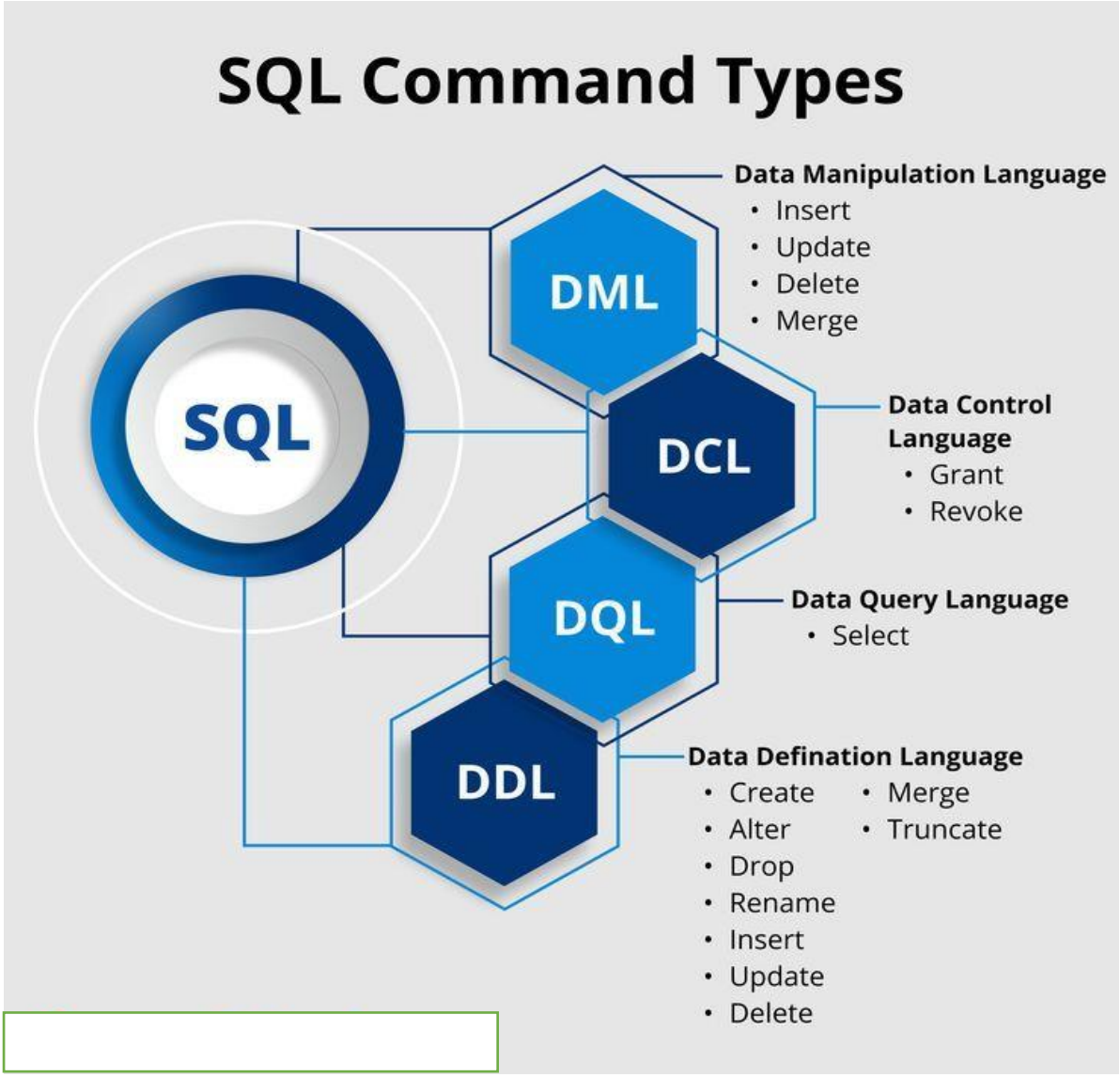
WHAT IS SQL?

SQL (STRUCTURED QUERY LANGUAGE) IS A PROGRAMMING LANGUAGE USED FOR MANAGING AND MANIPULATING DATA IN RELATIONAL DATABASES. IT ALLOWS YOU TO INSERT, UPDATE, RETRIEVE, AND DELETE DATA IN A DATABASE. IT IS WIDELY USED FOR DATA MANAGEMENT IN MANY APPLICATIONS, WEBSITES, AND BUSINESSES. IN SIMPLE TERMS, SQL IS USED TO COMMUNICATE WITH AND CONTROL DATABASES.

