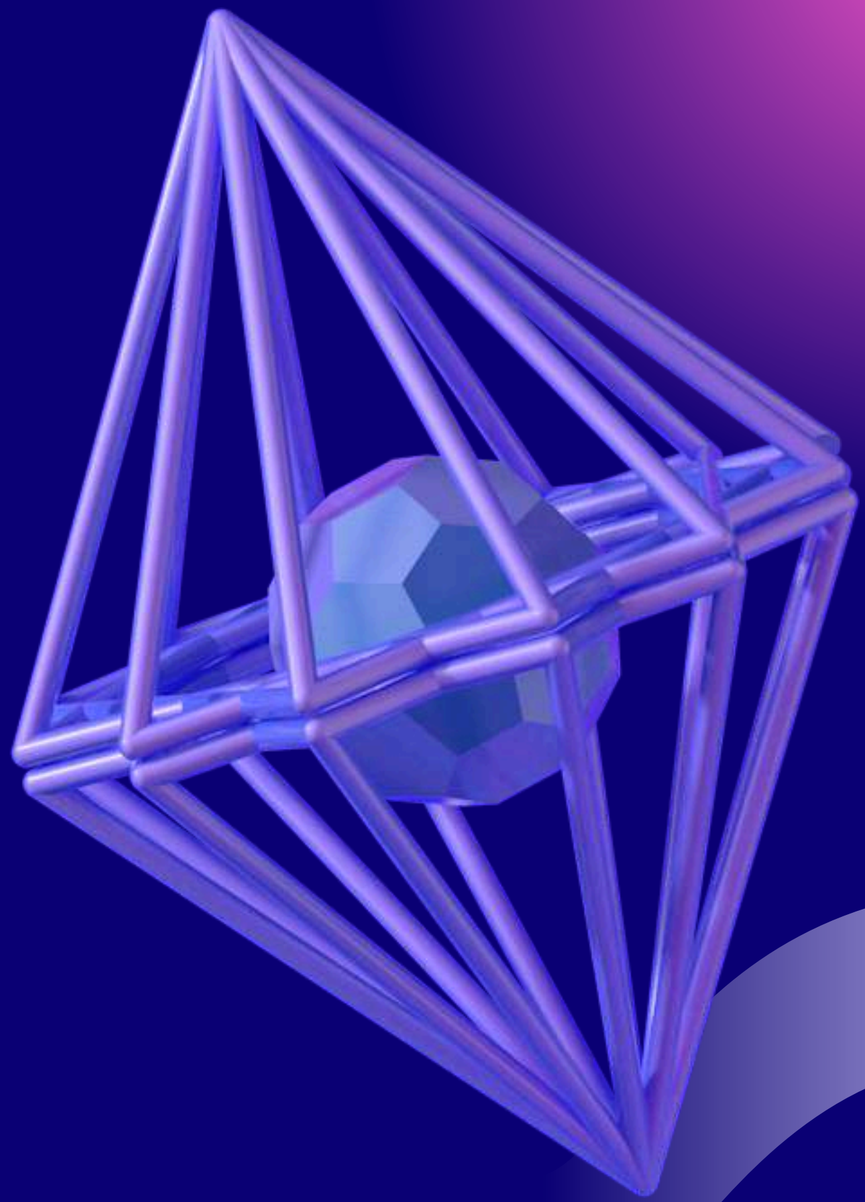
INTRODUCTION TO SQL

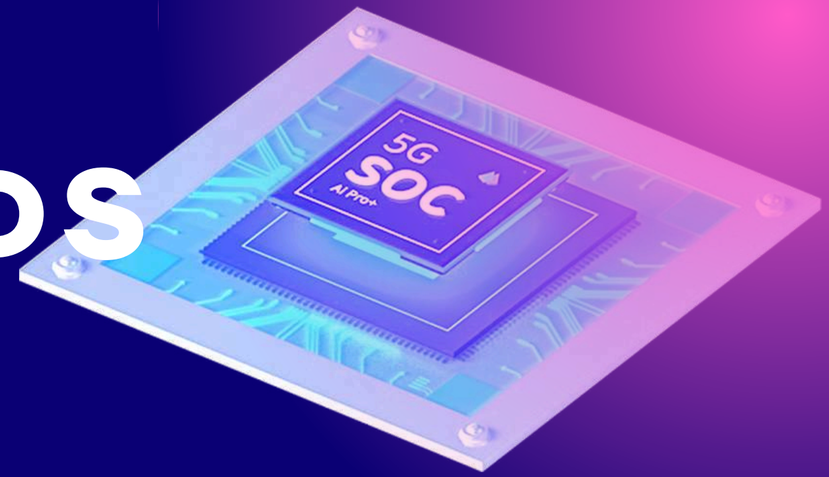
TEAM MAY DATING

ABOUT SQL

SQL (Structured Query Language) is a standard language used to manage and manipulate relational databases. It allows users to store, retrieve, update, and delete data within a database.



SQL COMMANDS



DATA DEFINITION LANGUAGE

DATA MANIPULATION LANGUAGE

DATA QUERY LANGUAGE

DATA CONTROL LANGUAGE

DATA DEFINITION LANGUAGE

SQL DDL, or Data Definition Language, comprises a set of SQL commands used to define, modify, and delete the structure of database objects. These commands are essential for managing the schema of a database, including tables, views, indexes, and other database components.

