Primary Key:

A primary key is a unique identifier for each record in a table. It ensures that each row in the table is uniquely identifiable. Only one primary key is allowed per table, and it cannot contain NULL values.

Example:

**CREATE TABLE students (**

**student\_id INT PRIMARY KEY,**

**student\_name VARCHAR(50),**

**age INT,**

**gender VARCHAR(10)**

**);**

Foreign Key:

A foreign key is a column or a set of columns in a table that refers to the primary key of another table. It establishes a relationship between two tables, ensuring referential integrity. The foreign key enforces that the values in the referencing column must exist in the referenced table.

**CREATE TABLE customers (**

**customer\_id INT PRIMARY KEY,**

**customer\_name VARCHAR(100),**

**email VARCHAR(100)**

**);**

**CREATE TABLE orders (**

**order\_id INT PRIMARY KEY,**

**order\_date DATE,**

**total\_amount DECIMAL(10, 2),**

**customer\_id INT, -- This column will be the foreign key**

**FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)**

**);**

In this example:

We created the customers table with three columns: customer\_id, customer\_name, and email. The customer\_id column is set as the primary key to ensure each customer has a unique identifier.

We created the orders table with four columns: order\_id, order\_date, total\_amount, and customer\_id. The order\_id column is set as the primary key for orders, ensuring each order has a unique identifier.

We added a customer\_id column to the orders table. This column is used to store the customer's identifier for whom the order is placed.

We used the FOREIGN KEY constraint to create a relationship between the customer\_id column in the orders table and the customer\_id column in the customers table. This establishes a one-to-many relationship between the customers and orders tables, where one customer can have multiple orders.

Unique Key:

A unique key ensures that the values in the column or a set of columns are unique and not repeated within the table. Unlike the primary key, a unique key can contain NULL values, but only one unique key is allowed per table.

**CREATE TABLE contact\_info (**

**contact\_id INT PRIMARY KEY,**

**employee\_id INT UNIQUE, --** employee\_id is also a UNIQUE constraint to ensure one-to-one relationship

**email VARCHAR(100),**

**phone VARCHAR(20),**

**FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)**

**);**

In this example, the employee\_id attribute in the employees table is the primary key, and it is also a unique constraint in the contact\_info table. This allows us to establish a one-to-one relationship between employees and their contact information.

When an employee is inserted into the employees table, a corresponding entry with the same employee\_id can be inserted into the contact\_info table. The foreign key constraint ensures that the employee\_id in the contact\_info table references a valid employee\_id in the employees table.

Having the same attribute serve as both a primary key and a foreign key allows for seamless integration between the two tables and helps maintain data integrity in the database.

Composite Key.

A composite key is a key that consists of two or more attributes (columns) in a table. Together, these attributes form a unique combination, and the combination must be unique for each row in the table. Composite keys are used when a single attribute is not sufficient to uniquely identify a row, but a combination of attributes does.

Let's create a table to demonstrate a composite key:

Suppose we have a table called orders, where we want to ensure that each order is uniquely identified by a combination of order\_id and customer\_id. The same customer can place multiple orders, but each order should have a distinct order\_id for that customer.

**CREATE TABLE orders (**

**order\_id INT,**

**customer\_id INT,**

**order\_date DATE,**

**total\_amount DECIMAL(10, 2),**

**PRIMARY KEY (order\_id, customer\_id)**

**);**

In this example:

We created the orders table with four columns: order\_id, customer\_id, order\_date, and total\_amount.

We used the PRIMARY KEY constraint to define a composite key on the combination of order\_id and customer\_id. This means that the combination of order\_id and customer\_id must be unique for each row in the table.

With this setup, each order placed by a customer will have a unique combination of order\_id and customer\_id. It ensures that there are no duplicate orders with the same order\_id for the same customer in the orders table. Using a composite key allows us to handle more complex scenarios where a single attribute is not enough to uniquely identify a row in a table.