**PROMPT:** Give me a documentation of Theory of CREATE TABLE statement, full syntax with examples, overview of data types, constraints with multiple examples for Beginner.

**CREATE Command in SQL**

The CREATE command is one of the basic SQL Data Definition Language (DDL) commands. It is used to create new database objects like:

* A database
* A table
* A schema
* A index
* A view

**Common CREATE Commands**

| **Command** | **What it does** | **Example** |
| --- | --- | --- |
| CREATE DATABASE | Makes a new database | CREATE DATABASE mydb; |
| CREATE TABLE | Makes a new table | CREATE TABLE students (...); |
| CREATE SCHEMA | Organizes tables under one group | CREATE SCHEMA school; |
| CREATE INDEX | Speeds up searches in big tables | CREATE INDEX idx ON students(name); |

1. **CREATE TABLE**:

This is the most common and important CREATE command. It is used to define a new table in a database, specifying its columns, data types, and constraints.

**What it does**: It creates the structure for where you will store your data.

**Example**:  
  
 CREATE TABLE Employees (

id INT PRIMARY KEY,

name VARCHAR(50) );

This command creates a table named Employees with two columns: id (a unique number for each employee) and name (a text field for their name).

1. **CREATE DATABASE**:

This command is used to create a new database. It's usually the very first step you take when starting a new project.

**What it does**: It creates a new, empty container (the database) to hold all your tables, views, and other objects.

**Example**:  
  
 CREATE DATABASE HR\_Management;

This command creates a new database named HR\_Management.

1. **CREATE VIEW**:

A view is a virtual table based on the result set of an SQL query. It doesn't store data itself but rather presents a predefined view of data from one or more tables.

**What it does**: It simplifies complex queries and enhances security by showing only a specific subset of data.

**Example**:  
  
 CREATE VIEW Active\_Employees AS

SELECT id, name

FROM Employees

WHERE status = 'Active';

This creates a virtual table called Active\_Employees that will only show the ID and name of employees whose status is 'Active'.

In addition to the common CREATE commands, CREATE SCHEMA and CREATE INDEX are also important for database management.

### 

### CREATE SCHEMA:

CREATE SCHEMA is a command used to create a **schema**, which is a logical container f or database objects like tables, views, and indexes. Think of a schema as a folder within a database. It helps organize objects and manage user permissions.

**Why use it?**

* **Organization**: It keeps related objects together. For example, you might have a sales schema for all sales-related tables and a payroll schema for HR tables, all within the same database.
* **Security**: You can grant or revoke permissions to users at the schema level, making it easier to control access to groups of tables.

**Example**

CREATE SCHEMA sales;

CREATE SCHEMA hr;

Now, you can create tables inside these schemas. For instance, to create a Customers table within the sales schema, you would use:

CREATE TABLE sales.Customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(100)

);

### CREATE INDEX:

CREATE INDEX is a command used to create an **index** on a table. An index is a data structure that improves the speed of data retrieval operations on a table. . It works similarly to the index in a book: instead of reading the entire book to find a topic, you can quickly look up the page number in the index.

**Why use it?**

* **Performance**: Indexes significantly speed up SELECT queries, especially on large tables, by allowing the database to find rows much faster.
* **Unique Constraints**: A unique index can be used to enforce a UNIQUE constraint on a column.

**Example**

Let's say you have an Employees table with a column for last\_name. If you frequently search for employees by their last name, creating an index on that column will make those searches much faster.

CREATE INDEX idx\_last\_name

ON Employees (last\_name);

Here, idx\_last\_name is the name of the index, and it is created on the last\_name column of the Employees table.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## Create Table

The CREATE TABLE statement is a fundamental command in **SQL** (Structured Query Language) used to define a new table in a database. A table is where data is organized into **rows** and **columns**.

## **Syntax**

The basic syntax for CREATE TABLE is as follows:

CREATE TABLE table\_name (  
 column1\_name data\_type constraint,  
 column2\_name data\_type constraint,  
 column3\_name data\_type constraint,  
 ...  
);

* **CREATE TABLE table\_name**: This part specifies that you're creating a new table and gives it a unique name.
* **column1\_name data\_type constraint**: This defines each column. You must specify a **column name**, a **data type**, and optionally a **constraint**. You separate each column definition with a comma.
* **( ... )**: Parentheses enclose the entire list of column definitions.

