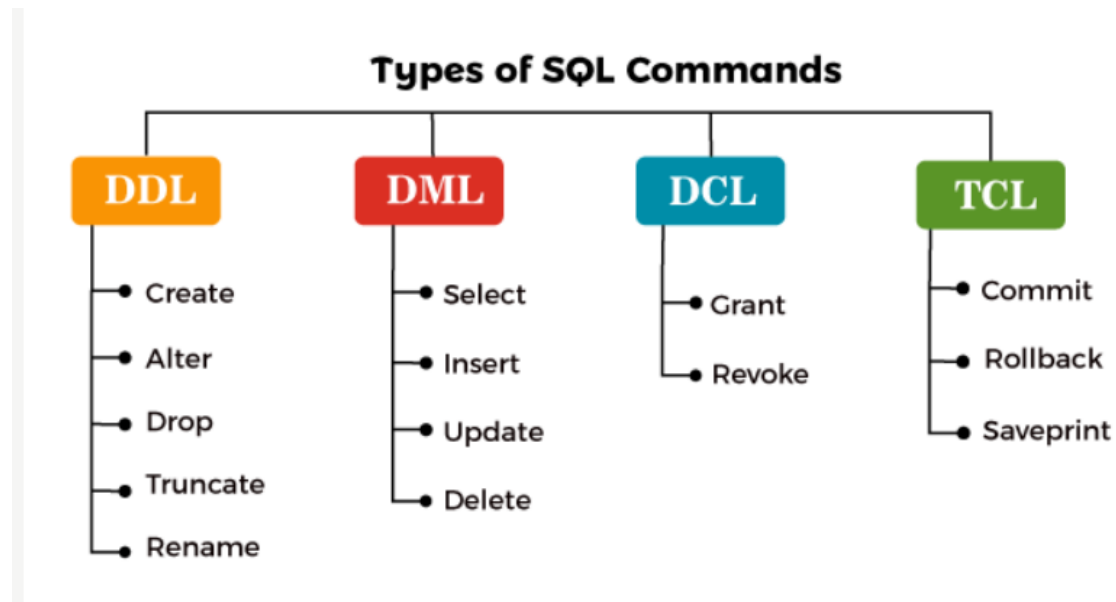




SQL Imp Concepts

Types of SQL Commands



Data Definition Language(DDL)

▼ Data Definition Language(DDL)

Data Definition Language command are used to define the database schema. DML command are used to create ,modify, and delete database structures but not data.

| DDL commands:

- **CREATE**: This command is used to create the database or its objects like table, index, function, views, store procedure, and triggers. The Syntax of CREATE query is follow:

```
CREATE TABLE employees (  
    ID int NOT NULL UNIQUE,  
    Name varchar(255),  
    Age int,  
    department varchar(255),  
    city varchar(255)  
);
```

- **ALTER:** This command is used to alter the structure of the database. It is also used to add and drop various constraints on an existing table. The Syntax of ALTER query is follow:

```
ALTER TABLE employees  
ALTER COLUMN phone VARCHAR(15);
```

- **DROP:** This command is used to remove a column from an existing table. The Syntax of DROP query is follow:

```
ALTER TABLE employees  
DROP COLUMN address;
```

- **TRUNCATE:** This command is used to remove all records from a table, including all spaces allocated for the records are removed. This command remove the data inside the data but not a table . It is faster than the DELETE command as it removes all rows in one operation rather than deleting each row one by one.

The Syntax of TRUNCATE query is follow:

```
TRUNCATE TABLE employees
```

- **RENAME:** This command is used to used to rename an existing table or column. The Syntax of CREATE query is follow:

```
ALTER TABLE employees
```

```
RENAME COLUMN Name to emp_Name
```

▼ Data Manipulation Language(DML)

Data Manipulation Language command are used to make **changes into database such as CRUD(create, read, update and Delete) operation.**

| DML commands:

- **SELECT:** The select is used to select data from a database. The Syntax of SELECT query is follow:

```
1)
SELECT * FROM employees
2)
SELECT Name, Age, department
FROM employees
```

- **INSERT:** This command is used to insert one or more rows of data into a database. Before inserting data should ensure that table is already created. The Syntax of INSERT query is follow:

```
INSERT INTO employees(ID, Name, Age, department, city)
VALUES (1, 'sadaf', 20, 'SE', 'ISLAMABAD');
```

