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

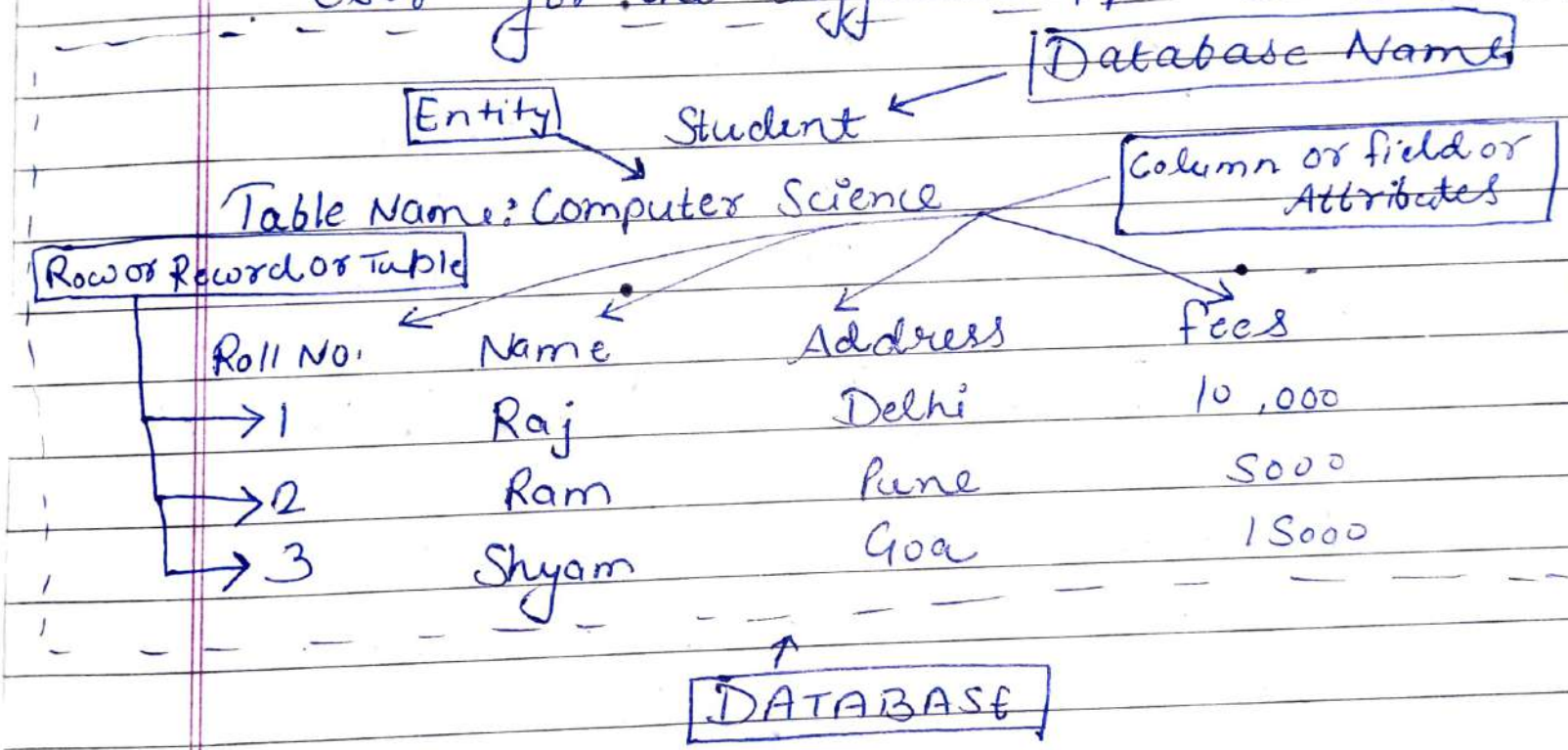


Table Name: MBA

Name	Roll	Address
Sam	01	Goa
Sum	02	Pune
Ron	03	Mumbai

Table Name: MCA

Name	Roll	Address
Amit	01	Bihar
Ram	02	UP
Shyam	03	MP

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.