TYPES OF SQL COMMANDS



Yogeshchouhan263@gmail.com

DDL COMMANDS FOR DATABASE

- CREATE : CREATE NEW DATABASES / TABLES OBJECTS

Creating database :

```
CREATE DATABASE database_name
```

Creating table in database :

```
CREATE TABLE students (
        Student_id INTEGER ,
        Name VARCHAR (255),
        cgpa INTEGER ,
        gmail VARCHAR (255)
    )
```

Student_id	Name	cgpa	gmail
1	yogesh	10	yogi@gmail.com
2	bhola	10	bhola@gmail.com

- DROP : DELETE EXISTING DATABASE OBJECTS

```
DROP DATABASE database_name
```

- TRUNCATE : DROP ALL ROWS FROM TABLE

```
TRUNCATE TABLE database_name.students
```

Student_id	Name	cgpa	gmail
null	null	null	null

DATA INTEGRITY

Data integrity in databases refers to the accuracy, completeness, and consistency of the data stored in a database. It is a measure of the reliability and trustworthiness of the data and ensures that the data in a database is protected from errors, corruption, or unauthorized changes.

There are various methods used to ensure data integrity, including:

CONSTRAINTS :

Constraints in databases are rules or conditions that must be met for data to be inserted, updated, or deleted in a database table. They are used to enforce the integrity of the data stored in a database and to prevent data from becoming inconsistent or corrupted.

TRANSACTIONS:

a sequence of database operations that are treated as a single unit of work.

NORMALIZATION:

a design technique that minimizes data redundancy and ensures data consistency by organizing data into separate tables.

CONSTRAINTS IN MYSQL

1. NOT NULL

```
create table users(
        user_id INTEGER  NOT NULL,
        name VARCHAR(255) NOT NULL,
        gmail VARCHAR(255),
        password VARCHAR(255)
)
```

User_id	name	gmail	password
1	Yogi	yogi@gmail.com	Alakla
2	Jogi	Null	Null
3	Mogi	Null	alkjslj
4	lofi	lofi@gmail.com	Null

2. UNIQUE(COMBO)

```
create table users(
    user_id INTEGER NOT NULL,
    name VARCHAR(255) NOT NULL,
    gmail VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) UNIQUE  NOT NULL
)
```

user_id	name	gmail	password
1	yogi	yogi@gmail.com	Ajio
2	yogi	yc@gmail.com	Akajl
3	ram	ram@gmail.com	Ajioe
4	ram	rc@gmail.com	Flsjljf

COMBO : if a single column does not contain unique values then combination of two columns make as unique.

3. PRIMARY KEY

a primary key in MySQL is like a special label for a table's row. Imagine you have a table with lots of rows, like a list of students in a class. Each row represents a student. Now, if you want to quickly find a specific student, you need something unique to identify them, right? That's where the primary key comes in.

Think of the primary key as the student ID number. It's unique for each student, so you can easily find any student by their ID. In MySQL, the primary key is a column (or combination of columns) that uniquely identifies each row in the table. It ensures that each row has a unique identifier, just like each student has a unique ID number. This helps in organizing and searching for data efficiently.

```
create table users(
    user_id INTEGER PRIMARY KEY NOT NULL,
    name VARCHAR (255) NOT NULL,
    gmail VARCHAR (255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
```

)

User_id	name	gmail	Password
1	Yogi	yogi@gmail.com	Ahakklj
2	Jogi	jogi@gmail.com	Ajkla
3	Rogi	rogi@gmail.com	Oeio
4	lofi	lofi@gmail.com	aklaja

4. AUTO INCREMENT

```
Create table users(
        User_id INTEGER PRIMARY KEY AUTO_INCREMENT,
        Name VARCHAR (255) NOT NULL,
        Gmail VARCHAR(255) UNIQUE NOT NULL,
        Password VARCHAR(255)
)
```

User_id	Name	Gmail	Password
1	Ram	ram@gmail.com	
2	Mohan	mohan@gmail.com	aijaaljlkjl
3	Shyam	shyam@gmail.com	ajaja
4	radhe	radhe@gmail.com	

