2.update:

--it is used to update the data within the table.

ex1: updating all the marks for all the students;

>update student set marks = marks+50;

ex2: updating marks for only one student: here we need to use 'where' clause.

>update student set marks = marks+50 where roll=12;

>update student set marks = marks+50 where sname='Sunil';

>update student set marks = marks+50 where marks <= 700;

> update student set marks = marks+50, sname='amit kumar' where sname='Amit';

>update student set marks=500 where marks IS NULL;

3.delete:

--it is used to delete the records/rows from the table.

>delete from student; //it will delete entire record from the table like truncate command.

>delete from student where sname IS NULL;

DRL (select)
==========

--this command is used to quering a table.

syntax:
----

select col1,col2,...

from tablename(s)
where condition
group by colname
having condition
order by colname [asc/desc]

ex1:

>select * from student; // all the columns and all the rows

ex2: restricting the number of rows by using where condition.

>select * from student where roll=10;

>select sname from student;

> select sname,roll from student;

>select sname from student where roll=10;

>select sname,roll from student where marks > 700;

using order by clause: to sort the records.

> select * from student order by marks;

> select * from student order by marks desc;

Operators:
==========

1. Arithmatic operators (*,/, + , -, %)

Note: mostly arithmatic operators are used after the select statements (90%) and all other operators are used in 'where clause' only.

2. relational operators: (= , >, <, <=, >= [!= or <>])

3.logical operators (AND, OR, NOT)

4. Special Operators (IS NULL, LIKE, BETWEEN, etc...)

examples:

Arithmatic operators:

>select sname,marks, marks+100 from student;

> select sname,marks, marks+100 Gracemarks from student;

Note: this temparory name of a column we can not use inside where clause.

Getting unique data(DISTINCT)
==========

>select DISTINCT marks from student;

Special Operators:
==========

IN ..... NOT IN

BETWEEN ..... NOT BETWEEN

IS NULL ..... IS NOT NULL

LIKE ..... NOT LIKE

>select * from student where marks IN(700,800,850);

>select * from student where marks BETWEEN 500 AND 800;
or
> select * from student where marks >=500 AND marks <= 800;

LIKE ----> NOT LIKE:

--it is used to retrive the data based on chrecter pattern.

1. % ---> it represents string or group of charecters.

2. _ ---> it represents a single charecter.

ex:

select * from student where sname LIKE 'r%'; // name will start with r.

ex: In name r can be any charecter.

> select * from student where sname LIKE '%r%';

Constraints:
==========

--constraints are created on the columns of a table.

--It prevents invalid data entry into our tables.

1. not null

2. unique

3. primary key

4. foreign key

5. check : Check constraint will not be supported by the mysql.

Note: some conatraints we can apply at column level and some contraints we can apply at table level.

column level: where we define the column.
not null,
unique,
primary key

table level: after defining all the column.
composit primary key (multi-column primary key)
foreign key

1. not null:

--null value is not allowed, that column value is mandatory.

2. unique:

--to that column duplicate values are not allowed.

--here we can insert null values, multiple time.

Note: whenever we define a unique constraint on a column then automatically DB engine will create an index on those columns. (searching based on unique column is super fast.)

3. primary key:

--here also DB engine will create index for that column.

--value can not be null.
--value can not be duplicate

another diff with the PK and unique is: in one table we can have multiple unique constraint but in one table we can have only one primary key.

--if we want to apply PK on multiple column then it will become a composit key.

*****Note: with the help of the PK column we can uniquly identify one record in a table.**

```
create table student
(
roll int primary key,
name varchar(12) not null unique,
address varchar(12) unique,
marks int
);
```

composit key:
----

teacher(tname, subject, age, phone, email);

create table teacher(

tname varchar(12),
subject varchar(12),
age int,
phone varchar(10),
email varchar(15),
primary key(tname,subject)
);

---here tname and subject will become a composit key, this combination can not be duplicate.

Foreign key:
=========

--with the help of FK we inforce the refrential integrity.

--with the help of FK we can establish relationship among 2 tables.

--Second table FK must refer to the first table PK.

--PK related FK column must belong to the same datatype but the column name can be different.

--FK can accept the duplicate and null value also.

Note: with the help of FK we can establish parent child relationship among tables.

create table dept
(
did int primary key,
dname varchar(12) not null,
location varchar(12)
);

create table emp
(

eid int primary key,
ename varchar(12),
address varchar(12),
salary int,
deptid int
);

lets achive the referential integrity:

> drop table emp;

create table emp
(
eid int primary key,
ename varchar(12),
address varchar(12),
salary int,
deptid int,
foreign key (deptid) references dept(did)
);

here dept table will act as a parent table
and emp table will act as child table.

--the table which contains the FK column will be considered as child table.

Note: whenever we try to establish a relationship using FK then DB violates following 2 rules:

1. insertion in the the child table.

2.deletion or updation in the parent table (even we can not drop the parent table also)

--to overcome this updation and deletion problem we should use

ON DELETE CASCADE
or
ON DELETE SET NULL

simillarly for update also

--while creating the child table.

```
create table emp
(
eid int primary key,
ename varchar(12),
address varchar(12),
salary int,
deptid int,
foreign key (deptid) references dept(did) ON UPDATE CASCADE
);
```

adding a contraint to an existing table:

```
> create table a1(id int, name varchar(12));

> alter table a1 modify id int primary key;
```

adding foreign key to the existing table:

```
>create table b1(bid int);

>alter table b1 add foreign key(bid) references a1(id) on delete set null;
```