🗃️ Example 1: Create a users Table

```
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

📥 Insert Data into users

```
INSERT INTO users (username, email)
VALUES
    ('alice', 'alice@example.com'),
    ('bob', 'bob@example.com'),
    ('carol', 'carol@example.com');
```

📚 Example 2: Create a books Table

```
CREATE TABLE books (
    book_id SERIAL PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(100),
    published_year INT,
    available BOOLEAN DEFAULT TRUE
);
```

📥 Insert Data into books

```
INSERT INTO books (title, author, published_year)
VALUES
    ('1984', 'George Orwell', 1949),
    ('Brave New World', 'Aldous Huxley', 1932),
    ('Fahrenheit 451', 'Ray Bradbury', 1953);
```

📎 Example 3: Create a loans Table (Relation Between users and books)

```
CREATE TABLE loans (
    loan_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(user_id),
    book_id INT REFERENCES books(book_id),
    loan_date DATE DEFAULT CURRENT_DATE,
    return_date DATE
);
```

📥 Insert Data into loans

```
INSERT INTO loans (user_id, book_id, return_date)
VALUES
    (1, 2, '2025-09-30'),
    (2, 1, NULL);
```

🔷 1. Basic Table Creation in PostgreSQL

Here's a simple example of a basic table to store **user information**:

```
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Explanation:

- SERIAL: Auto-incrementing integer.
- PRIMARY KEY: Uniquely identifies each row.
- NOT NULL: Field must be filled.
- UNIQUE: No duplicate values allowed.
- TIMESTAMP: Stores date and time.

🔷 2. Real-Time Table Used in MNC Projects

Let's take a **Human Resource Management System (HRMS)** as an example. These systems are common in multinational companies (MNCs). Below is a **real-world table structure** for storing **employee details**.

✅ Employee Table (Real-Time Use Case)

```
CREATE TABLE employees (
    employee_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15),
    hire_date DATE NOT NULL,
    job_id VARCHAR(10) NOT NULL,
    salary NUMERIC(10, 2),
    manager_id INT,
    department_id INT,
    status VARCHAR(20) DEFAULT 'Active',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Notes:

- salary NUMERIC(10, 2): For accurate financial data.
- manager_id: Self-referencing (manager is also an employee).
- department_id: Will usually be a **foreign key**.

- status: Active, Inactive, On Leave, etc.
- created_at and updated_at: For tracking changes (used in almost all real-time tables).

🔁 Example with Foreign Keys

Let's also add a **departments table** and add a **foreign key** to employees.

```
CREATE TABLE departments (
    department_id SERIAL PRIMARY KEY,
    department_name VARCHAR(100) NOT NULL UNIQUE,
    location VARCHAR(100)
);


-- Alter the employees table to add foreign key
ALTER TABLE employees
ADD CONSTRAINT fk_department
FOREIGN KEY (department_id)
REFERENCES departments(department_id);
```

🔗 Other Real-Time Tables Often Used in MNCs

Table Name	Purpose
projects	Track project assignments and status
timesheets	Log working hours per employee
payroll	Salary processing and tax details
users	App login and permissions
audit_logs	Track data changes for compliance

```
CREATE TABLE students (
    student_id SERIAL PRIMARY KEY,      -- Auto increment unique ID
    first_name VARCHAR(50) NOT NULL,    -- Student First Name
    last_name VARCHAR(50),              -- Last Name (optional)
    email VARCHAR(100) UNIQUE NOT NULL, -- Email should be unique
    phone VARCHAR(15),                  -- Contact number
    branch VARCHAR(10) NOT NULL,        -- Department (CSE, ECE, IT…)
    join_date DATE DEFAULT CURRENT_DATE -- Auto take today’s date
);

INSERT INTO students (first_name, last_name, email, phone, branch)
VALUES
('Ravi', 'Kumar', 'ravi.kumar@example.com', '9876543210', 'CSE'),
('Anitha', 'Reddy', 'anitha.reddy@example.com', '9123456780', 'ECE');

