DATA MANIPULATION LANGUAGE

IOTA ACADEMY

DML

- DML is a short name for Data Manipulation Language that deals with **data manipulation**, and includes the most common SQL statements such as **INSERT, UPDATE, DELETE** etc.
- DML statements are used for **managing data** in the database.
- DML commands are **not auto-committed,** It means changes made by the DML commands are **not permanent** to the database, they can be **rolled back**.
- It is used to **store, modify, delete and update data** in the database.

DML COMMANDS

Common DML Commands in SQL are:

S.No	Command	Description
1	**INSERT**	It is used to **insert data** into a table
2	**UPDATE**	It is used to **update existing data** within a table
3	**DELETE**	It is used to **delete records** from a database table

1) INSERT COMMAND

- INSERT command is used to **insert data** into a table.
- Consider a table **iota_student** with the following fields.

student_id	student_name	student_age

INSERT COMMAND…

- The below command will insert a new record into **iota_student** table.

Syntax :	Example :

INSERT INTO table_name
VALUES
(data1, data2, data3);
	INSERT INTO iota_student
	VALUES
	(100, 'Akash', 25);

- **Output(Result set)**:

student_id	student_name	student_age
100	Akash	25

INSERT: MULTIPLE VALUES

- We can insert **multiple values** with the help of insert command.

Syntax :	Example :

INSERT INTO table_name
VALUES
(data1, data1, data1),
(data2, data2, data2),
(data3, data3, data3);
	INSERT INTO iota_student
	VALUES
	(101, 'Anjali', 27),
	(102, 'Rahul', 25),
	(103, 'Rohit', 21);

- **Output**:

student_id	student_name	student_age
100	Akash	25
101	Anjali	27
102	Rahul	25
103	Rohit	21

INSERT COMMAND…

Insert value in only specific column:

- We **can specify only those column names** along with INSERT INTO statement where we want to insert values and then provide data values accordingly:

Syntax :	Example :
INSERT INTO table_name (column1, column2) **VALUES**(data1,data…);	**INSERT INTO** iota_student (student_id, student_name) **VALUES**(104, 'Anuj');

- **Output**:

student_id	student_name	student_age
continued…		
104	Anuj	*NULL*

INSERT COMMAND…

Insert **NULL value** to a column:

- Inserting record with a NULL Value in a table.

Example :

```
INSERT INTO iota_student
VALUES(105, 'Ankita', NULL);
```

- **Output**:

student_id	student_name	student_age
continued…		
104	Anuj	*NULL*
105	Ankita	*NULL*

INSERT COMMAND…

Insert **Default** value to a column:

Syntax :	Example :

INSERT INTO table_name
VALUES (data1, data2, **DEFAULT**) ;
	INSERT INTO iota_student
	VALUES (106, 'Bhuvan', **DEFAULT);**

- **Output**:

student_id	student_name	student_age
continued…		
105	Ankita	*NULL*
106	Bhuvan	*Default Value*

2) UPDATE COMMAND

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

In this syntax:

- First, specify the name of the **table** that you want to update data after the **UPDATE** keyword.
- Second, specify which **column** you want to update and the new value in the **SET** clause.
- Third, specify which rows to be updated using a **condition** in the **WHERE** clause.

NOTE:

- The WHERE clause is optional. If you omit it, the UPDATE statement will modify all rows in the table.
- You may encounter an error if you write SQL statements **without the WHERE clause** here if the **SAFE UPDATE MODE** is on in your MYSQL Workbench. You can turn it off using the following command.
  SET SQL_SAFE_UPDATES = 0;

UPDATE COMMAND…

Syntax :	Example :

UPDATE table_name
SET column_name = new_value
WHERE some_condition ;
	UPDATE iota_student
	SET student_age = 29
	WHERE student_id = 102;

student_id	student_name	student_age
100	Akash	25
101	Anjali	27
102	Rahul	29
Continued…		

In the above statement, if we do not use the WHERE clause, then our update query will update age for all the records of the table to **29**.

UPDATE COMMAND…

Updating Multiple Columns:

To update values in multiple columns, you use a list of comma-separated assignments by supplying a value in each column’s assignment in the form of a literal value, an expression, or a subquery.

