DDL(Data Definition Language)(Deals with the Structure):

1) **Database Related:**
    1) Show databases;
        1) This command will give me the list of all the databases
    2) CREATE DATABASE <<db-name>>;
        1) This command will create our database
    3) DROP DATABASE <<db-name>>;
        1) This command will remove my database
    4) USE <<db-name>>;
        1) This command will set my current-db to the db-name we have defined
2) **DDL Commands:**
    1) **CREATE:**
        1) Creates structure of a data object(eg: table)
        2) **CREATE TABLE:**
            1) Syntax(with Column-Level Constraints):
                1) CREATE TABLE <<table-name>>(
                <<col-name>> <<data-type>> <<col-constraint>>,
                ———————
                );
                2) Eg: CREATE TABLE student(
                student_id INT UNSIGNED NOT NULL,
                name VARCHAR(20) NOT NULL,
                major VARCHAR(15) NOT NULL,
                grade VARCHAR(2)
                );
            2) Syntax(With Table-Level Constraints):
                1) CREATE TABLE <<table-name>>(
                <<col-name>> <<data-type>> <<col-constraint>>,
                ————————,
                CONSTRAINT <<constraint_name>> <<constraint>>
                );
                2) EG: CREATE TABLE employee_data(
                employee_id INT NOT NULL,
                aadhar VARCHAR(12) UNIQUE,
                last_name VARCHAR(45) NOT NULL,
                email VARCHAR(25),
                salary DOUBLE(8,2),
                commision_percentage DOUBLE(2,2),
                hire_date DATE NOT NULL,
                CONSTRAINT emp_email_uk UNIQUE(email),
                CONSTRAINT emp_id_pk PRIMARY KEY(employee_id)
                );

3) EG: CREATE TABLE employees(
employee_id INT PRIMARY KEY,
last_name VARCHAR(20) NOT NULL,
email VARCHAR(15) NOT NULL UNIQUE,
salary DOUBLE(6,2),
commission_percentage DOUBLE(2,2),
hire_date DATE NOT NULL,
dept_id INT,
CONSTRAINT emp_dept_fk FOREIGN KEY(dept_id)
REFERENCES department(department_id)
);

2) Display And Describe Tables:

1) SHOW tables;
    1) It will display all the table names in the selected database
2) DESCRIBE <<table-name>>;
    1) It will give me the description i.e. col-name, constraints, data-type for each column of the table.

3) ALTER:

1) Alters the structure of the existing objects/tables

2) ALTER TABLE:

1) Syntax:
    1) ALTER TABLE <<table-name>> ADD <<constraint-name>>(<<col-name>>);
    2) EG:
        1) ALTER TABLE student ADD PRIMARY KEY(student_id);
        2) ALTER TABLE student ADD age INT;
        3) ALTER TABLE student DROP age;
        4) ALTER TABLE student MODIFY COLUMN major VARCHAR(10);

4) DROP:

1) Delete objects/tables from the database;

2) DROP TABLE:

1) Syntax:
    1) DROP TABLE <<table-name>>;
2) EG: DROP TABLE sample;

5) TRUNCATE:

1) Removes all the records from a table, including all spaces allocated for the records are removed.
2) Syntax:
    1) TRUNCATE <<table-name>>;
        1) EG: TRUNCATE sample;

6) Differences between DROP and TRUNCATE:

DROP	TRUNCATE
Used to remove table definitions and its contents	Used to delete all the rows from the table
Table space is freed from memory	Frees the record space
View of the table is also deleted	View of the table still remains
It is quick to perform but can give rise to complications	Faster than DROP
Integrity constraints will also be removed	The Integrity constraints are kept as it is

7) Constraints:

1) Enforce rules at table-level
2) Prevent data-manipulation, if there are dependencies
3) Constraint Name: its a variable to store the constraint-applied. It is not shown in the table structure.
    1) Eg: CONSTRAINT **emp_id_pk** PRIMARY KEY(employee_id);
4) Types:
    1) NOT NULL:
        1) This constraint is applied only at column-level
        2) It cannot be applied at table-level
    2) UNIQUE:
        1) Can be applied at column-Level as well as Table-Level
    3) PRIMARY KEY:
        1) Can be applied at column-Level as well as Table-Level
    4) FOREIGN KEY:
        1) Can be applied only at table-level
        2) It is also called referential integrity constraint
        3) The column that it references should be the PK or the UNIQUE key of the referenced table
        4) Constraint Keywords:
            1) FOREIGN KEY:
                1) Defines the column in the child table at the table-level
            2) REFERENCES:
                1) Identifies the table and the column in the parent table
            3) ON DELETE CASCADE:
                1) Deletes the dependent rows in the child table when the row in the parent table is deleted
            4) ON DELETE SET NULL:
                1) Convert the dependent foreign key values to NULL
    5) DEFAULT:
        1) It is used to assign the default value to the column when no value is inserted in it
        2) Column-Level

3) EG:
    1) CREATE TABLE sample(
    sample_id INT PRIMARY KEY,
    sample_name VARCHAR(20) DEFAULT 'name'
    );

8) ADDING & DROPPING Constraints:

1) Use ALTER command to:
    1) ADD or DROP a constraint but not to modify its(Constraints) structure
    2) Enable or Disable a constraint
    3) ADD a NOT NULL constraint using MODIFY clause
2) Syntax(ADD):
    1) ALTER TABLE <<table-name>> ADD [CONSTRAINT <<constraint-variable-name>>] <<constraint-type>> (<<col-name>>);
    2) EG: ALTER TABLE employees ADD CONSTRAINT emp_dept_fk FOREIGN KEY(dept_id) REFERENCES department(department_id);
3) Syntax(DROP):
    1) ALTER TABLE <<table-name>> DROP INDEX <<constraint-variable-name>>;
    2) EG:
        1) ALTER TABLE employees DROP INDEX emp_dept_fk;
        2) When Variable name is not given while defining the table structure:
            1) ALTER TABLE employees DROP PRIMARY KEY;

NOTE:

1) UNSIGNED -> Only positive numbers;

2) Listing All constraints:

1) #Listing All Constraints

```
SELECT column_name, constraint_name,
referenced_column_name, referenced_table_name
FROM information_schema.KEY_COLUMN_USAGE
where TABLE_NAME="employee_data";
```