**SQL-Structured Query Language**

* SQL is not case sensitive.
* SQL uses ; (semicolon) to separate two statement.

**What are the different types of SQL commands?**

DQL(Data Query Language):select

DDL(Data Definition Language):create,drop,alter

DML(Data Manipulation Language):insert,update,delete

DCL(Data Control Language):grant,revoke

* **Data Definition Language- CREATE,ALTER,DROP,TRUNCATE**

Basically DBA team uses create command & tester get readymade table.

**1 .CREATE:**It is used for creating table & Database.

**Syntax:**

Create table tablename( column 1 datatype,

column 2 datatype,

..column n datatype);

**Example:**

Create table MockResult(Firstnamevarchar(25),

Lastnamevarchar(20));

**Syntax for creating Database:**

Create database database name;

**Example:**

Create database Details;

**2.ALTER:** It is used to make changes in existing table.

* **Add-**

**Syntax:** Alter table tablename

Add columnnamedatatype;

**Example:** Alter table Mockresults

Add results varchar(25);

* **Drop**-

**Syntax:** Alter table tablename

Dropcolumn columnname;

**Example:** Alter table Mockresults

Dropcolumn results;

* **Modify**-

**Syntax:** Alter table tablename

Modifycolumnnamedatatype;

**Example:** Alter table Mockresults

ModifyLastnamevarchar(20);

* **Rename-**

**Syntax:** Alter table tablename

Rename column oldcolumnname to new columnname;

**Example:** Alter table Mockresults

Renamecolumn results to Feedback;

**3.Truncate:** Its used to delete data & structure remains same.

**Syntax:** Truncate table tablename

**Example:** Truncate table Mockresults;

**4.Drop:**To drop existing table from database.

**Syntax:** Drop table tablename

**Example:** Drop table Mockresults;

**Example:**

Drop database Studnet;

**DELETE** : It is used to delete existing records from the database.Where structure won’t change Delete command used to delete particular rows or all the rows from the table .

**Syntax**: Delete from tablename where condition ;

**Example: Delete from vctc where fname='rahul';**

**WHAT is the difference between delete drop &Truncate**

**DROP**: With the help of drop command we can drop complete table & Also database.

**DELETE**:With the help of delete command we can remove particular row .& also we can remove the multiple rows where structure will remain same.

**TRUNCATE**: With the help of truncate we can Remove the data from table but the structure will reamain same.

**DATA MANIPULATION LANGUAGE:** Insert,Select,Delete,Update

1. **Insert:** To insert the records into the table.

**Firstway**

**Syntax:** Insert into tablename(Column1,Column2)

Values(Value1,Value2);

**Example:** Insert into Mockresults(Firstname,Lastname)

Values(‘Akshay,’Kumar’);

Secondway: When you are aware about the table structure

**Syntax:** Insert into tablenameValues(Value1,Value2);

**Example:** Insert into MockresultsValues(‘Akshay,’Kumar’);

1. **SELECT:**To fetch the date from table & database.

* **To fetch all columns from the table.**

**Syntax:**Select\*from table name;

**Example:** Select\*fromMockresults;

* **To fetch Specified columns from the table.**

**Syntax:** Select column1,column2

From table name;

**Example:** SelectFirstname,Lastname

From Mockresults;

**WHERE CLAUSE:**

In where clause we will use relational & Logical operators.

**LOGICAL**: AND,OR,,NOT

1.AND operator: It displays the records when both the conditions are TRUE.

**Syntax:** Select \* from Tablename

wherecondition;

**Example**: Select \* from Products

whereproductname='chair' AND unit=10;

2.OR operator: : It displays the records when one of the conditions is TRUE.

**Syntax:** Select \* from Tablename

wherecondition;

**Example**: Select \* from Products

whereproductname='chair' OR unit=20;

1. NOT operator: : It displays the records when condition is not TRUE or FALSE.

**Syntax:** Select \* from Tablename

wherecondition;

**Example**: Select \* from products

where not unit=20;

**Relational operators**<,>,<=,>=,=

In this we apply condition by using operators.