Advantages of DDL

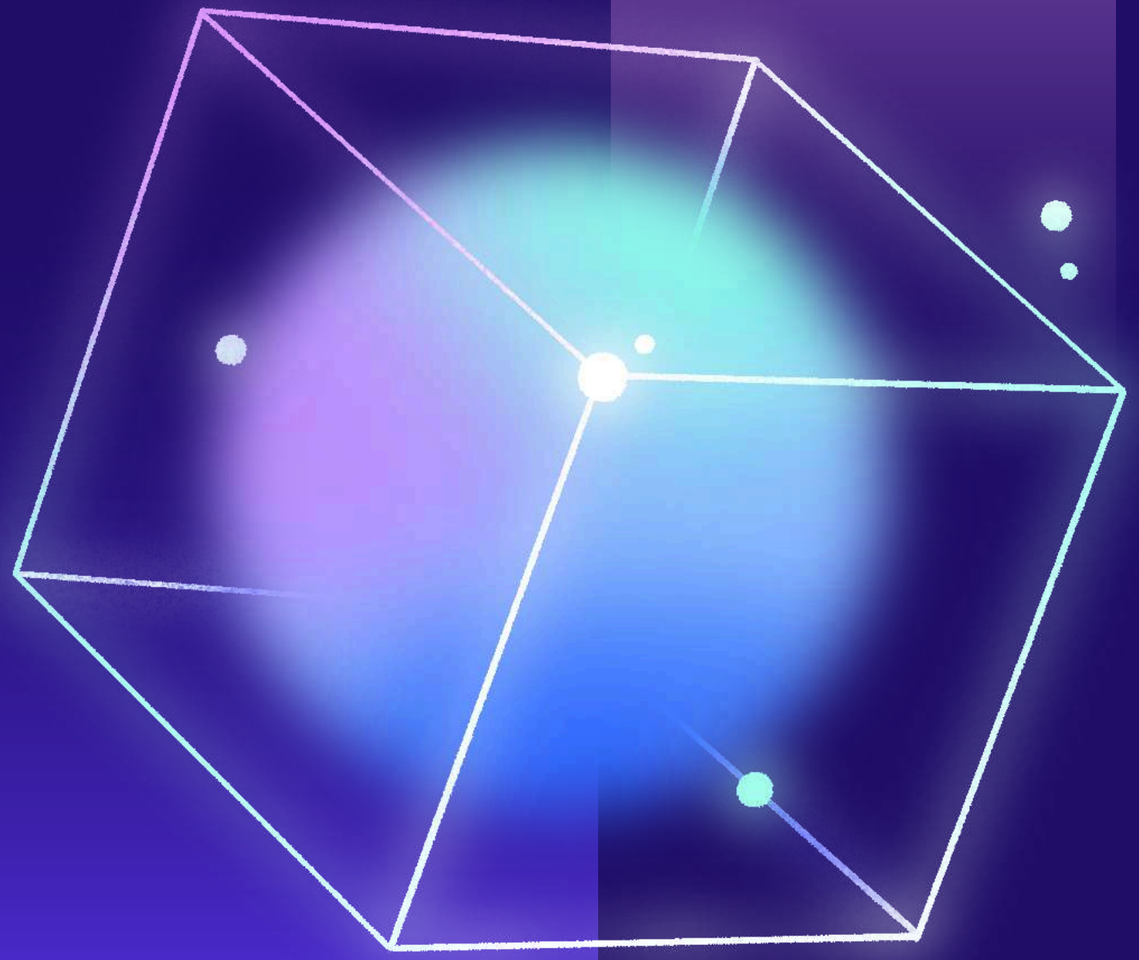
- **Defines Structure**: DDL commands like CREATE and ALTER allow you to set up and modify the structure of database objects (tables, indexes, views), organizing data effectively.
- **Manages Schema**: You can easily create, update, or remove entire database schemas, making it simpler to maintain and update the database structure over time.
- **Enforces Data Integrity**: DDL allows the implementation of rules and constraints (PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL) that ensure data remains accurate and consistent.

- **Boosts Performance**: By creating indexes and partitions using DDL commands, you can significantly improve query performance and speed up data retrieval
- **Standardized**: DDL commands are standardized across most SQL-based databases, making it easier to apply knowledge across different database systems.

Disadvantages of DDL

- **Irreversible Changes**: DDL commands, especially DROP and TRUNCATE, are auto-committed, meaning changes can't be undone once executed. This can result in permanent data loss if not used carefully.
- **Risk of Data Loss**: Misuse of commands like DROP can delete entire tables along with their data, leading to significant data loss, particularly if no backup is available.
- **Complex for Large Databases**: Altering the structure of large databases can be complicated and may require downtime or additional planning to avoid disrupting operations.

- **Causes Locking**: DDL operations can lock database objects during execution, which might slow down or block other operations, impacting overall database performance.
- **Compatibility Issues**: While DDL is standardized, different databases (like MySQL, Oracle, or PostgreSQL) may have variations in syntax and features, leading to compatibility issues during migration.



DDL COMMANDS

- CREATE
- ALTER
- RENAME
- COMMENT
- TRUNCATE
- DROP

CREATE

The CREATE command in SQL is used to create new database objects, such as databases, tables, views, or indexes. It defines the structure of the object — including the column names, data types, and constraints.

