

# MYSQL HANDWRITTEN NOTES

---



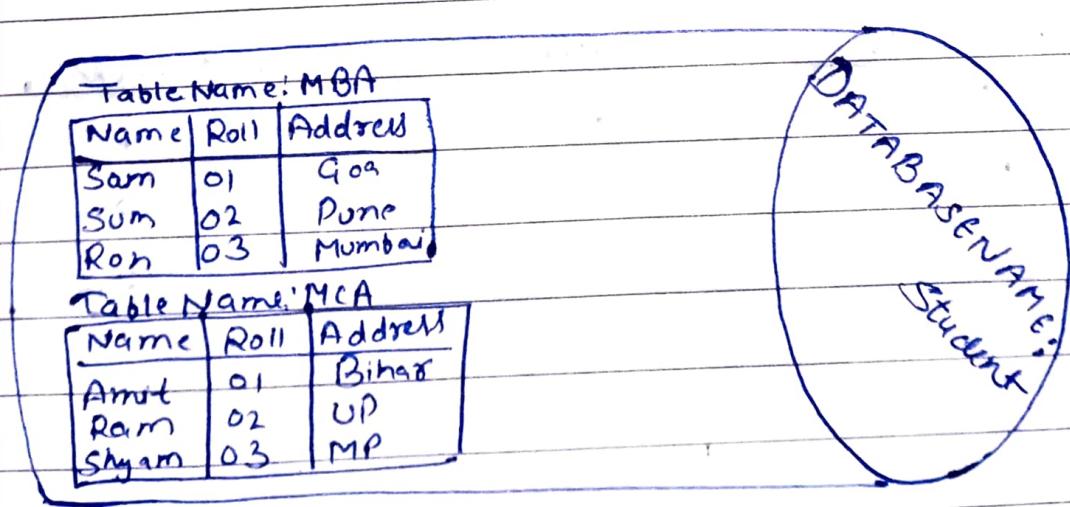
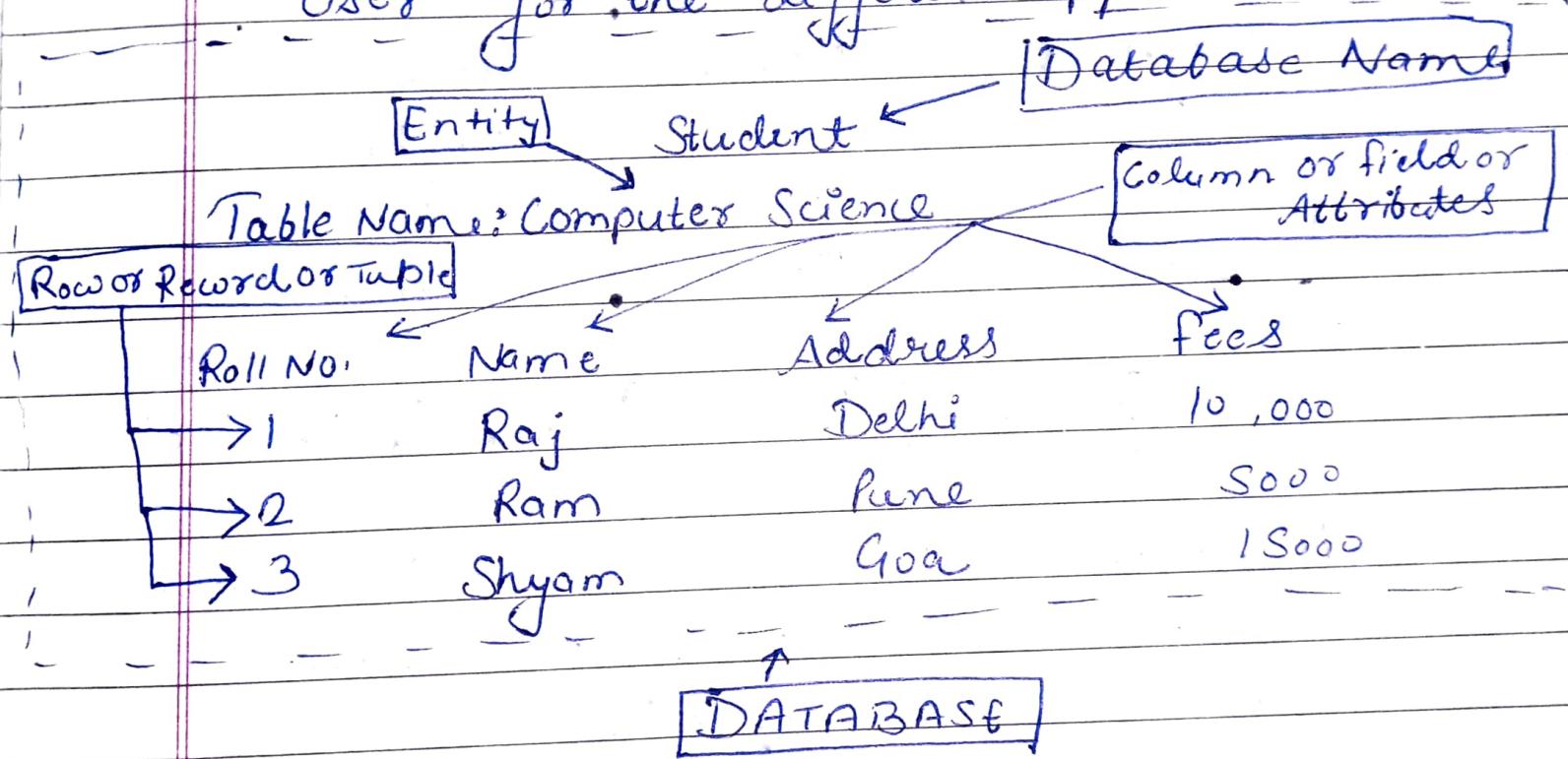
Vaishnavi Pandey

---



## Database

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



## Table / Entity

Table is the structure inside database that contains data, organized in columns & rows.

Student-ID	Name	Address
1	Ram	Delhi
2	Sam	Goa
3	Tom	Bihar

## Column / field / feature / Attributes

The name of each column in a table is used to interpret its meaning and is called an attributes.

