integer value: '' for column 'id' at row 1
mysql> INSERT INTO k_tab
   -> VALUES (NULL,'Raj');
mysql> SELECT * FROM k_tab;
```

id	name
10	Ram
11	Ali
12	Raj

Aliase - Aliase are used to temporarily rename a table name or a column name.

FOR Table-

Syntax- SELECT column_name FROM table_name AS alias_name;

Ex- SELECT name FROM student AS vidyarthi;

FOR Column-

Syntax- SELECT column_name AS alias_name FROM table_name;

Ex-i) SELECT ~~column~~ name AS student_name FROM student;

or SELECT name student_name FROM student

ii) SELECT name AS "Student Name" FROM student;

or SELECT name AS [Student Name] FROM student;

Note- यहाँ AS optional है। " ____ " or [____] का use name के बीच में Space देने के लिये use करते हैं पर [Student Name] MySQL Support नहीं करता है। जब तक ; देकर comman खत्म ना कर दें तो name को बदले रहोगा।

mysql>

→ SELECT name AS student FROM k_tab;

student
Ram
Ali
Raj

mysql> SELECT name student FROM k_tab;

student
Ram
Ali
Raj

mysql> SELECT name "Student Name" FROM table;

Student Name
Ram
Ali
Raj

Arithmetic Operators - (+, /, *, -)

Syntax- SELEC column_name, column_name Operator value FROM table_name;

Ex-(i) SELECT name, cost, cost + 100 FROM item_tab;

(ii) SELECT name, cost, cost + 100 AS "New Cost" FROM item_tab;

→ mysql> SELECT * FROM students;

stu_id	name	mark	result
1	Rahul	300	1st
2	Rani	250	2nd
3	Soni	320	1st
4	Jai	255	2nd
5	Rock	150	3rd
6	Sonam	145	fail

mysql> SELECT stu_id, name, mark, mark + 100 FROM students;

stu_id	name	mark	mark+100
1	Rahul	300	400
2	Rani	250	350
3	Soni	320	420
4	Jai	255	355
5	Rock	150	250
6	Sonam	145	245

"All the work you do, is done for your own salvation, is done for your own benefit." —Swami Vivekananda

mysql> SELECT stu_id, name, mark, mark+100 AS "new marks" FROM students;

stu_id	name	mark	new. marks
1	Rahul	300	400
2	Rani	250	350
3	Soni	320	420
4	Jai	255	355
5	Rock	150	250
6	Sonam	145	245

mysql> SELECT stu_id, name, mark, mark+stu_id AS "new marks" FROM students;

stu_id	name	mark	new marks
1	Rahul	300	301
2	Rani	250	252
3	Soni	320	323
4	Jai	255	259
5	Rock	150	155
6	Sonam	145	151

SELECT DISTINCT- The SELECT DISTINCT statement is used to display only distinct (different) value.

Syntax- SELECT DISTINCT column_name FROM table_name;

Ex- SELECT DISTINCT name FROM student;

→ mysql> SELECT * FROM student;

stu_id	name	City	pin
1	Raj	Ranchi	123456
2	Sona	Delhi	652345
3	Ram	Kolkata	879845
4	Sonu	NULL	764849
5	Shubham	HYD	542367
6	Amit	Kanput	564897
7	K.K'e .Company	Lucknow	234345
8	J. K'e Company	Azamgarh	785246
9	Anu	Delhi	652345

mysql> SELECT DISTINCT city FROM student;

City
Ranchi
Delhi
Kolkata
NULL
HYD
Kanput
Lucknow
Azamgarh

Data Definition.

ALTER TABLE - This command is used to Add/Change/Modify/Drop existing structure of the table.

- ADD Column
- Enable/Disable Constraints
- Change Column
- Modify Column
- Drop Column

ADD Column - When a new column is to be added to the table structure ~~when~~ without constraints.

Syntax- ALTER TABLE table_name
ADD COLUMN column_name datatype (size);

Note- Some RDBMS doesn't support ADD COLUMN

Ex- ALTER TABLE my tab
ADD COLUMN stu_id integer (5);

Name	Roll

Name	Roll	Stu_id

→ mysql> SELECT * FROM k tab;

id	name
10	Ram
11	Ali
12	Raj

mysql> ALTER TABLE k tab
→ ADD COLUMN m_no INT(10);
mysql> SELECT * FROM k tab;

Date ___/___/_____

id	name	m_no
10	Ram	NULL
11	Ali	NULL
12	Raj	NULL

- ADD more than one column-

Syntax- ALTER TABLE table_name
ADD COLUMN column_name datatype (size),
ADD COLUMN column_name datatype (size);

Ex- ALTER TABLE my_tab
ADD COLUMN city VARCHAR (50),
ADD COLUMN stu_id INTEGER(5);

→ mysql> SELECT * FROM n_tab;

id	name	city
1	Sumit	Patna

mysql> ALTER TABLE n_tab
->ADD COLUMN email VARCHAR(50),
->ADD COLUMN pin INT(8);

mysql> SELECT * FROM n_tab;

id	name	city	email	pin
1	Sumit	Patna	NULL	NULL

- ADD COLUMN by Position

1- Last (by default)
2- First
3- After

Sytax- ALTER TABLE table_name
ADD COLUMN column_name datatype (size) FIRST;

Ex- ALTER TABLE my_tab
ADD COLUMN city VARCHAR(50) FIRST;

→ mysql> SELECT * FROM k_tab;

id	name	m_no.
10	Ram	NULL
11	Ali	NULL
12	Raj	NULL

mysql> ALTER TABLE k_tab
-> ADD COLUMN roll INT(5) FIRST;
mysql> SELECT * FROM k_tab;

roll	id	name	m_no
NULL	10	Ram	NULL
NULL	11	Ali	NULL
NULL	12	Raj	NULL

mysql> ALTER TABLE k_tab
-> ADD COLUMN l_name VARCHAR(50) AFTER name;
mysql> SELECT * FROM k_tab;

roll	id	name	l_name	m_no
NULL	10	Ram	NULL	NULL
NULL	11	Ali	NULL	NULL
NULL	12	Raj	NULL	NULL

ADD COLUMN - when a new column is to be added to the table structure with constraints -
Syntax- ALTER TABLE table_name
ADD COLUMN column_name data_type (size)
Constraint_name. Constraint_name;

Date __/__/____

or ALTER TABLE table_name
ADD COLUMN column_name data_type(size) Constraint_name

ADD constraint name (column_name);

Ex- ALTER TABLE my_tab
ADD COLUMN roll INT(10) NOT NULL,
ADD PRIMARY KEY (roll);

→ mysql> SELECT * FROM items;

name
Mobile
Laptop
TV
PC
Cam

mysql> ALTER TABLE items
->ADD COLUMN item_no INT(5) NOT NULL
->ADD PRIMARY KEY (item_no);
ERROR 1062 (23000): Duplicate entry '0' for key 'PRIMARY'
mysql> ALTER TABLE items
->ADD COLUMN item_no INT(5) NOT NULL PRIMARY KEY;
ERROR 1062 (23000): Duplicate entry '0' for key 'PRIMARY'
mysql> ALTER TABLE items
->ADD COLUMN item_no INT(5) NOT NULL;
mysql> SELECT * FROM items;

name	item_no
Mobile	0
Laptop	0
TV	0
PC	0
Cam	0

```
mysql> ALTER TABLE items
    -> ADD COLUMN item_n INT(5) NOT NULL
                                AUTO_INCREMENT;
mysql> SELECT * FROM items;
```

name	item_no	item_n
Mobile	0	1
Laptop	0	2
TV	0	3
PC	0	4
Cam	0	5

Note- यहाँ UNIQUE & PRIMARY constraints के द्वारा AUTO_INCREMENT का use करना होगा।

ALTER TABLE for Adding Column by position with constraints -

```
mysql> SELECT * FROM item;
```

name
Mobile
Laptop
TV
PC
Cam

```
mysql> ALTER TABLE item
    -> ADD COLUMN item_no INT(5) NOT NULL
                    AUTO_INCREMENT FIRST,
    -> ADD PRIMARY KEY (item_no);
mysql> SELECT * FROM item;
```

item_no	name
1	Mobile
2	Laptop
3	TV
4	PC
5	Cam

```
mysql> ALTER TABLE item
    -> ADD COLUMN seller_id INT UNIQUE KEY
                    AFTER item_no;
mysql> SELECT * FROM item;
```

item_no	seller_id	name
1	NULL	Mobile
2	NULL	Laptop
3	NULL	TV
4	NULL	PC
5	NULL	Cam