TYPES OF CREATE STATEMENT

CREATE DATABASE - Makes a new database

CREATE TABLE - Makes a new table

CREATE VIEW - Makes a virtual table (query-based)

CREATE INDEX - Speeds up data searches

CREATE USER - Creates a database user

CREATE ROLE - Creates a role for access control

CREATE FUNCTION - Creates a custom function

CREATE PROCEDURE - Creates a stored procedure

CREATE TRIGGER - Defines an automatic database event

CREATE SCHEMA - Creates a logical structure for organizing objects

CREATE DATABASE:

```
CREATE DATABASE DaTechHub;
use DaTechHub;
```

OUTPUT:

CREATE TABLE:

```sql
CREATE TABLE Customers (
        customer_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    address VARCHAR(200)
);
```

```sql
INSERT INTO Customers (customer_name, phone, email, address)
VALUES
('Jose Rizal','09918203345','JRizal@gmail.com','Laguna Province'),
('Macario Sakay','09817209334','Sakay@gmail.com','Manila City'),
('Andres Bonifacio','09810229345','MrKatipunan@gmail.com','Manila City'),
('Emilio Aguinaldo','09920193305','ElPresidente@gmail.com','Kawit City'),
('Gabriela Silang','09019203345','SilangG@gmail.com','Santa City'),
('Antonio Luna','09102933045','MrKiller@gmail.com','Manila City');
```

OUTPUT:

customer_id	customer_name	phone	email	address
1	Jose Rizal	09918203345	JRizal@gmail.com	Laguna Province
2	Macario Sakay	09817209334	Sakay@gmail.com	Manila City
3	Andres Bonifacio	09810229345	MrKatipunan@gmail.com	Manila City
4	Emilio Aguinaldo	09920193305	ElPresidente@gmail.com	Kawit City
5	Gabriela Silang	09019203345	SilangG@gmail.com	Santa City
6	Antonio Luna	09102933045	MrKiller@gmail.com	Manila City
NULL	NULL	NULL	NULL	NULL

ALTER (ALTER TABLE)

SQL ALTER TABLE statement modify the structure of an existing table in a database. Whether adding new columns, modifying existing ones, deleting columns or renaming them. ALTER TABLE statement enables you to make changes without losing data stored in the table.

SYNTAX

```
ALTER TABLE table_name [ADD | DROP | MODIFY]
column_name datatype;
```

COMMON USES FOR ALTER

ADD

The ADD clause is used to add a new column to an existing table. You must specify the name of the new column and its data type

MODIFY

The MODIFY (or ALTER COLUMN in some databases like SQL Server) clause is used to modify the definition of an existing column, such as changing its data type or size.

DROP

The DROP clause allows you to remove a column from a table. Be cautious when using this command as it will permanently remove the column and its data.

RENAME

We can rename an existing column using RENAME COLUMN clause. This allows you to change the name of a column while preserving its data type and content.

ALTER (ADD):

```sql
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    address VARCHAR(200)
);

ALTER TABLE Customers
ADD address1 VARCHAR(200);
```

OUTPUT:

Result Grid | Filter Rows: | Edit:

	customer_id	customer_name	phone	email	address	address1
*	NULL	NULL	NULL	NULL	NULL	NULL

ALTER (MODIFY):

```sql
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    address VARCHAR(200)
);

ALTER TABLE Customers
RENAME COLUMN address1 to address2;

ALTER TABLE Customers
MODIFY COLUMN customer_name VARCHAR(200);
```

OUTPUT:

```
43  14:57:28  ALTER TABLE Customers MODIFY COLUMN customer_name VARCHAR(200)
```

ALTER (RENAME):

```
24 •  CREATE TABLE Customers (
25        customer_id INT PRIMARY KEY AUTO_INCREMENT,
26        customer_name VARCHAR(100),
27        phone VARCHAR(20),
28        email VARCHAR(100),
29        address VARCHAR(200)
30    );
31
32 •  ALTER TABLE Customers
33    RENAME COLUMN address1 to address2;
```

OUTPUT:

	customer_id	customer_name	phone	email	address	address2
•	NULL	NULL	NULL	NULL	NULL	NULL

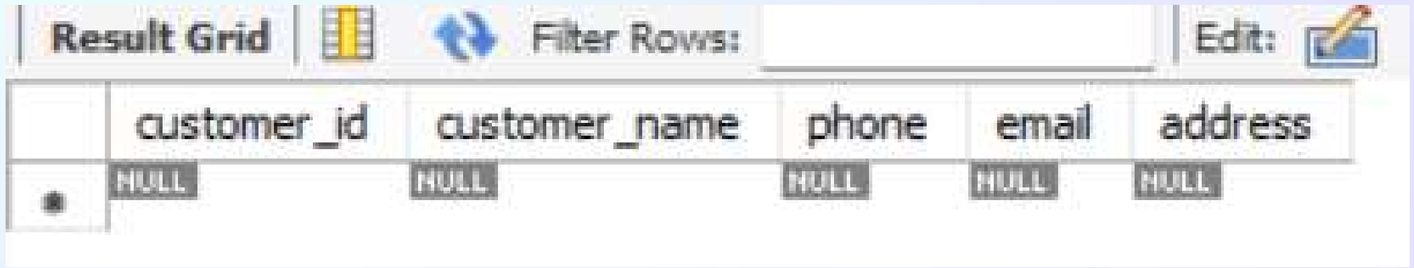
ALTER (DROP):

```sql
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    address VARCHAR(200)
);

ALTER TABLE Customers
RENAME COLUMN address1 to address2;

ALTER TABLE Customers
DROP COLUMN address2;
```

OUTPUT:



RENAME

is a simple yet powerful command that allows you to change the name of an existing table in your database. This can be useful for various reasons, such as updating table names to better reflect their content or restructuring your database without losing any data.

By using RENAME TABLE, you can easily and quickly rename one or more tables in a single command, ensuring that your database remains organized and up-to-date.

SYNTAX:

```sql
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    address VARCHAR(200)
);

RENAME TABLE Customers to VIP;
```

OUTPUT:

COMMENT/S

are annotations in our SQL code that are not executed by the database engine. They serve as notes or explanations for human readers, making it easier to understand and maintain the code. Whether we're explaining complex logic, providing context for a query, or temporarily disabling a part of our code, comments are indispensable tool for any developer.

