



# BIG DATA TOOLS FOR MANAGERS (N2MBA07)

## Unit -2 : Data Querying and Retrieval using SQL

---

# SQL- Structured Query Language

- SQL commands are instructions for the database. It is used to communicate with the database.
- SQL can perform various tasks like create a database/table, add data to tables, drop the table, modify the table, set permission for users.
- SQL commands are case insensitive, but table and column names are case sensitive.

# SQL- Structured Query Language

SQL statements are divided into two major categories.

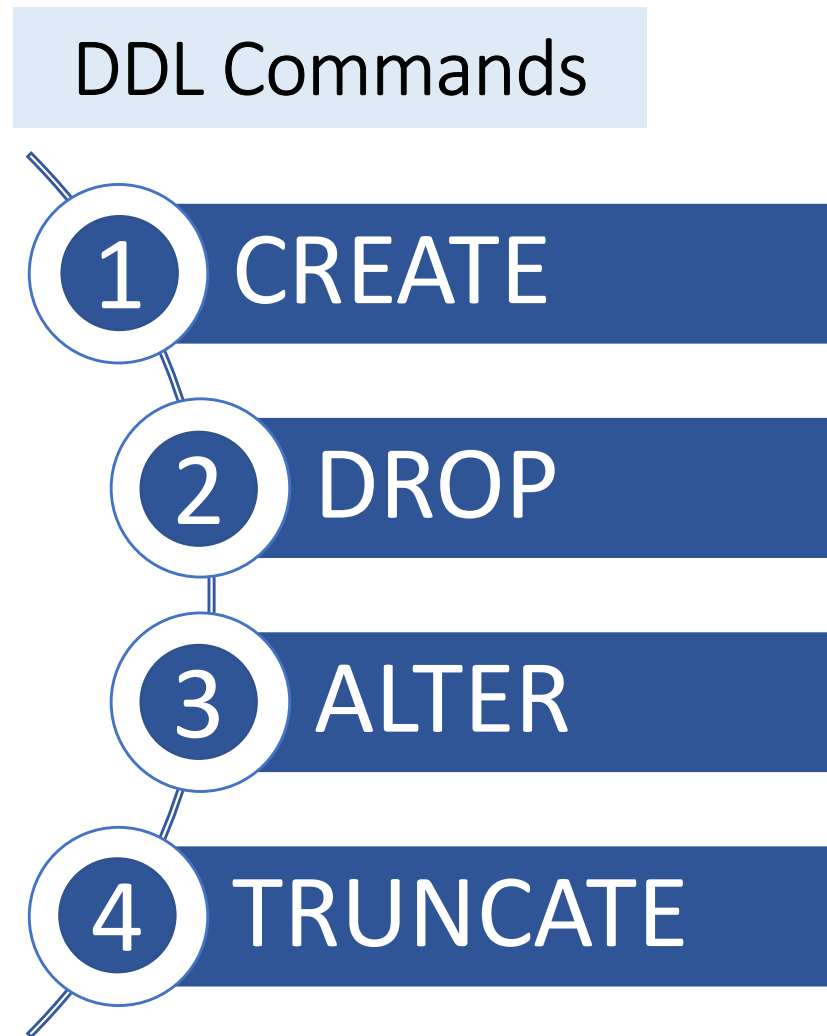
1. Data definition language (DDL)
2. Data manipulation language (DML)

# Data definition language (DDL)

- ✓ DDL statements are used to build and modify the structure of tables in the database.
- ✓ When you execute a DDL statement, it takes effect immediately.
- ✓ It is also known as data descriptive language.

# Data definition language (DDL)

- ✓ DDL statements are used to build and modify the structure of tables in the database.
- ✓ When you execute a DDL statement, it takes effect immediately.
- ✓ It is also known as data descriptive language.



# Data manipulation language (DML)

- ✓ Data Manipulation Language commands allow you to manage the data stored in the database.
- ✓ DML Command is used by the database user/ application programs to retrieve, add, remove or update the information in the database.

# Data manipulation language (DML)

- ✓ Data Manipulation Language commands allow you to manage the data stored in the database.
- ✓ DML Command is used by the database user/ application programs to retrieve, add, remove or update the information in the database.

