DML(Data Manipulation Language):

Basics About SQL:

1) **It is case-insensitive**
2) **To use a DB:**
    1) **Use <<db-name>>;**
3) **To show list of database:**
    1) **Show databases;**
4) **To show the list of tables in the selected DB:**
    1) **Show tables;**
5) **To look into the table structure:**
    1) **Describe <<table-name>>**

1) It stands for Data Manipulation Language
2) Operations:
    1) Insert
    2) Retrieve
    3) Update
    4) Delete
3) Commands:
    1) **Insert:**
        1) **Syntax:**
            1) INSERT INTO <<table-name>>(<<col1>>,<<col2>>,——)
               VALUES(<<val1>>,<<val2>>,———);
                1) Eg: INSERT INTO student(name, age)
                   VALUES(“James”,21);
            2) INSERT INTO <<table-name>> SET
               <<col1>>=<<val1>>, <<col2>>=<<val2>>, —— ;
                1) Eg: INSERT INTO student SET name="Ritu", age=22;
    2) **Retrieval:**
        1) Basic Syntax:
            1) SELECT * FROM <<table-name>>; -> fetch all the
               columns of the table
                1) Eg: SELECT * FROM student;
            2) SELECT <<col1>>, <<col2>> FROM <<table-name>>;
               -> fetch the data for the selected columns.
                1) Eg: SELECT name, age FROM student;
        2) Querying Syntax: WHERE clause

1) SELECT * FROM <<table-name>> WHERE <<condition>>
    1) Eg: SELECT * FROM student WHERE age>=22;
3) Aliasing:
    1) Column Aliasing: Used when we are adding any operation on the columns to modify the o/p so that the modified column name becomes more understandable.
        1) Syntax:
            1) SELECT <<operation>> AS <<alias-name>> FROM <<table-name>> {WHERE <<conditions>>};
            2) Eg:
                1) select count(*) as student_count from student;
                2) select count(*) as "student count" from student;
                3) select concat(name," ",age) as "Name And Age" from student where id>102 and age IS NOT NULL;
    2) Table Aliasing: Used when we want to make two or more different tables or two instances of the same table to interact.
        1) Syntax:
            1) SELECT * FROM <<table1>> <<alias1>>, <<table2>> <<alias2>>, ———- WHERE <<condition>>;
            2) Eg:
                1) SELECT * FROM student s, person p where s.name=p.name;
                2) SELECT s.id AS "Student Id", p.name AS "PersonName", s.age AS "Student Age" FROM student s, person p where s.name=p.name;
                3) SELECT * FROM student s1, student s2 WHERE s1.id=s2.id;

3) Update:

1) Syntax:
    1) UPDATE <<table-name>> SET <<col>>=<<val>> WHERE <<condition>>;
        1) Eg:
            1) UPDATE student SET age=27 WHERE id=101;
            2) UPDATE student SET text="Hello World" WHERE text=“text”; {*Not a Good Practice to write = in WHERE clause from String*}

3) UPDATE student SET text="Hello WORLD" WHERE text LIKE "text";

2) Syntax(Using Table Aliasing):
   1) UPDATE <<table-name>> <<alias-name>> SET <<alias-name>>.<<col>>=<<val>> WHERE <<condition>>;
3) Syntax(Update all columns in a single go):
   1) UPDATE <<table-name>> SET <<coll>>=<<val>>;
   2) Eg:
      1) UPDATE student SET text=“Bye";

4) DELETE:

1) Syntax:
   1) DELETE FROM <<table-name>> WHERE <<condition>>;
   2) Eg:
      1) DELETE FROM student WHERE id=106;
      2) DELETE FROM student WHERE name LIKE "Jacobs";
2) Syntax(Using Table Aliasing):
   1) DELETE FROM <<table-name>> <<alias-name>> WHERE <<condition>>;
      1) Eg:
         1) DELETE FROM student s WHERE s.age=21;
3) Syntax(To DELETE all records):
   1) DELETE FROM <<table-name>>;
   2) Eg:
      1) DELETE FROM student;

NOTE:

1) **In INSERT statement when we are not mentioning any column names the query expects the arguments that we are passing as VALUES should match the number of**

attributes that we are having in the table no matter whether Auto_Increment has been set on the PK field.

1) Eg:
    1) INSERT INTO student VALUES(105,"Ron",21,"Hi I am RON”);
2) SELECT keyword -> puts constraints on the columns
    1) EG:
        1) SELECT * -> all the columns
        2) SELECT <<col1>> -> it will show data corresponding only to <<col1>>
3) WHERE clause -> puts a constraint on the rows
    1) EG: I only want to show the person above age of 18
        1) SELECT * FROM <<table-name>> WHERE age>18;
4) When we are aliasing a column the we can do it in two ways:
    1) alias_name
    2) “Alias name”
5) While updating or deleting all the rows at one go in a table you may get ERROR CODE: 1175. To disable this run:
    1) SET SQL_SAFE_UPDATES=0;