① { What is SQL }

SQL (Structured Query language) is a PL used for performing CRUD operations in relational databases.

② { Types of SQL commands }

ⓐ DDL (Data Definition Language)

- CREATE { create new database objects }
- ALTER { modify existing database objects }
- DROP { delete existing database objects }
- TRUNCATE { delete all rows from a table }

ⓑ DML (Data Manipulation Language)

- INSERT (C)
- UPDATE (U)
- DELETE (D)
- SELECT (R)

ⓒ DCL (Data Control Language)

- GRANT [provide access rights on database]
- REVOKE [withdraw access rights on database]

ⓓ TCL (Transaction Control Language)

- COMMIT [save database changes and end transactions]
- ROLLBACK [undo changes that are not committed and end transaction]

③ {DDL commands}

ⓐ CREATE

ⓑ ALTER

ⓒ DROP

ⓓ TRUNCATE

④ { DDL commands for Databases }

ⓐ CREATE

- CREATE DATABASE campusx
- CREATE DATABASE IF NOT EXISTS campusx

ⓑ DROP

- DROP DATABASE campusx
- DROP DATABASE IF EXISTS campusx

⑤ { DDL commands for tables }

ⓐ CREATE

```
CREATE TABLE users (
    user_id  INTEGER,
    name     VARCHAR(255),
    email    VARCHAR(255),
    password VARCHAR(255)
)
```

(b) TRUNCATE — deletes all rows

↳ TRUNCATE TABLE users

(c) DROP

↳ DROP TABLE IF EXISTS users

(6) { Data Integrity }

↳ accuracy + completeness + consistency of data stored in the database

↳ It is a measure of the reliability and trustworthiness of the data.

↳ It ensures that data in a database is protected from errors, corruption or unauthorized changes.

How do we ensure data integrity?

↳ constraints

↳ Transactions

↳ Normalization

⋮

⑦ { Constraints in SQL }

Constraints in databases are rules or conditions that must be met for data to be inserted, updated or deleted in a table.

They are used to enforce the integrity of the data.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- AUTO INCREMENT
- CHECK
- DEFAULT
- FOREIGN KEY

⑧ {NOT NULL constraint}

When we do not provide a value for a particular field, then the default behaviour of SQL is to insert NULL values.

In order to stop this default behaviour, we use the NOT NULL constraint on that field/column.

This way now if we do not provide a value for that field during insertion time, MySQL would insert some value which is different from NULL depending on the data type of the field.

Example :-

```
CREATE TABLE users (
    user_id INTEGER NOT NULL,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255),
```

password VARCHAR(255)

)

Now if we insert (,,,,) into this table, (0, '', NULL, NULL) would be inserted into the table.

{why inserted NULL for email & password?}

The reason why NULL was inserted for email & password when no value was provided for them is because we did not enforce the NOT NULL constraint on these columns due to which SQL used these values as default value.

⑨ { UNIQUE constraint }

```
CREATE TABLE users
(
    user_id    INTEGER NOT NULL,
    name       VARCHAR(255) NOT NULL,
    email      VARCHAR(255) NOT NULL UNIQUE,
    password   VARCHAR(255) NOT NULL
)
```

⑩ { ANOTHER WAY OF WRITING CONSTRAINTS }

```
CREATE TABLE users
(
    user_id    INTEGER NOT NULL,
    name       VARCHAR(255) NOT NULL,
    email      VARCHAR(255) NOT NULL,
    password   VARCHAR(255) NOT NULL,

    CONSTRAINT users_email_unique
               UNIQUE (email)
)
```

But what is the benefit of this syntax?

- (a) Using this we can apply constraint on multiple columns simultaneously.
- (b) It gives flexibility to delete the constraint without actually deleting the table.

Ques: Suppose we want to make the combination of name & email unique?

Soln:- This can never be achieved thru the 1st syntax. To achieve this thru the second syntax, we write

```
CONSTRAINT users_name_email_unique
UNIQUE (name, email)
```

(11) { PRIMARY KEY constraint }

```
CREATE TABLE users (
    user_id  INTEGER NOT NULL PRIMARY KEY,
    name     VARCHAR(255) NOT NULL,
    email    VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL
)
```

When we write PRIMARY KEY, then there is NO need to write UNIQUE and NOT NULL constraints.

We can also use another syntax.

```
CONSTRAINT users_pk PRIMARY KEY (user_id)
```

(12) {AUTO_INCREMENT constraint}

```
CREATE TABLE users (
user_id INTEGER PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(255) NOT NULL,
email VARCHAR(255) NOT NULL UNIQUE,
password VARCHAR(255) NOT NULL
)
```

Can we define multiple constraints in a named constraint?

In mysql, you can only assign one constraint per named constraint. Each constraint whether it is primary key, unique constraint, foreign key constraint, or check constraint, should have its own named constraint.

For example, if you want to apply multiple constraints to a column or a set of columns, you need to define each constraint seperately with its own name.

```
CREATE TABLE users (
user_id INTEGER,
name VARCHAR(255) NOT NULL,
email VARCHAR(255) NOT NULL,
password VARCHAR(255)

CONSTRAINT users_pk PRIMARY KEY
                    (user_id),
CONSTRAINT users_id_AI AUTO_INCREMENT
                       (user_id)
)
```

(13) {CHECK constraint}

```
CREATE TABLE students (
student_id INTEGER PRIMARY KEY AUTO_INCR...,
name VARCHAR (50) NOT NULL,
age INTEGER
CONSTRAINT students_age_check CHECK
  (age > 6 AND age < 25)
)
```

(14) {DEFAULT constraint}

The default constraint is used to set a value in case we do not provide a value for that column.

