SQL (Structured Query Language) is used to perform operations on the records stored in the database or SQL statements are used to retrieve and update data in a database.,

Type of operation : such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.

to perform the queries of SQL language on the stored data in the database You are required to install any database management system in your systems like oracle,mysql,sqlserver,db2,postgresql,mangodb

why sql :

* The basic use of SQL for data professionals and SQL users is to insert, update, and delete the data from the relational database.
* SQL allows the data professionals and users to retrieve the data from the relational database management systems.
* It also helps them to describe the structured data.
* It allows SQL users to create, drop, and manipulate the database and its tables.
* It also helps in creating the view, stored procedure, and functions in the relational database.
* It allows you to define the data and modify that stored data in the relational database.
* It also allows SQL users to set the permissions or constraints on table columns, views, and stored procedures.

**difference bet sql and non sql**

Complex queries are easily managed in the SQL database\can’t handle complex query

 The schema of SQL databases is predefined, fixed, and static\ in non sql schema is non dynamic

SQL is a relational database management system\ non-sql is non relational dbms

These databases are vertically scalable\ horizontally mapped in non-sql

. The database type of SQL is in the form of tables, i.e., in the form of rows and columns.\ in file format,word,excel,notepad,notepad++ in non-sql

sql database is not the best choice for storing hierarchical data.\non-sql best for non hierarchical data

 SQLite, Ms-SQL, Oracle, PostgreSQL, and MySQL are examples of SQL database systems.\redis,mangodb are example of non-sql

Advantages of SQL

1. **No programming needed**
2. **High-Speed Query Processing**
3. **Standardized Language**
4. **Portability**
5. **Interactive language**
6. **More than one Data View**

## Disadvantages of SQL

1. **Cost**

**2. Interface is Complex**

**3. Partial Database control**

## Most Important SQL Commands and Statements

## Statements in sql

1. [Select Statement](https://www.javatpoint.com/sql-select)
2. [Update Statement](https://www.javatpoint.com/sql-update)
3. [Delete Statement](https://www.javatpoint.com/sql-delete)
4. [Create Table Statement](https://www.javatpoint.com/sql-create-table)
5. [Alter Table Statement](https://www.javatpoint.com/sql-alter-table)
6. [Drop Table Statement](https://www.javatpoint.com/sql-drop-table)
7. [Create Database Statement](https://www.javatpoint.com/sql-create-database)
8. [Drop Database Statement](https://www.javatpoint.com/sql-drop-database)
9. [Insert Into Statement](https://www.javatpoint.com/sql-insert)
10. [Truncate Table Statement](https://www.javatpoint.com/sql-truncate-table)
11. Describe Statement
12. [Distinct Clause](https://www.javatpoint.com/sql-select-distinct)
13. Commit Statement
14. Rollback Statement
15. Create Index Statement
16. Drop Index Statement
17. Use Statement

### **SELECT Statement :**

Used to access the records from one or more relational database table and views also retrieve the data that follow the condition we want.

we can also access the particular record from the particular column of the table

Syntax :

## SELECT Statement with WHERE clause

1. **SELECT** \* **FROM** Name\_of\_Table **WHERE** [condition];

## SQL SELECT Statement with GROUP BY clause

**SELECT** column\_Name\_1, column\_Name\_2, ....., column\_Name\_N aggregate\_function\_name(column\_Name2) **FROM** table\_name **GROUP** **BY** column\_Name1;

## SQL SELECT Statement with HAVING clause

**SELECT** column\_Name\_1, column\_Name\_2, ....., column\_Name\_N aggregate\_function\_name(column\_Name\_2) **FROM** table\_name **GROUP** **BY** column\_Name1 **HAVING** ;

## SELECT Statement with ORDER BY clause

1. **SELECT** Column\_Name\_1, Column\_Name\_2, ....., column\_Name\_N **FROM** table\_name **WHERE** [Condition] **ORDER** **BY**[column\_Name\_1, column\_Name\_2, ....., column\_Name\_N **asc** | **desc** ];