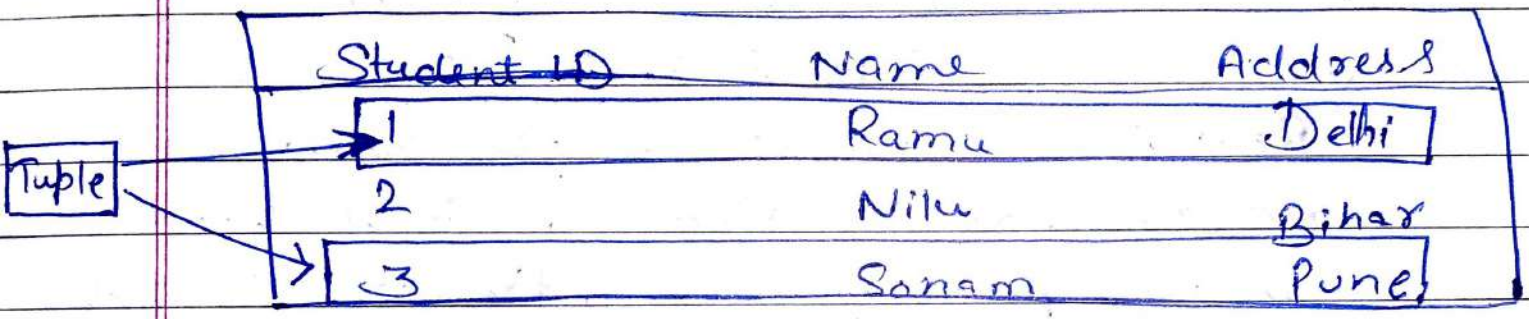
Attributes

Student_ID	Name	Address
1	Ram	Delhi
2	Sam	Goa
3	Tom	Bihar

DATE / /
PAGE NO.