Are NOT NULL constraint and DEFAULT constraint mutually exclusive?

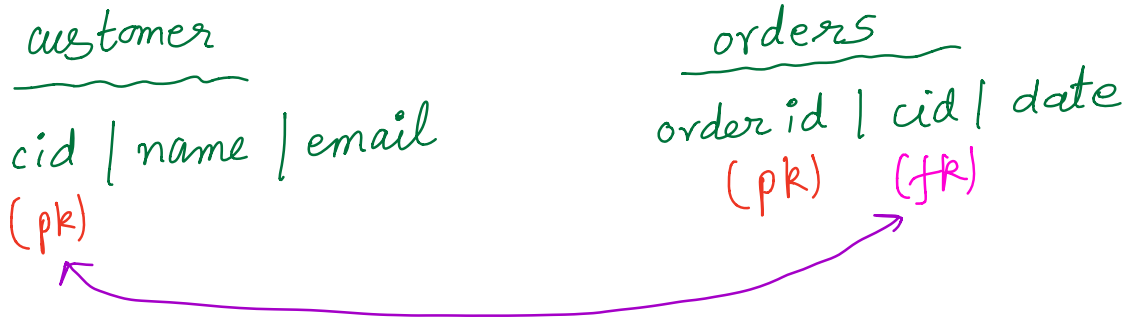
When we do not set NOT NULL on a column, then SQL would automatically insert a NULL value if we do not provide a value for that column during insertion. However, when we set NOT NULL on a column, now if we do not provide any value during insertion for that column, then SQL will use a default value on its own to fill that column. This is where DEFAULT comes. When we use DEFAULT and NOT NULL together, then we can manually insert NULL values for that column and even if we do not provide a value for that column, it would use the DEFAULT value set by us.

Example of DEFAULT constraint

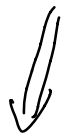
```
CREATE TABLE ticket (
    ticket_id INTEGER PRIMARY KEY,
    name VARCHAR (255) NOT NULL,
    travel_date DATETIME DEFAULT
    CURRENT_TIMESTAMP
)
```

(15) {FOREIGN KEY constraint}

A foreign key is a primary key of another table. It is used for referencing between 2 tables



```
CREATE TABLE customers (
cid INTEGER PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(255) NOT NULL,
email VARCHAR(255) NOT NULL UNIQUE,
)

CREATE TABLE orders (
order_id INTEGER PRIMARY KEY AUTO_INC..,
cid INTEGER NOT NULL,
date DATE NOT NULL DEFAULT CURRENT
                          _DATE

CONSTRAINT orders_fk FOREIGN KEY
(cid) REFERENCES customers (cid)
)
```



This now connects customers with orders. and now if we try to delete a customer then SQL would prevent us (by default) if that customer has any order in orders table.

This is called referential action. We have 4 referential actions.

- Restrict
- Cascade
- Set Null
- Set Default

(a) **Restrict mode** → Default mode

This is the default mode. This means that if we try to delete a customer, then SQL would restrict us if that customer has a order.

(b) Cascade.

If we have cascade mode and if we try to delete a customer, then all the orders of that particular customer would also be deleted.

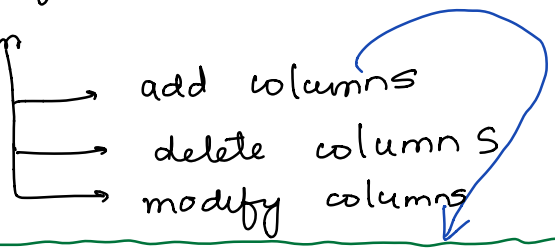
If we try to update a customer, then his/her information would also be updated in the orders table.

```
CREATE TABLE orders (
order_id INTEGER PRIMARY KEY,
cid INTEGER NOT NULL,
order_date DATETIME NOT NULL
        DEFAULT CURRENT_TIMESTAMP

CONSTRAINT orders_fk FOREIGN KEY
(cid) REFERENCES customers (cid)
ON DELETE CASCADE,
ON UPDATE CASCADE

)
```

Last DDL command - ALTER

The "ALTER TABLE" command in SQL is used to modify the structure of an existing table.

We can




```
ALTER TABLE customers ADD COLUMN
password VARCHAR(255) NOT NULL
```

We can also add a new column after a particular column.

```
ALTER TABLE customers ADD COLUMN
middle_name VARCHAR(50) NOT NULL
AFTER first_name
```

We can also add multiple columns.

```
ALTER TABLE customers
ADD COLUMN pan_number VARCHAR(255) AFTER
surname,
```

```
ADD COLUMN joining-date DATETIME NOT
NULL DEFAULT CURRENT_TIMESTAMP
```

DROPPING COLUMNS

```
ALTER TABLE customers
DROP COLUMN pan_number,
DROP COLUMN joining_date
```

MODIFYING COLUMNS

```
ALTER TABLE customers
MODIFY COLUMN pan_number VARCHAR
(50)
```

CONSTRAINTS USING ALTER TABLE

We can also add and delete constraints using ALTER TABLE command.

(a) adding constraint using ALTER TABLE

```
ALTER TABLE customers ADD CONSTRAINT
customer_age-check CHECK (age > 13)
```

(b) deleting constraint using ALTER TABLE

```
ALTER TABLE customers DROP CONSTRAINT
customer_age_check
```

In MySQL, We can NOT modify an existing constraint. You will have to delete the constraint and then add the new constraint.