## DML Commands

INSERT

SELECT

UPDATE

DELETE

# Data definition language (DDL)

## CREATE

Create command is a DDL command used to create a table or a database some objects like tables, Views, indexes, functions.

Syntax :

```
CREATE TABLE table_name (  
    col-name-1 data-type ,  
    col-name-2 data-type ,  
    col-name-3 data-type ,  
    ....  
);
```



# Data definition language (DDL)

## CREATE

```
CREATE TABLE table_name (  
    col-name-1    data-type ,  
    col-name-2    data-type ,  
    col-name-3    data-type ,  
    ....  
);
```

```
CREATE TABLE emp_details (  
    EMP_ID  TEXT,  
    EMP_NAME TEXT,  
    EMP_POST TEXT  
);
```

# Data definition language (DDL)

## DATA TYPE

The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

### Most common data types:

- String/Text
- Numeric
- Datetime

```
CREATE TABLE table_name (  
    col-name-1    data-type ,  
    col-name-2    data-type ,  
    col-name-3    data-type ,  
    ....  
);
```

# Data definition language (DDL)

DATA TYPE

STRING

## **CHAR(size)**

A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored

Example:

```
CREATE TABLE student(  
    usn CHAR(10)  
);
```

# Data definition language (DDL)

## DATA TYPE

## STRING TYPES

### CHAR(size)

A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length.

Example:

```
CREATE TABLE student(  
    usn CHAR(10)  
);
```

|   |   |   |   |   |   |   |   |         |         |
|---|---|---|---|---|---|---|---|---------|---------|
| 1 | S | I | 2 | 3 | M | B | A | 0       | 1       |
| 1 | S | I | 2 | 3 | M | B | A | 2       | (space) |
| 1 | S | I | 2 | 3 | B | A | 3 | (space) | (space) |

# Data definition language (DDL)

## DATA TYPE

## STRING

### **VARCHAR(size)**

A variable-length string between 1 and 255

Example:

```
CREATE TABLE student(  
    usn VARCHAR(10)  
);
```

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 1 | S | I | 2 | 3 | M | B | A | 0 | 1 |
| 1 | S | I | 2 | 3 | M | B | A | 2 |   |
| 1 | S | I | 2 | 3 | B | A | 3 |   |   |
| 1 | S | I | 2 | 3 | M | B | A | 0 | 4 |

# Data definition language (DDL)

## DATA TYPE

## STRING

---

### **TINYTEXT**

- TEXT column with a maximum length of 255 characters

### **MEDIUMTEXT**

- TEXT column with a maximum length of 16777215 characters

### **LONGTEXT**

- TEXT column with a maximum length of 4294967295 or 4 GB of characters

# Data definition language (DDL)

DATA TYPE

STRING

Example:

```
CREATE TABLE student (  
    USN          CHAR(10),  
    NAME         VARCHAR(50),  
    ADDRESS      TINYTEXT,  
    CITY         VARCHAR(20),  
    STATE        VARCHAR(20),  
    PINCODE      CHAR(5)  
);
```

# Data definition language (DDL)

## DATA TYPE

## NUMERIC

| Data Type | Signed range(Number with Sign)                         | Unsigned range            |
|-----------|--|---------------------------|
| TINYINT   | -128 to 127  | 0 to 255                  |
| SMALLINT  | -32768 to 32767  | 0 to 65535                |
| MEDIUMINT | -8388608 to 8388607                                    | 0 to 16777215             |
| INT       | -2147483648 to 2147483647                              | 0 to 4294967295           |
| BIGINT    | -9223372036854775808 to 9223372036854775807            | 0 to 18446744073709551615 |
| FLOAT     | Decimal precision can go to 24 places for a float type |                           |
| DOUBLE    | Decimal precision can go to 53 places for a double     |                           |



# Data definition language (DDL)

DATA TYPE

DATETIME

| Data Type | Maximum Size  |
|-----------|---|
| DATE      | Values range from '1000-01-01' to '9999-12-31'.                           |
| TIME      | Values range from '-838:59:59' to '838:59:59'.                            |
| DATETIME  | Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.         |
| TIMESTAMP | Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC. |