### **UPDATE Statement**

**UPDATE** table\_name

**SET** column\_name1 = new\_value\_1, column\_name2 = new\_value\_2, ...., column\_nameN = new\_value\_N

[ **WHERE**  CONDITION ];

|  |  |  |  |
| --- | --- | --- | --- |
| **Student\_Id** | **FirstName** | **LastName** | **User\_Name** |
| 1 | Ada | Sharma | sharmili |
| 2 | Rahul | Maurya | sofamous |
| 3 | James | Walker | jonny |

If I want to update username of ada as pk then update statement is as below

Update table\_name set user\_name=’pk’ where firstname=’ada’; updating on the basis of first name

Where id=1; updating on the basis of id

To update multiple column values

Update emp

Set fname=’a’, lname=’b’

Where id=1

If I want to update user name as baba for id 1 and 3 then update statement becomes

Update emp

Set username=’baba’

Where id in (1,3);

HOME WORK :IMPORTANT == SEE UPDATE WITH SELECT QUERY BRIFLY

HOW TO COPY COLUMN DATA OF ONEN TABLE TO ANOTER TABLE

UPDATE SECOND\_TABLE

SET LAST\_NAME = (SELECT LAST\_NAME FROM FIRST\_TABLE WHERE FIRST\_TABLE.STUDENT\_ID=SECOND\_TABLE.STUDENT\_ID);

### **DELETE Statement**

**DELETE** **FROM** table\_name

[ **WHERE** CONDITION ];

To delete specific row then we use where clause.

To delete all rows we write the query as

Delete from emp.

Imp : we cant use \* tp for deletion of all data

Delete \* from emp;

## Difference between DELETE and TRUNCATE statements

The **DELETE statement** only deletes the rows from the table based on the condition defined by WHERE clause or delete all the rows from the table when condition is not specified. But it does not free the space containing by the table.

The **TRUNCATE statement:** it is used to delete all the rows from the table **and free the containing space.**

Difference b/w DROP and TRUNCATE statements

When you use the drop statement it deletes the table's row together with the table's definition so all the relationships of that table with other tables will no longer be valid.

**When you drop a table:**

* Table structure will be dropped
* Relationship will be dropped
* Integrity constraints will be dropped
* Access privileges will also be dropped

On the other hand when we **TRUNCATE** a table, the table structure remains the same, so you will not face any of the above problems.

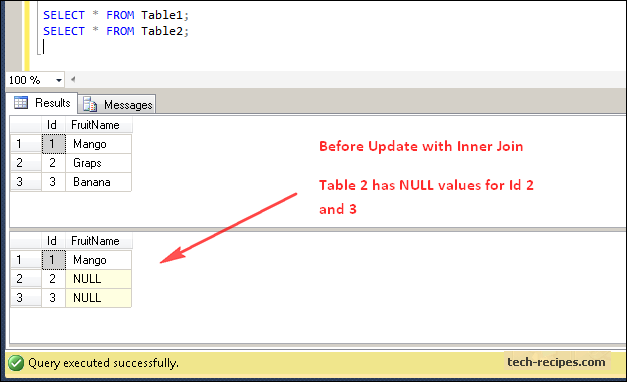
[**SQL DELETE Duplicate Rows**](https://www.javatpoint.com/sql-delete-duplicate-rows)

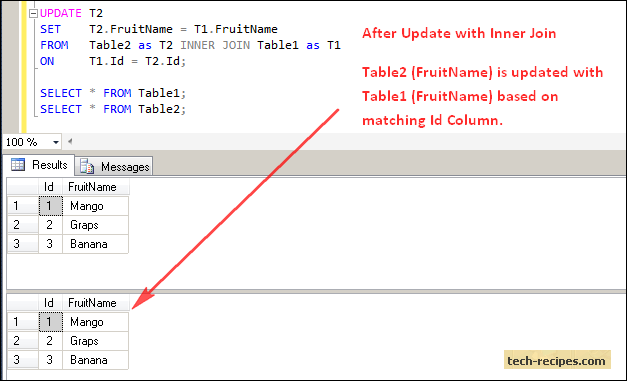
use distinct keyword to delete all the duplicate rows from the table