- **UPDATE:** This command is used to update existing data into one or multiple rows or columns by using UPDATE and WHERE Clause. The Syntax of UPDATE query is follow:

```
UPDATE employees
SET Name= 'Sadaf sultan'
WHERE Age= 20;
```

• **DELETE:** This command is used to remove single or multiple existing records from the database tables. The Syntax of DELETE query is follow:

```
DELETE FROM employees
WHERE Name ='sadaf sultan'
```

• **WHERE:** It is used to retrieve data from the database to specify a condition while fetching the data from a single table or by joining with multiple tables. The Syntax of WHERE query is follow:

```
SELECT * FROM employees
WHERE age > 50
```

```
SELECT * FROM employees
WHERE Name = 'sadaf'
```

• **ORDER BY:** This keyword to use sort data into ascending and descending orders. The syntax of this keyword is follow:

```
SELECT *
FROM employees
ORDER BY age;
---- by default in ascending order
```

```
SELECT *
FROM employees
ORDER BY age DESC;
---- the keyword of DESC is used for descending order
```

• **DISTINCT Keyword:** This command is used to remove the duplicate values. this command returns only on different values exiting in the column. The Syntax of DISTINCT query is follow:

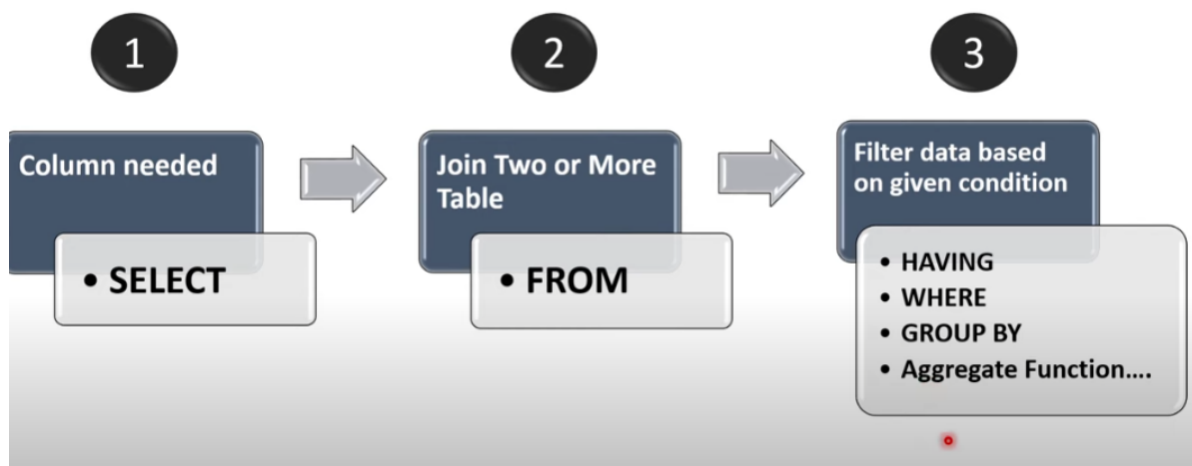
```
SELECT DISTINCT ProductName
FROM Products
```

```
SELECT DISTINCT ProductName, UnitPrice, UnitsOnOrder, ReorderLevel
FROM Products
WHERE ReorderLevel BETWEEN 1 AND 10
ORDER BY ReorderLevel
```

▼ Data Query Language(DQL)

DQL includes a variety of commands and clauses, such as SELECT, FROM, WHERE, GROUP BY, HAVING, and ORDER BY. These commands are used to specify which data should be retrieved, where to retrieve it from, how to filter it, how to group and aggregate it, and how to sort it.

3 Step Formula To Solve Join Based Query i



• **SELECT**: It is used to retrieve data from the database. The Syntax of SELECT query is follow:




```
SELECT * FROM employees
```

● **GROUP BY** : It is used to group the result set (used with aggregate functions: COUNT, MAX, MIN, SUM, AVG). Count is used to returns the number of rows that matches a specified criterion. AVG is used to returns the average value of a numeric column. SUM is used to returns the total sum of a numeric column. MAX is used to returns the largest value of the selected column. MIN is used to returns the Smallest value of the selected column.