CREATE TABLE faculty (
    faculty_id SERIAL PRIMARY KEY,      -- Unique Faculty ID
    full_name VARCHAR(100) NOT NULL,    -- Faculty Name
    email VARCHAR(100) UNIQUE NOT NULL, -- Unique Email
    phone VARCHAR(15),                  -- Contact Number
    department VARCHAR(50) NOT NULL,    -- Department Name
    join_date DATE DEFAULT CURRENT_DATE -- Auto insert joining date
);


INSERT INTO faculty (full_name, email, phone, department)
VALUES
('Dr. Suresh Reddy', 'suresh.reddy@college.com', '9876501234', 'CSE'),
('Prof. Lakshmi Devi', 'lakshmi.devi@college.com', '9123456789', 'ECE');
```

interview for **developers, DBAs, or data engineers**.

◈ Basic Level

1. How do you create a table in PostgreSQL?

Example answer: Using the CREATE TABLE statement with column names and data types.

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    age INT,
    hire_date DATE
);
```

---

2. What are data types supported in PostgreSQL when creating a table?

Example answer: Common types include:

- INT, SERIAL
- VARCHAR, TEXT
- DATE, TIMESTAMP
- BOOLEAN, NUMERIC, FLOAT

---

3. What is the difference between CHAR, VARCHAR, and TEXT in PostgreSQL?

---

4. What is a SERIAL data type? How is it used when creating a table?

---

5. How do you set a primary key when creating a table?

Example:

```
CREATE TABLE departments (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(100)
```

```
);
```

◈ Intermediate Level

6. How do you add foreign keys during table creation?

Example:

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
```

---

7. What are constraints? Name a few and show how to use them during table creation.

- NOT NULL
- UNIQUE
- CHECK
- DEFAULT

Example:

```
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    price NUMERIC CHECK (price > 0),
    stock INT DEFAULT 0
);
```

---

8. How do you define a composite primary key in PostgreSQL?

Example:

```
CREATE TABLE course_enrollments (
    student_id INT,
```

```
    course_id INT,
    PRIMARY KEY (student_id, course_id)
);
```

---

9. What is the difference between UNIQUE and PRIMARY KEY?

---

10. How do you define default values for a column in PostgreSQL?

◈ Advanced Level

11. How can you create a table with an inheritance in PostgreSQL?

Example:

```
CREATE TABLE vehicle (
    id SERIAL,
    name TEXT,
    PRIMARY KEY (id)
);

CREATE TABLE car (
    doors INT
) INHERITS (vehicle);
```

---

12. How do you use GENERATED ALWAYS AS IDENTITY in PostgreSQL 10+?

Example:

```
CREATE TABLE books (
    book_id INT GENERATED ALWAYS AS IDENTITY,
    title TEXT
);
```

---

13. How do you create temporary tables in PostgreSQL?

Example:

```
CREATE TEMP TABLE temp_sales (
    id INT,
    amount NUMERIC
);
```

---

14. What is the difference between TEMP, TEMPORARY, and UNLOGGED tables?

---

15. How do you create a table using the structure of another table?

Example:

```
CREATE TABLE new_table AS TABLE existing_table WITH NO DATA;
```

---

16. How do you partition a table in PostgreSQL?

Example:

```
CREATE TABLE measurement (
    logdate DATE NOT NULL,
    city_id INT NOT NULL,
    temperature NUMERIC,
    PRIMARY KEY (logdate, city_id)
) PARTITION BY RANGE (logdate);
```

📝 **Tip for Interviews:**

You may be asked to **write SQL queries**, **explain constraints**, or **debug a faulty CREATE TABLE statement**. Make sure you understand:

- Referential integrity
- Indexes & constraints
- Performance implications (e.g., UNLOGGED, partitioning)
- Schema design principles