[**SQL DELETE DATABASE**](https://www.javatpoint.com/sql-delete-database)

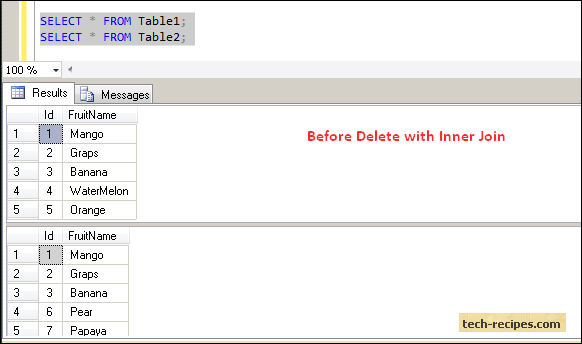
There is not used DELETE statement to delete the database. But, there is used DROP statement to delete the database.

**IMP QUE : commonly asked question that how to delete or update rows using join clause**

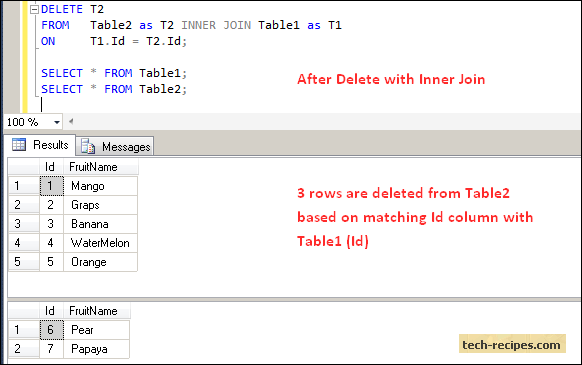
**Before Update with Inner Join**

**After Update with Inner Join**

**Before Delete with Inner Join**



**After Delete with Inner Join**



**What is SQL view?**

A view is a result set of a stored query on the data.

The SQL view is a table which does not physically exist. It is only a virtual table.

### **CREATE TABLE Statement**

**CREATE** **TABLE** table\_name

(

column\_name1 data\_type [column1 **constraint**(s)],

column\_name2 data\_type [column2 **constraint**(s)],

.....

.....,

column\_nameN data\_type [columnN **constraint**(s)],

**PRIMARY** **KEY**(one or more col)

);

### **ALTER TABLE Statement**

**ALTER** **TABLE** table\_name **ADD** column\_name datatype[(**size**)];

### **DROP TABLE Statement**

**DROP** **TABLE** [ IF EXISTS ]

table\_name1, table\_name2, ……, table\_nameN;

# **SQL Data Types: int ,varchar**

Data types mainly classified into three categories for every database.

* String Data types
* Numeric Data types
* Date and time Data types

**SQL Operator**

\*\* Exponentiation operator

+, - Identity operator, Negation operator

\*,/ Multiplication operator, Division operator

+, -, ||Addition (plus) operator, subtraction (minus) operator, String Concatenation operator

=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN : Comparison Operators

NOT : Logical negation operator

AND : conjunction operator

OR : Inclusion operator

## SQL Arithmetic Operators

1. SQL Addition Operator (+)
2. SQL Subtraction Operator (-)
3. SQL Multiplication Operator (+)
4. SQL Division Operator (-)
5. SQL Modulus Operator (+)

## SQL Comparison Operators

1. SQL Equal Operator (=)
2. SQL Not Equal Operator (!=)
3. SQL Greater Than Operator (>)
4. SQL Greater Than Equals to Operator (>=)
5. SQL Less Than Operator (<)
6. SQL Less Than Equals to Operator (<=)