3 TYPES OF COMMENTS

SINGLE

Single-line comments are used to annotate a single line of SQL code. They are often employed for brief explanations or to temporarily disable lines of code without removing them. Single-line comments begin with two hyphens (--), and the comment continues until the end of the line.

MULTI-LINE

are used when we need to comment out more than one line of SQL code. These comments are enclosed between /* and */ and can span multiple lines. They are typically used for longer explanations or to disable larger sections of code temporarily.

IN-LINE

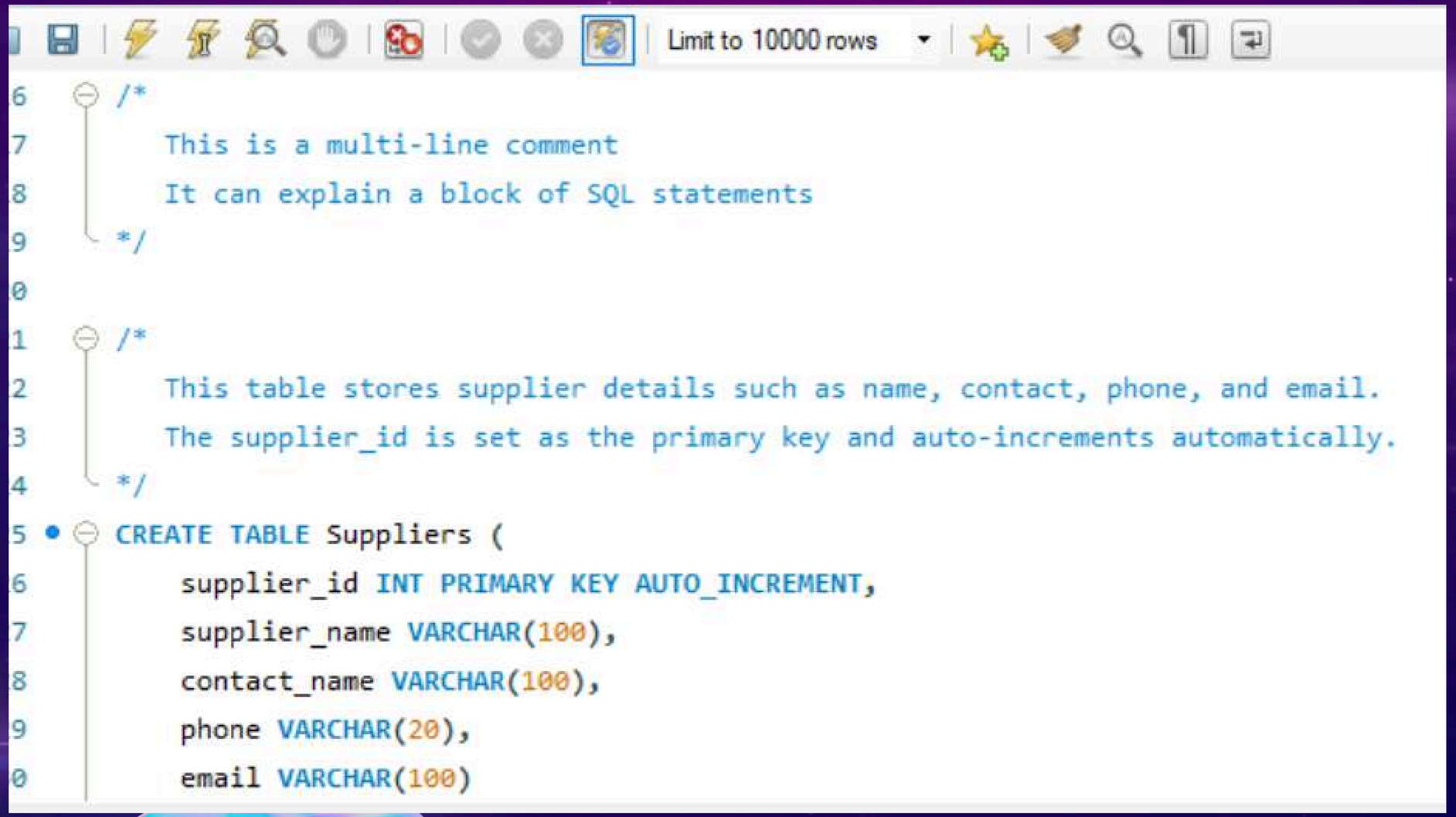
In-line comments allow us to add comments within a query itself. They are typically used to provide additional information or explanations for specific parts of the query, without interrupting the flow of the SQL statement. In-line comments start with /* and end with */.

SINGLE LINE COMMENT

```sql
-- This is an example of single line comment
-- It is used to explain one line of code

-- Creating a table for supplier information
CREATE TABLE Suppliers (
    supplier_id INT PRIMARY KEY AUTO_INCREMENT,
    supplier_name VARCHAR(100),
    contact_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100)
);
```

MULTI-LINE COMMENT



IN-LINE COMMENT

```
16  CREATE TABLE Suppliers (
17      supplier_id INT PRIMARY KEY AUTO_INCREMENT, -- Unique id foor each supplier
18      supplier_name VARCHAR(100),                 -- Supplier's company name
19      contact_name VARCHAR(100),                  -- Person to contact
20      phone VARCHAR(20),                          -- Supplier phone number
21      email VARCHAR(100)                          -- Supplier email add
```

DROP

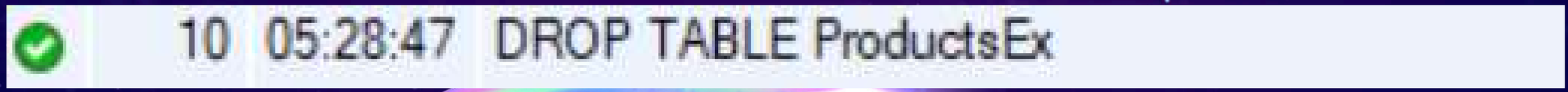
is used to permanently remove an object from a database, such as a table, database, index or view. When a table is dropped, both its data and structure are permanently removed, and the action cannot be undone without a backup.