Attributes

Student-ID	Name	Address
1	Ram	Delhi
2	Sam	Goa
3	Tom	Bihar

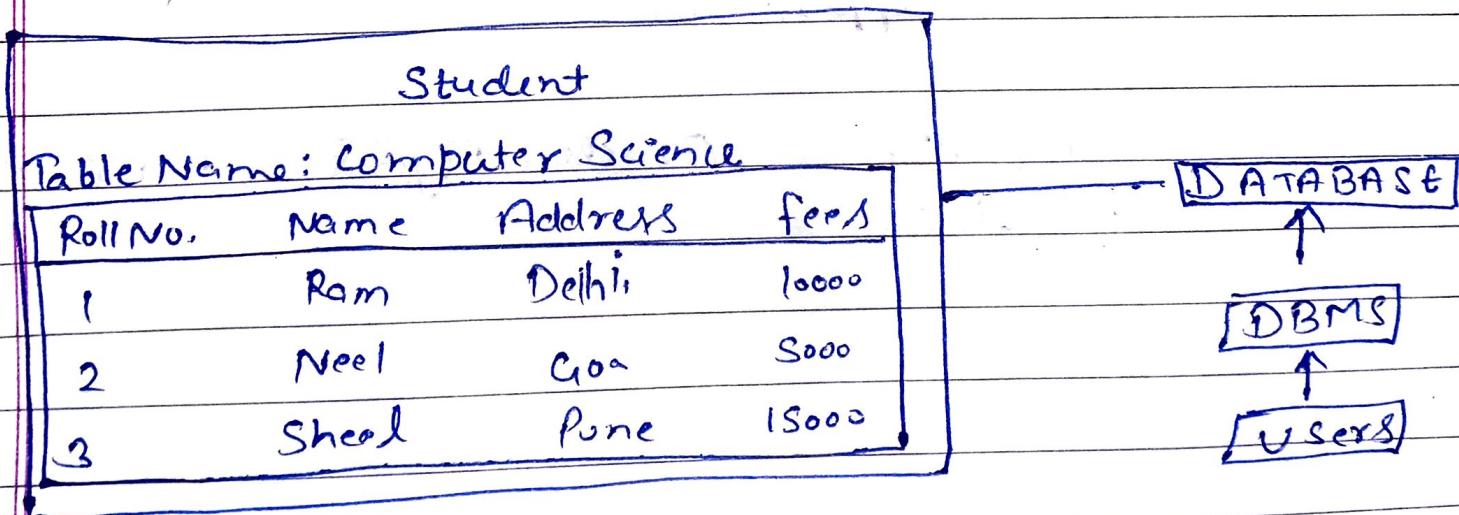
## Row / Record / Tuple

Each row in a table represents a record & is called a tuple

Student ID	Name	Address
1	Ramu	Delhi
2	Nitu	Bihar
3	Sonam	Pune

## Database Management System (DBMS)

DBMS is a software system that allows the access to the data in the database.



Ex - dBBase, Microsoft Access, Libre office Base, Fox Pro.

# Relational Database Management System (RDBMS)

RDBMS avoided the navigation model as in old DBMS and introduced Relational model. The relational model has Relationship b/w tables using primary keys, foreign keys & indexes. Thus the fetching and storing of data becomes 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.

# Structured Query language (SQL)

It is commonly used with all relational database for data definition & manipulation.

## feature of SQL:-

- It is a 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.

# MySQL

DATE / /

PAGE NO.

## CREATE DATABASE

It is used to create database.

Syntax:- CREATE DATABASE database-name;

Ex - CREATE DATABASE my-db;

Guidelines for creation of Database.

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

## USE

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

Syntax:-

CREATE TABLE table\_name

(

Column\_name1 data-type (size) [constraints],

Column\_name2 data-type (size) [constraints],

Column\_name3 data-type (size) [constraints]

);

Ex :-

CREATE TABLE my-tab

(

name varchar (30) NOTNULL,  
roll number (4) PRIMARY KEY,  
address varchar (100)

);

### Guidelines for creation of Table

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

USE sys; (Database changed)

DATE / /

PAGE NO.

## 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;

## SHOW DATABASES / TABLES

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

Syntax:- SHOW DATABASES;

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

Syntax:- SHOW TABLE;

## DATA Types

- INT or INTEGER - It holds whole number between -32,768 and 32,767 either it is negative or positive. It cannot hold a decimal numbers. 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, values have no decimal point. The maximum number of size for decimal is 65 & 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 (20);

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 :-** name VARCHAR (30);

- **TEXT** :- It holds a string with a maximum length of 65,535 characters.

**Syntax :-** column\_name TEXT;

**Ex :-** address TEXT;

- **DATE** :- It displays Date values in yy-mm-dd format.

**Syntax :-** column\_name DATE;

**Ex :-** age DATE;

- **DATETIME** :- It displays DATETIME Values in yyyy-mm-dd hh:mm:ss format.

**Syntax :-** column\_name DATETIME;

**Ex :-** Date\_of\_join DATETIME;

- **TIMESTAMP** - It also displays date and time.

Syntax :- column\_name TIMESTAMP;

Ex :- login\_dt TIMESTAMP;

Stu-id	Name	Address	DOB	fees

Stu-id — INT

Name — VARCHAR

Address — TEXT

DOB — DATE

fees — DEC.

help contents

help data manipulation

help data definition.

DATE / /

PAGE NO.

## DATA MANIPULATION LANGUAGE

### INSERT INTO

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

Syntax:-

```
INSERT INTO table_name (column1, column2,  
column3, column4, ...)  
VALUES (value1, 'value 2', 'value 3', value4...);
```

Ex:-

```
INSERT INTO my-tab (Stu-id, name, address,  
mobile-no).  
VALUES (03, 'Anu', 'Delhi', 982112);  
my-tab.
```

Stu-id	Name	Address	Mobile-no

Rules:-

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

Need single quotes  
CHAR

VARCHAR

DATE

DATETIME

TIMESTAMP

TIME

BLOB

TEXT

No need

INT or INTEGER

DEC or DECIMAL

## INSERT INTO

without specifying column name.