## SQL Logical Operators

1. SQL ALL operator : The ALL operator in SQL compares the specified value to all the values of a column from the sub-query in the SQL database.
2. SQL AND operator(CONJUCTIVE OPERATOR) : The **AND operator** in SQL would show the record from the database table if all the conditions separated by the AND operator evaluated to True
3. SQL OR operator: The OR operator in SQL shows the record from the table if any of the conditions separated by the OR operator evaluates to True
4. SQL BETWEEN operator : The **BETWEEN operator** in SQL shows the record within the range mentioned in the SQL query.
5. SQL IN operator : The **IN operator** in SQL allows database users to specify two or more values in a WHERE clause**. This logical operator minimizes the requirement of multiple OR conditions.**
6. SQL NOT operator : The **NOT operator** in SQL shows the record from the table if the condition evaluates to false
7. SQL ANY operator
8. SQL LIKE operator : The **LIKE operator** in SQL shows those records from the table which match with the given pattern specified in the sub-query.

The **Set Operators** in SQL combine a similar type of data from two or more SQL database tables

1. SQL Union Operator
2. SQL Union ALL Operator
3. SQL Intersect Operator
4. SQL Minus Operator

### **SQL Union Operator**

The SQL Union Operator combines the result of two or more SELECT statements and provides the single output.

The data type and the number of columns must be the same for each SELECT statement used with the UNION operator. This operator does not show the duplicate records in the output table.

### **SQL Union ALL Operator**

The SQL Union Operator is the same as the UNION operator, but the only difference is that it also shows the same record.

### **SQL Intersect Operator**

The SQL Intersect Operator shows the common record from two or more SELECT statements. The data type and the number of columns must be the same for each SELECT statement used with the INTERSECT operator.

### **SQL ALL Operator**

The ALL operator in SQL compares the specified value to all the values of a column from the sub-query in the SQL database

### **SQL AND Operator**

The **AND operator** in SQL would show the record from the database table if all the conditions separated by the AND operator evaluated to True. It is also known as the conjunctive operator and is used with the WHERE clause.

### **SQL OR Operator**

The OR operator in SQL shows the record from the table if any of the conditions separated by the OR operator evaluates to True. It is also known as the conjunctive operator and is used with the WHERE clause.

### **SQL BETWEEN Operator**

The **BETWEEN operator** in SQL shows the record within the range mentioned in the SQL query. This operator operates on the numbers, characters, and date/time operands.

### **SQL IN Operator**

The **IN operator** in SQL allows database users to specify two or more values in a WHERE clause. This logical operator minimizes the requirement of multiple OR conditions.

### **SQL NOT Operator**

The **NOT operator** in SQL shows the record from the table if the condition evaluates to false. It is always used with the WHERE clause.

### **SQL LIKE Operator**

The **LIKE operator** in SQL shows those records from the table which match with the given pattern specified in the sub-query.

The percentage (%) sign is a wildcard which is used in conjunction with this logical operator.

### **SQL ANY Operator**

The **ANY operator** in SQL shows the records when any of the values returned by the sub-query meet the condition.

The ANY logical operator must match at least one record in the inner query and must be preceded by any SQL comparison operator.

## SQL Set Operators

1. SQL Union Operator
2. SQL Union ALL Operator
3. SQL Intersect Operator
4. SQL Minus Operator

### **SQL Union Operator**

The SQL Union Operator combines the result of two or more SELECT statements and provides the single output.

The data type and the number of columns must be the same for each SELECT statement used with the UNION operator. This operator does not show the duplicate records in the output table.

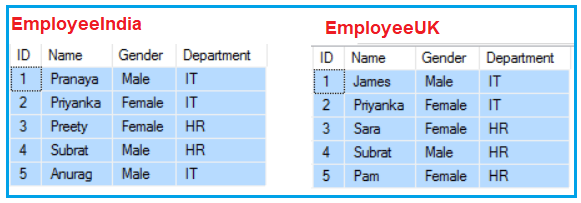
### **SQL Union ALL Operator**

The SQL Union Operator is the same as the UNION operator, but the only difference is that it also shows the same record.

