**EXPERIMENT NO 1.3**

**AIM:** To design and implement a normalized relational database schema using SQL for managing students, courses, and their enrollments with integrity constraints, perform data insertion, simulate transactions using SAVEPOINT and ROLLBACK, demonstrate error handling with faulty records, and retrieve meaningful reports using SQL joins to display student enrollment and grade details.

**THEORY:**

Database Normalization

* Normalization reduces redundancy and ensures consistency.
* In this schema:
  + Student Table (details of students)
  + Course Table (details of courses offered)
  + Enrollment Table (relationship between students and courses along with grades)

Integrity Constraints

* Primary Key (PK): Uniquely identifies each record.
* Foreign Key (FK): Maintains referential integrity across tables.
* NOT NULL & UNIQUE: Prevents invalid entries.

Transactions in SQL

* A transaction is a sequence of operations treated as a single unit.
* SAVEPOINT: Used to set intermediate points in a transaction.
* ROLLBACK: Undo changes up to a specific savepoint or the beginning of the transaction.

Joins for Reporting

* Joins combine data from multiple tables to generate meaningful reports, e.g., listing students with enrolled courses and grades.

**CODE:**

### 1. Create Database and Use It

CREATE DATABASE UniversityDB;

USE UniversityDB;

### Create Tables

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

email VARCHAR(50) UNIQUE NOT NULL

);

CREATE TABLE Courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(50) NOT NULL,

credits INT CHECK (credits > 0)

);

CREATE TABLE Enrollments (

enrollment\_id INT PRIMARY KEY,

student\_id INT,

course\_id INT,

grade CHAR(2),

FOREIGN KEY (student\_id) REFERENCES Students(student\_id),

FOREIGN KEY (course\_id) REFERENCES Courses(course\_id)

);

### Insert Records into Tables

INSERT INTO Students VALUES (1, 'Alice', 'alice@example.com');

INSERT INTO Students VALUES (2, 'Bob', 'bob@example.com');

INSERT INTO Courses VALUES (101, 'Database Systems', 4);

INSERT INTO Courses VALUES (102, 'Computer Networks', 3);

INSERT INTO Enrollments VALUES (1001, 1, 101, 'A');

INSERT INTO Enrollments VALUES (1002, 2, 102, 'B');

### Transactions with SAVEPOINT and ROLLBACK

START TRANSACTION;

INSERT INTO Students VALUES (3, 'Charlie', 'charlie@example.com');

SAVEPOINT sp1;

INSERT INTO Students VALUES (1, 'Duplicate', 'dup@example.com');

ROLLBACK TO sp1;

INSERT INTO Students VALUES (4, 'David', 'david@example.com');

COMMIT;