```
mysql> DESC item;
```

Field	Type	NULL	Key	Default	Extra
item_no	int(5)	NO	PRI	NULL	auto_incrument
seller_id	int(11)	YES	UNI	NULL	
name	varchar(20)	YES		NULL	

~~When After~~

ALTER TABLE for adding Constraint in Column- When integrity constraints have to be included.

Syntax- ALTER TABLE table_name
ADD CONSTRAINT constraint_name column_name;

Ex- ALTER TABLE my_tab
ADD CONSTRAINT PRIMARY KEY (roll);

→ mysql> DESC itab;

Field	Type	Null	Key	Default	Extra
seller_id	int(4)	YES		NULL	
name	varchar(50)	YES			

mysql> ALTER TABLE i_tab
-> ADD CONSTRAINT UNIQUE KEY (seller_id);

mysql> DESC i_tab;

Field	Type	Null	Key	Default	Extra
seller_id	int(4)	YES	UNI	NULL	
name	varchar(50)	YES		NULL	

Change Column- This is used to change name and data type of an existing column.

1- Without Constraints-

Syntax- ALTER TABLE table_name
CHANGE COLUMN old_column_name new_column_name new_data_type(size);

Ex- ALTER TABLE my_tab
CHANGE COLUMN name Student VARCHAR(5);

Note:- name change करना अनिवार्य है।

→ mysql> SELECT * FROM i_tab;

seller_id	name
NULL	Mobile
NULL	Laptop
NULL	TV
NULL	PC
NULL	Cam

mysql> ALTER TABLE i_tab
-> CHANGE COLUMN name p_name VARCHAR(50);
~~mysql> ALTER TABLE~~
~~-> CHAN~~
mysql> SELECT * FROM i_tab;

seller_id	p_name
NULL	Mobile
NULL	Laptop
NULL	TV
NULL	PC
NULL	Cam

- Change more than one column-
  Syntax- ALTER TABLE table name
  CHANGE COLUMN old_columnname1 new_column_name1 datatype (size),
  CHANGE COLUMN old_columnname2 new_column_name2 data_type (size), ——— ;

Ex- ALTER Table my_tab
CHANGE COLUMN name Student VARCHAR(5);
CHANGE COLUMN roll_id INT (5);

→ mysql> ALTER TABLE i_tab
→ CHANGE COLUMN seller_id sele_id INT(6),
→ CHANGE COLUMN p_name product_name VARCHAR(50);
mysql> SELECT * FROM i_tab;

sele_id	product_name
NULL	Mobile
NULL	Laptop
NULL	TV
NULL	PC
NULL	Cam

2- Change Column with Constraints:-
Syntax- ALTER TABLE table_name
CHANGE COLUMN old_column_name new_column_name
new_data_type (size) constrain_name
ADD constriaint_name (column_name);
Ex- ALTER TABLE my_tab
CHANGE COLUMN roll_id INT(5) NOT NULL,
ADD PRIMARY KEY (id);

→ mysql> DESC product_tab;

Field	Type	Null	Key	Default	Extra
sele_id	int(4)	YES	UNI	NULL	
name	varchar(30)	YES		NULL	

mysql> ALTER TABLE product_tab
→ CHANGE COLUMN sele_id id INT(10) NOT NULL,
→ ADD PRIMARY KEY (id);

mysql> DESC product_tab;

Date ___/___/___

Field	Type	Null	Key	Default	Extra
id	int (8)	YES	PRI	NULL	
name	varchar(30)	YES		NULL	

Note- हम चाहे तो ADD constraint न लिखकर भी line में constraint name लिख सकते हैं और UNI को PRI में बदलने से पहले यदि वहाँ value NULL है तो उसे unique बनाना होगा।

MODIFY COLUMN- This is used to modify size of the data type or the data type itself of an existing without changing column name.

Syntax- ALTER TABLE table-name
MODIFY COLUMN column-name data-type(size)

Note- हम चाहे तो साथ में constraint का भी use कर सकते हैं।

Ex- my_tab

Name	Roll
Ram	12345

ALTER TABLE my_tab
MODIFY COLUMN roll INTGER(10);

my_tab

Name	Roll
Ram	23456 7890

→ mysql>DESC product_tab;

Field	Type	Null	Key	Default	Extra
id	int(10)	NO	PRI	NULL	
name	varchar(30)	YES		NULL	

mysql> ALTER TABLE product tab
-> MODIFY COLUMN name CHAR(40);

mysql> DESC product_tab;

Field	Type	Null	Key	Default	Extra
id	int(10)	NO	PRI	NULL	
name	char(40)	YES		NULL	

mysql> ALTER TABLE product_tab
->MODIFY COLUMN name CHAR (4) NOT NULL;
mysql>DESC product_tab;

Field	Type	Null	Key	Default	Extra
id	int(10)	NO	PRI	NULL	
name	char(40)	NO		NULL	

DROP COLUMN- (i) when a column in a table need to delete.

Syntax- ALTER TABLE table_name
DROP COLUMN column_name;

Ex- ALTER TABLE my_tab
DROP COLUMN roll;

Note- सभी RDBMS इसे support नहीं करते हैं।

→ mysql> DESC product_tab;

Field	Type	Null	Key	Default	Extra
id	int(10)	NO	PRI	NULL	
name	char(40	NO		NULL	

mysql>ALTER TABLE product_tab
->DROP COLUMN name;
mysql>DESC product_tab

Field	Type	Null	Key	Default	Extra
id	int(10)	NO	PRI	NULL	

No religion has mandated killing others as a requirement for its advance or promotion. — Dr. A. P. J. Abdul Kalam

Date ___/___/___

(ii) When removing constraint from a column-

Syntax- ALTER TABLE table_name
DROP constraint_name column_name;

Ex- ALTER TABLE my_tab
DROP UNIQUE KEY(roll);

→ mysql>DESC p_tab;

Field	Type	Null	Key	Default	Extra
id	int(5)	NO	PRI	NULL	
name	varchar(40)	YES		NULL	
roll	int(5)	YES	UNI	NULL	
city	varchar(50)	YES		NULL	

mysql>ALTER TABLE p_tab
→ DROP UNIQUE KEY(roll);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'UNIQUE KEY(roll)' at line 2
mysql> SHOW INEX p_tab;

mysql>ALTER TABLE p_tab
->DROP INDEX roll;
mysql>DESC p_tab;

Field	Type	Null	Key	Default	Extra
id	int(5)	NO	PRI	NULL	
name	varchar(40)	YES		NULL	
roll	int(5)	YES		NULL	
city	varchar(50)	YES		NULL	

DROP TABLE - This command is used to delete/remove table from the database.

Syntax- DROP TABLE table_name;

Ex- DROP TABLE my_tab;

→ mysql> SHOW TABLES;

Tables_in_my_db
ap_tab
s_tab
i_tab
my_tab

mysql> DROP TABLE s_tab;

mysql> SHOW TABLES;

Tables_in_my_db
ap_tab
i_tab
my_tab

TRUNCATE TABLE - when we only want to delete the data inside the table, and not the table itself.

Syntax- TRUNCATE TABLE table_name;

Ex- TRUNCATE TABLE my_tab;

→ mysql> SHOW TABLES;

Tables_in_my_db
ap_tab
i_tab
my_tab

Date ___/___/_____

mysql> SELECT * FROM i_tab;

sale_id	product_name
NULL	Mobile
NULL	Laptop
NULL	TV
NULL	PC
NULL	Cam

mysql> TRUNCATE i_tab;
mysql> SELECT * FROM i_tab;
Empty set (0.00sec)
mysql> SHOW TABLES;

Tables_in_my_db
ap_tab
i_tab
my_tab

RENAME TABLE - This command is used to rename one or more tables

Syntax- RENAME TABLE old_table_name TO new_table_name;

Ex- RENAME TABLE my_tab TO your_tab;

→ mysql> SHOW TABES;

Tables_in_my_db
ap_tab
i_tab
my_tab

mysql> RENAME TABLE my_tab TO your_tab;
mysql> SHOW TABLES;

Table_in_my_db
ap_tab
i_tab
your_tab

ALTER DATABASE - ALTER DATABASE enables you to change the overall characteristics of a database. These characteristics are store in the db.opt file in the database directory. To use ALTER DATABASE, you need the ALTER privilege on the database.
ALTER SCHEMA is a synonym for ALTER DATABASE.
The database name can be omitted from the syntax, in which case the statement applies to the default database.
Syntax- ALTER DATABASE database_name;
Ex- ALTER DATABASE db;
Note- database_name न देने पर USE database पर ही work करेगा।

DROP DATABASE - The DROP DATABASE statement is used to delete a database.
Syntax- DROP DATABASE database_name;
Ex- DROP DATABASE db;

→ ~~mysql> SHOW DATABASES;~~

Date 24/6/19

→ mysql> SHOW DATABASES;

Database
information_schema
my_bd
mysql
performance_schema
sys
your_db

mysql> ~~SHOW~~ DROP DATABASE your_db;
mysql> SHOW DATBASES;

Database
information_schema
my_db
mysql
performance_schema
sys.

SHOW COLUMNS - It shows all the columns of table and their data type along with any other column specific details. It just like DESC. table_name;
Syntax- SHOW COLUMNS FROM table_name;
Ex- SHOW COLUMNS FROM emp;

→ mysql> SHOW COLUMNS FROM p_tab;

Field	Type	Null	Key	Default	Extra
id	int(5)	NO	PRI	NULL	
name	varchar(40)	YES		NULL	
roll	int(5)	YES	UNI	NULL	
city	varchar(50)	YES		NULL	

"All the work you do, is done for your salvation, is done for your own benefit." —Swami Vivekananda

SHOW CREATE DATABASE - It shows commands which you have written while creating your Database.

Syntax- SHOW CREATE DATABASE database_name;

Ex- SHOW CREATE DATABASE my_db;

→ mysql> SHOW CREATE DATABASE my_db;

Database	Create Database
my_db	CREATE DATABASE 'my_db' /* !40100 DEFAULT CHARACTER SET latin1 */

SHOW CREATE TABLE - It shows ~~which~~ commands which you have written while creating your table.

Syntax- SHOW CREATE TABLE table_name;

Ex- SHOW CREATE TABLE emp;

→ mysql> SHOW CREATE TABLE student;

Table	Create Table
student	CREATE TABLE 'student' (
'stid'	int(11) NOT NULL AUTO-INCREMENT,
'name'	varchar(50) DEFAULT NULL,
'city'	varchar(50) DEFAULT NULL,
'mark'	int(11) DEFAULT NULL,
'result	varchar(50) DEFAULT NULL,
PRIMARY KEY ('stid')	
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1	

Note- Right click करके mark select करके इसे copy कर सकते हैं।

"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr.A.P.J.Abdul Kalam

UPDATE- The UPDATE student is used to update existing records in a table.

Syntax- UPDATE table_name
SET column1 = value1, column2 = value2, ——
WHERE some_column = some_value;

Ex- UPDATE emp
SET emp_name = 'Arjun', salary = 35000
WHERE emp_id = 101;

Note- WHERE is necessary otherwise all records will be replace with given value.

→ SELECT * FROM students;

stu_id	name	mark	result
1	rahul	300	1st
2	Rani	250	2nd
3	Soni	320	1st
4	Jai	255	2nd
5	Rock	150	3rd
6	Sonam	145	fail

mysql> UPDATE students
→ SET mark = 100, result = 'fail'
→ WHERE stu_id = 1;
mysql> SELECT * FROM students;

stu_id	name	mark	result
1	rahul	100	fail
2	Rani	250	2nd
3	Soni	320	1st
4	Jai	255	2nd
5	Rock	150	3rd
6	Sonam	145	fail

"All the work you do, is done for your own salvation, is done for your own benefit." –Swami Vivekananda

```
mysql> UPDATE * FROM students;
    -> SET mark=340, result='1st';
mysql> SELECT * FROM students;
```

stu_id	name	mark	result
1	rahul	340	1st
2	Rani	340	1st
3	Soni	340	1st
4	Jai	340	1st
5	Rock	340	1st
6	Sonam	340	1st