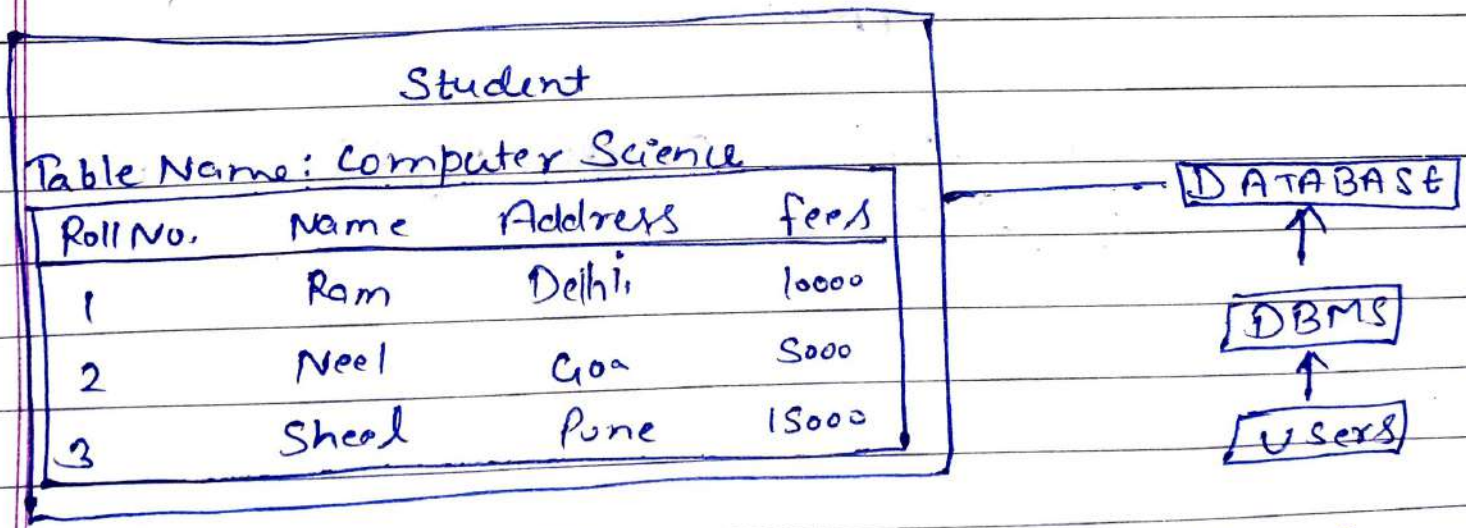
Row | Record | Tuple

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



Database Management System (DBMS)

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



Ex - dBase, Microsoft Access, Libre office Base, Foxpro.

DATE / /
PAGE NO.

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 DADABASE 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]
);
```

DATE / /
PAGE NO.

Ex :-

```
CREATE TABLE my_tab
(
name varchar (30) NOT NULL,
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/DBMS 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.

DATE / /
PAGE NO.

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);

DATE / /
PAGE NO.

- 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 (50);

- 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 yyyy-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;

DATETIME;

DATE / /
PAGE NO.

- 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, 'value2', 'value3', value4...);
```

Ex:-

```
INSERT INTO my_tab (Stu_id, name, address,
                    mobile_no).
VALUES (05, '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).

DATE / /
PAGE NO.

Need single quotes	No need
CHAR	INT or INTEGER
VARCHAR	DEC or DECIMAL
DATE	
DATETIME	
TIMESTAMP	
TIME	
BLOB	
TEXT	

INSERT INTO

without specifying column name.

Syntax :-

```
INSERT INTO table_name
VALUES (value1, 'value2', 'value3', value4);
```

Ex :-

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

my_tab :

Stu_id	Name	Address	Mobile no
05	Anu	Delhi	982112.

DATE / /
PAGE NO.

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 (column2, column1,
                        column4, column3).
VALUES ('value2', value1, value4, 'value3');
```

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 (Stu_id, name, address)
VALUES (05, 'Anu', 'Delhi');
```

my_tab.

Stu_id	Name	Address	Mobile_no.
05	Anu	Delhi	NULL

INSERT Multiple record at One time

Syntax:-

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

DATE / /
PAGE NO.

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_name
   FROM table-name;

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

1)

Name	Roll	Mobile
Rahul	01	8753
John	02	6534

↓

2)

Name	Mobile
Rahul	8753
John	6534

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'

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

- Use two time single quotes
  Ex:- 'k.k''s company'

DATE / /
PAGE NO.

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	Jai
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

DATE / /
PAGE NO.

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

2. 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

DATE / /
PAGE NO.

Ex:-

```
SELECT name
FROM new_tab
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.

DATE / /
PAGE NO.

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 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;
```

DATE / /
PAGE NO.

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.

Stu_id	Name
5	Anu
6	Sonu
7	Anu

Syntax:-
```
SELECT * FROM table_name
WHERE column_name = 'value'
AND column = 'value';
```

Stu_id	Name
5	Anu

Ex:-
```
SELECT * FROM new_tab
WHERE name = 'Anu'
AND Stu_id = 5;
```

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 = '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_name1
                    = 'value' );
```

Ex:-

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

DATE / /
PAGE NO.

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

DATE / /
PAGE NO.

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

DATE / /
PAGE NO.

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

DATE / /
PAGE NO.

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)

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

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

3. Between Date

Syntax:-

```
SELECT * FROM table_name
WHERE column_name
 BETWEEN 'yyyy/mm/dd' AND
  'yyyy/mm/dd';
```

Ex:-

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

(including both dates)

DATE / /
PAGE NO.

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 value 2;
```

Ex:-

```
SELECT * FROM Product.
WHERE stu_id
NOT BETWEEN 3 AND 7; (including 3 and 7)
```

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

Stu_id	Name
5	Anu
6	Sonu
7	Anu
8	Jai

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.

PAGE NO.

