**DATA DEFINITION LANGUAGE (DDL) COMMANDS:**

Data Definition Language (DDL) statements are used to define the database structure or schema. Data Definition Language describes how the data should exist in the database.

Data Definition Language (DDL) is used in different statements:

* **CREATE** - to create databases as well as objects in the database such as a table, view, index.
* **ALTER** - alters the structure of the database
* **DROP** - delete database as well as objects from the database
* **TRUNCATE** - remove all records from a table, including all spaces allocated for the records are removed
* **COMMENT** - add comments to the data dictionary
* **RENAME** – to change the name of the table.

**Create Database:**

CREATE DATABASE **D\_NAME;**

**Drop Database:**

DROP DATABASE **D\_NAME;**

**Use Database:**

USE **D\_NAME;**

**Create Table:**

CREATE TABLE **T\_NAME** (

ID **INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

NAME **VARCHAR (20),**

EMAIL **VARCHAR (255) UNIQUE,**

PHONE\_NO **BIGINT,**

CV **MEDIUMBLOB**);

**Delete Table:**

DROP TABLE **T\_NAME;**

**Delete Data of Table:**

TRUNCATE TABLE **T\_NAME;**

**Alter Table:**

ALTER TABLE **T\_NAME** AUTO\_INCREMENT = 101;

ALTER TABLE **T\_NAME** ADD FOREIGN KEY (ID) REFERENCES **T\_NAME** (ID);

ALTER TABLE **T\_NAME** ADD COLUMN **C\_NAME** VARCHAR (25);

ALTER TABLE **T\_NAME** MODIFY **C\_NAME** VARCHAR (26);

ALTER TABLE **T\_NAME** CHANGE **C\_NAME NEW\_CNAME** VARCHAR (20);

ALTER TABLE **T\_NAME** DROP **C\_NAME**;

ALTER TABLE **T\_NAME** MODIFY **C\_NAME** **C\_DATATYPE** AFTER **C\_NAME**; (change column position);

ALTER TABLE **T\_NAME** DROP FOREIGN KEY **FK\_NAME**;

RENAME TABLE **T\_NAME** TO **NEW\_TNAME**;

**DATA MANIPULATION LANGUAGE (DML) COMMANDS:**

Data Manipulation Language (DML) statements are used for managing data within schema objects DML deals with data manipulation, and therefore includes most common SQL statements such as SELECT, INSERT, etc. DML allows adding / modifying / deleting data itself.

DML is used to manipulate the existing data in the database objects (insert, select, update, delete).

**DML Commands**

1.INSERT

2.SELECT

3.UPDATE

4.DELETE

**SELECT:**

SELECT \* FROM **T\_NAME**; (Fetch all the records from the orders.)

SELECT **C\_NAME**, **C\_NAME** FROM **T\_NAME**; (Fetch multiple column values.)

SELECT **C\_NAME** AS **NEW\_CNAME**, **C\_NAME** AS **NEW\_CNAME** FROM **T\_NAME**; (AS stands for Alias (to rename column name))

* **WHERE CLAUSE:** (to fetch conditional based data)

SELECT **\*** FROM **T\_NAME** WHERE (C\_NAME = “RECORD”, C\_NAME < 20, C\_NAME < = 20, C\_NAME != “RECORD”, C\_NAME < > “RECORD”) **note:** using one condition at a time.

* FETCH DATA WITH **AND, OR, IN, BETWEEN, LIKE**  OPERATOR:

SELECT **\*** FROM **T\_NAME** WHERE

C\_NAME >= “RECORD” **AND** C\_NAME < = “RECORD”; (**Note: we can use multiple and different column to fetch record.)**

C\_NAME >= “RECORD” **OR** C\_NAME < = “RECORD”; (**Note: we can use multiple and different column to fetch record.)**

(C\_NAME >= “RECORD” **OR** C\_NAME < = “RECORD”) **AND** C\_NAME < = “RECORD” (**Note: we can use multiple and different column to fetch record.)**

C\_NAME **IN** (“RECORD”, “RECORD”) **(Note: We use IN operator to fetch multiple records in one column)**

C\_NAME **NOT IN** (“RECORD”, “RECORD”)

C\_NAME **BETWEEN** “RECORD” **AND** “RECORD”

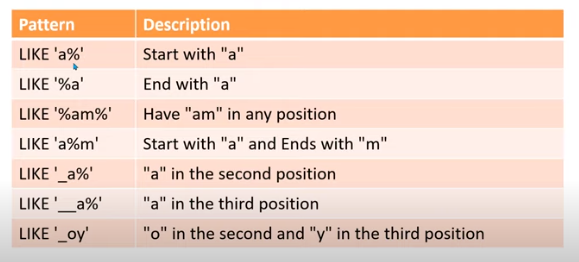
C\_NAME **NOT BETWEEN** “RECORD” **AND** “RECORD”

**LIKE OPERATOR:**

**Note:** Wildcard Characters:

**% Percentage Sign:** Represents zero, one, or multiple characters.

**\_ Underscore:** Represents a single character.



C\_NAME **LIKE** “PATTERN”;

C\_NAME **LIKE** “PATTERN” **OR** C\_NAME **LIKE** “PATTERN”;

C\_NAME **NOT LIKE** “PATTERN”;