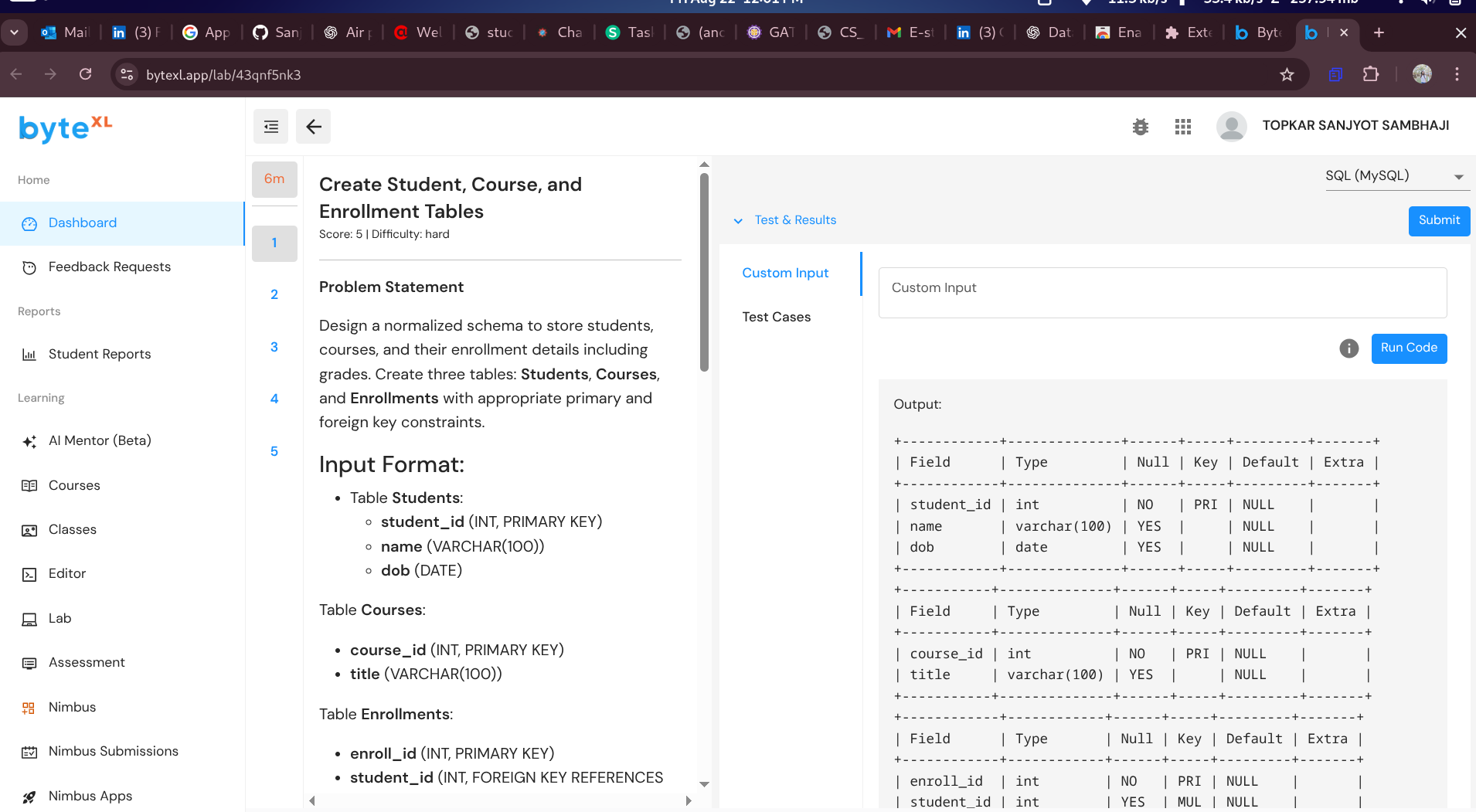
### Reports using Joins

