



INFORMATION SYSTEMS 3 – ASSIGNMENT 3

Designing a Database Accessible via a Web Application

Name of Qualification	Diploma in Information Technology: Software Development
Team Members	SS Thusi (Group Leader)
	A Dlamini
	L Nofemele
	K Sishi
	Y Magingxa
	SM Faye
Module name & Code	Information Systems 3 (ISYS300)
Due Date	14 May 2025
Lecturer	Mr Maguraushe

Table of Contents

1. Executive Summary	5
2. Conceptual Design	6
2.1 Entity-Relationship Diagram (ERD)	6
2.2 ERD Description	7
2.2.1 Entities and Attributes	7
2.2.2 Relationships	7
3. Physical Database Design	9
3.1 Database Schema	9
3.1.1 Database Initialization	9
3.1.2 Departments Table	9
3.1.3 Users Table	10
3.1.4 Courses Table	10
3.1.5 Course Content Table	11
3.1.6 Enrollments Table	11
3.1.7 Quiz Questions Table	12
3.1.8 Quiz Attempts Table	12
3.1.9 Insert Departments Data	12
3.1.10 Insert Users Data	13
3.1.11 Insert Courses Data	14
3.1.12 Insert Course Content Data	15
3.1.13 Insert Enrollments Data	17
3.1.14 Insert Quiz Questions Data	18
3.1.15 Insert Quiz Attempts Data	20
3.1.16 Enrollment Insert Trigger	20
3.1.17 Enrollment Delete Trigger	20
3.1.18 Enroll Student Procedure	21
3.1.19 Update Lesson Completion Procedure	22
3.1.20 Example Queries	22
3.1.21 Commit Transaction	24
3.2 Indexing and Storage	24
Indexing Justification (Theoretical)	24
Index Implementation (Practical)	25
Performance Test Demo	26
Data Type Selections	26
Storage Engine Choice	28
4. Normalization	28
4.1 UNF Table	28
4.2 Normalization Process	32
4.2.1 First Normal Form (1NF)	32

4.2.2 Second Normal Form (2NF)	33
4.2.3 Third Normal Form (3NF)	33
5. Client-Server Architecture	34
5.1 Application Overview	34
5.2 Architecture Diagram	35
5.3 Scenarios	36
5.3.1 Scenario 1: Student Enrollment (Pessimistic Locking)	36
5.3.2 Scenario 2: Quiz Submission (Optimistic Locking)	36
5.4 Frontend-Backend Interaction	37
5.5 Screenshots & Features	37
Student Dashboard:	37
My Active Courses:	38
Lesson:	38
My Academics:	39
6. Database Security	39
6.1 Role-Based Access Control (RBAC)	39
6.2 SQL Injection Prevention	40
7. Concurrency Control and Deadlock Handling	40
7.1 Multiple Students Enrolling in the Same Course Simultaneously	41
7.2 Instructor Editing Course Content While Students View It	42
7.3 Simultaneous Quiz Submission & Auto-Grading	42
8. Transaction Processing and ACID Compliance	43
8.1 Overview of Transaction Processing in SES	43
8.2 Enrollment Transaction (student/enroll.php)	43
Purpose	43
ACID Compliance	44
Implementation	44
8.3 Lesson Completion Transaction (student/course_content.php)	45
Purpose	45
ACID Compliance	45
Implementation	46
8.4 Quiz Submission Transaction (student/submit_quiz.php)	47
Purpose	47
ACID Compliance	47
Implementation	47
8.5 Challenges and Solutions	49
8.6 Conclusion	49
9. Integrity Constraints	49
9.1 Types of Integrity Constraints in SES	50
9.1.1 Primary Key Constraints	50
9.1.2 Foreign Key Constraints	50

9.1.3 Unique Constraints	51
9.1.4 Check Constraints	51
9.1.5 Not Null Constraints	52
9.2 Implementation of Integrity Constraints	52
9.3 Role of Integrity Constraints in Transaction Processing	52
10. Backup and Recovery	53
10.1 Backup Plan	53
Tools Used:	53
Storage Location:	53
Backup Purpose:	53
Step-by-Step Demonstration	53
10.2 Recovery Procedures	55
11. Conclusion	56

