SQL | UPDATE Statement

The UPDATE statement in SQL is used to update the data of an existing table in database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

Basic Syntax

```
UPDATE table_name SET column1 = value1, column2 = value2,...
WHERE condition;

table_name: name of the table
column1: name of first , second, third column....
value1: new value for first, second, third column..

condition: condition to select the rows for which the
values of columns needs to be updated.
```

1	Ram	Delhi	XXXXXXXXXX	18
2	RAMESH	GURGAON	XXXXXXXXXX	18
3	PRATIK	ROHTAK	XXXXXXXXXX	20
4	SURESH	Delhi	XXXXXXXXXX	18
3	PRATIK	ROHTAK	XXXXXXXXXX	20
2	RAMESH	GURGAON	XXXXXXXXXX	18

NOTE: In the above query the **SET** statement is used to set new values to the particular column and the **WHERE** clause is used to select the rows for which the columns are needed to be updated. If we have not used the WHERE clause then the columns in **all** the rows will be updated. So the WHERE clause is used to choose the particular rows.

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	xxxxxxxxxx	18
2	RAMESH	GURGAON	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
4	SURESH	Delhi	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
2	RAMESH	GURGAON	xxxxxxxxxx	18

Example Queries

- **Updating single column**: Update the column NAME and set the value to ‘PRATIK’ in all the rows where Age is 20.

```
UPDATE Student SET NAME = 'PRATIK' WHERE Age = 20;
```

Output:

This query will update two rows(third row and fifth row) and the table **Student** will now look like,

Updating multiple columns: Update the columns NAME to ‘PRATIK’ and ADDRESS to ‘SIKKIM’ where ROLL_NO is 1.

```
UPDATE Student SET NAME = 'PRATIK', ADDRESS = 'SIKKIM' WHERE ROLL_NO = 1;
```

Output:

The above query will update two columns in the first row and the table **Student** will now look like,

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	PRATIK	SIKKIM	XXXXXXXXXX	18
2	RAMESH	GURGAON	XXXXXXXXXX	18
3	PRATIK	ROHTAK	XXXXXXXXXX	20
4	SURESH	Delhi	XXXXXXXXXX	18
3	PRATIK	ROHTAK	XXXXXXXXXX	20
2	RAMESH	GURGAON	XXXXXXXXXX	18

```
UPDATE Student SET NAME = 'PRATIK';
```

Output:

The table **Student** will now look like,

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	PRATIK	Delhi	XXXXXXXXXX	18
2	PRATIK	GURGAON	XXXXXXXXXX	18
3	PRATIK	ROHTAK	XXXXXXXXXX	20
4	PRATIK	Delhi	XXXXXXXXXX	18

ROLL_NO	NAME	ADDRESS	PHONE	Age
3	PRATIK	ROHTAK	XXXXXXXXXX	20
2	PRATIK	GURGAON	XXXXXXXXXX	18

SQL | SELECT TOP Clause

SELECT TOP clause is used to fetch limited number of rows from a database. This clause is very useful while dealing with large databases.

- **Basic Syntax:**
- **SELECT TOP value column1,column2 FROM table_name;**
- **value:** number of rows to return from top
- **column1 , column2:** fields in the table
- **table_name:** name of table

- **Syntax using Percent**

- **SELECT TOP value PERCENT column1,column2 FROM table_name;**
- **value:** percentage of number of rows to return from top
- **column1 , column2:** fields in the table
table_name: name of table

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	xxxxxxxxxx	18
2	RAMESH	GURGAON	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
4	SURESH	Delhi	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
2	RAMESH	GURGAON	xxxxxxxxxx	18

- **To fetch first two data set from Student table.**

```
SELECT TOP 2 * FROM Student;
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18

ROLL_NO	NAME	ADDRESS	PHONE	Age
2	RAMESH	GURGAON	XXXXXXXXXX	18

To fetch 50 percent of the total records from Student table.

```
SELECT TOP 50 PERCENT * FROM Student;
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
2	RAMESH	GURGAON	XXXXXXXXXX	18
3	SUJIT	ROHTAK	XXXXXXXXXX	20

SQL | ORDER BY

The ORDER BY statement in sql is used to sort the fetched data in either ascending or descending according to one or more columns.

- By default ORDER BY sorts the data in **ascending order.**
- We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.

Syntax of all ways of using ORDER BY is shown below:

- **Sort according to one column:** To sort in ascending or descending order we can use the keywords ASC or DESC respectively.
  Syntax:

```
SELECT * FROM table_name ORDER BY column_name ASC|DESC

table_name: name of the table.
column_name: name of the column according to which the data is needed to be arranged.
ASC: to sort the data in ascending order.
DESC: to sort the data in descending order.
| : use either ASC or DESC to sort in ascending or descending order
```

- **Sort according to multiple columns:** To sort in ascending or descending order we can use the keywords ASC or DESC respectively. To sort according to multiple columns, separate the names of columns by (,) operator.
  Syntax:

```
SELECT * FROM table_name ORDER BY column1 ASC|DESC , column2 ASC|DESC
```

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	XXXXXXXXXX	18
2	PRATIK	BIHAR	XXXXXXXXXX	19
3	RIYANKA	SILIGURI	XXXXXXXXXX	20
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19

Queries:

- **Sort according to single column**: In this example we will fetch all data from the table **Student** and sort the result in descending order according to the column **ROLL_NO**.

ROLL_NO	NAME	ADDRESS	PHONE	Age
8	NIRAJ	ALIPUR	XXXXXXXXXX	19
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
3	RIYANKA	SILIGURI	XXXXXXXXXX	20
2	PRATIK	BIHAR	XXXXXXXXXX	19
1	HARSH	DELHI	XXXXXXXXXX	18

```
SELECT * FROM Student ORDER BY ROLL_NO DESC;
```

Output:

- To sort in ascending order we have to use ASC in place of DESC.
- **Sort according to multiple columns**:In this example we will fetch all data from the table **Student** and then sort the result in ascending order first according to the column **Age**. and then in descending order according to the column **ROLL_NO**.
- `SELECT * FROM Student ORDER BY Age ASC , ROLL_NO DESC;`

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
7	ROHIT	BALURGHAT	XXXXXXXXXX	18

ROLL_NO	NAME	ADDRESS	PHONE	Age
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
1	HARSH	DELHI	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
2	PRATIK	BIHAR	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
3	RIYANKA	SILIGURI	XXXXXXXXXX	20

- In the above output you can see that first the result is sorted in ascending order according to Age.

  There are multiple rows having same Age. Now, sorting further this result-set according to ROLL_NO will sort the rows with same Age according to ROLL_NO in descending order.

- **Note that:** ASC is the default value for ORDER BY clause. So, if we don't specify anything after column name in ORDER BY clause, the output will be sorted in ascending order by default.

  Example: The following query will give similar output as the above:

```
SELECT * FROM Student ORDER BY Age , ROLL_NO DESC;
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
1	HARSH	DELHI	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
2	PRATIK	BIHAR	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
3	RIYANKA	SILIGURI	XXXXXXXXXX	20