```
SQL : it is used for all the RDBMS s/w:


--there is slight variation of SQL are there in different types of RDMBS s/w.

varchar2 --- oracle db
varchar

auto_increment : mysql DB
sequence :    Oracle DB

Full join : Oracle DB
mysql : Union


DML operations:  these opearation work on the data of the tables
----------------------
(insert, update, delete)

--inserting all column values.

>insert into student values(10,'Ram',780);

--inserting partial column value:

1. insert into student values(14,'Amit',null);

2. insert into student(roll,name) values(15,'Ravi');

ex:

>insert into student(name,roll) values('pawan',18);

update:
=======

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

ex:- following comand will set the value (update ) for all the students.

>update student set marks = 500;

--to update marks for only one student, here we need to use 'where' clause.

ex:
update student set marks = 500 where roll = 14;

ex2:

>update student set marks = 500 where roll = 14 OR name='pawan';

> update student set marks = marks+50 where name = 'Ramesh';

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

> update student set marks = 600 where marks IS NULL;

>update student set name='Ram Kumar', marks = marks+20 where roll = 10;

delete :
=======


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

```
>delete from student;  // it will delete the all the records from the table, like the truncate
command.

Note: truncate is the DDL command where as delete is a DML command, DDL commands we can not rollback
where as DML commands can be rolledback

>delete from student where roll = 18;


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

--this command is used to quering a table(s).

syntax:

select col1, col2,....
from tablename(s)
where conditions
group by columnName
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 * from student where marks > 600;

ex3: projecting few/single columns:

>select name from student;

>select name, marks from student;

>select marks,roll, name from student;

 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 statments (90%) and all other types of
operators are used inside the where clause only.

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

3. logical operators : (AND, OR, NOT)

4. special operators  :( IS NULL, LIKE, BETWEEN, etc..)
```

```
examples:

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

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

>select name, marks, marks+100 UpdatedMarks from student;

****this temparory name of a column we can not use inside the where clause.


Getting unique data (DISINCT)
=========================

>select DISTINCT marks from student;

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

IN ... NOT IN

IS NULL .... IS NOT NULL

LIKE ... NOT LIKE

BETWEEN ..... NOT BETWEEN


> select * from student where marks IN(700, 550,600);

> select * from student where marks BETWEEN 500 AND 700;

or

>select * from student where marks >=500 AND marks <=700;

LIKE ... NOT LIKE:
----------------------

-- it is used to retrieve the data based on charecter patterns.

1. % ---> it represents the string or group of charecteres.

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


ex:

>select * from student where name LIKE 'r%';  // name should start with 'r'.

ex: In name r can be any charecter.

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

--r should be the 3rd charecter:

> select * from student where name LIKE '__r%';


Constraints in SQL:
================

--constraints are created on the column of a table.
```

```
--it prevents invalid data entry into our table.

1. not null

2. unique

3. primary key

4. foreign key

5. check : this contraint will not be supported by the mysql.


Note: some constraints we can apply at the column level and some constraints
we can apply at the table level.


column level : where we define the column

not null,
unique
primary key

table tavel : after defining all the columns

foreign key
composit key (multi-column primary key)


1. not null:
-------------

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

2. unique:
------------

--to that column duplicate values are not allowed.

--here we can insert null values multiple times.

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


3. primary key:
------------------

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

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

--another diff bt PK and unique is : inside one table we can have multiple
unique constraints but inside one table we can have only one Primary key.

--if we want to apply the PK on multiple columns of a table then it will become a composit key.

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



create table student
```

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

composit key:
-----------------

teacher (tname, subject, age, address, pincode)

create table teacher
(
tname varchar(12) not null,
subject varchar(12) not null,
age int not null,
address varchar(12),
pincode varchar(12),
primary key (tname, subject)
);

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


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

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

--with the help of a FK we establish the relationship among two tables.

--Second table(child table) FK column must refer to the PK column of the parent/first table.

--PK related FK column must belongs to the same datatype but the column names
can be different.

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


Note: with the help of FK we can establish the parent and child relationship among 2 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),
salary int,
deptId int
);


--lets achieve the referential integrity:

> drop table emp;
```

```
>create table emp
(
eid int primary key,
ename varchar(12),
salary int,
deptId int,
foreign key (deptId) references dept(did)
);


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



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

1. insertion inside the child table. (we can not insert a data which is not there inside the parent
table)

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

--so, in order to drop the parent table, we need to drop all the child tables then only we can drop
the parent table.

> delete from dept where did = 12; // error

> update dept set did = 18 where did =12; // error


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

ON DELETE CASCADE
or
ON DELETE SET NULL

similarly we can use for update also

ON UPDATE CASCADE
or
ON UPDATE SET NULL

--while creating the child table.


>create table emp
(
eid int primary key,
ename varchar(12),
salary int,
deptId int,
foreign key (deptId) references dept(did) ON UPDATE CASCADE ON DELETE SET NULL
);
```