Syntax :-

INSERT INTO table\_name

VALUES (value1, 'value2', 'value3', value4);

Ex :-

INSERT INTO my-tab

VALUES (03, 'Anu', 'Delhi', 982112);

my.tab

Stu-id	Name	Address	Mobile no.
05	Anu	Delhi	982112

Rules:-

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

## INSERT INTO

Changing the order of column

Syntax:-

```
INSERT INTO table_name(column1, column2,  
                      column3, column4),  
VALUES ('value1', 'value2', 'value3', 'value4');
```

Ex:-

```
INSERT INTO my-tab(name, stu-id, mobile-no,  
                   address),  
VALUES ('Anu', 05, 982112, 'Delhi');
```

my-tab				
Stu-id	Name	Address	Mobile-no.	
05	Anu	Delhi	982112	

## INSERT INTO

Insert Data Only in Specified columns

Syntax:-

INSERT INTO table\_name (column1, column2, column3),  
VALUES (value1, 'value2', 'value3');

Ex:-

INSERT INTO my-tab.C\_Student (name, address)  
VALUES ('Anu', 'Delhi');

my-tab.

Stu_id	Name	Address	Mobile_no.
03	Anu	Delhi	NULL

**INSERT Multiple records  
at One time**

Syntax:-

INSERT INTO table\_name  
(column1, column2, column3, column4),  
VALUES (value1, 'value2', 'value3', value4),  
(value1, 'value2', 'value3', value4),

Ex:-

INSERT INTO my-tab .

(Stu\_id , name, address, mobile\_no).

VALUES (01 , 'Anu' , 'Delhi' , 982112),

(02 , 'Rohan' , 'Mumbai' , 56149);

my-tab .

Stu_id	Name	Address	Mobile_no.
01	Anu	Delhi	98211
02	Rohan	Mumbai	56149

## SELECT

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

1. Select all columns from the table .

Syntax :- SELECT \* FROM table-name;

Ex:- SELECT \* FROM my-tab;

2. Select Particular columns from the table

Syntax :- SELECT column-name1 , column-name2  
FROM table-name;

Ex:- SELECT name, mobile  
FROM my-tab;

1)

Name	Roll	Mobile
Rahul	01	8783
John	02	6834

2)

2)

Name	Mobile
Rahul	8783
John	6834

### Single Quotes Problem

INSERT INTO my-tab(t-id, c-name, address)  
VALUES (142, 'K.K's Company', 'Delhi');

There are two way to solve this problem :-

- Use backslash

Ex:- 'K.K's Company'

- Use two time single quotes

Ex:- ''K.K''s Company'

T-id	C-name	Address
142	K.K's Company	Delhi

## 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;

Ex:-

SELECT stu\_id, name

FROM new-tab

LIMIT 3;

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

stu_id	Name
5	Anu
6	Sonu
7	Anu

## 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';`

Stu-id	Name
5	Anu
6	Sonu
7	Anu

Ex:-

`SELECT *`

`FROM new-tab`

`WHERE name = 'Anu';`

Stu-id	Name
5	Anu
7	Anu

Ex:-

`SELECT *`

`FROM new-tab`

`WHERE stu-id = 5;`

Stu-id	Name
5	Anu

## Q. Specific data from specific column

Syntax:-

`SELECT column-name`

`FROM table_name`

`WHERE column-name operator 'value';`

Ex:-

`SELECT name`

`FROM new-tab`

`WHERE name = 'Anu';`

Name
Anu
Anu

Ex:-

```
SELECT name
FROM newtab
WHERE stu_id=5;
```

Name
Anu

\* Note: Value can be text or numeric.  
If it is text then we have to put  
Single quotes.

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 values for a column.

## NULL Value

NULL values represent missing unknown data.

NULL ≠ 0

IS NULL - This is used to select only the records with NULL values 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;
```

IS NOTNULL - 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 city IS NOT NULL;
```

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 = 'Value';
```

stu_id	Name
5	Anu
6	Sonu
7	Anu

Ex:-

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

stu_id	Name
5	Anu

## 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';
```

Stu_id	Name
5	Anu
6	Sonu
7	Anu

Ex:-

```
SELECT * FROM new-tab  
WHERE name = 'Anu'  
OR stu_id = 5;
```

Stu_id	Name
5	Anu
7	Anu

## Combination of AND & OR

```
SELECT * FROM table-name  
WHERE column_name = 'value'  
AND column_name = 'value' OR column_name  
= 'value');
```

Ex:-

```
SELECT * FROM new-tab  
WHERE name = 'Anu'  
AND (stu_id = 5 OR address = 'Kolkata');
```

## IN

The IN Operator allows you to specify multiple values in a WHERE clause.

Syntax:-

SELECT \*

FROM table-name

WHERE column-name IN ('value1', 'value2', ..);

Ex:-

SELECT \*

FROM new-tab

WHERE name IN ('Anu', 'Sonu');

Stu-id	Name
5	Anu
6	Sonu
7	Jai
8	Anu
9	Bob
10	Candy

Stu-id	Name
5	Anu
6	Sonu
7	Anu

## NOT IN

Syntax:-

SELECT \*

FROM table-name

WHERE Column-name NOT IN ('value1', 'value2',  
...);

Ex:-

SELECT \*

FROM new\_tab

WHERE name NOT IN ('Anu', 'Sonu');

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

Stu_id	Name
8	Jai
9	Bob
10	Candy

## BETWEEN

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

### 1. Between Number.

Syntax :-

```
SELECT * FROM table_name
WHERE column_name BETWEEN value1
AND value2;
```

Ex:-

```
SELECT * FROM new-tab
WHERE stu-id BETWEEN 6 AND 8;
Including 6 and 8)
```

Stu-id	Name
5	Anu
6	Sonu
7	Anu
8	Jai

Stu-id	Name
6	Sonu
7	Anu
8	Jai

## 2. Between Text

Syntax:-

SELECT \* FROM table-name

WHERE column-name

BETWEEN 'value 1' AND 'value 2';

Ex:-

SELECT \* FROM new-tab

WHERE name BETWEEN 'B' AND 'J';

(including B but not J)

## 3. Between Date

Syntax:-

SELECT \* FROM table-name

WHERE column-name

BETWEEN 'yyyy/mm/dd' AND  
'yyy/mm/dd';

Ex:-

SELECT \* FROM new-tab;

WHERE date

BETWEEN '2000/01/04' AND '2001/02/14';

(including both dates).

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

## NOT BETWEEN

To display the data which is not in the range.

### 1. NOT BETWEEN Number

Syntax:-

SELECT \* FROM table-name

WHERE column-name

NOT BETWEEN value 1 AND value2;

Ex:-

SELECT \* FROM ProductS.

WHERE stu\_id

NOT BETWEEN 3 AND 7; (including 3 and 7).