**Syntax:** Select column 1,column2

From tablename

wherecondition;

**Example**: Select\* from products where unit>10;

Select\* from products where unit>=10;

Select\* from products where unit<=10;

Select\* from products where unit>=20;

Select \* from employee where job='clerk' and sal<12000;

Select \* from employee where salary<12000;

**DESC**: It is used for getting description of table.

**Syntax**desctablename;

**Example**: desc students;

**IN:**when we need to fetch a multiple data from single column then we use in operator's.

**Syntax:** Select column1 ,column2

fromtablename

where INcondition;

**Example**: Select productname,unit

from products where unit in(10,5,20);

**Not IN:**

**Syntax:** Select column1 ,column2

fromtablename

where Not INcondition;

**Example**: Select productname,unit

from products where unit not in(10);

**BETWEEN:**

It select values within the range. Values can be number,text or date( Between operator fetch a data btw two ranges)

**Syntax**: Select \* from tablename

where price BETWEENcolumn1 AND column2;

**Example**: Select \* from products

whereprice BETWEEN 100 and 300 and unit=10;

**NOT BETWEEN:**

It select values not within the range

**Syntax**: Select \* from tablename

where price not BETWEENcolumn1 AND column2;

**Example**: Select \* from products

whereprice not BETWEEN 100 and 300 and unit=10;

**LIKE** :- Operator basically used to search specifies condition in table. it is used to search pattern in a columns.

Percentage (%) – It represents Zero,one& multiple char.

Underscroll(\_)- It will represent one single character.

**Syntax:**

|  |  |
| --- | --- |
| Select \* from tablename where columnname Like 'F%'; | Find any value which start with ‘F’ |
| Select \* from tablename where columnname Like '%F'; | Find any value which ends with ‘F’ |
| Select \* from tablename where columnname Like ‘%F%'; | Find any value that have ‘F’ in any position |
| Select \* from tablename where columnname Like '\_\_F%'; | Find any value that have ‘F’ in third position |

**Example**:

Select \* from products where productname like '\_\_air%';

Select \* from products where productname like '%A%';

Select \* from products where productname Like 'F%';

Select \* from products where productname like '%E';

* **Select Distinct:** It returns distinct records & values.

**Syntax:** Select distinctcolumnname

From table name;

**Example:** Select Distinctproductname

from products;

* **Top Clause:** To fetch top records from table.

**Syntax:** Select Top number

\*From table name;

**Example:** Select Top (3)

\*From products;

But This Top clause does not support in MYSQL So we use LIMIT in MYSQL

**Syntax:** Select \*From table name

LIMIT 3;

**Example:** Select \*Fromproductname

LIMIT 3;

**NULL:** A field with no value

Blank value,null value is different from zero.A field which kept blank during record creation is called Null.

**Syntax:**  Select column name

From table name

Where columnname IS NULL;

**Example:** Select productid

From product

Where productid IS NULL;

**IS NOT NULL:**

.**Syntax:**  Select column name

From table name

Where columnname IS NOT NULL;

**Example:** Select productid

From product

Where productid IS OT NULL;

**GROUP BY CLAUSE:**

Group by used in SQL with SELECT statement toorganize data in groups. It is used with Aggregate function.

.**Syntax:** Select columnname ,aggregate fun

fromtablename

group by columnname;

**Example:** Select productid ,count(\*)

from products

group by productid;

**Order BY CLAUSE:**

It is used to sort result in ASC or in DESC.

Descending-

.**Syntax:** Select \* from tablename

order by columnnamedesc/asc;

**Example:**Select \* from teams

order by saldesc;

Ascending- BYdefault ASC

**Example:**Select \* from teams

order by sal;

**Example:**Select \* from teams

order by salasc;

**Aliases:**

Column or table can be renamed temporary.

It only exists for the duration of particular query.

.**Syntax:** Select columname as Alias name

fromtablename;

**Example:**Select productname as productcollection

from products;

**Aggregate Function:**An Aggregate function performs calculations on set of values & returns a single value.

**Max ,Min,Sum,Avg,Count.**

* **MAX:** It returns max value from table.