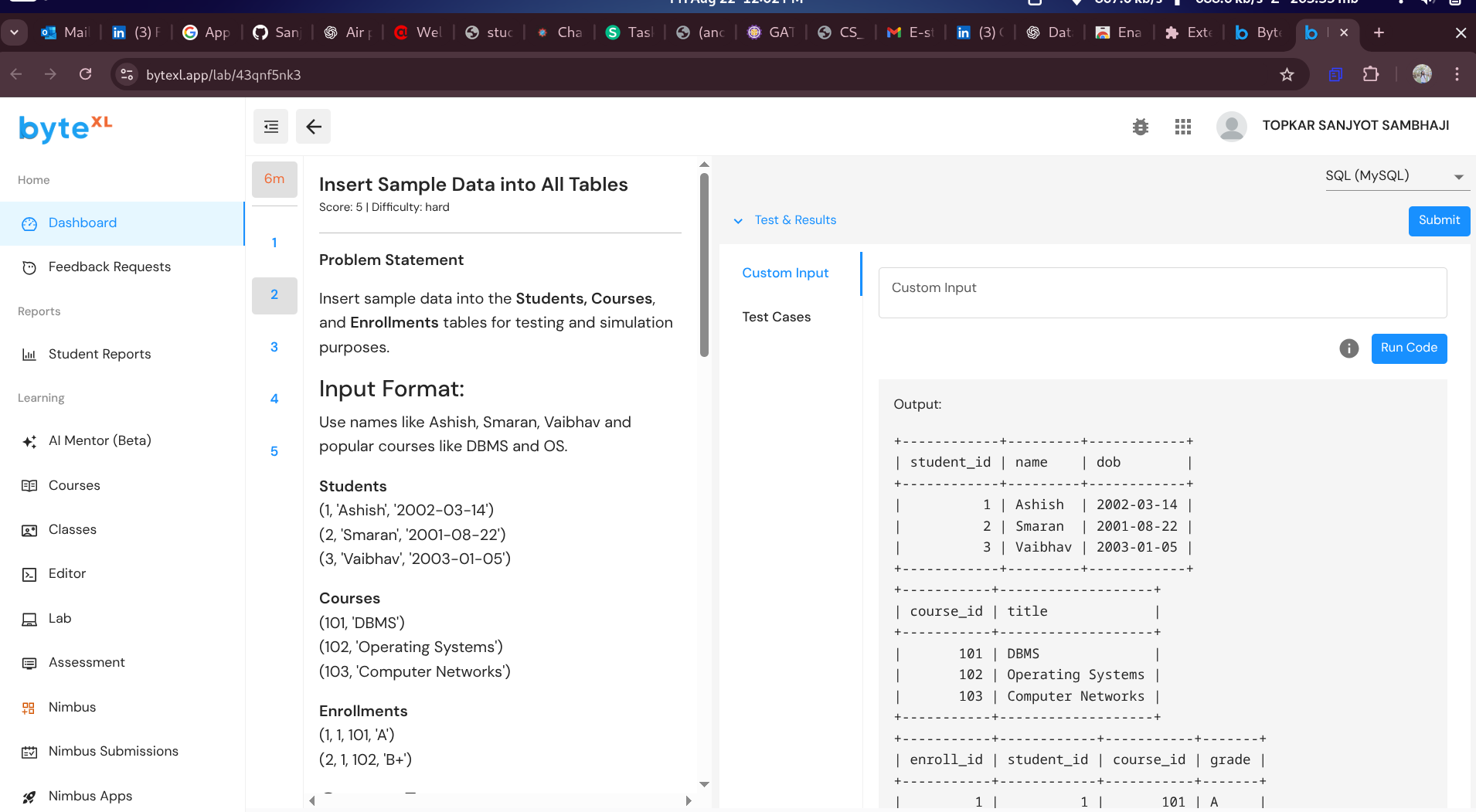
SELECT s.student\_id, s.name, c.course\_name, e.grade