```
mysql> UPDATE * FROM students
    -> SET mark=400
    -> WHERE stu_id=1;
mysql> SELECT * FROM students;
```

stu_id	name	mark	result
1	rahul	400	1st
2	Rani	340	1st
3	Soni	340	1st
4	Jai	340	1st
5	Rock	340	1st
6	Sonam	340	1st

UPDATE with CASE -

```
UPDATE table_name
SET new_column =
CASE
    WHEN column_name1 = some_value1 THEN new_value1
    WHEN column_name2 = some_value2 THEN new_value2
    ELSE new_value3.
END;
```

Date ___/___/_____

Ex- UPDATE student
SET Result =
CASE
WHEN mark >= 300 THEN 'First'
WHEN mark < 300 AND mark >= 250 THEN 'Second'
WHEN mark < 250 AND mark >= 150 THEN 'Third'
ELSE 'Fail'
END;

→ mysql> SELECT * FROM students;

stu_id	name	mark	result
1	rahul	100	1st
2	Rani	340	1st
3	Soni	260	1st
4	Jai	340	1st
5	Rock	170	1st
6	Sonam	340	1st

```
mysql> UPDATE students
    -> SET result =
    -> CASE
    -> WHEN mark >= 300 THEN 'First'
    -> WHEN mark < 300 AND mark >= 250 THEN
                                    'Second'
    -> WHEN mark < 250 AND mark >= 150 THEN
                                    'Third'
    -> END;
mysql> SELECT * FROM students;
```

stu_id	name	mark	result
1	rahul	100	Fail
2	Rani	340	First

3	Soni	260	Second
4	Jai	340	First
5	Rock	170	Third
6	Sonam	340	First

DELETE - The DELETE statement is used to delete records in a table.

1- Delete a Specific Record-

Syntax- DELET FROM table_name
WHERE some_column = some_value;

Ex- DELETE FROM ~~table_name~~ emp
WHERE ~~FROM~~ empname = 'Sonam' AND empid = 105

Note- WHERE is necessary otherwise all records will be delete.

-> mysql > SELECT * FROM students;

stu_id	name	mark	result
1	Rahul	100	Fail
2	Rani	340	First
3	Soni	260	Second
4	~~Rock~~ Jai	340	First
5	Rock	170	Third
6	Sonam	340	First

mysql> DELETE FROM students
-> WHERE stu_id = 6;

mysql> SELECT * FROM students;

stu_id	name	mark	result
1	Rahul	100	Fail
2	Rani	340	First
3	Soni	260	Second
4	Jai	340	First
5	Rock	170	Third

"No religion has mandated killing other human beings for the enhancement or promotion." -Dr.A.P.J.Abdul Kalam

2- Delete All Records-

Syntax- DELETE FROM table_name;
or DELETE * FROM table_name;

Ex- DELETE FROM emp;
or DELETE * FROM emp;

Note- We can not undo this.

→ mysql> SELECT * FROM student;

stu_id	name	mark	result
1	rahul	100	Fail
2	Rani	340	First
3	Soni	260	Second
4	Jai	370	First
5	Rock	170	Third

mysql> DELETE FROM students;
mysql> SELECT * FROM student;
Empty set (0.00 sec)
mysql > SHOW TABLES

Tables_in_my_db
ap_tab
i_tab
items
students
u_tab

mysql >DESC students;

Field	Type	Null	Key	Default	Extra
stu_id	int(4)	YES		NULL	
name	varchar(20)	YES		NULL	
mark	int(10)	YES		NULL	
result	varchar(10)	YES		NULL	

"All the work you do, is done for your own salvation, is done for your own benefit." — Swami Vivekananda

Copy Old Table to new table-

1- With-in Same Database -

Syntax- CREATE TABLE new-table LIKE old-table;
INSERT new-table SELECT * FROM old-table;

Ex- CREATE TABLE teacher LIKE student;
INSERT teacher SELECT * FROM student;

→ mysql> SHOW TABLES;

Table_in_mydb
emp
student

mysql> SELECT * FROM emp;

roll	name	~~city~~
101	Ram	~~Delhi~~
102	Sam	~~Kolkata~~
103	Rani	~~Jaipur~~

mysql> DESC emp;

Field	Type	Null	Key	Default	Extra
roll	int(5)	YES	UNI	NULL	
name	varchar(40)	YES		NULL	

mysql> CREATE TABLE new_emp LIKE emp;
mysql> SELECT * FROM new_emp;
Empty set (0.00 sec)
mysql> SHOW TABLES;

Table_in_my_db;
emp
new_emp
student

"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr.A.P.J.Abdul Kalam

mysql> DESC new_emp;

Field	Type	Null	Key	Default	Extra
roll	int(5)	YES	UNI	NULL	
name	varchar(40)	YES		NULL	

mysql> INSERT new_emp SELECTE * FROM emp;
mysql> SELECT * FROM new_emp;

roll	name
101	Ram
102	Sam
103	Ravi

2- Within different Database - Use the dabase where you want to copy the old table.

Syntax- CREATE TABLE new_tab LIKE old_db.old_tbl
INSERT new_table SELECT * FROM old_bd.old_table;

Ex- CREATE TABLE teacher LIKE my_db.student;
INSERT teacher SELECT * FROM my_db.student

-> mysql> CREATE DATABASE newdb;
mysql> SHOW DATABASES;

Database
information_schema
my_db
mysql
newdb
performance_schema
sys

mysql> SHOW TABLES;

Tables_in_my_db
emp
new_emp
student

mysql> SLECTE * FROM emp;

roll	name
101	Ram
102	Sam
103	Rani

mysql> USE new_db;
Database changed
mysql> SHOW TABLES;
Empty set (0.00 sec)
mysql> CREATE TABLE emp LIKE my_db.emp;
mysql> SHOW TABLES;

Tables_in_new_db
emp

mysql> DESC emp

Field	Type	Null	Key	Default	Extra
roll	int(5)	YES	UNI	NULL	
name	varchar(40)	YES		NULL	

mysql> INSRT emp SELECT * FROM my_db.emp;
mysql> SELECT * FROM emp;

roll	name
101	Ram
102	Sam
103	Rani

Date 26 / 6 / 19

Functions-

MIN (column name) - Smallest value of the selected column.

Syntax- SELECT MIN(column name) FROM table name;

MAX (column name) - Largest value of the selected column.

Syntax- SELECT MAX (column name) FROM table name;

→ mysql>SELECT * FROM sch;

stuid	name	marks
1	Ram	100
2	Rani	340
3	Soni	260
4	Jai	390
5	Sam	170

mysql> SELECT MIN(marks) FROM sch;

MIN (marks)
100

mysql>SELECT MAX(marks) FROM sch;

MAX (marks)
390

~~mysql>SELECT MAX(marks) FROM sch AS num~~

mysql>SELECT MAX(marks) AS num FROM sch;

num
390

~~SQRT~~

SUM (column_name) - The total sum value of a numeric column.

Syntax - SELECT SUM(column_name) FROM table_name;

AVG (column_name) - The average value of a numeric column.

Syntax - SELECT AVG(column_name) FROM table_name;

→ mysql> SELECT SUM(marks) FROM sch;

SUM(marks)
1260

mysql> SELECT SUM(marks) AS total_marks FROM sch;

total_marks
1260

mysql> SELECT AVG(marks) AS av_marks FROM sch;

av_marks
252.0000

SQRT (column_name) - The square root of a numeric column.

→ ~~mysql> SELEC~~

mysql -> SELECT SQRT(marks) FROM sch;

SQRT(marks)
10
18.490889145857 74
16.124515496597 1
19.748417658131498
13.038404810405298

"No person has ever made it through life without others as a requirement for its sustenance or promotion." —Dr. A. P. J. Abdul Kalam

mysql> SELECT marks, SQRT(marks) FROM sch;

marks	SQRT(marks)
100	10
340	18.43908891458577 4
260	16.12451549659 71
390	19.74841765813149 8
170	13.03840481040529 8

DECIMAL DATATYPE-

Syntax- column-name DECIMAL (T,D);

where- T- Total digits. Range 1-62

D- digits after decimal. Rang 0-30 and must not be more than T.

Ex- price DECIMAL (7,2);

= 16999.81

we use DECIMAL, DEC, NUMERIC, FIXED.

- Column_name DECIMAL (T)
  = Column_name DECIMAL (T,0)
- column_name DECIMAL; (Default T=10)

Size in memory-

No. of Digits	Number of Bytes
0	0
1-2	1
3-4	2
5-6	3
7-9	4

Ex- 1434978 76.7678

9 4 2 4 + = 6 Byte

"All the work you do, is done for your own salvation, is done for your own benefit." —Swami Vivekananda

```
mysql> CREATE TABLE product
    -> (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> pname VARCHAR(40),
    -> price DECIMAL(7,2) NOT NULL
    -> );
mysql> INSERT INTO product (pname, price)
    -> VALUES ("Mobile", 69999.56),
    -> ("Computer", 16000.56),
    -> ("Laptop", 25000.00),
    -> ("Printer", 10000.23);
mysql> SELECT * FROM product;
```

id	pname	price
1	Mobile	69999.56
2	Computer	16000.56
3	Laptop	25000.00
4	Printer	10000.23

- Zerofill -

Syntax- column-name DECIMAL (T,D) ZEROFILL;

Ex 250.65, T=4 D=4 then 0250.6500