# Data definition language (DDL)

## DATA TYPE

---

### String Type

CHAR

VARCHAR

TINYTEXT

MEDIUMTEXT

LONGTEXT

### Numeric Type

TINYINT

SMALLINT

MEDIUMINT

INT

BIGINT

FLOAT

DOUBLE

### Date Type

DATE

TIME

DATETIME

TIMESTAMP

# Data definition language (DDL)

## DATA TYPE

---

Example : String, Int, and DateTime

```
CREATE TABLE student_details (  
    USN          CHAR(10),  
    NAME         VARCHAR(50),  
    DOB          DATE,  
    AGE          TINYINT,  
    CGPA         FLOAT,  
    ADDRESS      MEDIUMTEXT  
);
```

# Data definition language (DDL)

## CREATE with Constraints

- SQL constraints are used to specify rules for the data in a table.
- Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table.

Syntax :

```
CREATE TABLE table_name (  
    col-name-1 data-type constraint ,  
    col-name-2 data-type constraint ,  
    col-name-3 data-type constraint ,  
    ....  
);
```

# Data definition language (DDL)

## CREATE with Constraints

---

Some basic SQL constraints:

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. DEFAULT

# Data definition language (DDL)

## CREATE with Constraints

### NOT NULL Constraint

- By default, a column can hold NULL values
- The NOT NULL constraint enforces a column to NOT accept NULL values.

### Syntax :

```
CREATE TABLE student_details (  
    USN          CHAR(10)    NOT NULL,  
    NAME         VARCHAR(50) NOT NULL,  
    DOB          DATE,  
    AGE          TINYINT,  
    CGPA         FLOAT,  
    ADDRESS      MEDIUMTEXT  
);
```

# Data definition language (DDL)

## CREATE with Constraints

### UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are different.

Syntax :

```
CREATE TABLE student_details (  
    USN          CHAR(10)    UNIQUE,  
    NAME         VARCHAR(50),  
    DOB          DATE,  
    AGE          TINYINT,  
    CGPA         FLOAT,  
    ADDRESS      MEDIUMTEXT  
);
```

# Data definition language (DDL)

## CREATE with Constraints

---

### PRIMARY KEY Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).



# Data definition language (DDL)

## CREATE with Constraints

---

### PRIMARY KEY Constraint

Syntax :

```
CREATE TABLE student_details (  
    USN          CHAR(10) PRIMARY KEY,  
    NAME         VARCHAR(50),  
    DOB          DATE,  
    AGE          TINYINT,  
    CGPA         FLOAT,  
    ADDRESS      MEDIUMTEXT  
);
```

# Data definition language (DDL)

## CREATE with Constraints

---

### **DEFAULT Constraint**

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

# Data definition language (DDL)

## CREATE with Constraints

---

### DEFAULT Constraint

```
CREATE TABLE student_details (  
    USN          CHAR(10) UNIQUE,  
    NAME         VARCHAR(50) NOT NULL,  
    DOB          DATE,  
    AGE          TINYINT,  
    CGPA         FLOAT,  
    ADDRESS      MEDIUMTEXT DEFAULT 'TUMKUR,INDIA'  
);
```

# Data definition language (DDL)

## DROP

---

- The DROP TABLE statement is used to drop an existing table in a database.
- It permanently removes objects from the database or MySQL server.

Syntax:

**DROP TABLE** table\_name;

Example:

**DROP TABLE** students;

# Data definition language (DDL)

## TRUNCATE

---

- TRUNCATE TABLE statement is used to delete the data inside a table, but not the table.
- It permanently removes records/observation from the table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE TABLE students;
```

# Data definition language (DDL)

## ALTER

---

Alter command is used to alter the structure of the tables in the database.

For Example :

- ✓ To add a column to existing table.
- ✓ To rename any existing Column.
- ✓ Alter is also used to drop a column.
- ✓ To Change datatype of any Column or to modify its size

# Data definition language (DDL)

## Example:

---

Create employee table with following columns :  
emp\_id, ename, post, country