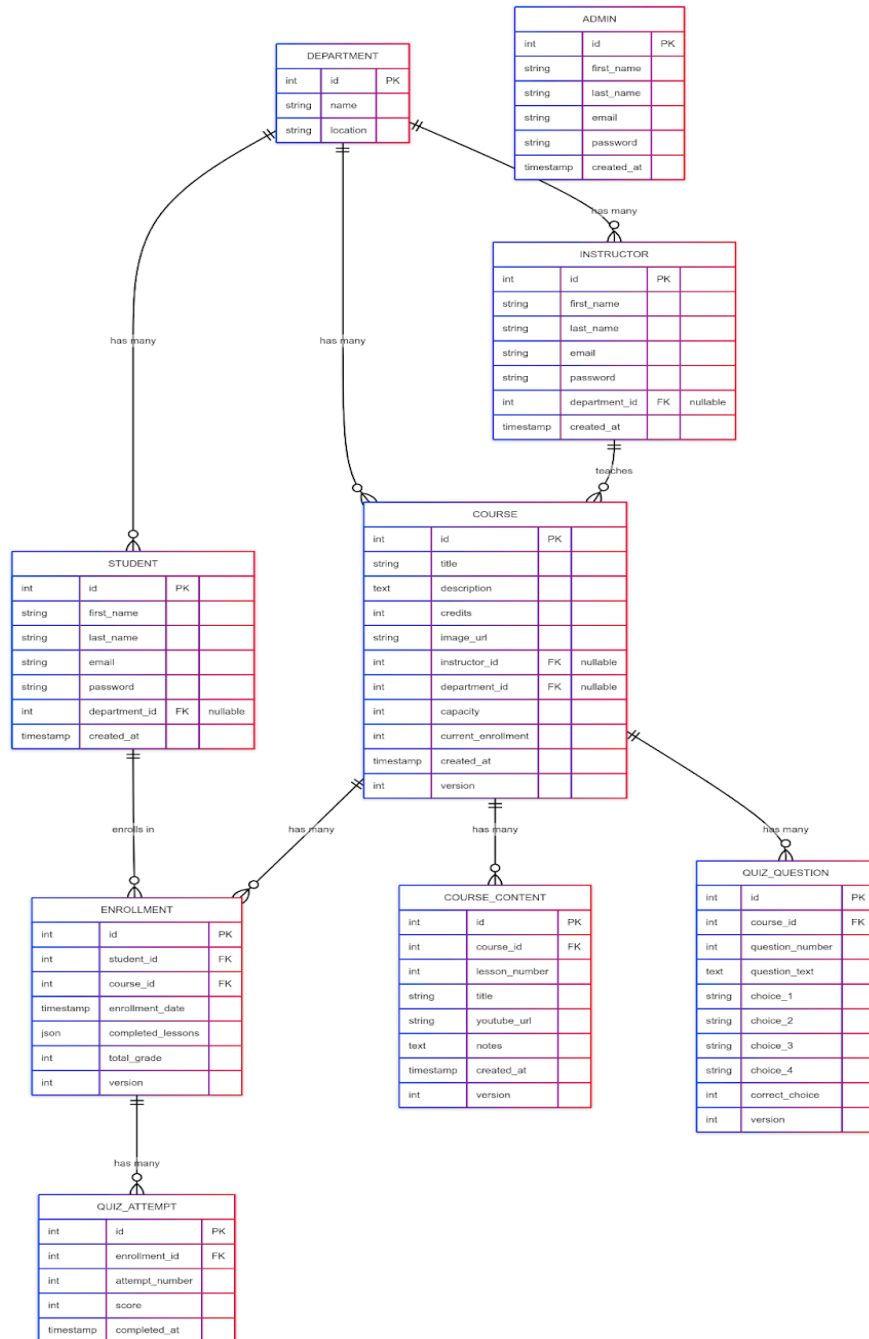
1. Executive Summary

The Student Enrollment System (SES) is a scalable online learning platform developed by a team of six students for ISYS300, built on a robust client-server architecture using

the WAMP stack (Apache, PHP, MySQL 8.0). SES enables students, instructors, and administrators to manage course enrollments, lesson progress, and quiz submissions via a web interface accessible on browsers like Chrome. The `ses_db` database, normalized to 3NF, supports concurrent access with pessimistic and optimistic locking to ensure data consistency during critical operations like enrollment and quiz submission. Key features include a dynamic front-end (HTML, CSS, JavaScript), secure back-end (PHP with PDO), and comprehensive security measures (RBAC, SQL injection prevention). The system was rigorously tested by 14 May 2025, demonstrating ACID-compliant transactions, efficient indexing, and reliable backup/recovery processes. This report details SES's design, implementation, and compliance with project requirements, showcasing its scalability and robustness for multi-user environments.

2. Conceptual Design

2.1 Entity-Relationship Diagram (ERD)



2.2 ERD Description

Student 1 designed the entity-relationship diagram (ERD) for the `ses_db` database, providing the conceptual foundation for the Student Enrollment System (SES), an online learning platform. The ERD defines entities, their attributes, relationships, cardinality, business rules, and assumptions, guiding the logical and physical database design (section 3). This section details the ERD's components, ensuring a clear blueprint for SES's data structure.