### 2. NOT BETWEEN Text

Syntax:-

SELECT \* FROM table-name

WHERE column-name

NOT BETWEEN 'value1' AND 'value2';

Ex:-

SELECT \* FROM new-tab

WHERE name

NOT BETWEEN 'B' AND 'J';

(including B but not include J).

Stu-id	Name	Stu-id	Name
5	Anu	5	Anu
6	Sonu	6	Sonu
7	Anu	7	Anu
8	Jai	8	Jai
9	Bob		
10	Candy		

## NOT BETWEEN

### 3. NOT BETWEEN Date

Syntax:-

SELECT \* FROM table\_name

WHERE column\_name

NOT BETWEEN 'yyyy/mm/dd' AND 'yyyy/mm/dd';

Ex:-

SELECT \* FROM new\_tab

WHERE date

NOT BETWEEN '2000/01/01' AND '2001/02/31';

(including both dates).

## BETWEEN with IN

Syntax:-

SELECT \* FROM table\_name

WHERE (column\_name BETWEEN value1 AND  
value2).

AND column\_name IN (value1, value2);

Ex:-

SELECT \* FROM emp

WHERE Csalary BETWEEN 25000 AND 50000)

AND dept IN ('IT', 'HR');

## 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 CSalary BETWEEN 25000 AND 50000  
AND NOT dept IN ('IT', 'HR');

## 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 '%mu';

## Wildcards

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

Wildcard	Description
%.	zero or more characters
_	one single character
[charlist]	sets & ranges of characters to match
[^charlist] or [!charlist]	Matches only a character NOT specified within the brackets

1. % - Zero or more characters.

'Geek %' - All starting with geek  
Ex:- Geekyshows.

'% shows' - All ending with shows.

Ex:- Geekyshows.

'%.sh%' - All containing with sh.

Ex:- Geekyshows.

2.  $\textcircled{2}$  — One Single character

'Show\_'

'Show?' — Starting with show then any character. Ex:- shows

'\_eek' — any character then eek. Ex:- geek.

'G-e\_k' — G then any character , then e then any character , then k,  
Ex:- Geek.

### NOT LIKE

Syntax:-

SELECT \*

FROM tablename

WHERE columnname NOTLIKE 'pattern';

Ex:-

SELECT \*

FROM newtab

WHERE name NOT LIKE '%.nu';

## ORDER BY

This is used to sort the records

ASC - It sorts in ascending order (by default)

DESC - It sorts in descending order.

1. Sort in descending order.

Syntax :-

```
SELECT * FROM table_name
```

```
ORDER BY column-name DESC;
```

Ex:-

```
SELECT * FROM emp
```

```
ORDER BY emp-name DESC;
```

2. Sort in ascending order

Syntax:-

```
SELECT * FROM table_name
```

```
ORDER BY column-name ASC; [ASC is optional]
```

Ex:-

```
SELECT * FROM emp
```

```
ORDER BY emp-name;
```

## 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 cannot insert a new record, or update a record without adding a value to this field.

Ex:-

CREATE TABLE Student

C

Name Varchar (30),

Roll integer (5),

Mobile\_no integer (10) NOT NULL

);

## 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 <sup>uniquekey</sup> NULL values.

Name	Roll	Mobile No.
A	1	99 98
B	2	
C	3	97

CREATE TABLE Student

(

Name varchar(30),

Roll integer (S),

Mobile\_no integer (10) UNIQUE KEY  
);

### 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.

CREATE TABLE Student

(

Name Varchar (30),

Roll integer (S) NOTNULL PRIMARY KEY,

Mobile\_no integer (10)

);

CREATE TABLE Student

C

Name varchar (30),

Roll integer (5) NOT NULL,

Mobile\_no integer (10),

PRIMARY KEY (Roll)

;

Name	Roll	Mobile No

## AUTO INCREMENT

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

Syntax:-

CREATE TABLE table\_name

(

column\_name int NOT NULL AUTO\_INCREMENT,

Column\_name1 varchar (50) NOT NULL,

Column\_name2 varchar (50),

PRIMARY KEY (column\_name)

);

Stu_id	Name	City
1	Raj	Delhi
2	Rani	Groa
3	Sonam	Pune

Ex:-

⇒ CREATE TABLE emp;

(  
 Emp-id int NOT NULL AUTO\_INCREMENT,  
 Emp-name varchar (50) NOT NULL,  
 City varchar (50),  
 PRIMARY KEY (Emp-id)  
 );

No need to write emp-id

⇒ INSERT INTO emp (emp-name, city)

VALUES

('Subham', 'Delhi')

('Anky', 'Mumbai');

⇒ INSERT INTO emp (emp-id, emp-name, city)

VALUES

(NULL, 'Sumit', 'Delhi'),

(NULL, 'Joe', 'Pune');

⇒ INSERT INTO emp (emp-id, emp-name, city)

VALUES

(NULL, 'Subham', 'Delhi'),

(1, 'Joe', 'Pune');

Starting AUTO INCREMENT with a particular Number in SQL

Ex:-

`ALTER TABLE ktab AUTO_INCREMENT=10;`

## Aliases

Aliases 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 widyarthi;`

For Column

Syntax:- `SELECT column_name AS alias_name FROM table_name;`

Ex:- `SELECT name AS student_name FROM Student;`

`SELECT name student_name FROM student;`

`SELECT name AS "Student Name" FROM student;`

`SELECT name AS [Student Name] FROM student;`

## Arithmetic Operators

\* / + -

Syntax:- SELECT column\_name, column\_name  
Operator value FROM table\_name;

Ex:-

```
SELECT name, cost, cost+100 FROM items_tab;  
SELECT name, cost, cost+100 AS "NewCost"  
FROM items_tab;
```

## SELECT DISTINCT

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

Syntax:- SELECT DISTINCT column\_name  
FROM table\_name;

Ex:- SELECT DISTINCT name FROM student;

# DATA DEFINITION LANGUAGE

This command is used to Add/changed Modify / Drop existing structure of the table .