```
CREATE TABLE employee (  
    emp_id  text,  
    ename   text,  
    post    text,  
    country text  
);
```

# Data manipulation language (DML)

- ✓ Data Manipulation Language commands allow you to manage the data stored in the database.
- ✓ DML Command is used by the database user/ application programs to retrieve, add, remove or update the information in the database.

## DML Commands

INSERT

SELECT

UPDATE

DELETE



# Data manipulation language (DML)

## INSERT

---

INSERT INTO statement is used to insert new records in a table.

Syntax:

**INSERT INTO**

**table\_name**

**VALUES** ('value-1', 'value-2',.....);

# Data manipulation language (DML)

## INSERT

---

INSERT INTO statement is used to insert new records in a table.

Syntax:

**INSERT INTO**

**table\_name**

**VALUES** ('value-1', 'value-2',.....);

# Data manipulation language (DML)

## INSERT

---

Example :

```
INSERT INTO  
    employee  
VALUES ( 'E1', 'John', 'Manager', 'USA');
```

```
INSERT INTO  
    employee  
VALUES ( 'E2', 'Nick', 'AVP', 'UK');
```

```
INSERT INTO  
    employee  
VALUES ( 'E3', 'John', 'VP', 'INDIA');
```

# Data manipulation language (DML)

## SELECT

---

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

Syntax:

**SELECT**

*column1, column2, column3,...*

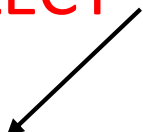
**FROM** *table\_name*;

# Data manipulation language (DML)

SELECT

---

SELECT \* FROM table\_name;

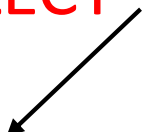
 (asterisk) represent all the columns from table

# Data manipulation language (DML)

SELECT

---

SELECT \* FROM table\_name;

 (asterisk) represent all the columns from table

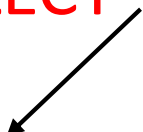
SELECT \* FROM employee;

# Data manipulation language (DML)

## SELECT

---

**SELECT** \* **FROM** table\_name;

 (asterisk) represent all the columns from table

**SELECT** \* **FROM** employee;

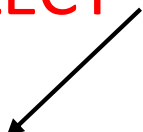
**SELECT** ename **FROM** employee;

# Data manipulation language (DML)

## SELECT

---

**SELECT** \* **FROM** table\_name;

 (asterisk) represent all the columns from table

**SELECT** \* **FROM** employee;

**SELECT** ename **FROM** employee;

**SELECT** ename, country **FROM** employee;



# Data manipulation language (DML)

## SELECT with WHERE

WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

Syntax:

**SELECT**

*column1, column2, column3,...*

**FROM** *table\_name*

**WHERE** **condition;**

# SQL Operators

## SQL WHERE

| Operator | Description  |
|----------|--|
| =        | Equal  |
| >        | Greater than   |
| <        | Less than  |
| >=       | Greater than or equal  |
| <=       | Less than or equal   |
| <>       | Not equal. <b>Note:</b> In some versions of SQL this operator may be written as != |
| BETWEEN  | Between a certain range  |
| LIKE     | Search for a pattern   |
| IN       | To specify multiple possible values for a column                                   |

# SQL Operators

## SQL WHERE

---

Condition can be applied in multiple columns using AND OR operators.

- AND operator displays a record if all the conditions separated by AND are TRUE.
- OR operator displays a record if any of the conditions separated by OR is TRUE.

# SQL Operators

## SQL WHERE

---

Example:

```
SELECT * FROM employee WHERE emp_id = 'E1';
```

```
SELECT * FROM employee  
WHERE emp_id = 'E2' AND country = 'UK';
```

```
SELECT * FROM employee  
WHERE post = 'AVP' OR country = 'UK';
```

# GROUP BY

## Aggregation

---

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country."

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

# GROUP BY

## Aggregation

---

### Syntax :

```
SELECT col-1, col-2, function(col-3)  
FROM table_name  
WHERE condition  
GROUP BY col-1, col-2;
```

### Example:

```
SELECT USN, Sections, max(CGPS)  
FROM student_details  
GROUP BY USN, Sections;
```

# GROUP BY with HAVING

## Aggregation

---

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

HAVING clause is equivalent to WHERE clause but HAVING used with only GROUP BY clause.

# GROUP BY

## Aggregation

---

### Syntax :

```
SELECT col-1, col-2, function(col-3)  
FROM table_name  
WHERE condition  
GROUP BY col-1, col-2  
HAVING condition;
```

### Example:

```
SELECT USN, Sections, max(CGPA)  
FROM student_details  
GROUP BY USN, Sections  
HAVING max(CGPA) > 7.5;
```