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 (salary BETWEEN 25000 AND 50000)
AND dept IN ('IT', 'HR');
```

DATE / /
PAGE NO.

BETWEEN with NOT IN

Syntax:-
SELECT * FROM table_name
WHERE (column_name BETWEEN value 1 AND value 2).
AND NOT column_name IN (value 1, value 2);

Ex:-
SELECT * FROM emp
WHERE (Salary 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 '%m';

DATE / /
PAGE NO.

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. _ — 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 table_name
WHERE column_name NOT LIKE 'pattern';

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

DATE / /
PAGE NO.

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;
```

DATE / /
PAGE NO.

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 NULL values.

Unique Key

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

```
CREATE TABLE Student
(
Name varchar(30),
Roll integer(5),
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(5) NOTNULL PRIMARY KEY,
Mobile no integer(10)
);
```

DATE / /
PAGE NO.

```
CREATE TABLE Student
(
Name varchar (30),
Roll integer (5) NOT NULL,
Mobile_no integer (10),
PRIMARY KEY (Roll)
);
```

Name	Roll	Mobile No.

AUTO INCREAMENT

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

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

Syntax:-

```
CREATE TABLE table_name
(
column_name int NOT NULL AUTO_INCREAMENT,
Column_name1 varchar (50) NOT NULL,
Column_name2 varchar (50),
PRIMARY KEY (column_name)
);
```

DATE / /
PAGE NO.

Ex:-

=> CREATE TABLE emp;
(
Emp_id int NOT NULL AUTO_INCREAMENT,
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');

DATE / /
PAGE NO.

Starting AUTO INCREMENT with a particular Number in SQL

Ex:-

```
ALTER TABLE k_tab 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;
```

DATE / /
PAGE NO.

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 "New Cost"
  FROM items_tab;
```

SELECT DISTINCT

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

Syntax:- SELECT DISTINCT column_name
FROM table_name;

Ex:- SELECT DISTINCT name FROM student;

DATE / /
PAGE NO.

DATA DEFINITION LANGUAGE

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.

ALTER TABLE

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

Name	Roll

Syntax :-

ALTER TABLE table_name
ADD COLUMN column_name datatype (Size);

Ex:-

ALTER TABLE my_tab
ADD COLUMN Stu_id integer (5);

Name	Roll	Stu_id

DATE / /
PAGE NO.

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;
```

DATE / /
PAGE NO.

(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 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);

DATE / /
PAGE NO.

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);
```

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);
```

Date
Page No.

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

Chang 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
	1234567890

DATE / /
PAGE NO.

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);
```

DATE / /
PAGE NO.

DROP TABLE

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

Syntax:-

```
DROP TABLE table_name;
```

Ex:-

```
DROP TABLE my_tab;
```

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.

DATE / /
PAGE NO.

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;

Date / /
Page No.

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 DATABASE

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;
```

DATE / /
PAGE NO.