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

## ALTER TABLE

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

Syntax :-

ALTER TABLE table-name

ADD COLUMN column-name datatype (size);

Ex:-

ALTER TABLE my-tab

ADD COLUMN stu-id integer (5);

Name	Roll

Name	Roll	Stu-id

## 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);

## ADD COLUMN by Position

- Last (by default)
- First
- After

Syntax:-

① ALTER TABLE table\_name

ADD COLUMN column-name datatype(size) FIRST;

Ex:-

ALTER TABLE my-tab

ADD COLUMN City varchar(50) FIRST;

(2) Syntax :-

ALTER TABLE table\_name

ADD COLUMN column\_name datatype(size)  
AFTER column\_name;

Ex:-

ALTER TABLE my-tab

ADD COLUMN L-name varchar (50) AFTER name;

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 datatype(size)  
constraint\_name;

ADD constraint\_name column\_name;

Ex:-

ALTER TABLE my-tab

ADD COLUMN roll int (10) NOT NULL

ADD PRIMARY KEY (roll);

When integrity constraints have to be included.

Syntax:-

ALTER TABLE table\_name

ADD CONSTRAINT constraint\_name column\_name;

Ex:-

ALTER TABLE my-tab;

ADD const CONSTRAINT PRIMARY KEY (roll);

Change Column Name and Its Data Type without constraints

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

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(3);

Change more than one Column Name and its data type without constraints

Change more than one column

Syntax:-

ALTER TABLE table\_name

CHANGE COLUMN old\_column\_name

new\_column\_name new\_data\_type (size);

CHANGE COLUMN old\_column\_name new\_colu  
-mn\_name new\_data\_type (size);

Ex:-

ALTER TABLE my-tab

CHANGE COLUMN name student varchar (5);

CHANGE COLUMN roll\_id int (5);

Change Column name and its Data type with constraint

Change column with constraints :-

Syntax:-

ALTER TABLE table\_name

CHANGE COLUMN old-column-name new\_column-name

new\_data-type (size) constraint\_name;

ADD constraint\_name (column-name);

Ex:-

```
ALTER TABLE my-tab
CHANGE COLUMN roll id int (5) NOT NULL,
ADD PRIMARY KEY (id);
```

MODIFY COLUMN Data type And Its Size with or without constraint

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

Syntax:-

```
ALTER TABLE table_name
```

```
MODIFY COLUMN column-name datatype (size);
```

Ex:-

```
ALTER TABLE my-tab
```

```
MODIFY COLUMN roll integer (10);
```

Name	Roll
	12345

Name	Roll
	12345678910

# DROP COLUMN

DROP COLUMN with or without constraint.

- 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;

- 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);

## DROP TABLE

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

Syntax:-

`DROP TABLE table-name;`

Ex:-

`DROP TABLE my-tab;`

1

## TRUNCATE TABLE

Delete Data of Table using 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;`

## RENAME TABLE

This command is used to rename one or more table.

Syntax:- RENAME TABLE old\_table\_name to new\_table\_name;

Ex:- RENAME TABLE my-tab to your-tab;

## ALTER DATABASE

ALTER DATABASE enables you to change the overall characteristics of a database. These characteristics are stored 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;

## DROP DATABASE

The DROP DATABASE statement is used to delete a database.

Syntax:- DROP DATABASE database-name;

Ex:- DROP DATABASE db;

## SHOW COLUMNS

It shows all the columns of table and their data type along with any other column specific details. It is just like DESC table\_name.

Syntax:-

SHOW COLUMNS FROM table\_name;

Ex:-

SHOW COLUMNS FROM emp;

## SHOW CREATE DATA BASE

It shows commands which you have written while creating your Database.

Syntax:-

SHOW CREATE DATABASE database\_name;

Ex:-

SHOW CREATE DATABASE my\_db;

## SHOW CREATE TABLE

It shows commands which you have written while creating your table.

Syntax:-

SHOW CREATE TABLE table-name;

Ex:- SHOW CREATE TABLE emp;

## UPDATE

The UPDATE statement 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 = 'Argun', salary = 35000

WHERE emp\_id = 101;

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

## UPDATE with CASE

UPDATE table\_name

SET new\_column =

CASE

WHEN column\_name1 = some\_value1 THEN

THEN new\_value1

WHEN column\_name2 = some\_value2

THEN new\_value2

Date \_\_\_\_\_  
Page No. \_\_\_\_\_

ELSE new\_value3;

END;

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;

## DELETE

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

1. Delete a Specific Record.

Syntax:-

DELETE FROM table\_name

WHERE Some\_column = Some\_value;

Ex:-

DELETE FROM emp

WHERE emp\_name = 'Sonam' AND emp\_id = 105.

NOTE:- WHERE is necessary otherwise all records will be deleted.

## 2. Delete All Records

Syntax:-

`DELETE FROM table-name;`

Or

`DELETE * FROM table-name;`

Ex:-

`DELETE FROM emp;`

Or `DELETE * FROM emp;`

Note:- We cannot undo this.

## COPY OLD TABLE TO NEW TABLE

Copy Old Table to New Table within Same Database.

Within 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;`

Within Different Database use the database where you want to copy the old table.

Syntax:-

`CREATE TABLE new-table LIKE old-db.old-table;`

`INSERT new-table SELECT * FROM old-db.old-table;`

SELECT AVG(Salary) AS av-salary FROM emp  
                        |  
                        | Date  
                        |  
                        | Alias (Optional)

Ex:-

CREATE TABLE teacher LIKE my-db.Student;  
INSERT Teacher SELECT \* FROM my-db.Student;

### MIN and MAX function

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

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

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

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

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

ROUND(column-name, decimal) - Function is used to round a numeric field to the number of decimals specified.

## DECIMAL

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

Where

T = Total digits. Range 1-65

D = Digits after decimal. Range 0-30 and must not be more than T

Ex:-

price DECIMAL (7,2)

16999.81

also {  
use as  
as }  
DEC  
NUMERIC  
FIXED

- column\_name DECIMAL (T)
- column\_name DECIMAL (T,0)
- Column\_name DECIMAL;

- Default T=10

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

Ex - 16999.81 = 3 byte + 1 byte = 4 bytes.