- Permanent Removal: Deletes object and data permanently.
- Deletes Dependencies: Removes related constraints and indexes.
- Schema Deleted: Removes table structure and schema.
- No Integrity Check: Skips constraints and triggers.

SYNTAX:

```
4 •  CREATE TABLE ProductsEx (
5        product_id INT PRIMARY KEY AUTO_INCREMENT);
6
7 •  DROP TABLE ProductsEx;
```

OUTPUT:



DROP - FK SITUATION

```sql
CREATE TABLE Suppliers (
    supplier_id INT PRIMARY KEY AUTO_INCREMENT, -- Unique id foor each supplier
    supplier_name VARCHAR(100),                 -- Supplier's company name
    contact_name VARCHAR(100),                  -- Person to contact
    phone VARCHAR(20),                          -- Supplier phone number
    email VARCHAR(100)                          -- Supplier email add
);

-- DROP completely deletes the table and its structure from the database
DROP TABLE Suppliers;
```

14 05:35:46 DROP TABLE Suppliers

Error Code: 3730. Cannot drop table 'suppliers' referenced by a foreign key constrai...

WHAT THE HELLY?

EXPLANATION:

- While testing the DROP TABLE Suppliers; command, an error appeared:
- Error Code 3730: Cannot drop table 'suppliers' referenced by a foreign key constraint 'fk_Suppliers' on table 'products'.
- This means another table (Products) is dependent on Suppliers through a foreign key relationship.
- MySQL prevents dropping tables that are still referenced by other tables to protect data consistency.

HOW TO FIX AND DROP SAFELY

Option 1: Drop the child table first

Since Products depends on Suppliers, drop it first:

- DROP TABLE Orders; -- Orders depends on Products and Customers
- DROP TABLE Products; -- Products depends on Suppliers
- DROP TABLE Suppliers; -- Now safe to drop

WHAT THE HELLY? WHY WOULD WE DROP EVERYTHING YUNGSTUNNA?

HOW TO FIX AND DROP SAFELY

Option 2: Remove the foreign key constraint first

If you only want to drop Suppliers, remove its foreign key from Products first:

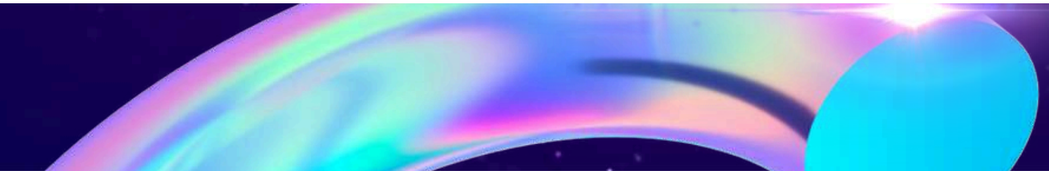
- ALTER TABLE Products DROP FOREIGN KEY fk_Suppliers;
- DROP TABLE Suppliers;

SYNTAX:

```
-- DROP completely deletes the table and its structure
ALTER TABLE Products DROP FOREIGN KEY fk_Suppliers;
DROP TABLE Suppliers;
```

OUTPUT:

```
15  05:48:38  ALTER TABLE Products DROP FOREIGN KEY fk_Suppliers
```



TRUNCATE

is a Data Definition Language (DDL) action that removes all rows from a table but preserves the structure of the table for future use.

Although TRUNCATE is similar to the DELETE command (without the WHERE clause), it is much faster because it bypasses certain integrity constraints and locks. It was officially introduced in the SQL:2008 standard.

SYNTAX:

```
19      ('Maria Reyes', 'Designer', 42000.00);
20
21 •    TRUNCATE TABLE Employees;
```

OUTPUT:

```
23 •    SELECT * FROM Employees;
24
```

Result Grid | Filter Rows:

	emp_id	emp_name	position	salary
*	NULL	NULL	NULL	NULL

DATA MANIPULATION LANGUAGE

SQL DML, or Data Manipulation Language, comprises a set of SQL commands used to manage and manipulate data within a database. Unlike DDL (Data Definition Language) which focuses on the database structure, DML commands operate on the actual data stored in tables.

Advantages of DML

- DML statements could alter the data that is contained or stored in the database.
- It delivers effective human contact with the machine.
- User could specify what data is required.
- DML aims to have many different varieties and functionalities between vendors providing databases.

Disadvantages of DML

- We cannot use DML to change the structure of the database.
- Limit table view i.e., it could conceal some columns in tables.
- Access the data without having the data stored in the object.
- Unable to build or erase lists or sections using DML.

DML COMMANDS

- SELECT STATEMENT
- INSERT STATEMENT
- UPDATE STATEMENT
- DELETE STATEMENT

SELECT

SQL SELECT is used to retrieve data from one or more tables, either all records or specific results based on conditions. It returns output in a tabular format of rows and columns.

- Extracts data from tables.
- Targets specific or all columns (*).
- Supports filtering, sorting, grouping, and joins.
- Results are stored in a result set.