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 = 'Arjun', 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
```

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 delete.

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_salaray FROM emp;

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

- DEC • column_name DECIMAL (T)
- NUMERIC • column_name DECIMAL (T, D)
- FIXED • 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 = 3byte + 1byte = 4 bytes.

1456 4345 6699 9.81 = 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(CU
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;

DATE
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 empname, 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

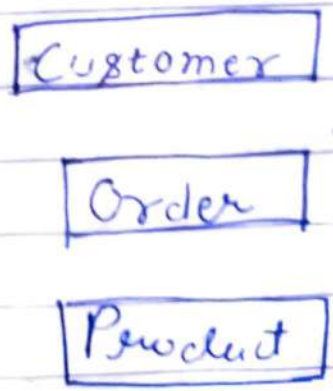
- SELECT emp_name, MIN(salary) FROM emp GROUP BY emp_name HAVING MIN(salary)>2500;
- SELECT CustID, COUNT(*) FROM orders GROUP BY CustID HAVING COUNT(*)>2;

WHY DO WE NEED MULTI TABLE

- To maintain atomicity
- To reduce redundancy

Relationship

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



One to One

Person

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

Aadhar

ID	Aadhaar No	State	Issue Date	RID
1	5987	Bihar	12-12-2000	1
2	59812S	West Bengal	11-09-2003	2
3	541263S	MH	07-01-2001	3

One to Many Relationship

Customer

CustID	Name	City	Mobile
1	Rahul	Patna	25698
2	Sonam	kolkata	9678
3	kunal	Mumbai	R345C

Order

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

Foreign key

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	Data	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 {1NF}

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

Composite key वो key होती है जहाँ हम दो key को मिलाकर एक key बनाते है।

UserID	Name
1	Rahul
2	Sonam
3	Sumit
4	Rohit

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

2NF (Normalization form)

Course Code	Date	CName	Seat	Remain	Room	Rcapa
C12S	12-01-2014	CProg	12	5	101	15
DS144	12-01-2014	DataStr.	45	22	102	50
C12S	21-07-2014	CProg	15	11	101	15
J678	08-11-2014	JavaProg	15	2	102	50

↓

Course Code	Date	~~CName~~	Seat	Remain	Room	Rcapa
C12S	12-01-2014		12	5	101	15
DS144	12-01-2014		45	22	102	50
C12S	21-07-2014		15	11	101	15
J678	08-11-2014		15	2	102	50

CourseCode	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	Reapa
C12S	12-01-2014	12	5	101	15
DS144	21-01-2014	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

DATE / /
PAGE NO.

Course code	Data	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	5	2	102

Courcode	Cname
C125	C Prog
DS144	Data Str.
J678	Java Prog

Room	Rcap
101	15
102	50

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.

DATE / /
PAGE NO.

- 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
FOREIGN KEY (E_id) REFERCES employee (Empid)
                                    (emp_id)
);
```

Emp_id	Name	Addres
1	Rani	Delhi
2	Jay	Mum
3	Veeru	kol

Date / /
Page No.

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
(
eid INT NOT NULL AUTO_INCREMENT
  PRIMARY KEY,
ename VARCHAR (40),
address VARCHAR (40)
);
```

DATE / /
PAGE NO.

Now create table with foreign key

→
```
CREATE TABLE department
(
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
(
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).
);
```

DATE / /
PAGE NO.

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);
```

DATE / /
PAGE NO.

Unable to Delete Parent Table

we have to delete first child 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 record from child table will delete automatically.

DATE / /
PAGE NO.

```
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
);
```

ON DELETE SET NULL

```
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 DELETE SET NULL
);
```

ON DELETE NO ACTION or ON DELETE RESTRICT

```
CREATE TABLE department
(
did int not null auto_increament primary key,
dname varchar(40),
empid int,
constraint employee_eid_fk.
Foreign key (empid) references employee (eid)
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
(
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
(
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_eid-fk.
foreign key (empid) references employee (eid)
ON UPDATE RESTRICT
);
```

COMPOSITE KEY IN SQL

coursecode	Date	Cname	Seat	Remain	Room	Rcapa
C125	12-01-2017	CProg	12	5	101	15
DS144	12-01-2017	Data Str	45	22	102	50
C125	21-07-2017	CProg	15	11	101	15
J678	08-11-2017	JavaProg	15	2	102	50

यह दोनो मिलकर
Primary key bnti है
तो इसे Composite key कहते हैं।

DATE / /
PAGE NO.

```
CREATE TABLE course (
Coursecode varchar(40),
Date date,
Cname 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	Scode	Marks	Bcode
1	Abc	CS11	50	CS
1	cde	CS12	60	CS
1	Efg	CS13	55	CS

Student.rollno
Student.name

Exam.rollno
Exam.name

DATE / /
PAGE NO.

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

Empid	Name	city
1	Rahul	Delhi
2	krish	kol
3	Jay	Mum

Department

did	DepName
101	IT
102	HR
103	Admin