### **Example**s

Here’s a simple example of a CREATE TABLE statement for a Students table.

CREATE TABLE Students (  
 student\_id INT,  
 first\_name VARCHAR(50),  
 last\_name VARCHAR(50),  
 date\_of\_birth DATE  
);

CREATE TABLE employees (

emp\_id SERIAL PRIMARY KEY,

emp\_name VARCHAR(100),

dept TEXT,

salary NUMERIC(10, 02),

join\_date DATE DEFAULT CURRENT\_DATE

);

**Data Types**

**Data types** dictate the kind of data a column can hold. Choosing the right data type helps save storage space and ensures data integrity. Here's an overview of common data types:

| Data Type | Description | Example |
| --- | --- | --- |
| **INT** | Stores whole numbers (integers). | 100, 25 |
| **DECIMAL(10,2)** | Fixed-point (p=precision, s=scale) Prices, salaries | 52599.73 |
| **FLOAT/ REAL** | Approximate decimals, Scientific values | 3.14 |
| **VARCHAR(100)** | Variable length up to n chars, Names, emails | 'John', 'Marketing' |
| **CHAR(1)** | Fixed length n characters, | ‘M’, ‘F’ Gender (M, F) |
| **TEXT** | Stores large text strings. | 'This is a long description.' |
| **DATE** | (Joining date)Stores a date in 'YYYY-MM-DD' format | '2024-08-07' |
| **TIME** | Store time only(HH:MM:SS), Shift start time | ‘04:36:02’ |
| **DATETIME** | Stores both date and time. | '2024-08-07 15:30:00' |
| **TIMESTAMP** | Auto-updating date-time, Record creation time | '2024-08-07 15:30:00' |
| **BOOLEAN** | Stores a true or false value, Status, is\_active | TRUE, FALSE |

## **Constraints**

**Constraints** are rules enforced on data columns to limit what data can be inserted. They ensure the accuracy and reliability of the data within a table.

### **1. NOT NULL**

Ensures a column cannot have a NULL (empty) value. This is useful for columns that must have a value, like a user's first name.

**Example:**

CREATE TABLE Users (  
 user\_id INT NOT NULL,  
 username VARCHAR(50) NOT NULL  
);

### **2. UNIQUE**

Ensures all values in a column are different. It allows NULL values, but if a value is present, it must be unique.

**Example:**

CREATE TABLE Products (  
 product\_id INT NOT NULL UNIQUE,  
 product\_name VARCHAR(100)  
);

### **3. PRIMARY KEY**

A PRIMARY KEY is a **unique identifier** for each row in a table. It's a combination of NOT NULL and UNIQUE, meaning it must be both unique and not empty. A table can have only one primary key.

**Example:**

CREATE TABLE Employees (  
 employee\_id INT PRIMARY KEY,  
 first\_name VARCHAR(50)  
);

**4. FOREIGN KEY**

A FOREIGN KEY is a field in one table that refers to the PRIMARY KEY in another table. It establishes a link between two tables, ensuring **referential integrity**.

**Example:**

First, we need a Departments table with a primary key:

CREATE TABLE Departments (  
 department\_id INT PRIMARY KEY,  
 department\_name VARCHAR(50)  
);

Then, we can create the Employees table with a FOREIGN KEY that references the Departments table.

CREATE TABLE Employees (  
 employee\_id INT PRIMARY KEY,  
 employee\_name VARCHAR(50),  
 department\_id INT,  
 FOREIGN KEY (department\_id) REFERENCES Departments(department\_id)  
);

### **5. DEFAULT**

Sets a default value for a column if no value is explicitly specified during an INSERT operation.

**Example:**

CREATE TABLE Orders (  
 order\_id INT PRIMARY KEY,  
 order\_date DATE DEFAULT CURRENT\_DATE  
);

### **6. CHECK**

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

**Example:**

CREATE TABLE Students (  
 student\_id INT PRIMARY KEY,  
 age INT,  
 CHECK (age >= 18)  
);