Important Points-

- ✓ GROUP BY clause is used with the SELECT statement.
- ✓ In the query, GROUP BY clause is placed after the WHERE clause.
- ✓ In the query, GROUP BY clause is placed before HAVING & ORDER BY clause if used any.
- ✓ We use GROUP BY clause in conjunction with an aggregate expression.

WHERE → GROUP BY → HAVING → ORDER BY

	WHERE Clause	HAVING Clause
	WHERE Clause cannot contain aggregate functions	HAVING Clause can contain aggregate functions
	The WHERE clause is used to filter rows <i>before</i> the grouping is performed.	The HAVING clause is used to filter rows <i>after</i> the grouping is performed.
	GROUP BY Clause is used after WHERE Clause	GROUP BY Clause is used before HAVING Clause
	WHERE Clause is used with single row function like UPPER, LOWER etc.	HAVING Clause is used with multiple row function like SUM, COUNT etc.

```
-- // sql scripts
```

```
select d.dept_name
from departments d left join employees e
on d.dept_id = e.dept_id
group by d.dept_name
having count(e.dept_id)>2;
```

```
select d.dept_name, avg(e.salary)
from departments d left join employees e
on d.dept_id = e.dept_id
group by d.dept_name
having avg(e.salary)>75000
order by d.dept_name DESC;
```

```
select e.emp_name , m.manager_name , e.city
from employees e left join managers m
on e.manager_id=m.manager_id
where e.city = m.city;
```

```
select m.manager_name , d.dept_name , e.emp_name , e.salary
```

```
from employees e inner join departments d
on e.dept_id = d.dept_id inner join managers m
on m.manager_id = e.manager_id
where e.salary between 350000 and 90000;
```

```
-- // sql scripts
select dept_id , count(emp_id)
from employees
group by dept_id;
```

```
select d.dept_name , count(e.emp_id)
from departments d left join employees e
on d.dept_id = e.dept_id
group by d.dept_name
having count(e.emp_id)>2;
```

```
select d.dept_name , max(e.salary) , e.emp_name
from departments d left join employees e
on d.dept_id = e.dept_id
group by d.dept_name;
```

The Syntax GROUP BY query is follow:

```
---- COUNT
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;

SELECT COUNT(ProductID)
FROM Products;
```



```

----- AVERAGE
SELECT AVG(ProductID)
FROM Products;

---- SUM
SELECT SUM(ProductID)
FROM Products;

---- MAXIMUM
SELECT MAX (ProductID)
FROM Products;

----- MINIMUM
SELECT MIN(ProductID)
FROM Products;

```

• **COLUMN ALIASES:** It is often used to make column names more readable. An **alias** only exists for the duration of that query. An **alias** is created with the AS keyword. The Syntax ALIASES is follow:

```

SELECT TerritoryID, TerritoryDescription AS TerritoryDetails
FROM Territories

```

• **HAVING:** This clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

```

SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;

```

• **BETWEEN , AND & OR :** used to select values within a given range. The values can be numbers, text, or dates. The AND is used when we want only on the given range or specified range in this case both values must be satisfied

the condition. But by using OR operator if one condition or value is exiting the result be will be shown. The Syntax of WHERE query is follow:

```
SELECT OrderID, CustomerID, ShipName
FROM Orders
WHERE OrderID BETWEEN 10270 AND 10300
```

```
SELECT OrderID, CustomerID, ShipName
FROM Orders
WHERE ShipName BETWEEN 'EFG' OR 'ABC'
```

● **IN & LIKE:** The IN command use to allow multiple values into WHERE clause. The Syntax of IN query is follow:

```
SELECT * FROM employees
WHERE department IN ('Software', 'Computer Science', 'information Technology');
```

LIKE operator is used in WHERE clause to search a specified pattern in a Column. The Syntax of LIKE query is follow:

```
--This query show all the employees whose names started with A
SELECT * FROM employees
WHERE Name LIKE 'a%';
```

```
--This query show all the employees whose names ending with A
SELECT * FROM employees
WHERE Name LIKE '%a';
```

```
--This query show all the employees whose names contain A in second position
SELECT * FROM employees
```

```
WHERE Name LIKE '%_a';
```

--This query show all the employees whose names started with S but ended at F

```
SELECT * FROM employees  
WHERE Name LIKE 'S%f';
```

