You are tasked with building a PostgreSQL-backed database for an EdTech company that manages online training and certification programs for individuals across various technologies.

The goal is to:

Design a normalized schema

Support querying of training data

Ensure secure access

Maintain data integrity and control over transactional updates

Database planning (Nomalized till 3NF)

A student can enroll in multiple courses

Each course is led by one trainer

Students can receive a certificate after passing

Each certificate has a unique serial number

Trainers may teach multiple courses

Tables to Design (Normalized to 3NF):

1. \*\*students\*\*

\* `student\_id (PK)`, `name`, `email`, `phone`

2. \*\*courses\*\*

\* `course\_id (PK)`, `course\_name`, `category`, `duration\_days`

3. \*\*trainers\*\*

\* `trainer\_id (PK)`, `trainer\_name`, `expertise`

4. \*\*enrollmentsnrollment\*\*

\* `enrollment\_id (PK)`, `student\_id (FK)`, `course\_id (FK)`, `enroll\_date`

5. \*\*certificates\*\*

\* `certificate\_id (PK)`, `enrollment\_id (FK)`, `issue\_date`, `serial\_no`

6. \*\*course\\_trainers\*\* (Many-to-Many if needed)

\* `course\_id`, `trainer\_id`

---

Phase 2: DDL & DML

\* Create all tables with appropriate constraints (PK, FK, UNIQUE, NOT NULL)

\* Insert sample data using `INSERT` statements

\* Create indexes on `student\_id`, `email`, and `course\_id`

---

Phase 3: SQL Joins Practice

Write queries to:

1. List students and the courses they enrolled in

2. Show students who received certificates with trainer names

3. Count number of students per course

---

Phase 4: Functions & Stored Procedures

Function:

Create `get\_certified\_students(course\_id INT)`

→ Returns a list of students who completed the given course and received certificates.

Stored Procedure:

Create `sp\_enroll\_student(p\_student\_id, p\_course\_id)`

→ Inserts into `enrollments` and conditionally adds a certificate if completed (simulate with status flag).

---

Phase 5: Cursor

Use a cursor to:

\* Loop through all students in a course

\* Print name and email of those who do not yet have certificates

---

Phase 6: Security & Roles

1. Create a `readonly\_user` role:

\* Can run `SELECT` on `students`, `courses`, and `certificates`

\* Cannot `INSERT`, `UPDATE`, or `DELETE`

2. Create a `data\_entry\_user` role:

\* Can `INSERT` into `students`, `enrollments`

\* Cannot modify certificates directly

---

Phase 7: Transactions & Atomicity

Write a transaction block that:

\* Enrolls a student

\* Issues a certificate

\* Fails if certificate generation fails (rollback)

```sql

BEGIN;

-- insert into enrollments

-- insert into certificates

-- COMMIT or ROLLBACK on error

```

---