$\overline{1456\ 434566\ 999.81} = 4 + 2 + 1 = 7$

```
CREATE TABLE product
Id INT AUTO_INCREMENT PRIMARY KEY,
Pname VARCHAR(40),
Price DECIMAL(7,2) NOT NULL
);
```

```
INSERT INTO product (pname, price)
VALUES ("Mobile", 16999.25),
("computer", 25000.65);
```

- Zerofill

```
CREATE TABLE product
Id INT AUTO_INCREMENT PRIMARY KEY,
Pname VARCHAR(40),
Price DECIMAL(10,4) ZEROFILL
);
```

Ex:- 250.65

→ 000250.6500

### ROUND FUNCTION

Ex:- SELECT \* , ROUND(price, 1) FROM product;

If we want  
to count  
no. of times  
a person  
orders  
(Same person)

```
SELECT COUNT(CO
WHERE CustID = 101
```

## COUNT()

The COUNT(column\_name) function returns the number of values (NULL values will not be counted) of the specified column;

```
SELECT COUNT(column_name) FROM table-name;
```

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

```
SELECT COUNT(*) FROM table-name;
```

The COUNT(DISTINCT column\_name) Function returns the number of distinct values of the specified column

```
SELECT COUNT(DISTINCT column_name)
FROM table-name;
```

## UPPER and LOWER function

- UPPER(column\_name) or UCASE(column\_name)
  - Converts the value of a field to uppercase.
- LOWER(column\_name) or LCASE(column\_name)-
  - Converts the value of a field to lowercase.

SELECT emp-name, CONCAT(city, ',', pin) AS  
New-Address FROM emp;

PAGE NO.

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

Ex:- SELECT MID(city, 1, 3) AS shortcity FROM emp;

- LENGTH(column\_name) - the length of the value in a text field.

Ex:- SELECT city, LENGTH(city) FROM emp;

## CONCAT function

- CONCAT(column\_name1, column\_name2, ...)

- It joints two column.

Ex:- SELECT emp-name, CONCAT(city, pin) FROM emp;

- REVERSE(column\_name) - It reverse the order of letter in string.

Ex:- SELECT REVERSE(city) FROM emp;

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

Ex:- SELECT emp-name, salary, NOW() AS DateTime FROM emp;

- FORMAT(column\_name, format\_type) - function is used to format how a field is to be displayed.

## GROUP BY

Ex:-

- `SELECT emp_name, MIN(salary) FROM emp GROUP BY emp_name;`
- `SELECT custID, COUNT(*) FROM orders GROUP BY CustID;`

## HAVING Clause

- `SELECT emp_name, MIN(salary) FROM emp GROUP BY emp_name HAVING MIN(salary)>25000;`
- `SELECT CustID, COUNT(*) FROM orders GROUP BY CustID HAVING COUNT(*)>2;`

## WHY DO WE NEED MULTI TABLE

- To maintain atomicity
- To reduce redundancy

## Relationship

- One to one
- One to many
- Many to many

Customer

Order

Product

One to One

Person			
ID	Name	City	Mobile
1	Rahul	Petna	25698
2	Sonam	kolkata	45879
3	kunal	Mumbai	47898

## Aadhar

ID	Aadhaar No	State	Issue Date	RID
1	S987	Bihar	12-12-2000	1
2	S9812S	West Bengal	11-09-2003	2
3	S41263S	MH	07-01-2001	3

One to Many Relationship

## Customer

CustID	Name	City	Mobile
1	Rahul	Petna	25698
2	Sonam	kolkata	9618
3	kunal	Mumbai	923456

## order

foreignkey

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

## Many to Many

Author

AuthID	Name	City	Mobile
101	Rahul	Patna	256789
102	Sonam	Kolkata	378910
103	Kunal	Mumbai	67890

AuthID	BookID
101	151
101	152
103	152

Book

BookID	Name	Date	Price
151	DBMS	12-1-2000	300
152	Data	14-2-2003	200
153	C	16-3-2006	400

## Normalization

→ To reduce redundancy

→ To remove duplications (minimise)

- 1NF
- 2NF
- 3NF

User

1 NF

UserID	Name	Interest	Interest1	Interest2
1	Rahul	PC Games	Cricket	-
2	Sam	Tennis	Chat	-
3	Joe	Cricket	Chat	-
4	Joe	Singing	Chat	-

Composite key of primary key  
 hoti  $\Rightarrow$  CT8T 8M 97 key  
 and Atm1072 207 key

User ID	Name	ID	User ID	Interest
1	Rahul	101	1	PC Game
2	Sonam	102	1	Crickey
3	Sumit	103	1	Chat
4	Punit	104	2	Tennis
		105	2	Gaming
		106	2	Study

## 2NF (Normalization form)

Course Code	Date	CName	Seat	Remain	Room	Repa
C12S	12-01-2014	C Prog	12	5	101	15
DS14Y	12-01-2014	Data Str.	45	22	102	50
C12S	21-07-2014	C Prog	15	11	101	15
J678	08-11-2014	JavaProg	15	2	102	50

Course Code	Date	CName	Seat	Remain	Room	Repa
C12S	12-01-2014	C Prog	12	5	101	15
DS14Y	12-01-2014	Data Str.	45	22	102	50
C12S	21-07-2014	C Prog	15	11	101	15
J678	08-11-2014	JavaProg	15	2	102	50

Course code	Cname
C12S	C Prog
DS144	Data Str.
J678	Java Prog

- A table is in 2NF when it is already in 1NF
- And their non-key fields must depend on primary key

### 3NF

Course code	Date	Seat	Remain	Room	Recept
C12S	02-01-2014	12			
DS144	12-01-2014	45	22	102	50
C12S	21-07-2014	15	11	101	15
J678	08-11-2014	15	2	102	50

Course code	Cname
C12S	C Prog
DS144	Data Structure
J678	Java Programming

Course Code	Date	Seat	Remaining	Room
C12S	12-01-2014	12	9	101
DS144	12-01-2014	45	22	102
C12S	21-07-2014	15	11	101
J678	08-11-2014	5	2	102

Course Code	Chano	Room	Reape
C12S	Class	101	15
DS144	Data Str.	102	50
J678	JavaProg		

## FOREIGN KEY

- A FOREIGN KEY in one table points to a PRIMARY KEY in another table
- A foreign key can have a different name than the primary key it comes from.
- The primary key 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.

Emp-ID	Name	Address
1	Rani	Delhi
2	Jay	Mum
3	Veeru	Kol

Dep-ID	DName	Emp-ID
1	IT	1
2	HR	1
3	Admin	2

Create Table with foreign key in SQL

CREATE TABLE department

(

D\_id int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

D\_Name varchar(40),

E\_id int,

CONSTRAINT employee\_Eid\_fk

Emp-id	Name	Address
1	Rani	Delhi
2	Jay	Mum
3	Veeru	Kol

FOREIGN KEY (E\_id) REFERENCES employee (Empid)  
(emp-id)  
);

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.