```
Select * FROM Employee CROSS JOIN
    Department;
SELECT Name, Depname FROM Employee
    CROSS JOIN Department;
```

Also write,

```
SELECT Employee.Name, Department.Depname
  FROM Employee CROSS JOIN Department;

SELECT * FROM Employee, Department;
```

DATE / /
PAGE NO.

SELECT * from department, employee;

Employee

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

department

did	DepName
101	IT
102	HR
103	Admin

SELECT * from employee, department;

Employee

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

Department

did	DepName
101	IT
102	HR
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

DATE / /
PAGE NO.

- ## EQUIJOIN

```
SELECT column_name FROM table1
INNER JOIN table2
ON column_name = column_name;
```

Employee

Empid.	Name	City
1	Rahul	Delhi
2	krish	Kol
3	Jay	Mum

Department

did.	DepName	Empid
101	IT	3
102	HR	1
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

DATE / /
PAGE NO.

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

name	depname
Rahul	HR
krish	Admin
Jay	IT

- NON-EQUIJOIN <> means Not equal to

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

→ 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

These name should be same in case of Natural JOIN

• NATURAL JOIN

Employee

Empid	Name	City
1	Rahul	Delhi
2	krish	kol
3	Jay	Pune

Department

did	DepName	Empid
101	IT	3
102	HR	1
103	Admin	2

⇒ SELECT column_name FROM table1
NATURAL JOIN table2;

⇒ 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.

- LEFT OUTER JOIN / LEFT JOIN
- RIGHT OUTER JOIN / RIGHT JOIN
- FULL OUTER JOIN / FULL JOIN.

DATE / /
PAGE NO.

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 table
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	Empid
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

DATE / /
PAGE NO.

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
Soram	Null
Mona	Null

DATE / /
PAGE NO.

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;
```

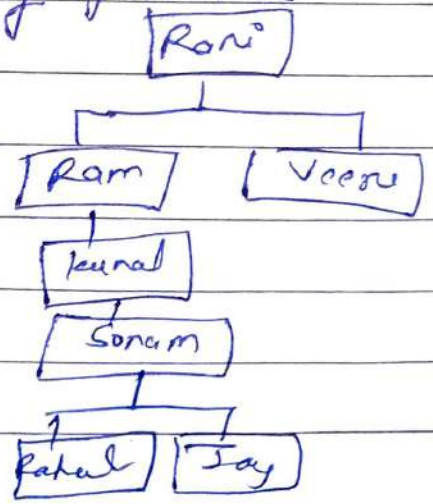
SELF JOIN

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	Soram	4
4	kunal	5
5	Ram	6
6	Rani	Null
7	Veeru	6



DATE / /
PAGE NO.

Aliases

```
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;
```

```
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 S1 here because it duplicate data

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

```
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
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.manid = m.empid;
```

ii) COALESCE()

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

Ex:-

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

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

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 student Name employee ORDER
                                BY StudentName;
  ```

StudentName
Jay
krish
kunal
Rahul
Ram
Rani
Sonam
Veeru
  ```

Date / /
Page No.

UNION ALL

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

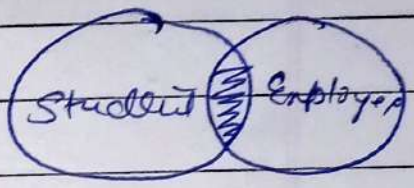
Ex:- SELECT name FROM student
UNION ALL
SELECT name FROM employee ORDER BY name;

name
Jay
Jay
krish
Kunal
Rahul
Rahul
Rani
Soram
Veeru

DATE / /
PAGE NO.

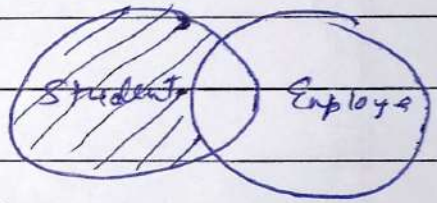
INTERSECT & EXCEPT

- 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;
```

- 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;
```