SELECT

```
1    SELECT customer_name, address FROM customers;
```

OUTPUT:

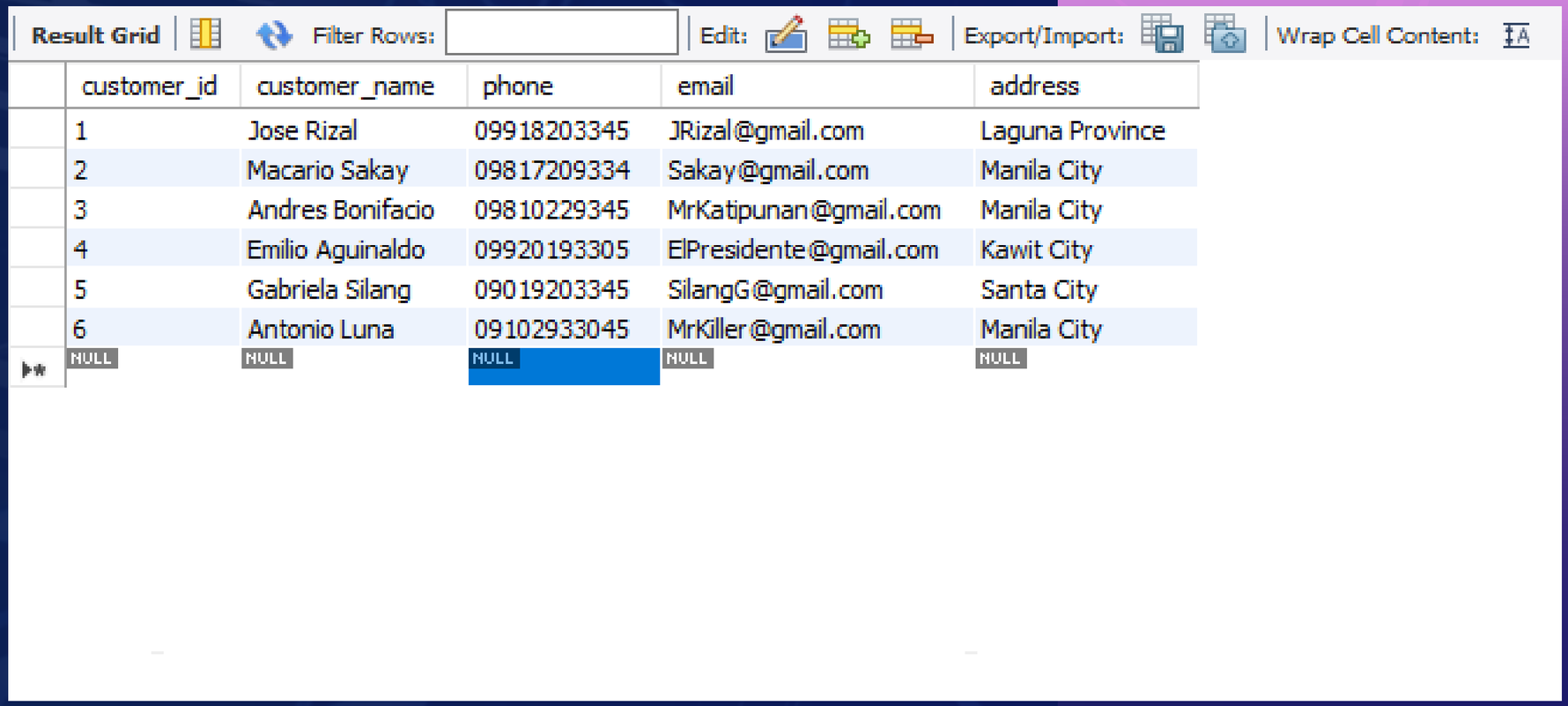
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	address
Jose Rizal	Laguna Province
Macario Sakay	Manila City
Andres Bonifacio	Manila City
Emilio Aguinaldo	Kawit City
Gabriela Silang	Santa City
Antonio Luna	Manila City

SELECT (ALL COLUMN)

```
1    SELECT * FROM customers;
```

OUTPUT:



customer_id	customer_name	phone	email	address
1	Jose Rizal	09918203345	JRizal@gmail.com	Laguna Province
2	Macario Sakay	09817209334	Sakay@gmail.com	Manila City
3	Andres Bonifacio	09810229345	MrKatipunan@gmail.com	Manila City
4	Emilio Aguinaldo	09920193305	ElPresidente@gmail.com	Kawit City
5	Gabriela Silang	09019203345	SilangG@gmail.com	Santa City
6	Antonio Luna	09102933045	MrKiller@gmail.com	Manila City
NULL	NULL	NULL	NULL	NULL

INSERT

is used to add new rows to an existing table. It can insert values into all columns, specific columns, or even copy data from another table. This command is essential for populating databases with meaningful records such as customers, employees or students.

INSERT

```
1    INSERT INTO customers ( customer_id , customer_name, phone, email, address )
2    VALUES ( 7, 'Jahmell Dorias', 09121233455, 'minamina20@gmail.com', 'taga taytay ata rizal');
```

OUTPUT:

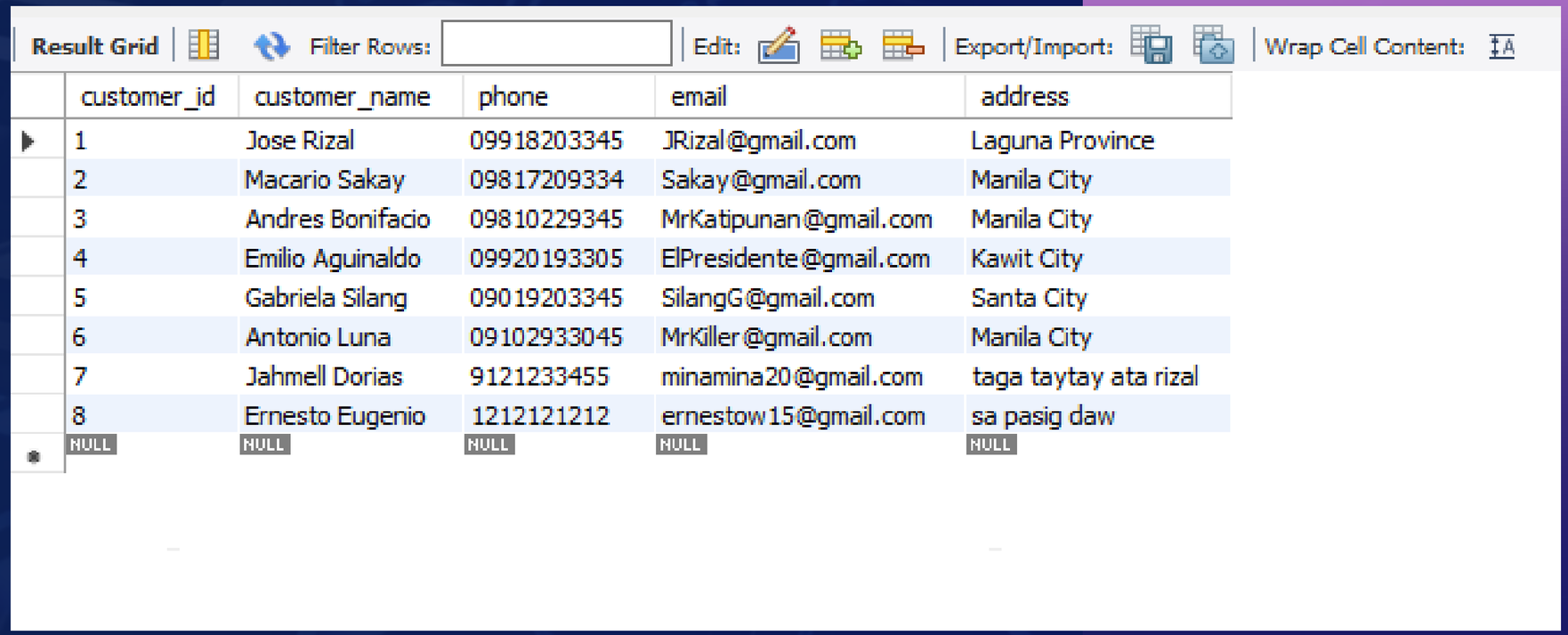
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	customer_name	phone	email	address
1	Jose Rizal	09918203345	JRizal@gmail.com	Laguna Province
2	Macario Sakay	09817209334	Sakay@gmail.com	Manila City
3	Andres Bonifacio	09810229345	MrKatipunan@gmail.com	Manila City
4	Emilio Aguinaldo	09920193305	ElPresidente@gmail.com	Kawit City
5	Gabriela Silang	09019203345	SilangG@gmail.com	Santa City
6	Antonio Luna	09102933045	MrKiller@gmail.com	Manila City
7	Jahmell Dorias	09121233455	minamina20@gmail.com	taga taytay ata rizal
NULL	NULL	NULL	NULL	NULL

INSERT

```
1    INSERT INTO customers
2    VALUES ( 8, 'Ernesto Eugenio', 01212121212, 'ernestow15@gmail.com', 'sa pasig daw' );
```

OUTPUT:



customer_id	customer_name	phone	email	address
1	Jose Rizal	09918203345	JRizal@gmail.com	Laguna Province
2	Macario Sakay	09817209334	Sakay@gmail.com	Manila City
3	Andres Bonifacio	09810229345	MrKatipunan@gmail.com	Manila City
4	Emilio Aguinaldo	09920193305	ElPresidente@gmail.com	Kawit City
5	Gabriela Silang	09019203345	SilangG@gmail.com	Santa City
6	Antonio Luna	09102933045	MrKiller@gmail.com	Manila City
7	Jahmell Dorias	9121233455	minamina20@gmail.com	taga taytay ata rizal
8	Ernesto Eugenio	1212121212	ernestow15@gmail.com	sa pasig daw
NULL	NULL	NULL	NULL	NULL

INSERT

```
INSERT INTO Customers (customer_name, phone, email, address)
VALUES
('Jyle Ultrik? Flores', '03120301235','congjyle23@gmail.com','sa manda daw'),
('Sean Santiago','09345346353','sean32@gmail.com','sa manda din daw'),
('Jomar Durante','09443523453','parkjomar@gmail.com','sa manda ulet');
```

OUTPUT:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	customer_name	phone	email	address
1	Jose Rizal	09918203345	JRizal@gmail.com	Laguna Province
2	Macario Sakay	09817209334	Sakay@gmail.com	Manila City
3	Andres Bonifacio	09810229345	MrKatipunan@gmail.com	Manila City
4	Emilio Aguinaldo	09920193305	ElPresidente@gmail.com	Kawit City
5	Gabriela Silang	09019203345	SilangG@gmail.com	Santa City
6	Antonio Luna	09102933045	MrKiller@gmail.com	Manila City
7	Jahmell Dorias	9121233455	minamina20@gmail.com	taga taytay ata rizal
8	Ernesto Eugenio	1212121212	ernestow15@gmail.com	sa pasig daw
9	Jyle Ultrik? Flores	03120301235	congjyle23@gmail.com	sa manda daw
10	Sean Santiago	09345346353	sean32@gmail.com	sa manda din daw
11	Jomar Durante	09443523453	parkjomar@gmail.com	sa manda ulet
NULL	NULL	NULL	NULL	NULL

TRIVIA

Did you When MySQL is operating in safe update mode, there is an error happens when you try to run Data Manipulation Commands and it is called **Error:1175** and this is caused by the Safety feature of SQL, inorder to proceed manipulating table informations, you'll need to turn it off by disabling it.

```
SET SQL_SAFE_UPDATES = 0;
```

UPDATE

The UPDATE statement in SQL is used to modify existing records in a table without deleting them. It allows updating one or multiple columns, with or without conditions, to keep data accurate and consistent.