```
-> mysql> CREATE TABLE item_tab;
    -> (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> pname VARCHAR(40),
    -> price DECIMAL(12,3) ZEROFILL
    -> );
mysql> INSERT INTO item_tab (pname, price)
    -> VALUES ("Mobile", 69999.55),
    -> ("Computer", 16000.56),
    -> ("Laptop", 25000.00),
    -> ("Printer", 10000.23);
```

"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr.A.P.J.Abdul Kalam

mysql> SELECT * FROM item tab;

id	pname	price
1	Mobile	00006999.556
2	Computer	0000016000.5600
3	Laptop	0000250000.000
4	Printer	0000010000.230

ROUND function function is used to round a numeric field to the number of decimal specified.

Syntax SELECT ROUND (column name, decimal) FROM table name;

→ mysql> SELECT * FROM product;

id	pname	price
1	Mobile	69999.56
2	Computer	16000.56
3	Laptop	25000.00
4	Printer	10000.23

mysql> SELECT *, ROUND(price, 1) FROM product;

id	pname	price	ROUND(price, 1)
1	Mobile	69999.56	69999.6
2	Computer	16000.56	16000.6
3	Laptop	25000.00	25000.0
4	Printer	10000.23	10000.2

COUNT() - The COUNT (column name) function returns the number of values (NULL values will not be counted) of the specified column

Syntax- SELECT COUNT(column_name) FROM table_name;

COUNT(*) - function returns the number of records in a table.

Syntax- SELECT COUNT(*) FROM table_name;

COUNT (DISTINCT column_name) - function returns the number of distinct values of the specified column.

Syntax- SELECT COUNT (DISTINCT column_name) FROM table_name;

→ mysql> SELECT * FROM order;

OrderID	CustID	SellerID
1	101	201
2	102	202
3	103	204
4	101	205
5	101	301
6	102	401
7	103	301
8	103	345

mysql> SELECT COUNT (CustID) FROM order;

COUNT(CustID)
8

mysql> SELECT COUNT(*) FROM order;

COUNT(*)
8

Date ___/___/_____

mysql> SELECT COUNT(DISTINCT CustID) FROM order;

COUNT (DISTINCT CustID)
3

mysql> SELECT COUNT(CustID) FROM order
-> WHERE CustID = 101;

COUNT (CustID)
3

mysql> SELECT COUNT(CustID) FROM order
-> WHERE CustID = 102;

COUNT (CustID)
2

UPPER(column_name) or UCASE(column_name) - converts the value of a field to uppercase

Syntax- ~~SELECT~~ SELECT UPPER(column_name) FROM tbl_name;
OR SELECT UCASE(column_name) FROM tbl_name;

LOWER(column_name) or LCASE(column_name) - Converts the value of a field to lowercase.

Syntax- SELECT LOWER(column_name) FROM tbl_name
OR SELECT LCASE(column_name) FROM tbl_name;

→ mysql> SELECT * FROM sch;

stu_id	name	marks
1	Ram	100
2	Rani	340
3	Soni	260
4	Jai	390
5	Sam	170

"All the work you do, is done for your own benefit, is done for your own benefit." –Swami Vivekananda

```
mysql> SELECT UPPER(name) FROM sch;
```

UPPER(name)
RAM
RANI
SONI
JAI
SAM

```
mysql> SELECT UPPER(name) AS Name, marks FROM sch;
```

Name	marks
RAM	100
RANI	340
SONI	260
JAI	390
SAM	170

```
mysql> SELECT UCASE(name) FROM sch;
```

UCASE(name)
RAM
RANI
SONI
JAI
SAM

```
mysql> SELECT LOWER(name) FROM sch;
```

LOWER(name)
ram
rani
soni
jai
sam

Date ___/___/___

mysql> SELECT LCASE(name) FROM sch;

LCASE(name)
ram
rani
roni
jai
ram

MID (column_name, start, length) or SUBSTRING (column_name, start, length) - function is used to extract characters from a text field.

Syntax - SELECT MID (column_name, start, length) FROM table_name;
or SELECT SUBSTRING (column_name, start, length) FROM table_name;

LENGTH(column_name): the length of the value in a text field.

Syntax - SELECT LENGTH(column_name) FROM table_name;

Note: ये space को भी count करता है।

→ mysql> SELECT * FROM product;

id	pname	price
1	Mobile	69999.56
2	Computer	16000.56
3	Laptop	25000.00
4	Printer	10000.23

mysql> SELECT MID (pname, 1, 3) FROM product;

MID(pname, 1, 3)
Mob
Com
Lap
Pri

"Whenever you take a step forward, you are bound to disturb something." –Indira Gandhi

mysql> SELECT SUBSTRING(pname,1,3) FROM product;

SUBSTRING(pname, 1,3)
Mob
Cop
Lap
Pri

mysql> SELECT pname, LENGTH(pname) FROM product;

pname	LENTH(pname
Mobile	6
Computer	7
Laptop	6
Printer	7

CONCAT (column_name1, column_name2, ____) - It joints two column.

Syntax- SELECT ~~table_name column_name~~ CONCAT(column_name1, column_name2, ____) FROM table_name;

→ mysql> SELECT * FROM product;

id	pname	price
1	Mobile	69999.56
2	Computer	16000.56
3	Laptop	25000.00
4	Printer	10000.23

mysql> SELECT CONCAT (pname, price) FROM product;

CONCAT (pname, price)
Mobile69999.56
Computer16000.56
Laptop25000.00
Printer10000.23

Date ___/___/___

mysql> SELECT id, CONCAT (pname, ' ', price) FROM product;

id	CONCAT (pname, ' ', price)
1	Mobile 69999.56
2	Computer 16000.56
3	Laptop 25000.00
4	Printer 10000.23

mysql> SELECT id, CONCAT (price, ',', (SELECT MID (pname, 1, 3))) FROM product;

id	CONCAT (price, ',', (SELECT MID (pname, 1, 3)))
1	69999.56, Mob
2	16000.56, Cop
3	25000.00, Lap
4	10000.23, Pri

REVERSE ((column_name) - It reverse the order of letter in string.

Syntax - ~~SELECT column name~~
SELECT REVERSE(column_name) FROM table name;

→ mysql> SELECT * FROM sch;

stu_id	name	marks
1	Ram	100
2	Rani	340
3	Soni	260
4	Jai	390

mysql> SELECT REVERSE(name) FROM sch;

REVERSE(name)
maR
inaR
inoS
iaJ

"Whenever you take a step forward, you are bound to disturb something." —Indira Gandhi

NOW()- function returns the current system date and time.

Syntax- SELECT column_name, NOW() FROM table_name;

→ mysql> SELECT stu_id, marks, NOW() AS DateTime FROM sch;

stu_id	marks	DateTime
1	100	2019-06-27 03:13:17
2	340	2019-06-27 03:13:17
3	260	2019-06-27 03:13:17
4	390	2019-06-27 03:13:17

GROUP BY- Generally हम इसका use function के साथ करते है ये group बनाकर दिखाता है।

→ mysql> SELECT * FROM emp;

empid	emp_name	dept	salary
101	Ram	IT	50000
102	Amit	Research	45000
103	Pooja	IT	30000
104	Sonam	HR	50000
105	Pooja	Trainer	10000
106	Sonam	Accountant	25000

mysql> SELECT emp_name, MIN(salary) FROM emp GROUP By emp_name;

emp_name	MIN(salary)
Amit	45000
Pooja	25000
Ram	50000
Sonam	25000

mysql> SELECT * FROM order;

OrderID	CustID	SellerID
1	101	201
2	102	202
3	103	204
4	101	205
5	101	301
6	102	401
7	103	301
8	103	345

mysql> SELECT CustID, COUNT(*) FROM order
~~GROUP BY CustID~~ WHERE CustID = 101;

Cust_id	COUNT(*)
101	3

mysql> SELECT CustID, COUNT(*) FROM order
GROUP By CustID;

Cust_id	COUNT(*)
101	3
102	2
103	3

HAVING Clause- इसका use GROUP BY के साथ करते हैं। (condition)

→ mysql> SELECT emp_name, MIN(salary) FROM emp
GROUP By emp_name HAVING MIN(salary) > 2500;

emp_name	MIN(salary)
Amit	45000
Ram	50000

```
mysql> SELECT CustID, COUNT(*) FROM order
GROUP BY CustID;
```

CustID	COUNT(*)
101	3
102	2
103	3

```
mysql> SELECT CustID, COUNT(*) FROM order
GROUP BY CustID HAVING COUNT(*) >2;
```

CustID	COUNT(*)
101	3
103	3

Why do we need Multi Table in SQL - जब हमारे table के किसी Column में value atomic नहीं रहती है तो उसके लिये Query करने में दिक्कत होती है तब हम उस Column के लिये अलग table बनाते हैं

Id	Sname	Course Enrolled
1	Nilu	Ms-office, Linux, Windows
2	Rahul	Linux, PotoEditing, windows
3	Sonam	Linux, Windows
4	Sujit	Windows
5	Jay	Linux

① table ② table

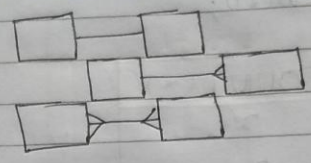
~~Relat~~

Note - foreign key unique नहीं भी हो सकता है ~~या~~ ~~है~~ ~~सकता~~ ~~है~~ ~~पर~~ Primary key हमेशा unique रहता है

Date ___/___/___

Relationship - एक table से अधिक table से relation बना सकता है वो भी different types के।

1- One to One
2- One to Many
3- Many to Many



(ज्यादा use होता है।)

1- **One to One** - Ex- Person to Aadhaar

Person

ID	Name	City	Mobile
1	Rahul	Patna	2569874
2	Sonam	Kolkata	4587985
3	Kunal	Mumbai	478954

Aadhaar

ID	Aadhaar No.	State	Issue Date
1	598 7458	Bihar	12-12-2000
2	5981254	West Bengal	11-09-2003
3	5412635	MH	07-01-2001

Drawback -

ID	Name	City	Mobile	Aadhaar No.	State	Issue Date
1	Rahul	Patna	2569874	5987458	Bihar	12-12-2000
2	Sonam	Kolkata	4587985	5981254	West Bengal	11-09-2003
3	Kunal	Mumbai	478954	5412635	MH	07-01-2001

2- **One to Many** - Ex- Customer to Order

Customer

CustID	Name	City	Mobile
1	Rahul	Patna	2569874
2	Sonam	Kolkata	4587985
3	Kunal	Mumbai	478954

Order

OrID	Quantity	Date	CustID
451	12	12-12-2000	1
298	54	11-09-2003	1
569	17	07-01-2001	3

3- Many to Many - Ex- Author to Book

Author

AuthID	Name	City	Mobile
101	Rahul	Patna	2569874
102	Sonam	Kolkata	4587985
103	Kunal	Mumbai	4789854

Book

BookID	Name	Date	Price
151	DBMS	12-01-2000	300
152	Data	12-02-2003	200
153	C	16-3-2006	400

How to relate - use third table called link table or junction table.

AuthBook.

AuthID	BookID
151	~~350~~ 151
152	152
153	153

Normalization -

1- 1NF (1st normal form)
2- 2NF (2nd normal form)
3- 3NF (3rd normal form)

others - 4NF, 5NF, CNF, HigherNF etc.

"Failure comes only when we forget our ideals and objectives and principles." --Jawaharlal Nehru

1- 1NF- No redundancy.

User

UserID	Name
1	Rahul
2	Sonam
3	Sumit
4	Priti
5	Ram
6	Sam

Interest

ID	UserID	Interest
101	1	PC Game
102	1	Cricket
103	1	Chat
104	2	Tennis
105	2	Game
106	2	Study

2- 2NF- ~~Any~~ Non-Key fields are depend only Primary Key

Course Code	Date	Cname	Seat	Remain	Room	Rcapa
C125	12-01-2014	C Programming	12	5	101	15
DS144	12-01-2014	Data Structure	45	22	102	50
C125	21-07-2014	C Programming	15	11	101	15
J678	08-11-2014	Java Programming	15	2	102	50

Non Key field Cname does not depend upon Primary Key Course Code.

Course Code	Date	Seat	Remain	Room	Rcapa
C125	12-01-2014	12	5	101	15
DS144	12-01-2014	45	22	102	50
C125	21-07-2014	15	11	101	15
J678	08-11-2014	15	2	102	50

Course Code	Cname
C125	C Programming
DS144	Data Structure
J678	Java Programming

Relationship- One to many.

Note- 2NF में होने के लिये 1NF में होना required है।

2. 3NF- Any Non-key does not depend upon another non-key field. 3NF के लिये 2NF होना required है।

In the above given example of 2NF, the Non key field Rcapa is ~~depend~~ depend upon the another Non key field Room.

Course Code	Date	Seat	Remain	Room
C125	12-01-2014	12	5	101
DS144	12-01-2014	45	22	102
C125	21-07-2014	15	11	101
J678	08-11-2014	15	2	102

Course Code	Cname
C125	C Programming
DS144	Data Structure
J678	Java Programming

Room	Rcapa
101	15
102	50

FOREIGN KEY-

- A FOREIGN KEY is one table points to a PRIMARY KEY in another table.
- A FOREIGN KEY can have a different name than the primary key it came from.
- The primary used by a foreign key is also known as a parent key. The table where the

primary key is from is known as a parent table.

- The foreign key can be used to make sure that the row in one table have corresponding row in another table.
- Foreign key value can be null, even though primary key value can't.
- Foreign key don't have to be unique in fact, they often aren't.

Primary Key

Emp_ID	Name	Add
1	Rani	Delhi
2	Jay	Mum
3	Veera	Kol

. Parent Table

Foreign Key

DepID	DName	Emp_ID
1	IT	1
2	HR	1
3	Admin	2

. Child Table

How to CREAT TABLE with FOREIGN KEY-

Syntax- CREATE TABLE table_name
(
column_name datatype PRIMARY KEY,
column_name2 datatype, ____,
foreign key column name same datatype,
CONSTRAINT cname FOREIGN KEY
(foreign key column name) REFERENCES
parent_table_name (Parent table Primary Key_name)
);

- The CONSTRAINT clause allows to define constraint name for the foreign key constraint. If we omit it, MySQL will generate a name automatically. It is optional.
- The REFERENCES clause specifies the parent table and its columns to which the columns in

the child table refer. The number of columns in the child table and parent table specified in the FOREIGN KEY and REFERENCES must be the same.