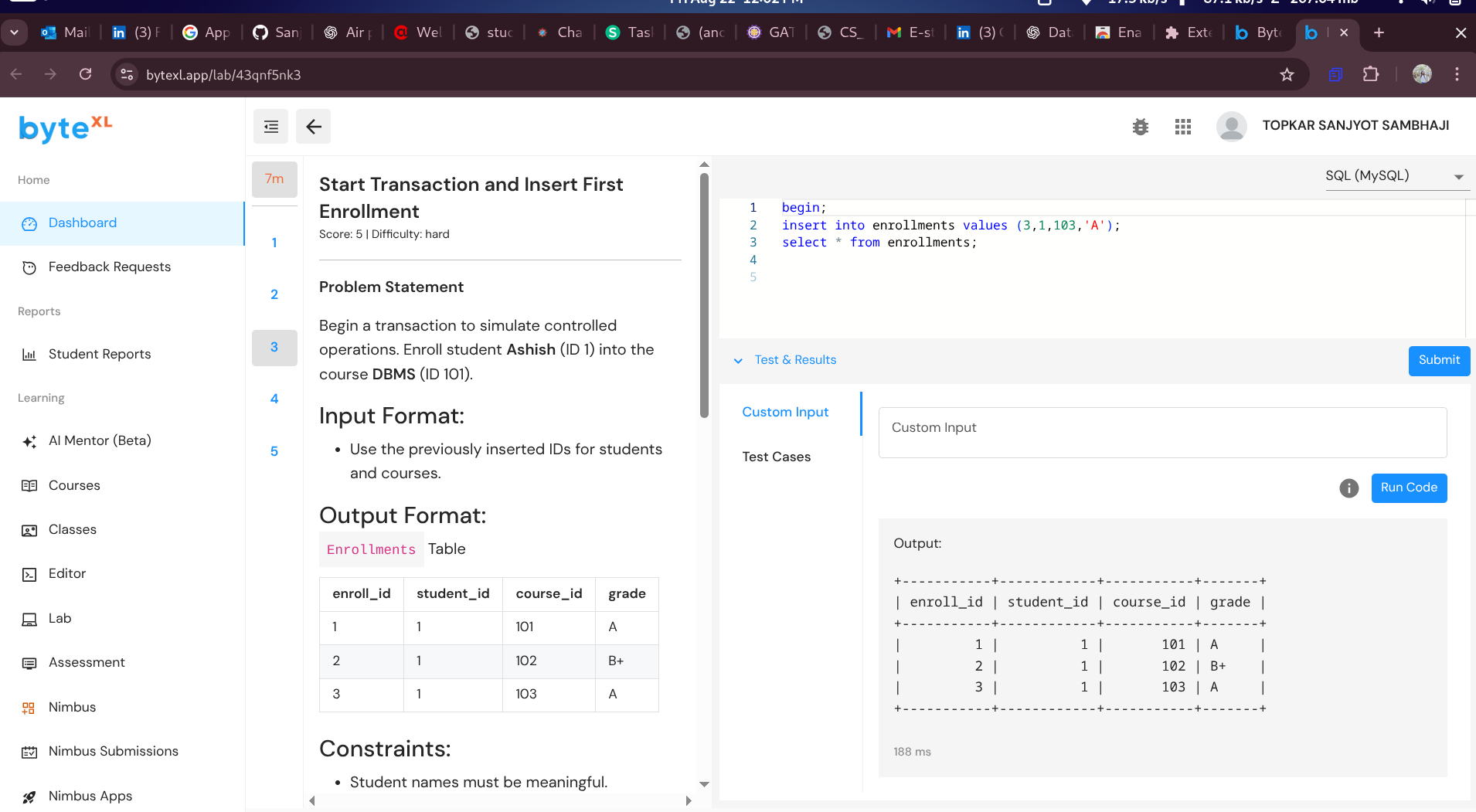
FROM Students s

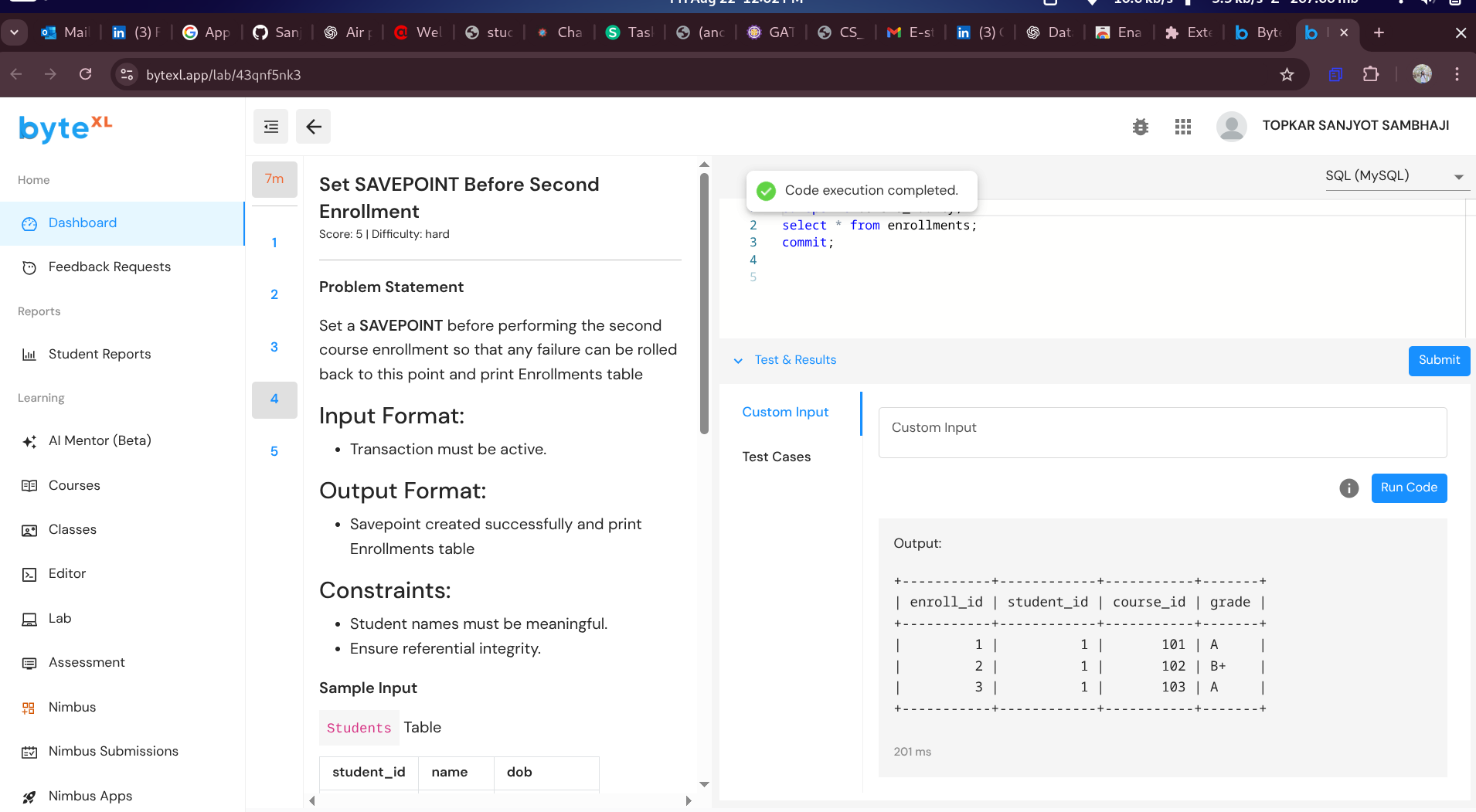
JOIN Enrollments e ON s.student\_id = e.student\_id