Example :

```
UPDATE iota_student
SET student_age = 23, student_name = 'Karan'
WHERE student_id = 101;
```

- **Output:**

student_id	student_name	student_age
100	Akash	25
101	Karan	23
Continued…		

UPDATE COMMAND…

- When we have to update any integer value in a table, then we can fetch and update the value in the table in a single statement.
- For example, if we have age column and want to update the age column of the **iota_student** table every year for every student, then we can simply run the following UPDATE statement to perform the operation:

```
UPDATE iota_student SET student_age = student_age+1;
```

- As you can see, we have used age = age + 1 to increment the value of age by 1.
- **NOTE:** You may encounter an error if you write SQL statements **without a WHERE clause** here **if SAFE UPDATE MODE** is on in your MYSQL Workbench. You can turn it off using the following command `SET SQL_SAFE_UPDATES = 0;` (Value = 1, to turn it again).

DELETE COMMAND

- DELETE command is used to **delete data** from a table.

Syntax :	**Example :**
DELETE FROM table_name;	**DELETE FROM** iota_student;

- The above command will **delete all the records** from the table **iota_student**.

DELETE COMMAND…

Deleting a particular Record from a Table

In our **iota_student** table if we want to delete a single record, we can use the WHERE clause to provide a condition in our DELETE statement.

Syntax :	**Example :**

DELETE FROM student
WHERE column_name=value;
	DELETE FROM iota_student
	WHERE student_id=103;

The above command will delete the record where **student**_id is **103** from the table.

DELETE, DROP AND TRUNCATE

DELETE:

Basically, it is a DML Command. With the help of the "DELETE" command, we can either delete all the rows in one go or can delete rows one by one. i.e., as per the requirement or the condition using the Where clause.

DROP:

It is a DDL Command. With the help of the "DROP" command, we can drop (delete) the whole structure in one go. By using this command the existence of the whole table is finished or say lost.

TRUNCATE:

It is also a DDL Command. It is used to delete all the rows of a relation (table) in one go. With the help of the "TRUNCATE" command, we can't delete the single row as here WHERE clause is not used. By using this command the existence of all the rows of the table is lost.

COMPARISON TABLE

	DELETE Command	DROP Command	TRUNCATE Command
Language	The DELETE command is **DML** Command.	The DROP command is **DDL** Command.	The TRUNCATE command is a **DDL** Command.
Use	**deletes one or more** existing records from the table in the database.	**drops the complete table** from the database.	**deletes all the rows** from the existing table, leaving the row with the column names.
Transition	We **can restore any deleted row** or multiple rows from the database using the ROLLBACK command.	By default, it **can not be reverted** but there are special cases when this is possible.	By default, it **can not be reverted** but there are special cases when this is possible.
Memory Space	**does not free** the allocated space of the table from memory.	**removes the space allocated** for the table from memory.	**does not free the space** allocated for the table from memory.
Performance Speed	Performs **slower than the DROP command and TRUNCATE command** as it deletes one or more rows based on a specific condition.	faster performance than DELETE Command but not as compared to the Truncate Command because the DROP command deletes the table from the database after deleting the rows.	works **faster than the DROP command and DELETE command** because it deletes all the records from the table without any condition.

AUTO INCREMENT FEATURE

- AUTO_INCREMENT keyword allows the user to create a unique number to get generated when a new record is inserted into a table.
- Often this is the primary key field, that we would like to be created automatically every time a new record is inserted.

NOTE:

- Only a KEY column can be made an auto_increment column and there can only be one auto_increment column in a table.
- One can initialise the Autoincrement value by altering the table as given in Example 2.

Example 1:

```
CREATE TABLE iota_student
( student_id INT PRIMARY KEY AUTO_INCREMENT,
student_name VARCHAR(255) );
```

Example 2:

```
ALTER TABLE iota_student
AUTO_INCREMENT = 100;
```

AUTO INCREMENT FEATURE WITH TRUNCATE AND DELETE COMMAND

- TRUNCATE command is different from DELETE command.

The delete command will ***delete all the rows*** from a table whereas truncate command not only deletes all the records stored in the table, but it also ***re-initializes the table*** (like a newly created table).

For example: If you have a table with 10 rows and an **auto increment** primary key, and if you use DELETE command to delete all the rows, it will delete all the rows, but will not re-initialize the primary key, hence if you will insert any row after using the DELETE command, the auto increment primary key will start from 11. But in case of TRUNCATE command, primary key is re-initialized, and it will again start from 1.

THANK YOU