UPDATE, DELETE, ALTER COMMAND

Update Query:

The UPDATE statement updates data in the existing table. It allows you to change the values in one or more columns of a single row or multiple rows.

Syntax of the UPDATE statement is shown below:

UPDATE table_name **SET** column_name = new_value [WHERE condition];

HERE

- UPDATE table_name is the command that tells MySQL to update the data in a table .
- SET column_name = new_value are the names and values of the fields to be affected by the update query.
- [WHERE condition] is optional and can be used to put a filter that restricts the number of rows affected by the UPDATE MySQL query.

Let's considered the student table below to understand the update query

student

s_id	name	age	grade
1	John	21	72
2	Mike	20	68
3	Kate	22	86
4	Andy	21	94

Scenario 1: Update all the students grade to 90 with the students grade greater than 80

```
UPDATE student SET grade = 90 WHERE grade > 80;
```

You can verify weather grade updated for all the students with grade > 80 by executing the query below

```
SELECT * FROM student;
```

Output:

student

s_id	name	age	grade
1	John	21	72
2	Mike	20	68
3	Kate	22	90
4	Andy	21	90

If you observe the table kate and andy grade is updated with grade = 90

Scenario 2: Update all the students grade to 70 with the students grade equal to 68

```
UPDATE student SET grade = 70 WHERE (grade = 68);
```

You can verify weather grade updated for all the students with grade = 68 by executing the query below

```
SELECT * FROM student;
```

Output:

student

s_id	name	age	grade

1	John	21	72
2	Mike	20	68
3	Kate	22	90
4	Andy	21	90

If you observe the table kate grade is updated with grade = 68

Scenario 3: Update all the students grade to 90 with the student id equal to 1

```
UPDATE student SET grade = 90 WHERE (s_id = 1);
```

You can verify weather grade updated for all the students with student id is 1 by executing the query below

```
SELECT * FROM student;
```

Output:

student

s_id	name	age	grade
1	John	21	90
2	Mike	20	68
3	Kate	22	90
4	Andy	21	90

If you observe the table John grade is updated with student id is 1

Now let's try to see how to update multiple columns and understand how it works

Scenario 4: Update the students name = 'Alex' and age = 23 with the student id equal to 3

```
UPDATE student SET name = 'Alex', age = 23 WHERE (s_id = 3);
```

You can verify weather name and grade updated for all the students with student id is 3 by executing the query below

```
SELECT * FROM student;
```

Output:

student

s_id	name	age	grade
1	John	21	90
2	Mike	20	68
3	Alex	23	90
4	Andy	21	90

If you observe the table name and age is updated with student id 3.

DELETE Query

The **DELETE** statement is used to delete existing records in a table.

Syntax of the **DELETE** statement is shown below:
DELETE FROM table_name **WHERE** condition;

Consider the table given below:

student

s_id	name	age	grade
1	John	21	90
2	Mike	20	68
3	Alex	23	90
4	Andy	21	90

Scenario 1: DELETE the student with student id 4

```
DELETE FROM student WHERE s_id = 4;
```

You can verify weather student with student id is 3 deleted by executing the query below

```
SELECT * FROM student;
```

Output:

student

s_id	name	age	grade
1	John	21	90

2	Mike	20	68
3	Alex	23	90

If you observe the table there is no row with student id 4

Scenario 2: DELETE the student with student id 3 and age 22

```
DELETE FROM student WHERE s_id = 3 AND age = 22;
```

You can verify weather student with student id is 3 and age 22 is deleted by executing the query below

```
SELECT * FROM student;
```

Output:

student

s_id	name	age	grade
1	John	21	90
2	Mike	20	68
3	Alex	23	90

If you observe there was no record which match both student id and age so no row affected

ALTER QUERY

The **ALTER TABLE** statement is used to add, delete, or modify columns in an existing table.

Syntax of the **ALTER** statement is shown below:

ALTER TABLE table_name **ADD** column_name datatype;

Consider the table given below:

student

s_id	name	age	grade
1	John	21	90
2	Mike	20	68
3	Alex	23	90

Scenario 1: Alter the student table by deleting the column grade

```
ALTER TABLE student DROP grade;
```

You can verify weather grade column is deleted by executing the query below

```
SELECT * FROM student;
```

Output:

student

s_id	name	age
1	John	21
2	Mike	20

3	Alex	23

If you observe the table there is no column grade

Scenario 1: Write a query to delete a particular value present in the table

```
UPDATE student SET age = NULL WHERE s_id = 2;
```

You can verify weather age is updated with null by executing the query below

```
SELECT * FROM student;
```

Output:

student

s_id	name	age
1	John	21
2	Mike	Null
3	Alex	23

If you observe the table age of mike with s_id is 2 is updated

Let us understand the difference between **DELETE, DROP and TRUNCATE**

DELETE Table:

DELETE is a DML (Data Manipulation Language) command. This command removes records from a table. It is used only for deleting data from a table, not to remove the table from the database.

You can delete **all records** with the syntax:

```
DELETE FROM table_name;
```

Or you can delete a **group of records** using the WHERE clause:

```
DELETE FROM table_name WHERE col=value;
```

TRUNCATE Table:

TRUNCATE TABLE is similar to **DELETE**, but this operation is a DDL (Data Definition Language) command. It also deletes records from a table without removing table structure, but it doesn't use the WHERE clause.

Here's the syntax:

```
TRUNCATE TABLE table_name;
```

TRUNCATE is faster than **DELETE**, as it doesn't scan every record before removing it. TRUNCATE TABLE locks the whole table to remove data from a table; thus, this command also uses less transaction space than DELETE.

DROP Table:

The **DROP TABLE** is another DDL (Data Definition Language) operation. But it is not used for simply removing data from a table; it deletes the table structure from the database, along with any data stored in the table.

Here is the syntax of this command:

```
DROP TABLE table_name;
```