SAMPLE SYNTAX:

```
UPDATE Suppliers
SET email = 'SuntukanWithSara@gmail.com'
WHERE supplier_name = 'TheHague Ph'
AND contact_name = 'Rodrigo Duterte'
AND email = 'TheHaguePh@gmail.com';
```

```
INSERT INTO Suppliers (supplier_name, contact_name, phone, email)
VALUES
('DavaoTech Ph','Apollo Quiboloy','09123456789','davaotech@gmail.com'),
('Technologia Ph','Alice Guo Hua Ping','09203396502','wowtechnologia@gmail.com'),
('BarrelTech','Janet Napoles','09876301945','BarrelTech@gmail.com'),
('TheHague Ph','Rodrigo Duterte','0912930043','TheHaguePh@gmail.com'),
('MeowMeowTech','Kiko Barzaga','09102239457','CongMeowMeow@gmail.com');
```

OUTPUT:

Result Grid | Filter Rows: | Edit: | Export/Import:

supplier_id	supplier_name	contact_name	phone	email
41	DavaoTech Ph	Apollo Quiboloy	09123456789	davaotech@gmail.com
42	Technologia Ph	Alice Guo Hua Ping	09203396502	wowtechnologia@gmail.com
43	BarrelTech	Janet Napoles	09876301945	BarrelTech@gmail.com
44	TheHague Ph	Rodrigo Duterte	0912930043	SuntukanWithSara@gmail.com
45	MeowMeowTech	Kiko Barzaga	09102239457	CongMeowMeow@gmail.com
NULL	NULL	NULL	NULL	NULL

DELETE

The SQL DELETE statement is used to remove specific rows from a table while keeping the table structure intact. It is different from DROP, which deletes the entire table.

SAMPLE SYNTAX:

```
DELETE FROM Suppliers
WHERE supplier_name = 'DavaoTech Ph'
AND contact_name = 'Apollo Quiboloy'
AND phone = '09123456789'
AND email = 'davaotech@gmail.com';
```

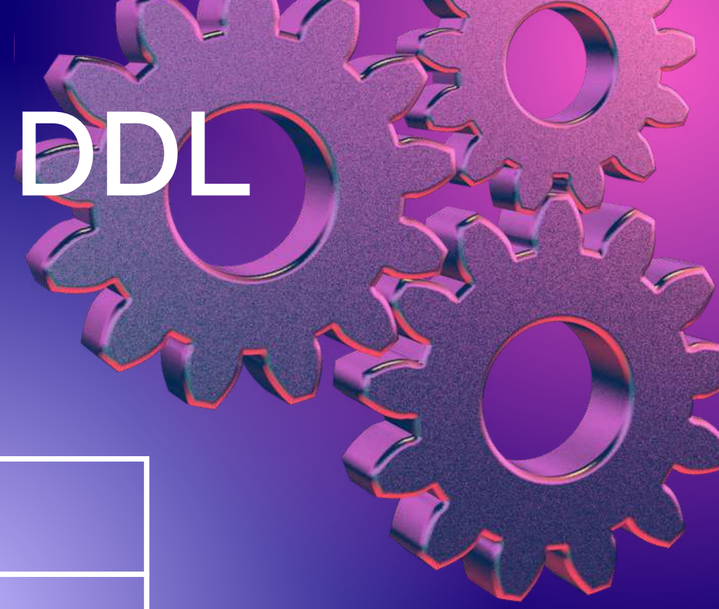
```
INSERT INTO Suppliers (supplier_name, contact_name, phone, email)
VALUES
('DavaoTech Ph','Apollo Quiboloy','09123456789','davaotech@gmail.com'),
('Technologia Ph','Alice Guo Hua Ping','09203396502','wowtechnologia@gmail.com'),
('BarrelTech','Janet Napoles','09876301945','BarrelTech@gmail.com'),
('TheHague Ph','Rodrigo Duterte','0912930043','TheHaguePh@gmail.com'),
('MeowMeowTech','Kiko Barzaga','09102239457','CongMeowMeow@gmail.com');
```

OUTPUT:

Result Grid | Filter Rows: | Edit: | Export/Import:

supplier_id	supplier_name	contact_name	phone	email
42	Technologia Ph	Alice Guo Hua Ping	09203396502	wowtechnologia@gmail.com
43	BarrelTech	Janet Napoles	09876301945	BarrelTech@gmail.com
44	TheHague Ph	Rodrigo Duterte	0912930043	SuntukanWithSara@gmail.com
45	MeowMeowTech	Kiko Barzaga	09102239457	CongMeowMeow@gmail.com
NULL	NULL	NULL	NULL	NULL

DIFFERENCES BETWEEN DDL AND DML



DDL	DML
Used to define database objects like tables, indexes, views, etc.	Used to manipulate data within the database.
Changes made using DDL affect the structure of the database.	Changes made using DML affect the data stored in the database.
DDL statements are not transactional, meaning they cannot be rolled back.	DML statements are executed by application developers or end-users.
DDL statements are typically used during the design and setup phase of a database.	DML statements are used during normal operation of a database.

SUMMARY

- SQL has multiple categories for different database tasks.
- DDL defines and manage structures.
- DML manipulates the actual data.
- Both are essentials for building and maintaining databases.

THANK YOU