- **IS NULL:** It is used to find the null values in a specific column. The Syntax IS NULL query is follow:

```
SELECT * FROM Employees WHERE department IS NULL;
```

- **TOP N:** It is used to specify the number of records which want to return. The Syntax IS NULL query is follow:

```
SELECT TOP 20*  
FROM Orders  
ORDER BY EmployeeID
```

```
SELECT TOP 10 *  
FROM EmployeeTerritories  
WHERE EmployeeID = 5  
ORDER BY TerritoryID DESC
```

- **UNION :** It is used to combines the results of two or more SELECT statements into a single result set, without duplicates. The Syntax of UNION query is follow:

```
SELECT *  
FROM Customers  
WHERE Country = 'germany' AND Region IS NULL
```

```
UNION
SELECT *
FROM Customers
WHERE Country = 'Mexico' AND Region IS NULL
```

- **INTERSECT:** It is used to returns only the rows that are common between two SELECT statements, without duplicates. The Syntax of INTERSECT query is follow:

```
SELECT *
FROM Customers
WHERE Country = 'germany' AND Region IS NULL
INTERSECT
SELECT *
FROM Customers
WHERE City LIKE 'A%'
```

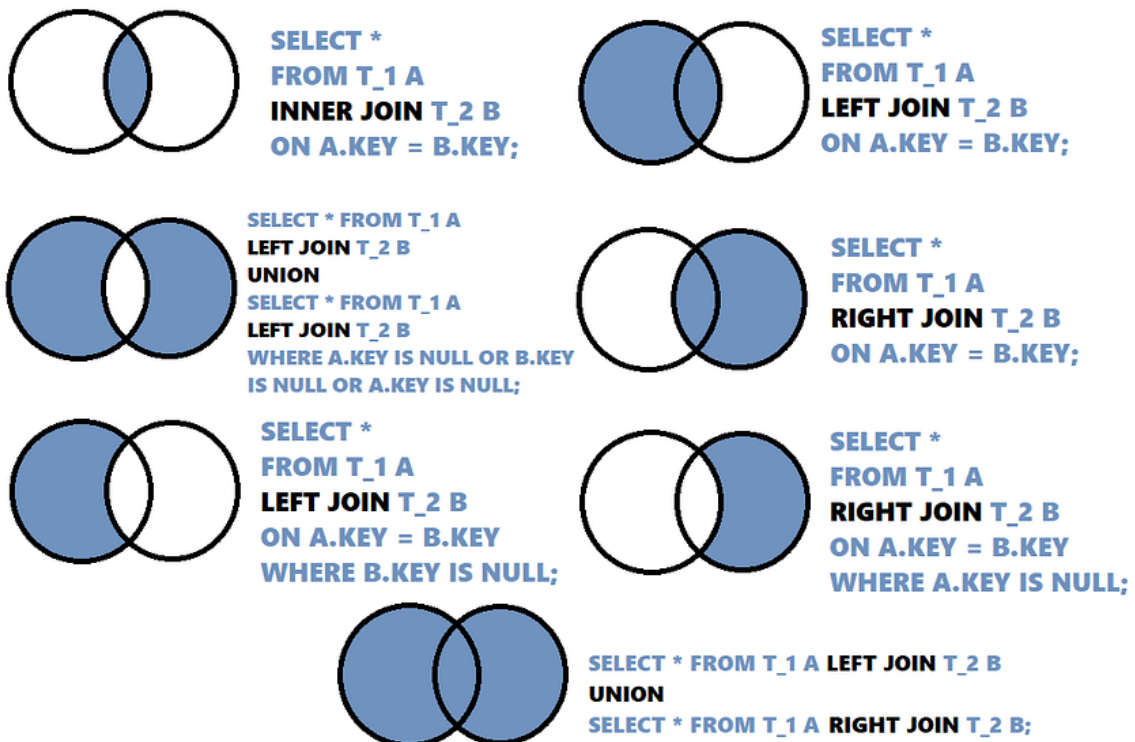
- **EXCEPT:** It is used returns only the rows that are present in the first SELECT statement but not in the second SELECT statement, without duplicates. The Syntax of EXCEPT query is follow:

```
SELECT *
FROM Customers
WHERE Country = 'germany' AND Region IS NULL
EXCEPT
SELECT *
FROM Customers
WHERE Country = 'Mexico' AND Region IS NULL
```