```
mysql>CREATE TABLE employee
    ->(
    -> eid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -> ename VARCHAR(40),
    -> address VARCHAR(40)
    ->);
mysql>DESC employee;
```

Field	Type	Null	Key	Default	Extra
eid	int(11)	NO	PRI	NULL	auto_increment
ename	varchar(40)	YES		NULL	
address	varchar(40)	YES		NULL	

```
mysql>CREATE TABLE department
    ->(
    -> did INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -> dname VARCHAR(40),
    -> empid INT NOT NULL,
    -> FOREIGN KEY(empid) REFERENCES employee
(eid)
    ->);
mysql>DESC department;
```

Date ___/___/___

Field	Type	Null	Key	Default	Extra
did	int(11)	NO	PRI	NULL	auto_increment
dname	varchar(40)	YES		NULL	
empid	int(11)	NO	MUL	NULL	

```
mysql> DROP TABLE department;

mysql> CREATE TABLE department
    -> (
    -> did INT NOT NULL AUTO_INCREMENT PRIMARY
KEY,
    -> dname VARCHAR(40),
    -> empid INT NOT NULL,
    -> CONSTRAINT employee_did_fk
    -> FOREIGN KEY(empid) REFERENCES employee(did)
    -> );

mysql> DESC department;
```

Field	Type	Null	Key	Default	Extra
did	int(11)	NO	PRI	NULL	auto_increment
dname	varchar(40)	YES		NULL	
empid	int(11)	NO	MUL	NULL	

HOW to Find CONSTARINTS Of a table-

Syntax- SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS WHERE TABLE_NAME = 'table_name';

```
-> mysql> SELECT * FROM INFORMATION_SCHEMA
.TABLE_CONSTRAINTS
    -> WHERE TABLE_NAME = 'department';
```

CONSTRAINT_CATALOG	CONSTRINT_SCHEMA	CONSTRAINT_NAME
def	my_db	PRIMARY
def	my_db	employee_id_fk

TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	ENFORCED
my_db	department	PRIMARY KEY	YES
my_db	department	FOREIGN KEY	YES

HOW to remove FOREIGN KEY from a table -

Syntax- ALTER TABLE table_name
DROP FOREIGN KEY constraint_name;

for some other RDBMS- ALTER TABLE table_name
DROP CONSTRAINT constraint_name;

कुछ RDBMS ये syntax को support करते हैं

→ mysql> ALTER TABLE department
→ DROP FOREIGN KEY employee_id_fk;

mysql> SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
→ WHERE TABLE_NAME='department';

CONSTRAINT_CATALOG	CONSTRINT_SCHEMA	COSTRINT_NAME
def	my_db	PRIMARY

TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	ENFORCED
my_bd	department	PRIMARY KEY	YES

ADD FOREIGN KEY CONSTRAINT in a table -

1- Make any existing column as a FOREN KEY -

Syntax- ALTER TABLE table_name
ADD CONSTRAINT constraint_name
FOREIGN KEY (column_name) REFERENCES
parent_table_name (parent_table primary_key_name);

"Failure comes only when we forget our ideals and objectives and principles." –Jawaharlal Nehru

2- Make non existing column as a FOREIGN KEY in a table-

Syntax- ALTER TABLE table_name
-> ADD COLUMN column_name same_datatype,
-> ADD CONSTRAINT constraint_name
-> FOREIGN KEY(column_name) REFERENCES parent_table_name(parent_table primary key_name);

-> mysql> CREATE TABLE departments
-> (
-> did INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
-> dname VARCHAR (40),
-> empid INT NOT NULL
->);
mysql> DESC departments;

Field	Type	Null	Key	Default	Extra
did	int(11)	NO	PRI	NULL	auto_increment
dname	varchar(40)	YES		NULL	
empid	int(11)	NO		NULL	

mysql> ALTER TABLE departments
-> ADD CONSTRAINT employee_id_fk
-> FORIGN KEY(empid) REFERENCES employee(id);
mysql> DESC departments;

Field	Type	Null	Key	Default	Extra
did	int(11)	NO	PRI	NULL	auto_increment
dname	varchar(40)	YES		NULL	
empid	int(11)	NO	MUL	NULL	

mysql> DROP TABLE departments;

```
mysql> CREATE TABLE departments
    -> (
    -> did INT NOT NULL AUTO_INCRIMENT
PRIMARY KEY,
    -> dname VARCHAR(40)
    -> );
mysql> ALTER TABLE departments
    -> ADD COLUMN empid INT NOT NULL,
    -> ADD CONSTRAINT employee(eid);
mysql> DESC departments;
```

Field	Type	Null	Key	Default	Extra
did	int(11)	NO	PRI	NULL	auto_increment
dname	varchar(40)	YES			
empid	int(11)	NO	MUL	NULL	

Note- हमें Parent table को delete करने से पहले उसके child table को delete ~~करने से~~ करना होगा।

```
mysql> SELECT * FROM employee;
```

eid	ename	address
1	Rani	Delhi
2	Sumit	Kol
3	Rahul	Mum
4	Raj	Delhi
5	Jay	Hyd
6	Veeru	Patna

```
mysql> SELECT * FROM ~~dept~~ department;
```

PTO.

Note- यहाँ employee Parent table है ~~जिसका~~ Primary key eid department table का foreign key empid है।

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exs	2
6	IT	3
7	Exs	3
8	IT	4
9	HR	5
10	Admin	6

```
mysql> DELETE FROM employee
    -> WHERE eid = 1;
ERROR 1451 (23000): Cannot delete or update a parent
row: a foreign key constraint fails ('geekyshows'.
'department', CONSTRAINT 'employee_eid_fk' FOREIGN KE
mysql> DELETE FROM department
    -> WHERE did = 1;
mysql> DELETE FROM department
    -> WHERE did = 2;
mysql> DELETE FROM department
    -> WHERE did = 3;
mysql> SELECT * FROM department;
```

did	dname	empid
4	IT	2
5	Exs	2
6	IT	3
7	Exs	3
8	IT	4
9	HR	5
10	Admin	6

~~mysql> SELECT * FROM employee;~~

mysql > DELETE FROM employee
→ WHERE eid = 1;

mysql > SELECT * FROM employee;

eid	ename	address
2	Sumit	Kol
3.	Rahul	Mum
4	Raj	Delhi
5	Jay	Hyd
6	Veeru	Patna

ON DELETE Clause - (for T-) इसका use करने पर यदि हम record को direct delete कर सकते हैं और यदि parent table के किसी भी foreign key record, child table में होगा तो automatic delete कर दिया जायेगा।

1- ON DELETE CASCADE
2- ON DELETE SET NULL
3- ON DELETE NO ACTION
4- ON DELETE RESTRICT

1- ON DELETE CASCADE - इसका use करने से Parent table का कोई record delete होने पर child table में उपस्थित उस record से संबंधित record automatic delete हो जायेगा।

→ mysql > SELECT * FROM employee;

eid	ename	address
1	Rani	Delhi
2	Sumit	Kol
3	Sonam	Hdy
4	Priti	Mum
5	Kunal	Kol

Date ___/___/_____

```
mysql>CREATE TABLE department
    ->(
    -> did INT NOT NULL AUTO_INCREMENT
PRIMARY KEY,
    ->dname VARCHAR(40),
    ->empid INT NOT NULL,
    -> CONSTRAINT employee_eid_fk
    ->FOREIGN KEY(empid) REFERENCES
employee(eid)
    ->ON DELETE CASCADE
    ->);
mysql> INSERT INTO department (dname, empid)
    ->VALUES ('HR',1),('Admin',1),('IT',2),('Exe',2
('IT',3),('IT',4),('HR',5);
mysql>SELECT * FROM department
```

did	dname	empid
1	HR	1
2	Admin	1
3	IT	2
4	Exe	2
5	IT	3
6	Exe	3
7	IT	4
8	HR	5