BINARY C\_NAME **LIKE** “PATTERN”; (Note: BINARY used for case-sensitive)

**INSERT:**

INSERT INTO **T\_NAME** (F\_NAME, EMAIL, PHONE\_NO) VALUES

("IKHLAAS", "IKHLAS@GMAIL.COM”, 0300422345),

("ALI", "ALI@GMAIL.COM”, 0300482345),

("ZAFAR", "ZAFAR@GMAIL.COM”, 0300426345);

**UPDATE:**

UPDATE **T\_NAME** SET **C\_NAME** = **C\_DATA** WHERE **C\_ID** = 1;

(Modify record in a table)

UPDATE **T\_NAME** SET **C\_NAME** = **NULL** WHERE **C\_ID** = 2; (Set null record in a table)

**DELETE:**

DELETE FROM **T\_NAME** WHERE **C\_ID** = 3; (To delete record)

DELETE FROM **T\_NAME** WHERE **C\_NAME** = **C\_RECORD**; (To delete particular record.)

**LIST OF CONSTRAINTS:**

**SQL constraints are**used to specify rules for the data in a table

* NOT NULL
* UNIQUE
* DEFAULT
* CHECK
* FOREIGN KEY
* PRIMARY KEY

**AGGREGATE FUNCTIONS:**

* COUNT
* MAX
* MIN
* SUM
* AVG

SELECT COUNT(**C\_NAME**) FROM T\_NAME; **Note:** you can use \* instead of c\_name, we can use **Distinct** to find unique record.

SELECT MAX(**C\_NAME**) FROM T\_NAME;

SELECT MIN(**C\_NAME**) FROM T\_NAME;

SELECT SUM(**C\_NAME**) AS **NEWT\_NAME** FROM T\_NAME;

SELECT AVG(**C\_NAME**) FROM T\_NAME;

**ORDER BY OR DISTINCT CLAUSE:**

SELECT \* FROM **T\_NAME** ORDER BY **C\_NAME**;

SELECT \* FROM **T\_NAME** ORDER BY **C\_NAME DESC**;

SELECT \* FROM **T\_NAME WHERE C\_NAME = “RECORD”** ORDER BY **C\_NAME DESC**;

SELECT \* FROM **T\_NAME** ORDER BY **C\_NAME, C\_NAME**;

SELECT DISTINCT **C\_NAME** FROM **T\_NAME;**

SELECT DISTINCT **C\_NAME** FROM **T\_NAME** ORDER BY **C\_NAME;**

**LIMIT OR OFFSET CLAUSE:**

**Note:** OFFSET is an optional parameter to fetch data. Give Offset before limit.

SELECT \* FROM **T\_NAME** LIMIT 2;

SELECT \* FROM **T\_NAME** WHERE **C\_NAME** = “RECORD” LIMIT 2;

SELECT \* FROM **T\_NAME** WHERE **C\_NAME** = “RECORD” ORDER BY C\_NAME LIMIT 2;

SELECT \* FROM **T\_NAME** WHERE **C\_NAME** = “RECORD” ORDER BY C\_NAME LIMIT 2, 3; (Offset)

**COMMIT & ROLLBACK WORKS FOR:**

* Insert
* Update
* Delete

**Note:**

We use commit to save our previously changes.

We use rollback to revert our update, insert, and delete query.

COMMIT;

ROLLBACK;

**JOINS:**

* Inner join
* Left join
* Right join
* Cross join

**INNER JOIN:**

The Inner join selects records that have matching values in both tables.

The left join returns all records from the left table (table1), and the matched record from the right table (table2).

The right join returns all records from the right table (table2), and the matched record from the left table (table1).

Table 2

Table 1

SELECT **\*** FROM **T\_NAME1** INNER JOIN **T\_NAME2** ON **T\_NAME1**. C\_NAME(FK) = **T\_NAME2**.C\_NAME(PK) WHERE C\_NAME = “RECORD”

SELECT **\*** FROM **T\_NAME1** RIGHT JOIN **T\_NAME2** ON **T\_NAME1**. C\_NAME(FK) = **T\_NAME2**.C\_NAME(PK)

SELECT **\*** FROM **T\_NAME1** LEFT JOIN **T\_NAME2** ON **T\_NAME1**. C\_NAME(FK) = **T\_NAME2**.C\_NAME(PK)

SELECT **\*** FROM **T\_NAME1** CROSS JOIN **T\_NAME2**

**SET OPERATORS:**

* UNION/UNION ALL
* INTERSECT
* EXCEPT

**Note: Rules for Union/Union All**

* Each select statement within union must have the same numbers of columns.
* The columns must have also similar data types.
* The columns in each select statement must also be in the same order.

SELECT \* FROM T\_NAME UNION SELECT \* FROM T\_NAME;

SELECT \* FROM T\_NAME UNION ALL SELECT \* FROM T\_NAME;

SELECT \* FROM T\_NAME WHERE C\_NAME = “RECORD” UNION SELECT \* FROM T\_NAME WHERE C\_NAME = “RECORD”;

SELECT \* FROM T\_NAME WHERE C\_NAME = “RECORD” UNION ALL SELECT \* FROM T\_NAME WHERE C\_NAME = “RECORD”;