.**Syntax:** Select MAX(column name)

From table name;

**Example:** Select MAX(Salary)

From products;

* **MIN:** It returns min value from table.

.**Syntax:** Select MIN(column name)

From table name;

**Example:** Select MIN(Salary)

From products;

* **SUM:** It give SUM from table.

.**Syntax:** Select SUM(column name)

From table name;

**Example:** Select SUM(Salary)

From products;

* **AVG:** It returns Average value from table.

.**Syntax:** Select AVG(column name)

From table name;

**Example:** Select AVG(Salary)

From products;

* **Count:** It returns total of records.

.**Syntax:** Select count (\*)

From table name;

**Example:** Select count(\*)

From products;

**Having Clause:** Utilized as conditional statement.

It is used as option for Where clause. Bcz we cannot combine where clause with aggregate function so having clause is used .

**Syntax:**

Select columnname

From tablename

Where condition

GroupBycolumnname

Having condition

Order by column name;

**Example:**Select city,count(\*)

from Students where city ='pune'

group by city having count(\*)>=1

order by city ;

**Union & Union All:**

SQL operator used to combine two or more result sets.

* **Union:** Remove any dupe records,

only keeps unique records.

Combine result sets of 2or more select statement.

**Syntax:**

Select \* from table1

union

Select \* from table2

order by condition;

**Example:**

Select \* from Students

union

Select \* from students1

order by id;

* **Union all:**

Keeps all records including dupe

**Syntax:**

Select \* from table1

Union all

Select \* from table2

order by condition;

**Example:**

Select \* from Students

Union all

Select \* from students1

order by id;

**Constraints:** Limitation

It is used to limit the data.

**Types of constraints:**

Not null,unique,primarykey,foreignkey,check

1.Not Null: Ensure that the column has the null value it should not be blank.

**Syntax:**

create table tablename

(column1datatype not null, column2datatype);

**Example**:

create table faculty

(employeeIdint(2) not null,

Name varchar (12),Salaryint);

2.Unique: All values in a column should be unique.

3.Primary Key: Primary key basically combination of not null and unique key , only one primary key we can declare in a table . Not allow duplicate values and not allow the null values and used for joining the two tables

**Syntax:**

CREATE TABLE tablename

(column1datatype not null,

column2datatype    PRIMARY KEY ());

**Example:**

CREATE TABLE Students( ID int NOT NULL,  
    LastNamevarchar(255) NOT NULL,  
    FirstNamevarchar(255),  
    Age int,  
    PRIMARY KEY (ID));

4.Foreign Key:

Foreign key is used to join the two tables , it allows duplicate values and it allow the null values , multiple

Foreign key can be declare in a same table

**Syntax:**CREATE TABLE tablename (

(column1datatype not null,

column2datatype

 PRIMARY KEY ()  
 FOREIGN KEY () REFERENCES Persons());

**Example:**

CREATE TABLE facultydept (

DeptidIDint NOT NULL,

Name varchar (12),

CONSTRAINT FK\_deptname FOREIGN KEY (Name)

REFERENCES faculty(N**ame))**

**Check :-**

Check is give the condition along with column w.r.to the customer requirement .Ensure that all values in a column satisfied specificed.

**Syntax:**

Create table Employeesalary (Employee id int(12),

Name varchar(30),Salary int(15)

check(Salary>2000));

**Default :-**

Default provides default value for a column when nothing is specified.

**Syntax:**

Create table StudentCity(Name varchar(15),City varchar(20)default ‘Pune’);

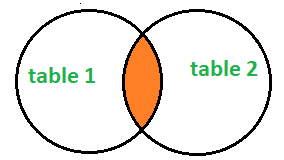
**JOINS:-**

A join clause is used to combine rows from two or more tables based on related column between them.

Used to join two or more tables.