```
mysql>DELETE FROM employee
    ->WHERE eid=1;
mysql>SELECT * FROM employee;
```

eid	ename	address
2	Sumit	Kol
3	Sonam	Hdy
4	Priti	Mum
5	Kunal	Kol

"Whenever you take a step forward, you are bound to disturb something." --Indira Gandhi

mysql> SELECT * FROM department;

did	dname	empid
3	IT	2
4	Exe	2
5	IT	3
6	Exe	3
7	IT	4
8	HR	5

2- ON DELETE SET NULL - इसका use करने से Parent table से कोई record delete करने पर उससे संबन्धित Child table के record में NULL set हो जायेगा। Child table में foreign key वाले column को NOT NULL नहीं लगाना है।

-> mysql> SELECT * FROM employee;

eid	ename	address
1	Rani	Delhi
2	Sumit	Kol
3	Sonam	Hdy
4	Priti	Mum
5	Kunal	Kol

```
mysql>CREATE TABLE department
    ->(
    ->did INT NOT AUTO_INCREMENT PRIMARY KEY,
    ->dname VARCHAR(40),
    ->empid INT,
    ->CONSTRANT employee_eid_fk
    ->FOREIGN KEY(empid) REFERENCES employee(eid),
    ->ON DELETE SET NULL
    ->);
```

```
mysql>INSERT INTO department (dname, empid)
    ->VALUES ('IT',1), ('HR',1), ('Admin',1), ('IT',2),
('Exe',2), ('IT',3), ('Exe',3), ('IT',4), ('HR',5);
mysql> SELECT * FROM department;
```

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

```
mysql>DELETE FROM employee
    ->WHERE eid=1;
mysql> SELECT * FROM employee;
```

eid	ename	address
2	Sumit	Kol
3	Sonam	Hdy
4	Priti	Mum
5	Kunal	Kol

```
mysql>SELECT * FROM department;
```

did	dname	empid
1	IT	NULL
2	HR	NULL
3	Admin	NULL
4	IT	2

5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

3&4 ON DELETE NO ACTION & ON DELETE RESTRICT - जब delete use करने पर पहले child table की foreign key की value को delete करने के बाद होता Parent table की Primary key के record को delete कर सकते हैं value by default रहती है।

-> mysql> SELECT * FROM employee;

eid	ename	address
1	Rani	Delhi
2	Sumit	kol
3	Sonam	Mdy
4	Priti	Mum
5	Kunal	kol

mysql> CREATE TABLE department
->(
-> eid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
->ename VARCHAR(40),
->address VARCHAR(40)
->);

~~mysql> INSERT INTO employee (name, address
-> VALUES('Rani', 'Delhi'),('Sumit', 'Kol'), ('Sonam'), ('Priti~~

mysql> INSERT INTO department (dname, empid)
-> VALUES ('IT', 1), ('HR', 1), ('Admin', 1), ('IT', 2), ('Exe', 2), ('IT', 3), ('IT', 4), ('HR', 5);

mysql> SELECT *FROM department;

Date ___/___/___

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exu	2
6	IT	3
7	Exu	3
8	IT	4
9	HR	5

```
mysql> DELETE FROM employee
    -> WHERE eid=1;
ERROR 1451 (23000): Cannot delete or update row
a foreign key constraint fails ('my_db', 'department',
CONSTRAINT 'employee_eid_fk' FOREIN KEY (empid)
REFERENCES 'employee' ('eid') ON DELETE RESTRICT
mysql> DELETE FROM department
    -> WHERE did=1;
mysql> DELETE FROM department
    -> WHERE ~~FROM~~ did=2;
mysql> DELETE FROM department
    -> WHERE did=3;
~~mysql> DELETE FROM employee~~
    ~~-> WHERE~~
mysql> SELECT * FROM department;
```

did	dname	empid
4	IT	2
5	Exu	2
6	IT	3
7	Exu	3
8	IT	4
9	HR	5

```
mysql> DELETE FROM employee
    -> WHERE eid=1;
mysql> SELECT * FROM employee;
```

eid	name	address
2	Sumit	Kol
3	Sonam	Hdy
4	Priti	Mum
5	Kunal	Kol

ON UPDATE -

1- ON UPDATE CASCADE
2- ON UPDATE SET NULL
3- ON UPDATE NO ACTION
4- ON UPDATE RESTRICT

1- **ON UPDATE CASCADE** - इसका use करने पर यदि हम Parent table के PRIMARY KEY के किसी record को change करते है तो उसके अनुसार child table के foreign key में भी changing हो जायेगा।

```
-> mysql> SHOW TABLES;
```

Tables_in_my_db
employee

```
mysql> CREATE TABLE department
    -> (
    -> did INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -> dname VARCHAR(40),
```

Date ___/___/_____

```
-> empid INT
-> CONSTRAINT employee_id_fk
->FOREIGN KEY (empid) REFERENCES
employee(eid)
-> ON UPDATE CASCADE
->);
mysql>INSERT INTO department (dname, empid)
-> VALUES ('IT',1),('HR',1),('Admin',1),('IT',2),
('Exe',2),('IT',3),('Exe',3),('IT',4),('HR',5);
mysql> SELECT * FROM employee;
```

eid	ename	address
1	Rani	Delhi
2	Sumit	Kol
3	Sonam	Hdy
4	Preeti	Mum
5	Kunal	Kol

```
mysql> SELECT * FROM department;
```

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

```
mysql> UPDATE employee
-> SET eid=10
->WHERE eid=1;
```

mysql> SELECT * FROM department;

did	dname	empid
1	IT	10
2	HR	10
3	Admin	10
4	IT	2
5	Ex	2
6	IT	3
7	Ex	3
8	IT	4
9	HR	5

mysql> SELECT * FROM employee;

eid	ename	address
2	Sumit	Kol
3	Sonam	Hdy
4	~~Kunal~~ Priti	Mum
5	~~Rani~~ Kunal	Kol
10	Rani	Delhi

2- ON UPDATE NULL- इसका use करने पर यदि Parent table के Primary key के Value को किसी Column के लिये update करते है तो उसके corresponding child table के foreing key की value को NULL set कर दिया जाता है।

→ mysql> SHOW TABLES

Tables_in_mydb
employee

Date ___/___/___

```
mysql> CREATE TABLE department
    ->(
    -> did INT NOT NULL AUTO_INCREMENT
PRIMARY KEY,
    -> dname VARCHAR(40),
    -> empid INT,
    -> CONSTRAINT employee_eid_fk
    -> FOREIGN KEY(empid) REFERENCES
employee(eid)
    -> ON UPDATE SET NULL
    ->);
mysql> INSERT INTO department (dname, empid)
    -> VALUES ('IT',1), ('HR',1), ('Admin',1), ('IT',2),
('Exe',2), ('IT',3), ('Exe',3), ('IT',4), ('HR',5);
mysql> SELECT * FROM employee;
```

eid	ename	address
1	Rani	Delhi
2	Sumit	Kol
3	Sonam	Hdy
4	Priti	Mum
5	Kunal	Kol

```
mysql> SELECT * FROM department;
```

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

"Whenever you take a step forward, you are bound to disturb something." —Indira Gandhi

```
mysql> UPDATE employee
    -> SET eid=10
    -> WHERE eid = 1;
mysql> SELECT * FROM employee;
```

eid	ename	address
~~*~~	~~Rani~~	~~Delhi~~
2	Sumit	Kol
3	Sonam	Hyd
4	Priti	Mum
5	Kunal	Kol
10	Rani	Delhi

```
mysql> SELECT * FROM department;
```

did	dname	mgrid
1	IT	NULL
2	HR	NULL
3	Admin	NULL
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

314- ON UPDATE NO ACTION & ON UPDATE RISTRICT - इसका use करने पर यदि हमें Parent के किसी parent row को update करना है तो पहले हमें उसके corresponding child table के foreign row को update करना होगा, तो value bydefault set रहती है।

→ mysql> SHOW TABLES;

Table_in_my_db
employee

- mysql> CREATE TABLE department
  ->(
  -> did INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  -> ~~empid empid INT~~ dname VARCHAR(40),
  -> empid INT
  -> CONSTRAINT employee_eid_fk
  -> FOREIGN KEY(empid) REFERENCES employee(eid)
  -> ON UPDATE NO ACTION
  ->);

mysql> INSERT INTO department (dname, empid)
  -> VALUES ('IT',1),('HR',1),('Admin',1),('IT',2),('Exa',2),('IT',3),('Exa',3),('IT',4),('HR',5);

mysql> SELECT * FROM employee;

eid	ename	address
1	Rani	Delhi
2	Sumit	Kol
3	Sonam	Hdy
4	Priti	Mum
5	Kunal	Kol

mysql> SELECT * FROM department;

did	dname	empid
1	IT	1
2	HR	1

3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

```
mysql> UPDATE employee
    -> SET eid=10
    -> WHERE eid=1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('my_db', 'department', CONSTRAINT 'employee_eid_fk' FOREIGN KEY ('empid') REFERENCES 'employee' ('eid'))
```

Composite Key- जब दो या दो से अधिक column को मिलाकर हम Primary key बनाते हैं तो उसे composite key कहते हैं। यहाँ इस example में Course Code और Date को मिलाकर हम पूरे table के data को uniquely identify कर सकते हैं और इन दोनों से मिलकर जो key बनेगा उसे Composite key कहेंगे।

Course Code	Date	Cname	Seat	Remain	Room	Rcapa
C125	12-01-2014	C Programming	12	5	101	15
DS144	12-01-2014	Data Structure	45	22	102	50
C125	21-07-2014	C Programming	15	11	101	15
J678	08-11-2014	Java Programming	15	2	102	50

Date ___/___/___

How to CREATE Composite Key-

Syntax- CREATE table_name
(
column_name1 datatype (size), column_name2 datatype (size) -
PRIMARY KEY (column_name, 2 columnname, ---)
);