### **SQL Intersect Operator**

The SQL Intersect Operator shows the common record from two or more SELECT statements. The data type and the number of columns must be the same for each SELECT statement used with the INTERSECT operator.

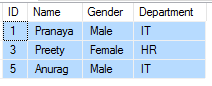
### **SQL Except Operator**



The following SQL Query will return the unique rows from the left query (the select statement before the EXCEPT operator) that is not present in the right query (the select statement after the EXCEPT operator).

**SELECT ID, Name, Gender, Department FROM EmployeeIndia**  
**EXCEPT**  
**SELECT ID, Name, Gender, Department FROM EmployeeUK**

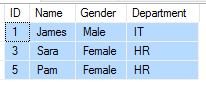
###### **Results:**



To retrieve all of the rows from Table EmployeeUK that do not exist in Table EmployeeIndia, reverse the two queries as shown below.

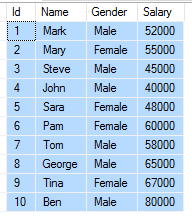
**SELECT ID, Name, Gender, Department FROM EmployeeUK**  
**EXCEPT**  
**SELECT ID, Name, Gender, Department FROM EmployeeIndia**

###### **Result:**

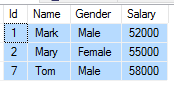


##### **EXCEPT Operator in SQL Server on a single table:**

You can also use the EXCEPT operator on a single table. Let’s use the following Employees table for this example.

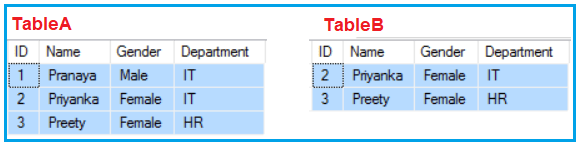


###### **Result:**



##### **Difference between EXCEPT and NOT IN Operator SQL server**

Now, you may have one question on your mind i.e. what is the difference between EXCEPT and NOT IN Operator in SQL Server. Let’s understand the difference between EXCEPT and NOT IN operators in SQL Server with one example. We are going to use the following two tables (TableA and TableB).



**Example: Using EXCEPT Operator**

**SELECT ID, Name, Gender, Department FROM TableA**  
**EXCEPT**  
**SELECT ID, Name, Gender, Department FROM TableB**

###### **Result:**

Using EXCEPT Operator in SQL Server

###### **Example: Using NOT IN Operator**

The same result can also be achieved using the NOT IN operator using the following query.

**SELECT ID, Name, Gender, Department FROM TableA Where ID NOT IN (Select ID from TableB)**

###### **Result:**

Using NOT IN Operator in SQL Server

##### **So, what is the difference between EXCEPT and NOT IN operators in SQL Server?**

The EXCEPT operator filters duplicate rows and return only DISTINCT rows from the left query that aren’t in the right query’s results, whereas NOT IN does not filter the duplicates rows. To understand this, insert the following row into TableA

**INSERT INTO TableA VALUES(1, ‘Pranaya’, ‘Male’,’IT’)**

Now execute the following EXCEPT query. Notice that we get only the DISTINCT rows in the result set.

**SELECT ID, Name, Gender, Department FROM TableA**  
**EXCEPT**  
**SELECT ID, Name, Gender, Department FROM TableB**

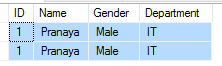
###### **Result:**

what is the difference between EXCEPT and NOT IN operators in SQL Server

Now execute the following query. Notice that the duplicate rows are not filtered in the result set.

**SELECT ID, Name, Gender, Department FROM TableA Where ID NOT IN (Select ID from TableB)**

###### **Result:**



The EXCEPT operator expects the same number of columns in both the queries, whereas NOT IN, compares a single column from the outer query with a single column from the subquery. In the following example, the number of columns is different.