**Types of Joins:**

**INNER JOIN:** Returns the records that have matching values in both the table (table1 & table2).

****

**Syntax:**

**Select** table 1. Column1,table 2.Column 2

**From** table 1

**INNER JOIN** table 2

**ON** table 1. Common Field(pk)= table 2.Commin Field(fk);

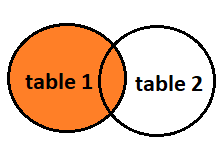
**Example:**

Select f1.NAME,f2.DEPTID from Faculty as F1

INNER join

Facultydept as F2on f1.Name=f2.name;R

**LEFT JOIN:**Returns all records from left table..& matching results from right table.

****

**Syntax:**

**Select** table 1. Column1,table 2.Column 2

**From** table 1

**LEFT JOIN** table 2

**ON** table 1. Common Field(pk)= table 2.Commin Field(fk);

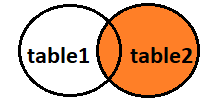
**Example:**

Select f1.NAME,f2.DEPTID from Faculty as F1

Left join

Facultydept as F2on f1.Name=f2.name;

**RIGHT JOIN:**Returns all records from Right table..& matching results from left table.

****

**Syntax:**

**Select** table 1. Column1,table 2.Column 2

**From** table 1

**RIGHT JOIN** table 2

**ON** table 1. Common Field(pk)= table 2.Commin Field(fk);

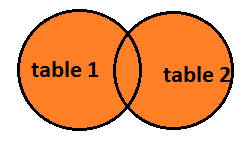
**Example:**

Select f1.NAME,f2.DEPTID from Faculty as F1

RIGHT join

Facultydept as F2on f1.Name=f2.name;

**FULL JOIN:**Returns all the records when there is match either in left or right table.



Right join query + union +

Left join query = full join

**Syntax:**

**Select** table 1. Column1,table 2.Column 2

**From** table 1

**RIGHT JOIN** table 2

**ON** table 1. Common Field(pk)= table 2.Commin Field(fk);

**Example:**

Select \* from faculty as F

right join facultydept as f1

on f.name=f1.name

union

Select \* from faculty as F

left join facultydept as f1

on f.name=f1.name

**SELF JOIN:** It is normal SQL join that joins itself.

Each row joins itself by other rows

**Example**: Select \*from faculty as f1,faculty as f2

where f1.employeeid=f2.employeeid;

**1 table ---- join perform**

**To join the three tables: -**

**Select t1.columname , t2.columname ,t3.columname**

**from Table1 t1 inner join table2 t2**

**on t1.PK = T2.FK**

**inner join table3 t3**

**on t2.pk =t3.fk**

**Ques.9. How are Joins different from the Union clause?**  
Ans. Some features of Joins are-

1. Join combines attributes of the rows present in the two tables that share some common fields or attributes.
2. Join is applicable when the two tables have at least one common attribute.
3. There are many types of joins like INNER JOIN, OUTER JOIN, LEFT JOIN, RIGHT JOIN, etc.
4. The length of the resultant rows is more as compared to the length of rows of the tables involved.

Whereas in case of Union-

1. Join combines rows of the tables that are present in the query.
2. Join is applicable when the number of columns present in the query is the same and the corresponding attributes have the same domain.
3. There are two types of a union like UNION and UNION ALL.
4. The number of the resultant rows is more as compared to the number of rows present in each table involved in the query.

**Ques.10. Write the SQL command to join 3 tables.**  
Ans. Consider the below 3 tables. Now in order to fetch EmpName**,**DeptName, Gender information of all the employees, we will join the 3 tables.

| **EmpNo** | **EmpName** | **DeptId** | **GenderId** |
| --- | --- | --- | --- |
| **1886** | **John Snow** | **101** | **1** |
| **1889** | **Amara Giselle** | **102** | **2** |
| **1890** | **Richie Tung** | **102** | **1** |

**Table – Employees**

| **DeptId** | **DeptName** |
| --- | --- |
| **101** | HR |
| 102 | IT |

**Table – Department**

| **GenderId** | **Gender** |
| --- | --- |
| 01 | Male |
| 02 | Female |

**Table – Gender**

SELECT EmpName, DeptName, Gender

FROM Employees

JOIN Departement ON Employees.DeptId=Department.DeptId

JOIN Genders ON Employees.GenderId=Genders.GenderID;