```
-> mysql> CREATE TABLE course (
   -> Courscode VARCHAR (40),
   -> Date DATE,
   -> Cname VARCHAR (40),
   -> Seat INT,
   -> Remain INT,
   -> Room INT,
   -> Rcapa INT,
   -> PRIMARY KEY (courscode, date)
   -> );
mysql> DESC course;
```

Field	Type	Null	Key	Default	Extra
Courscode	VARCHAR(40)	NO	PRI	NULL	
Date	date	NO	PRI	NULL	
Cname	varchar(40)	YES		NULL	
Seat	int(11)	YES		NULL	
Remain	int(11)	YES		NULL	
Room	int(11)	YES		NULL	
Rcapa	int(11)	YES		NULL	

Note- Realword database में हम Composite key का use कम से कम करते हैं और हम single column को ही Primary key बनाते हैं।

Shorthand Notation – जब हम Multiple table के साथ काम करते हैं और एक ही नाम के column दोनों ही table में हो तो हम उस column को access करने के लिये shorthand का use करते हैं।

Syntax - columnname.table_name.

Ex -

Student.

Rollno	Name	Branch
1	Jay	Computer Science
2	Suhani	Electronic and Comm.

Exam

Rollno	Name	S_code	Marks	P_code
1	Abc	CS11	50	CS
1	cde	CS12	60	CS
1	Efg	CS13	55	CS
1	Ghi	CS14	70	CS
2	xyz	EC101	66	EC
2	Ab	EC102	70	EC
2	cd	EC103	50	EC

→ Student.rollno, Student.name, Exam.rollno, Exam.name

How to Join Tables - दो table को एक-दूसरे से Join करने के दोनों में से किसी एक table में एक Primary key होना चाहिये और दूसरे table में उसके respect में एक foreign key बना कर दोनों table के बीच relation बनाकर उन्हें Join कर सकते हैं। एक table में एक ही Primary key हो सकता है पर Foreign key एक से अधिक भी हो सकता है। ऊपर दिये गये example में Student table में Rollno Primary key है और Exam table में Rollno foreign key है। ये जरूरी नहीं है की हम दूसरे table में foreign key का name पहले table के Primary key का नाम same रखें।

CROSS JOIN / Cartesian JOIN / Cartesian Product

Cross Product – This Cross Join returns every row from one table crossed with every row from second table.

Syntax – for all table
```
SELECT * FROM table_one_name CROSS JOIN table_two_name;
SELECT * FROM table_one_name, table_two_name;
```

for one column
```
SELECT column1_name, column_name_table2 FROM table1_name CROSS JOIN table2_name;
```

Note – जब हम दो table पर काम करते हैं तो हमें fully-qualified name यानी shorthand का use करना चाहिये।

Ex –

Employee

Empid	Name	City
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum

Department

did	DepName
101	IT
102	HR
103	Admin

```
SELECT Name, Depname FROM Department, Employee;
```

Name	Depname
Rahul	IT
Rahul	HR
Rahul	Admin
Krish	IT
Krish	HR
Krish	Admin
Jay	IT
Jay	HR
Jay	Admin

Note – table1 के हम row का 1 to many relation table2 के सभी row से होगा।

"Whenever you take a step forward, you are bound to disturb something." –Indira Gandhi

```
-> mysql> CREATE TABLE employee
    ->(
    -> empid INT ~~NOT~~ AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR (40),
    -> city VARCHAR (40)
    -> );
mysql> INSERT INTO employee (name, city)
    ->VALUES ('Rahul','Delhi'), ('Krish','Kol'), ('Jay',
                                                 'Mum');
mysql>CREATE TABLE department
    ->(
    -> did INT ~~NOT~~ PRIMRY KEY
    -> depname VARCHAR(40)
    -> );
mysql> INSERT INTO department (did, depname)
    ->VALUES (101,'IT'), (102,'HR'),(103,'Admin');
mysql> SELECT * FROM employee;
```

empid	name	city
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum

```
mysql>SELECT * FROM department;
```

did	depname
101	IT
102	HR
103	Admin

```
mysql>SELECT name, depname FROM employee CROSS
JOIN department;
```

Date ___/___/_____

name	depname
Rahul	IT
Krish	IT
Jay	IT
Rahul	HR
Krish	HR
Jay	HR
Rahul	Admin
Krish	Admin
Jay	Admin

mysql> SELECT name, depname FROM department CROSS ~~department~~ JOIN employee;

name	depname
Rahul	IT
Rahul	HR
Rahul	Admin
Krish	IT
Krish	HR
Krish	Admin
Jay	IT
Jay	HR
Jay	Admin

mysql> SELECT name, depname FROM department, employee;

name	depname
Rahul	IT
Rahul	HR
Rahul	Admin
Krish	IT
Krish	HR
Krish	Admin

Jay	IT
Jay	HR
Jay	Admin

mysql>SELECT * FROM department, employee;

did	depname	empid	name	City
101	IT	1	Rahul	Delhi
102	HR	1	Rahul	Delhi
103	Admin	1	Rahul	Delhi
101	IT	2	Krish	Kol
102	HR	2	Krish	Kol
103	Admin	2	Krish	Kol
101	IT	3	Jay	Mum
102	HR	3	Jay	Mum
103	Admin	3	Jay	Mum

CROSS

INNER JOIN - An INNER JOIN with some result rows removed by a condition in the query.

1- EQUIJOIN
2- NON-EQUI JOIN
3- NATURL JOIN

1- EQUIJOIN(=)-

employee

Syntax- SELECT column-name FROM table1
INNER JOIN table2
ON 1column-name = 2 Column-name;

Note- यदि syntax में table2 का order at Print होगा तो table1 के order में

-> mysql>CREATE TABLE emp
->(
-> empid INT AUTO_INCREMENT PRIMARY KEY,
-> name VARCHAR(40),
-> City VARCHAR(40).
->);

"Failure comes only when we forget our ideals and objectives and principles." —Jawaharlal Nehru

Date ___/___/___

```
mysql> INSERT INTO imp (name, city)
    -> VALUES ('Rahul', 'Delhi'), ('Krish', 'Kol'), ('Jay', 'M
mysql> CREATE TABLE dep
    -> (
    -> did INT PRIMARY KEY,
    -> depname VARCHAR(40),
    -> empid INT,
    -> CONSTRAINT employee_did_fk
    -> FOREIGN KEY(empid) REFERENCES emp(em
    -> );
mysql> INSERT INTO dep (did, depname, empid)
    -> VALUES (101, 'IT', 3), (102, 'HR', 1), (103, 'Admi
mysql> SELECT * FROM emp;
```

empid	name	city
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum

```
mysql> SELECT * FROM dep;
```

did	depname	empid
101	IT	3
102	HR	1
103	Admin	2

```
mysql> SELECT emp.name, dep.dename FROM emp
    -> INNER JOIN dep
    -> ON emp.empid = dep.empid;
```

name	depname
Jay	IT
Rahul	HR
Krish	Admin

```
mysql> SELECT emp.name, dep.depname FROM emp
    ->INNER JOIN dep
    ->ON ~~emp.empid = dep.empid~~;
    ->ON dep.empid = emp.empid;
```

name	depname
Jay	IT
Rahul	HR
Krish	Admin

```
mysql> SELECT emp.name, dep.depname FROM dep
    ->INNER JOIN emp
    -> ON emp.empid = dep.empid;
```

name	depname
Rahul	HR
Krish	Admin
Jay	IT

```
mysql> SELECT emp.name, dep.depname FROM dep
    -> INNER JOIN emp
    -> ON dep.empid = emp.empid;
```

name	depname
Rahul	HR
Krish	Adimn
Jay	IT

```
mysql> SELECT emp.name, dep.depname FROM dep
    ->INNER JOIN emp
    ->ON ~~dep~~ emp.empid = dep.empid;
```

name	depname
Rahul	HR
Krish	Admin
Jay	IT

"Failure comes only when we forget our ideals and objectives and principles." – Jawaharlal Nehru

2- ~~INNER~~ NOT-EQUI JOIN (<>) -

Syntax- SELECT column_name FROM table1
INNER JOIN table2
ON column_name <> column_name;

-> mysql>SELECT * FROM emp;

empid	name	city
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum

mysql> SELECT * FROM dep;

did	depname	empid
101	IT	3
102	HR	1
103	Admin	2

mysql> SELECT emp.name, dep.depname FROM dep
-> INNER JOIN emp
-> ON emp.empid <> dep.empid;

name	depname
Rahul	IT
Rahul	Admin
Krish	IT
Krish	HR
Jay	HR
Jay	Admin

3- NATURAL JOIN- ये बिलकुल EQUI JOIN की तरह ही work करता है पर इसे use करने के लिये PRIMARY KEY (table1) का और FOREING KEY (table2) का same name का होना चाहिये।

Syntax- SELECT column name FROM table1 NATURAL JOIN table2;

-> mysql> SELECT emp.name, dep.depname FROM dep
-> NATURAL JOIN emp;

name	depname
Rahul	HR
Krish	Admin
Jay	IT

Outer Joins- An outer joins returns all rows from one of the tables, along with matching information from another table.

1- LEFT OUTER JOIN / LEFT JOIN
2- RIGHT OUTER JOIN / RIGHT JOIN
3- FULL OUTER JOIN / FULL JOIN

1- LEFT OUTER JOIN / LEFT JOIN - The LEFT JOIN keyword returns all rows from the left table, with the matching rows in the right table. The result is NULL in the right side when there is no match.
In a LEFT OUTER JOIN the table that comes before the join is the left table, and the table that comes after the join is the ~~right~~ right table.

Syntax- SELECT column_name FROM table1 (Left)
LEFT JOIN table2 (Right Table)
ON column_name = column_name;

Note किसी column के लिये जो कि left table का है उसके लिये Right table में कोई record नहीं मिलता है तो वहाँ NULL value show होता है

-> mysql> SELECT * FROM emp;

empid	name	city
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum
4	Sonam	Hyd
5	Mona	Patna

mysql> SELECT * FROM dep;

did	depname	empid
101	IT	3
102	HR	1
103	Admin	2
104	IT	1

mysql> SELECT emp.name, dep.depname FROM emp
->LEFT JOIN dep
->ON emp.empid = dep.empid;

name	depname
Jay	IT
Rahul	HR
Krish	Admin
Rahul	IT
Sonam	NULL
Mona	NULL

"Whenever you take a step forward, you are bound to disturb something." —Indira Gandhi

mysql> SELECT emp.name, dep.depname FROM emp
-> LEFT JOIN emp
-> ON emp.empid = dep.empid;

name	depname
Jay	IT
Rahul	HR
Krish.	Admin
Rahul	IT

2- RIGHT OUTER JOIN / RIGHT JOIN - The RIGHT JOIN keyword ~~sep~~ returns all rows from the right table, with the matching rows in the left table. The result is NULL in the left side when there is no match. In a RIGHT OUTER the table that comes before the join is the Right table, and the table that comes after the ~~join~~ join is the Left table.

Syntax- SELECT column_name FROM table1 (Right Table)
RIGHT JOIN table2 (Left Table)
ON column_name = column_name;

-> mysql> SELECT emp.name, dep.depname FROM emp
-> RIGHT JOIN dep
-> ON emp.empid = dep.empid;

name	depname
Jay	IT
Rahul	HR
Krish	Admin
Rahul	IT