Dep-ID	DName	Emp-ID
1	IT	1
2	HR	1
3	Admin	2

→ CREATE Table employee  
(

    cid INT NOT NULL AUTO\_INCREMENT  
    PRIMARY KEY,

    ename VARCHAR (40),

    address VARCHAR (40)

) ;

Now create table with foreign key

→ CREATE TABLE department

C

    did INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,  
    dname VARCHAR (40),  
    empid INT NOT NULL,  
    FOREIGN KEY (empid) REFERENCES employee(eid)  
);

Also; we write

→ CREATE TABLE department

C

    did INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,  
    dname VARCHAR (40),  
    empid INT NOT NULL,  
    FOREIGN KEY (empid) REF  
    CONSTRAINT employee\_eid\_FK  
    FOREIGN KEY (empid) REFERENCES employee(eid).  
);

## HOW To Find Constraint Name in SQL

Syntax:- Select \* from INFORMATION\_SCHEMA.  
TABLE\_CONSTRAINTS.

WHERE TABLE\_NAME = 'department';

## Drop foreign key from Table

ALTER TABLE department

DROP FOREIGN KEY employee\_eid-fk;

## Add foreign key constraint in a Table

ALTER TABLE department

ADD CONSTRAINT employee\_eid-fk

FOREIGN KEY (empid) REFERENCES employee(eid);

## Add column with foreign key

ALTER TABLE department

ADD COLUMN empid int not null,

ADD CONSTRAINT employee\_eid\_fk

FOREIGN KEY (empid) REFERENCES employee(eid);

## Unable to Delete Parent Table

we have to delete first <sup>child</sup> parent table  
then parent table.

## Unable to Delete Parent Table records

DELETE FROM department  
where did = 1;

DELETE FROM employee  
where eid = 1;

## ON Delete Clause in SQL

- ON DELETE CASCADE
- ON DELETE SET NULL
- ON DELETE NO ACTION
- ON DELETE RESTRICT.

### ON DELETE CASCADE

ON Delete Cascade is used when we want to delete any record from parent table ~~first~~ and the child table will delete automatically.

CREATE TABLE department

C

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  
);

ON DELETE SET NULL

CREATE TABLE department

C

did int not null auto-increment primary  
key,  
dname varchar (40),  
empid int,  
constraint employee\_eid\_fk  
foreign key (empid) references employee (eid)  
ON DELETE SET NULL  
);

ON DELETE NO ACTION or ON DELETE RESTRICT

CREATE TABLE department

C

did int not null auto-increment primary key,  
dname varchar(40),  
empid int,

constraint employee\_cid\_fk .

Foreign key (empid) references employee (cid)

ON DELETE RESTRICT

);

## ON UPDATE CLAUSE

- ON UPDATE CASCADE
- ON UPDATE SET NULL
- ON UPDATE NO ACTION
- ON UPDATE RESTRICT

## ON UPDATE CASCADE

CREATE TABLE department

C

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 CASCADE

);

UPDATE employee

SET eid = 10

WHERE eid = 1;

ON UPDATE SET NULL

CREATE TABLE department

C

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

);

## ON UPDATE NO ACTION OR ON UPDATE RESTRICT

CREATE TABLE department

(

did int not null auto-increment primary key,  
 dname varchar(40),  
 empid int,  
 constraint employee\_empid\_fk  
 foreign key (empid) references employee (eid)  
 ON UPDATE RESTRICT

);

## COMPOSITE KEY IN SQL

coursecode	Date	Cname	Seat	Remain	Room	Capacity
C12S	12-01-2014	ClProg	12	5	101	15
DS14Y	12-01-2014	DataStr	45	22	102	50
C12S	21-07-2014	CProg	15	11	101	15
J878	08-11-2014	JavaProg	15	2	102	50

→ 2nd col. Distinct

Primary key bnti  
 Composite key anafi

CREATE TABLE course (

Coursecode varchar(40),

Date date,

Name varchar(40),

Seat int,

Remain int,

Room int,

Recapa int,

PRIMARY KEY (coursecode, date)

) ;

## Shorthand Notation

Suppose we have two Tables

Student

Roll no	Name	Branch
1	Jay	CS
2	Veeru	E & C

Exam

Roll No.	Name	Score	Marks	Code
1	Abc	CS11	50	CS
1	Cde	CS12	60	CS
1	Efg	CS13	55	CS

Student.rollno

Exam.rollno

Student.name

Exam.name

## How To Join Tables in SQL

① CROSS JOIN / Cartesian Join / Cartesian Product / Cross Product.

The cross join returns every row from one table crossed with every row from the second.

Employee			Department	
Empid	Name	City	did	DapName
1	Rahul	Delhi	101	IT
2	Krish	Kol	102	HR
3	Jay	Mum	103	Admin

Select \* FROM Employee CROSS JOIN Department;

SELECT Name, DapName FROM Employee  
CROSS JOIN Department;

Also write,

SELECT Employee.Name, Department.DapName  
FROM Employee CROSS JOIN Department;

SELECT \* FROM Employee, Department;

SELECT \* from department, employee;

Employee			department	
Empid	Name	City	did	DepName
1	Rahul	Delhi	101	IT
2	Ikeish	Kol	102	HR
3	Jay	Mum	103	Admin.

SELECT \* from employee, department;

Employee			Department	
Empid	Name	city	did	DepName
1	Rahul	Delhi	101	IT
2	Ikeish	Kol	102	HR
3	Jay	Mum	103	Admin

## INNER JOIN

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

- EQUIJOIN
- NON - EQUIJOIN
- NATURAL JOIN

## EQUIJOIN

SELECT column\_name FROM table1  
 INNER JOIN table2  
 ON column\_name = column\_name;

Employee			Department		
Empid.	Name	city	did.	DepName	Empid
1	Rahul	Delhi	101	IT	3
2	Krish	Kol	102	HR	1
3	Jay	Mum	103	Admin	2

• SELECT emp.name, dep.depname FROM emp  
 INNER JOIN dep  
 ON emp.empid = dep.empid;

name	depname
Jay	IT
Rahul	HR
Krish	Admin

• SELECT emp.name, dep.depname FROM emp  
 INNER JOIN dep  
 ON dep.empid = emp.empid;

name	depname
Jay	IT
Rahul	HR
Krish	Admin

→ SELECT emp.name, dep.depname FROM dep  
 → INNER JOIN emp  
 ON dep.empid = emp.empid;

name	depname
Rahul	HR
Keish	Admin
Jay	IT

means Not equal to

• NON-EQUITY JOIN  $\neq$

SELECT column-name FROM table1

INNER JOIN table2

ON column-name  $\neq$  column-name;

→ SELECT emp.name, dep.depname FROM dep  
 INNER JOIN emp  
 ON emp.empid  $\neq$  dep.empid;

name	depname
Rahul	IT
Rahul	Admin
Keish	IT
Keish	HR
Jay	HR
Jay	Admin

These name  
should be same  
in case of Natural /  
**JOIN**  
PAGE NO.