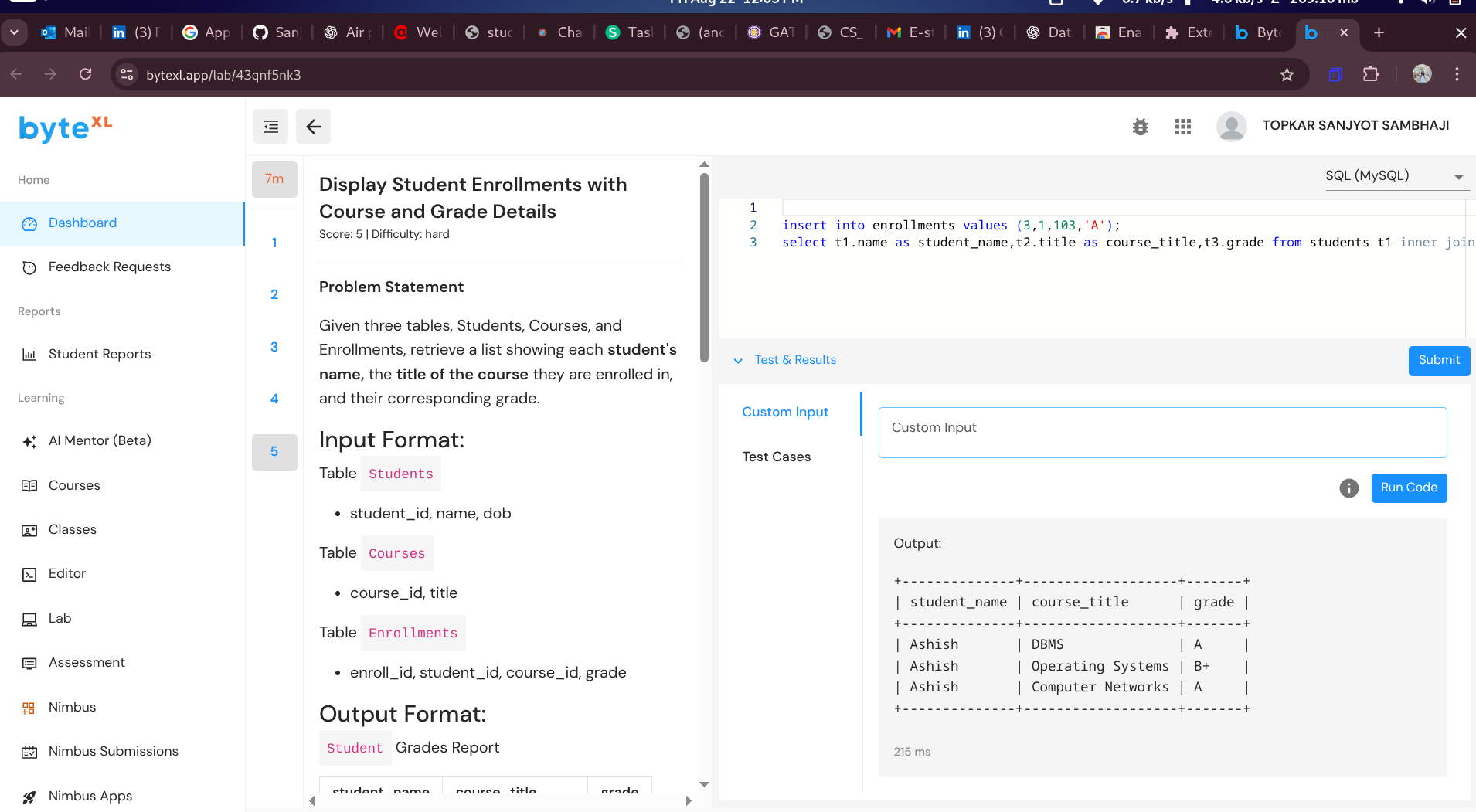
JOIN Courses c ON e.course\_id = c.course\_id;

****

****

****

****

****