▼ Joins

In SQL joins are used to combine rows from two or more tables based on a related column between them. There are several types of joins available in SQL :

JOIN



Inner join

The inner join will return only the record where a key find a match in both tables.

```
SELECT * FROM Orders AS o
INNER JOIN [Order Details] AS OD
```

```
ON o.OrderID = OD.OrderID
```

```
SELECT o.OrderID, o.CustomerID, o.OrderDate FROM Orders AS o  
INNER JOIN Customers AS c ON o.CustomerID = c.CustomerID
```

Outer join

The OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

```
SELECT * FROM Orders AS o  
FULL OUTER JOIN [Order Details] AS OD  
ON o.OrderID = OD.OrderID
```

Self join

A self join is a regular join, but the table is joined with itself

```
SELECT e1.TitleOfCourtesy, e1.FirstName, e1.LastName, e2.City, e2.Address  
FROM Employees e1  
INNER JOIN Employees e2 ON e1.Title = e2.Title AND e1.EmployeeID = e2.EmployeeID
```

Cross join

The Cross Join returns all records from both tables.

```
SELECT P.ProductName, c.companyName, c.ContactTitle  
FROM Products P  
CROSS JOIN Customers c
```

Right join

The Right Join keyword returns all records from the right table, and the matching records from the left table . The result is 0 records from the left side, if there is no match.

```
SELECT * FROM Orders AS o
RIGHT JOIN [Order Details] AS OD
ON o.OrderID = OD.OrderID
```

```
SELECT o.OrderID, o.CustomerID, o.OrderDate FROM Orders AS o
RIGHT JOIN Customers AS c ON o.CustomerID = c.CustomerID
```

Left join

The Left Join keyword returns all records from the left table (Table1), and the matching records from the right table (Table2). The result is 0 records from the right side, if there is no match.

```
SELECT * FROM Orders AS o
LEFT JOIN [Order Details] AS OD
ON o.OrderID = OD.OrderID
```

```
SELECT o.OrderID, o.CustomerID, o.OrderDate FROM Orders AS o
LEFT JOIN Customers AS c ON o.CustomerID = c.CustomerID
```

Left anti-join

One of the join kinds available in the Merge dialog box in Power Query is a *left anti join*, which brings in only rows from the left table that don't have any

matching rows from the right table

```
SELECT c.ContactName, c.CompanyName,o.ShipCountry
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.CustomerID IS NULL
```

| Right anti-join

One of the join kinds available in the Merge dialog box in Power Query is a *right anti join*, which brings in only rows from the right table that don't have any matching rows from the left table

```
SELECT c.ContactName, c.CompanyName,o.ShipCountry
FROM Orders o
RIGHT JOIN Customers c ON c.CustomerID = o.CustomerID
WHERE o.CustomerID IS NULL
```

▼ Views

A view is a virtual table that is based on the result of a SELECT statement. A view can be used to simplify complex queries or to present a subset of data to users without exposing the underlying tables.

| Purpose Of Views

The purpose of views is to provide a customized and simplified view of the data for the end-users, without them needing to know the underlying structure of the data. Views providing a layer of abstraction that protects sensitive data or schema changes. These are also known as virtual tables.

| Types of Views

There are two types of view :

- **Simple view:** A simple view is based on a single table and selects all the columns or a subset of the columns from that table.

```
CREATE VIEW shippersView
AS
SELECT* FROM Shippers WHERE CompanyName='United Package'
```

- **Complex view:** A complex view, is based on multiple tables and includes one or more joins or subqueries in the SELECT statement.

```
CREATE VIEW orders_View_Detail
AS
SELECT OrderID,OrderDate , ShipName, ShipCountry FROM Orders WHERE CustomerID IN
(SELECT CustomerID FROM Customers WHERE Country IN
(SELECT Country FROM Suppliers WHERE Country= 'japan' OR Country='uk' ))
```

Creating view

```
CREATE VIEW OrderView
AS
SELECT *
FROM Orders
```

Altering view

```
ALTER VIEW orderView
AS
SELECT orderID,OrderDate
FROM Orders
```

Dropping view

```
DROP VIEW orderView
```