Auto-matic generated user-id

5. CHECK

```
Create table student(
        St_id INTEGER PRIMARY KEY AUTO_INCREMENT,
        Name VARCHAR(255) NOT NULL,
        Age INTEGER CHECK (Age < 25),
        Cgpa INTEGER
)
```

St_id	Name	Age	Cgpa
null	null	null	null

INSERT INTO database_name.table_name values

(1,'yogi',22,8.9),

(2,'jogi',24,7.8),

(3,'rogi',37,3.4), # this row did not store because of check constraints

(4,'rohan',45,6.7) # 45>25 so this row also not stored

So final table we achieve:

St_id	Name	Age	Cgpa
1	Yogi	22	8.9
2	jogi	24	7.8

6. DEFAULT

```
Create table orders(
        Order_id INTEGER PRIMARY AUTO_INCREMENT,
        Customer_id INTEGER NOT NULL,
        Order_date DATETIME DEFAULT CURRENT_DATETIMESTAMP()
)
```

Order_id	Customer_id	Order_date
1	234	Current date and time

7. FOREIGN KEY

```
Create table customers(
        Customer_id INTEGER PRIMARY KEY ,
        Name VARCHAR (255) NOT NULL,
        Gmail VARCHAR (255) UNIQUE NOT NULL
)
```

Customer_id	Name	Gmail
1	Yogesh	Yogi@gmail.com

```
Create table orders(
Order_id INTEGER PRIMARY KEY,
Customer_id INTEGER NOT NULL,
Order_date DATETIME DEFAULT CURRENTDATETIMESTAMP(),

CONSTRAINT constraint_name FORIEN KEY (Customer_id) REFERENCES customers (Customer_id)
)
```

Order_id	Customer_id	Order_date
234	1	Current date and time
244	2	Current date and time

REFERENTIAL ACTIONS

1.RESTRICT : THIS ACTION DOES NOT ALLOW THE CHANGES MADE ON THE PARENT TABLE TO THE CHILD TABLE.

2.CASCADE: THIS ACTION PROPAGATES THE CHANGES MADE ON THE PARENT TABLE TO THE CHILD TABLE. FOR EXAMPLE, IF A RECORD IN THE PARENT TABLE IS DELETED, ALL RELATED RECORDS IN THE CHILD TABLE WILL ALSO BE DELETED.

3.SET NULL: WHEN A RECORD IN THE PARENT TABLE IS DELETED OR UPDATED, THE CORRESPONDING FOREIGN KEY VALUES IN THE CHILD TABLE ARE SET TO NULL. THIS IS OFTEN USED WHEN YOU WANT TO MAINTAIN REFERENTIAL INTEGRITY BUT DON'T WANT TO AUTOMATICALLY DELETE OR UPDATE RELATED RECORDS

4.SET DEFAULT:

ALTER TABLE command

The "ALTER TABLE" statement in SQL is used to modify the structure of an existing table. Some of the things that can be done using the ALTER TABLE statement include:

1. Add columns

2. Delete columns

3. Modify columns

1. ADD COLUMNS:

Customer_id	name	Gmail
1	yogi	yogi@gmail.com

Adding a new column password :

ALTER TABLE customer ADD COLUMN password VARCHAR(50) NOT NULL

Customer_id	name	Gmail	Password
1	Yogi	yogi@gmail.com	anything

Adding a new column into a specific place:

ALTER TABLE customer ADD COLUMN last_name VARCHAR(45) NOT NULL AFTER name

Customer_id	name	last_name	Gmail	Password
1	yogi	chouhan	yogi@gmail.com	anything

Adding multiple columns :

ALTER TABLE customer

ADD COLUMNS age INTEGER not null AFTER last_name,

ADD COLUMNS gender VARCHAR (255) NOT NULL AFTER age

Customer_id	name	Last_name	age	gender	Gmail	Password
1	yogi	chouhan	20	male	yogi@gmail.com	anything

2.DELETE COLUMNS:

```
ALTER TABLE customer
DROP COLUMN age,
DROP COLUMN gender
```

Customer_id	name	last_name	gmail	password
1	yogi	chouhan	yogi@gmail.com	anything

3.MODIFY COLUMNS:

```
ALTER TABLE customer MODIFY COLUMNS password INTEGER NOT NULL
```

Customer_id	name	Last_name	gmail	Password
1	yogi	chouhan	yogi@gmail.com	1234

HAPPY LEARNING