Date: 4th, January 2025.

**1. PRIMARY KEY**

**Introduction:**

A **PRIMARY KEY** uniquely identifies each record in a table. It ensures no duplicate values and does not allow `NULL`.

**Syntax:**
CREATE TABLE TableName (ColumnName DataType PRIMARY KEY );

**Example:**
CREATE TABLE Students ( StudentID INT PRIMARY KEY, Name VARCHAR(50), Age INT );

**Exercises:**

1. Create a table `Employees` with `EmployeeID` as the primary key and columns `Name` and `Department`.

ANS:

2. Insert duplicate values for `EmployeeID` in the `Employees` table and observe the error.

ANS:

## 2. FOREIGN KEY

**Introduction:**

A **FOREIGN KEY** links two tables, ensuring referential integrity. It references the **PRIMARY KEY** in another table.

**Syntax:**

CREATE TABLE TableName ( ColumnName DataType, FOREIGN KEY (ColumnName) REFERENCES OtherTable(PrimaryKeyColumn) );

**Example**:

CREATE TABLE Orders ( OrderID INT PRIMARY KEY, CustomerID INT, FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) );

**Exercises:**

1. Create two tables, `Departments` and `Employees`. Add a foreign key in `Employees` referencing `Departments`.

ANS:

2. Try to insert a record in `Employees` with a `DepartmentID` that doesn't exist in `Departments`.

ANS:

**3. UNIQUE**

**Introduction:**

A **UNIQUE** constraint ensures all values in a column are different. It allows a single `NULL` value.

**Syntax:**
CREATE TABLE TableName ( ColumnName DataType UNIQUE );

**Example**:

CREATE TABLE Users ( UserID INT PRIMARY KEY, Email VARCHAR(100) UNIQUE );

**Exercises:**

1. Create a `Library` table with a `BookTitle` column having a unique constraint.

ANS:

2. Try to insert duplicate values in the `BookTitle` column and observe the error.

ANS:

**4. NOT NULL**

**Introduction:**

The **NOT NULL** constraint ensures a column cannot have `NULL` values.

**Syntax:**
CREATE TABLE TableName ( ColumnName DataType NOT NULL );

**Example**:
CREATE TABLE Products ( ProductID INT PRIMARY KEY, ProductName VARCHAR(50) NOT NULL );

**Exercises:**

1. Create a `Courses` table with `CourseName` as a NOT NULL column.

ANS:

2. Try inserting a record without specifying a value for `CourseName`.

ANS:

**5. CHECK**

**Introduction:**

The **CHECK** constraint ensures all column values satisfy a specific condition.

**Syntax:**
CREATE TABLE TableName ( ColumnName DataType CHECK (Condition) );

**Example:**
CREATE TABLE Employees ( EmployeeID INT PRIMARY KEY, Age INT CHECK (Age >= 18) );

**Exercises:**

1. Create a `Students` table with an `Age` column that must be at least 5.

ANS:

2. Try inserting a record with `Age` less than 5 and observe the error.

ANS:

**6. DEFAULT**

**Introduction:**

The DEFAULT constraint assigns a default value to a column if no value is provided during insertion.

**Syntax:**
CREATE TABLE TableName ( ColumnName DataType DEFAULT DefaultValue );

**Example:**
CREATE TABLE Accounts ( AccountID INT PRIMARY KEY, Balance DECIMAL(10, 2) DEFAULT 0.00 );

**Exercises:**

1. Create a `Users` table where the `Status` column has a default value of `Active`.

ANS:

2. Insert a record without specifying `Status` and check the default value.

ANS:

---

**7. Exercises for Practice**

1. Combine constraints: Create a `Flights` table where `FlightID` is a primary key, `Source` and `Destination` are NOT NULL, and `Seats` must be greater than 0.

ANS:

2. Create a `Books` table with `BookID` as the primary key, `Title` as unique, and `Price` that must be greater than 0.

ANS:

**Practise Session:**

| Table Name | Column Name | Data Type | Constraints |
|---|---|---|---|
| **Passengers** | PassengerID | INT | Primary Key |
| | Name | VARCHAR(100) | NOT NULL |
| | Age | INT | CHECK (Age > 0) |
| | Gender | CHAR(1) | None |
| | Email | VARCHAR(100) | UNIQUE |
| **Trains** | TrainID | INT | Primary Key |
| | TrainName | VARCHAR(100) | NOT NULL |
| | Source | VARCHAR(50) | NOT NULL |
| | Destination | VARCHAR(50) | NOT NULL |
| | TotalSeats | INT | CHECK (TotalSeats > 0) |
| **Tickets** | TicketID | INT | Primary Key |
| | PassengerID | INT | Foreign Key referencing `Passengers(PassengerID)` |
| | TrainID | INT | Foreign Key referencing `Trains(TrainID)` |
| | BookingDate | DATE | DEFAULT CURRENT_DATE |
| | Status | VARCHAR(20) | DEFAULT 'Booked' |
| **Stations** | StationID | INT | Primary Key |
| | StationName | VARCHAR(100) | NOT NULL |
| | Location | VARCHAR(100) | None |
| | TrainID | INT | Foreign Key referencing `Trains(TrainID)` |

0. Create a above tables with the specified constraints and do the following commands.

ALTER Statements

1. Add a `ContactNumber` column to the `Passengers` table.

2. Add a `DepartureTime` column to the `Trains` table with a default time.
3. Add a unique constraint to the `StationName` column in the `Stations` table.

MODIFY Statements
4. Modify the `ContactNumber` column in `Passengers` to make it `NOT NULL`.
5. Change the `Status` column in `Tickets` to have a default value of `Confirmed`.

DROP Statements
6. Drop the `DepartureTime` column from the `Trains` table.
7. Drop the unique constraint on `StationName` in the `Stations` table.

INSERT Statements
8. Insert a record into the `Passengers` table.
9. Insert a record into the `Trains` table.
10.Insert a ticket into the `Tickets` table.

DELETE and UPDATE
11. Delete a passenger from the `Passengers` table.
12. Update the `TotalSeats` in the `Trains` table to increase by 10.