### Employee

EmpId	Name	City
1	Rahul	Delhi
2	Krush	Kol
3	Jay	Punam

### Department

DeptId	DepName	EmpId
101	IT	3
102	HR	
103	Admin	1

## NATURAL JOIN

SELECT column\_name FROM table1  
NATURAL JOIN table2;

→ SELECT emp.name, dep.depname FROM dep  
NATURAL JOIN emp;

name	depname
Rahul	HR
Krush	Admin
Jay	IT

## Outer Joins

An outer join returns all rows from one of the tables, along with matching information from another table.

- LEFT OUTER JOIN | LEFT JOIN
- RIGHT OUTER JOIN | RIGHT JOIN
- FULL OUTER JOIN | FULL JOIN.

## 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 table.

Syntax:-

SELECT column-name FROM table1  
LEFT JOIN table2 — Right Table.

ON column-name = column\_name;

Emp

Empid	Name	City
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum
4	Sonam	Hyd
5	Mona	Patna

Dep

did	DepName	Embid
101	IT	3
102	HR	1
103	Admin	2
104	IT	1

SELECT emp.name, dep.name FROM emp  
 LEFT JOIN emp  
 ON emp.empid = dep.empid;

Name	Depname
Jay	IT
Rahul	HR
krish	Admin
Rahul	IT
Sonam	NULL
Mona	NULL

\* SELECT emp.name, dep.depname FROM dep  
 LEFT JOIN emp  
 ON emp.empid = dep.empid;

Name	Depname
Jay	IT
Rahul	HR
krish	Admin
Rahul	IT

## RIGHT OUTER JOIN | RIGHT JOIN

The RIGHT JOIN keyword 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 JOIN the table that comes before the join is the Right table, and the table that comes after the join is the Left table.

Syntax:-

```
SELECT column-name FROM table1 ← Right table  
RIGHT JOIN table2. ← Left table  
ON column-name = column-name;
```

Ex:-

```
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
Priya	Null

## FULL OUTER JOIN | FULL JOIN

The FULL OUTER JOIN returns all rows from the left table and from the right table.

The FULL OUTER JOIN combines the result of both LEFT and RIGHT joins.

Syntax:-

SELECT column\_name FROM table1  
FULL JOIN table2

ON Column\_name = Column\_name;

## SELF JOIN

Self Join is a table joined to itself.

Q:- why do we need Self Join?

It is used in hierarchy problems

Empid	Name	Managerid	
1	Rahul	3	
2	Jay	3	
3	Sonam	4	
4	Kunal	5	
5	Ram	6	
6	Rani	Null	
7	Neeru	6	

```

graph TD
    Roni[Roni] --> Ram[\"Ram\"]
    Roni --> Veenu[\"Veenu\"]
    Ram --> Kunal[Kunal]
    Kunal --> Sonam[Sonam]
    Sonam --> Rahul[Rahul]
    Sonam --> Jay[\"Jay\"]
  
```

Alias

```

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	Rani
Veeru	Rani

```

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

~~✓~~ SELECT e.name Name, m.name Manager

FROM empman e

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

3 SELECT e.name Name, m.name Manager

FROM empman e

INNER JOIN empman m

ON m.manid = e.empid;

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

we don't write  
DATE here because  
it duplicate data

cid	Name	city	SELECT s2.name Name, s2.city City FROM student s1 INNER JOIN student s2 ON s1.city = s2.city AND s1.name = 'Rahul'
1	Rahul	Delhi	
2	Jay	Kol	
3	Sonam	Delhi	
4	Kunal	Hyd	
5	Ram	Delhi	
6	Rani	Patna	
7	Veeru	Kol	

;

Name	city
Rahul	Delhi
Ram	Delhi
Sonam	Delhi

## Replacing Null

### 1) IFNULL()

Syntax:- IF NULL (exp1, exp2)

If exp1 is not NULL then IFNULL()

returns exp1 otherwise it returns exp2.

Ex:-

SELECT IFNULL ('GEEKY', 'SHOWS') AS Example

SELECT e.name Name, IFNULL(m.name, 'No Manager') AS Manager  
FROM empman e

LEFT JOIN empman m  
ON e.man\_id = m.emp\_id;

## ii) COALESCE()

Syntax:- COALESCE (exp1, exp2 ... expn);

Ex:-

SELECT COALESCE (NULL, 'Geekyshows') AS Example;

SELECT e.ename Name, COALESCE (m.name,  
 'No Manager') AS Manager  
 FROM empmane  
 LEFT JOIN empman m  
 ON e.man\_id = m.emp\_id;

## UNION

A UNION Combines the results of  
 two or more queries into one table.

Rules -

- The number of columns in each SELECT Statement must match.
- The data types in the columns should be same.
- The columns in each SELECT Statement 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;

SELECT name AS studentName FROM Student

UNION

SELECT name AS studentName employee ORDER  
BY studentName;

Student Name
Jay
Keish
Kunal
Rahul
Ram
Rani
Sonam
Veevee

## UNION ALL

UNION ALL returns every match whether it is duplicate or distinct ones.

Ex:- SELECT name FROM student

UNION ALL

SELECT name FROM employee ORDER BY name;

name
Jay
Jay
Krushna
Icorel
Rahul
Rahul
Rani
Sonam
Veeva

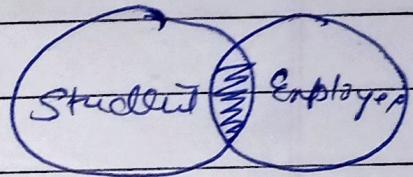
## INTERSECT & EXCEPT

- INTERSECT returns only those columns that are in the first query & also in the second query.

SELECT name FROM student

INTERSECT

SELECT name FROM Employee;



records

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

SELECT name FROM student

EXCEPT

SELECT name FROM Employee;