```
mysql> SELECT emp.name, dep.depname FROM dep
    ->RIGHT JOIN emp
    ->ON emp.empid=dep.empid;
```

name	depname
Jay	IT
Rahul	HR
Krish	Admin
Rahul	IT
Sonam	NULL
Mona	NULL

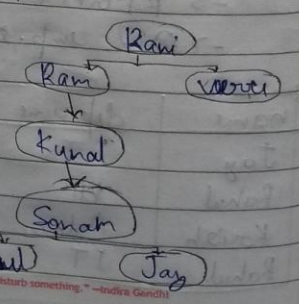
3- FULL OUTER JOIN - The FULL OUTER JOIN returns all rows from the left table and from right table. (Note) - ये सभी RDBMS में काम नहीं करता (MySQL में भी नहीं)
The FULL OUTER JOIN combines the result of both LEFT and RIGHT joins.

Syntax - SELECT column_name FROM table
FULL JOIN table.2
ON column_name = column_name;

SELF JOIN - Self join is a table joined to itself. ये तब use होता है जब हमारी table में Hairchy problem हो।

Ex

Empid	Name	Manager ID
1	Rahul	3
2	Jay	3
3	Sonam	4
4	Kunal	5
5	Ram	6
6	Rani	NULL
7	Veeru	6



→ mysql > DESC empman;

Field	Type	Null	Key	Default	Extra
empid	int(11)	NO	PRI	NULL	
name	varchar(40)	YES		NULL	
manid	int(11)	YES		NULL	

mysql > SELECT * FROM empman;

empid	name	manid
1	Rahul	3
2	Jay	3
3	Sonam	4
4	Kunal	5
5	Ram	6
6	Ravi	NULL
7	Veeru	6

mysql > SELECT e.name Name, m.name Manager
→ FROM empman e
→ INNER JOIN empman m
→ ON e.manid = m.empid;

Name	Manager
Rahul	Sonam
Jay	Sonam
Sonam	Kunal
Kunal	Ram
Ram	Ravi
Veeru	Ravi

mysql > SELECT e.name Name, m.name Manager
→ FROM empname e

Date ___/___/___

-> INNER JOIN empman m
-> ON m.empid = e.manid;

Name	Manager
Rahul	Sonam
Jay	Sonam
Sonam	Kunal
Kunal	Ram
Ram	Rani
Veeru	Rani

mysql> SELECT e.name Name, m.name Manager
-> FROM empman e
-> INNER JOIN empman m
-> ON e.empid = m.manid;

Name	Manager
Sonam	Rahul
Sonam	Jay
Kunal	Sonam
Ram	Kunal
Rani	Ram
Rani	Veeru

mysql> SELECT e.name Name, e.name Manager
-> FROM empman e
-> INNER JOIN empman m
-> ON ~~m.name~~ m.manid = e.empid

Name	Manager
Sonam	Rahul
~~Kunal~~ Sonam	Jay
~~Ram~~ Kunal	Sonam

Rain	Kunal
Rani	Ram
Rani	Veeru

```
mysql > SELECT e.name Name, m.name Manager
    -> FROM empman e
    -> LEFT JOIN empman m
    -> ON e.manid = m.empid
```

Name	Manager
Rahul	Sonam
Jay	Sonam
Sonam	Kunal
Kunal	Ram
Ram	Rani
Rani	NULL
Veeru	Rani

```
mysql > DESC student;
```

Field	Type	Null	Key	Default	Extra
stuid	int(11)	YES		NULL	
name	varchar(40)	YES		NULL	
city	varchar(40)	YES		NULL	

```
mysql > SELECT * FROM student;
```

stuid	name	city
1	Rahul	Delhi
2	Jay	Kol
3	Sonam	Delhi
4	Kunal	Hyd
5	Ram	Delhi
6	Rani	Patna
7	Veeru	Kol

"Failure comes only when we forget our ideals and objectives and principles." ~Jawaharlal Nehru

```
mysql> SELECT s2.name Name, s2.city City
    -> FROM student s1
    -> INNER JOIN student s2
    -> ON s1.city = s2.city AND s1.name = 'Rahul';
```

Name	City
Rahul	Delhi
Sonam	Delhi
Ram	Delhi

```
mysql> SELECT s1.name Name, s1.city City ~~AND s1.name = 'R~~
    -> FROM student s1
    -> INNER JOIN student s2
    -> ON s1.city = s2.city AND s1.name = 'Rahul';
```

Name	City
Rahul	Delhi
Rahul	Delhi
Rahul	Delhi

Note- हम यहाँ आवश्यकता के अनुसार INNER, LEFT, RIGHT और CROSS JOIN use कर सकते है।

How to Replace NULL -

1- IFNULL() - If exp1 is not NULL the IFNULL return exp1 otherwise it return exp2

Syntax- IFNULL(exp1, exp2)

Ex- SELECT IFNULL("GEEKY", "SHOWS") As Exmp

-> SELECT * FROM empman;

empid	name	manid
1	Rahul	3
2	Jay	3
3	Sonam	4
4	Kunal	5
5	Ram	6
6	Rani	NULL
7	Vuru	6

```
mysql> SELECT IFNULL('Geeky','Shower') AS Example;
```

Example
Geeky

```
mysql> SELECT IFNULL(NULL,'Shower') AS Example;
```

Example
Shower

```
mysql> SELECT e.name Name, IFNULL(m.name,'No
Manager') AS Manager
    -> FROM empman e
    -> LEFT JOIN empman m
    -> ON e.manid = m.empid
```

Name	Manager
Rahul	Sonam
Jay	Sonam
Sonam	Kunal
Kunal	Ram
Ram	Rani
Rani	No Manager
Vuru	Rani

~~2. COTA~~

2- COALESCE() -

Syntax- COALESCE(exp1, exp2, ——— expn);

Ex- SELECT COALESCE (NULL, 'Geekyshows') AS Example

-> mysql>SELECT COALESCE(NULL,'NO Manager') AS Example;

Example
NO Manager

mysql> SELECT COALESCE('Geekyshows','NO Manager') AS Example;

Example
Geekyshows

mysql> SELECT e.name Name, COALESCE (m.name,'No Manager') AS Manager
-> FROM empman e
-> LEFT JOIN empman m
-> ON e.manid = m.empid;

Name	Manager
Rahul	Sonam
Jay	Sonam
Sonam	Kunal
Kunal	Ram
Ram	Rani
Rani	NO Manager
Veera	Rani

UNION - A UNION combines the results of two or more quires into one table.

Rule -

- The number of column in each SELECT statement must match.
- The data types in the columns should be same.
- The columns in each SELECT statement ~~must~~ must be in the same order.
- Must have the same expressions and aggregate function in each SELECT statement.
- The UNION operator selects only distinct values by default and it doesn't select duplicate values.

Ex- SELECT name FROM Student
UNION
SELECT name FROM Employee;

-> mysql> SELECT * FROM student;

stid	name	city
1	Rahul	Delhi
2	Jay	Kol
3	Sonam	Delhi
4	Kunal	Hyd
5	Ram	Delhi
6	Rani	Patna
7	Veeru	Kol

mysql> SELECT * FROM employee;

Date / /

empid	name	city
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum

```
mysql> SELECT name FROM student
    -> UNION
    -> SELECT name FROM employee;
```

~~Name~~ name
Rahul
Jay
Sonam
Kunal
Rani
Vivu
Krish

```
mysql> SELECT name AS StudentName FROM student
    -> UNION
    -> SELECT name AS StudentName FROM employee
```

StudentName
Rahul
Jay
Sonam
Kunal
Rani
Vivu
Krish.

```
mysql> SELECT name AS StudentName FROM student
    -> UNION
    -> SELECT name AS EmployeeName FROM employee
```

Student Name
Rahul
Jay
Sonam
Kunal
Ram
Rani
Veeru
Krish

```
mysql> SELECT name AS StudentName FROM Student
    -> UNION
    -> SELECT name AS StudentName FROM employee
ORDER BY StudenName;
```

Student Name
Jay
Krish
Kunal
Rahul
Ram
Rani
Sonam
Veeru

UNION ALL- UNION ALL returns every match whether it is duplicate or distinct ones. (UNION के Rule यहाँ भी apply होंगे)

```
mysql> SELECT name FROM student
    -> UNION ALL
    -> SELECT name FROM employee;
```

name
Rahul
Jay
Sonam
Kunal
Ram
Rani
Varu
Rahul
Krish
Jay

```
mysql> SELECT name FROM student
    -> UNION ALL
    -> SELECT name FROM employee ORDER
BY name;
```

name
Jay
Jay
Krish
Kunal
Rahul
Rahul
Ram
Rani
Sonam
Varu

INTERSECT- INTERSECT returns only those colums that are in the first query and also in the second query. (Note मत mysql इसे support नहीं करता है)

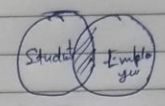
Ex-

Student

Stuid	Name	City
1	Rahul	Delhi
2	Jay	Kol
3	Sonam	Delhi
4	Kunal	Hyd
5	Ram	Delhi
6	Ravi	Patna
7	Varu	Kol

Employee

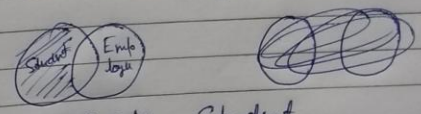
Empid	Name	City
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum




```
SELECT name FROM Student
INTERSECT
SELECT name FROM Employee;
```

Name
Rahul
Jay

EXCEPT- EXCEPT returns only those records that are in the first query but not in the second query



Ex-
```
SELECT name FROM Student
EXCEPT
SELECT name Employee;
```

Name
Sonam
Kunal
Ram
Ravi
Varu