**SELECT ID, Name, Gender, Department FROM TableA**  
**EXCEPT**  
**SELECT ID, Name, Gender FROM TableB**

**The above query would produce the following error.**

**Msg 205, Level 16, State 1, Line 1**

**All queries combined using a UNION, INTERSECT, or EXCEPT operator must have an equal number of expressions in their target lists.**

The NOT IN operator compares a single column from the outer query with a single column from the subquery. In the following example, the subquery returns multiple columns

**SELECT ID, Name, Gender, Department FROM TableA Where ID NOT IN (Select ID, Name from TableB)**

The above query would produce the following error.

**Msg 116, Level 16, State 1, Line 2**

**Only one expression can be specified in the select list when the subquery is not introduced with EXISTS.**

#### Problem:

You have duplicate rows in your table, with only the IDs being unique. How do you find those duplicate entries?

Solution :

SELECT  name,

 category,

FROM product

GROUP BY name, category

HAVING COUNT(id) >1;

#### **TRUNCATE TABLE Vs DROP TABLE**

Drop table command can also be used to delete complete table but it deletes table structure too. TRUNCATE TABLE doesn't delete the structure of the table.

#### **TRUNCATE TABLE Vs DELETE TABLE**

Truncate table is faster and uses lesser resources than DELETE TABLE command.

**Note:** The rollback process is not possible after truncate table statement. Once you truncate a table you cannot use a flashback table statement to retrieve the content of the table.

Actually, there is no difference between DISTINCT and UNIQUE.

NULL values are excluded from the count function

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

DIFFERENCE BETWEEN HAVING AND WHERE CLAUSE

1. The HAVING clause is used in database systems to fetch the data/values from the groups according to the given condition.

1. The WHERE clause is used in database systems to fetch the data/values from the tables according to the given condition.

2. The HAVING clause is always executed with the GROUP BY clause.

2. The WHERE clause can be executed without the GROUP BY clause.

3. The HAVING clause can include SQL aggregate functions in a query or statement.

3. We cannot use the SQL aggregate function with WHERE clause in statements.

4. We can only use SELECT statement with HAVING clause for filtering the records.

4. Whereas, we can easily use WHERE clause with UPDATE, DELETE, and SELECT statements.

5. The HAVING clause is used in SQL queries after the GROUP BY clause.

5. The WHERE clause is always used before the GROUP BY clause in SQL queries.

6. We can implements this SQL clause in column operations.

6. We can implements this SQL clause in row operations.

7. It is a post-filter.

7. It is a pre-filter.

8. It is used to filter groups.

8. It is used to filter the single record of the table.

We can make INNERJOIN to provide same result set like INTERSECT by using DISTINCT

Q) 4 th highest salary depat wise  
  
Select \* from (  
Select department,salary,dense \_ rank( ) over ( partition by department order by salary desc) as ranking from table\_ name ) as ranking = 4;

# **SQL COPY TABLE**

If you want to copy the data of one SQL table into another SQL table in the same SQL server, then it is possible by using the SELECT INTO statement in SQL.

The SELECT INTO statement in Structured Query Language copies the content from one existing table into the new table. SQL creates the new table by using the structure of the existing table.

Syntax : SELECT \* INTO New\_table\_name FROM old\_table\_name;

# **SQL TEMP TABLE**

The concept of temporary table is introduced by SQL server. It helps developers in many ways:

**Temporary tables** can be created at run-time and can do all kinds of operations that a normal table can do. These temporary tables are created inside tempdb database.

There are two types of temp tables based on the behavior and scope.

1. Local Temp Variable
2. Global Temp Variable

## Local Temp Variable

Local temp tables are only available at current connection time. It is automatically deleted when user disconnects from instances. It is started with hash (#) sign.

**CREATE** **TABLE** #**local** **temp** **table** (

User id **int**,

Username **varchar** (50),

User address **varchar** (150)

)

## Global Temp Variable

Global temp tables name starts with double hash (##). Once this table is created, it is like a permanent table. It is always ready for all users and not deleted until the total connection is withdrawn.

**CREATE** **TABLE** ##new **global** **temp** **table** (

User id **int**,

User **name** **varchar** (50),

User address **varchar** (150)

)