**Prompt : -Give the detailed document for CREATE command along with theory, syntax, constraints and two examples for beginner.**

The **CREATE** command is a fundamental Data Definition Language (DDL) command in SQL used to create new database objects. This command is essential for defining the structure of your database.

**Theory**

The CREATE command is a crucial part of the database schema definition. It's used to construct the framework for storing data, but it doesn't add any data itself. Think of it as building a new, empty file cabinet: you've created the cabinet and defined the drawers, but you haven't put any files inside yet. The most common use of CREATE is to define a new **table**, which is a collection of related data organized in rows and columns.

**Key Concepts**

* **Database:** A structured collection of data, typically stored electronically.
* **Table:** A specific type of database object that organizes data into rows and columns.
* **Column (or field):** A vertical entity in a table that contains all the values for a single attribute. For example, "First Name" could be a column.
* **Row (or record):** A horizontal entity in a table that represents a single, complete entry. For example, all the information for one person.
* **Data Type:** Specifies the type of data a column can hold (e.g., INTEGER, VARCHAR, DATE).

**Syntax**

The basic syntax for creating a table is as follows:

SQL

CREATE TABLE table\_name (

column1\_name data\_type [constraint],

column2\_name data\_type [constraint],

column3\_name data\_type [constraint],

...

);

**Breakdown**

* CREATE TABLE are the keywords that initiate the command.
* table\_name is the name you give to your new table.
* column\_name is the name of each column you're defining.
* data\_type is the type of data the column will store.
* [constraint] is an optional rule applied to a column to limit the type of data that can be entered.

**Constraints**

**Constraints** are rules enforced on data columns in a table. They are used to limit the type of data that can go into a table, ensuring the accuracy and reliability of the data.

* NOT NULL: Ensures that a column cannot have a NULL value. It must always contain a value.
* UNIQUE: Ensures all values in a column are different.
* PRIMARY KEY: A combination of NOT NULL and UNIQUE. It uniquely identifies each row in a table. A table can have only one primary key.
* FOREIGN KEY: Uniquely identifies a row in another table, creating a link between two tables.
* CHECK: Ensures that all values in a column satisfy a specific condition.
* DEFAULT: Sets a default value for a column if no value is specified.

**Examples for Beginners**

**Example 1: Creating a Simple Students Table**

Let's create a table to store basic student information like their ID, first name, and last name.

SQL

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL

);

**Explanation:**

* **Students** is the name of our new table.
* **StudentID** is a column of type INT (an integer). The PRIMARY KEY constraint ensures that each student has a unique ID and it can't be empty.
* **FirstName** is a column of type VARCHAR(50). VARCHAR is used for variable-length strings, and 50 is the maximum number of characters it can hold. The NOT NULL constraint means that a student's first name must always be provided.
* **LastName** is another VARCHAR(50) column, also with a NOT NULL constraint.

**Example 2: Creating an Employees Table with More Constraints**

This example demonstrates creating a table for employees with more complex constraints, including a default value and a check constraint.

SQL

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Department VARCHAR(50) DEFAULT 'Unassigned',

Salary DECIMAL(10, 2) CHECK (Salary > 0)

);

**Explanation:**

* **EmployeeID** is the PRIMARY KEY.
* **FirstName** and **LastName** are VARCHAR(50) with NOT NULL.
* **Department** is a VARCHAR(50). The DEFAULT 'Unassigned' constraint automatically sets the department to 'Unassigned' if no value is provided when a new employee is added.
* **Salary** is a DECIMAL(10, 2) data type, which stores numbers with up to 10 total digits and 2 digits after the decimal point. The CHECK (Salary > 0) constraint ensures that no employee can have a salary of zero or less.