2.2.1 Entities and Attributes

The ERD comprises the following entities with their primary attributes:

- **Departments:**
 - `id` (PK, int, auto_increment)
 - `name` (varchar(100), not null)
 - `location` (varchar(100), nullable)
- **Students:**
 - `id` (PK, int, auto_increment)
 - `first_name` (varchar(50), not null)
 - `last_name` (varchar(50), not null)
 - `email` (varchar(100), not null, unique)
 - `password` (varchar(255), not null)
 - `department_id` (FK to Departments, int, nullable)
 - `created_at` (timestamp, default: CURRENT_TIMESTAMP)
- **Instructors:**
 - `id` (PK, int, auto_increment)
 - `first_name` (varchar(50), not null)
 - `last_name` (varchar(50), not null)
 - `email` (varchar(100), not null, unique)
 - `password` (varchar(255), not null)
 - `department_id` (FK to Departments, int, nullable)
 - `created_at` (timestamp, default: CURRENT_TIMESTAMP)
- **Admins:**
 - `id` (PK, int, auto_increment)
 - `first_name` (varchar(50), not null)
 - `last_name` (varchar(50), not null)
 - `email` (varchar(100), not null, unique)
 - `password` (varchar(255), not null)
 - `created_at` (timestamp, default: CURRENT_TIMESTAMP)
- **Courses:**
 - `id` (PK, int, auto_increment)
 - `title` (varchar(100), not null)
 - `description` (text, nullable)
 - `credits` (int, nullable)

- image_url (varchar(255), nullable)
- instructor_id (FK to Instructors, int, nullable)
- department_id (FK to Departments, int, nullable)
- capacity (int, nullable)
- current_enrollment (int, default: 0)
- created_at (timestamp, default: CURRENT_TIMESTAMP)
- version (int, default: 0)
- **Enrollments:**
 - id (PK, int, auto_increment)
 - student_id (FK to Students, int, not null)
 - course_id (FK to Courses, int, not null)
 - enrollment_date (timestamp, default: CURRENT_TIMESTAMP)
 - completed_lessons (json, nullable)
 - total_grade (int, default: 0)
 - version (int, default: 0)
- **Course_Content:**
 - id (PK, int, auto_increment)
 - course_id (FK to Courses, int, not null)
 - lesson_number (int, not null)
 - title (varchar(100), not null)
 - youtube_url (varchar(255), nullable)
 - notes (text, not null)
 - created_at (timestamp, default: CURRENT_TIMESTAMP)
 - version (int, default: 0)
 - Unique: (course_id, lesson_number)
- **Quiz_Questions:**
 - id (PK, int, auto_increment)
 - course_id (FK to Courses, int, not null)
 - question_number (int, not null)
 - question_text (text, not null)
 - choice_1 (varchar(255), not null)
 - choice_2 (varchar(255), not null)
 - choice_3 (varchar(255), not null)
 - choice_4 (varchar(255), not null)
 - correct_choice (int, not null)
 - version (int, default: 0)
 - Unique: (course_id, question_number)
- **Quiz_Attempts:**
 - id (PK, int, auto_increment)
 - enrollment_id (FK to Enrollments, int, not null)
 - attempt_number (int, default: 1)
 - score (int, default: 0)
 - completed_at (timestamp, default: CURRENT_TIMESTAMP)

2.2.2 Relationships

The ERD establishes the following relationships:

- **Departments:**
 - Has many **Students** (via department_id)
 - Has many **Instructors** (via department_id)
 - Has many **Courses** (via department_id)
- **Students:**
 - Belongs to **Department** (via department_id, nullable)
 - Enrolls in **Courses** (via student_id in Enrollments)
- **Instructors:**
 - Belongs to **Department** (via department_id, nullable)
 - Teaches **Courses** (via instructor_id)
- **Admins:**
 - Manages system (no direct relationships, functionality handled in application logic)
- **Courses:**
 - Belongs to **Department** (via department_id, nullable)
 - Taught by **Instructor** (via instructor_id, nullable)
 - Has many **Enrollments** (via course_id)
 - Has many **Course_Content** (via course_id)
 - Has many **Quiz_Questions** (via course_id)
- **Enrollments:**
 - Belongs to **Student** (via student_id)
 - Belongs to **Course** (via course_id)
 - Has many **Quiz_Attempts** (via enrollment_id)
- **Course_Content:**
 - Belongs to **Course** (via course_id)
- **Quiz_Questions:**
 - Belongs to **Course** (via course_id)
- **Quiz_Attempts:**
 - Belongs to **Enrollment** (via enrollment_id)

3. Physical Database Design

3.1 Database Schema

Student 1 (Scelo Thusi) implemented the `ses_db` schema, translating the conceptual ERD (section 2.2) into a MySQL 8.0 database for the Student Enrollment System (SES). The schema comprises seven tables (`departments`, `users`, `courses`, `course_content`, `enrollments`, `quiz_questions`, `quiz_attempts`), with constraints, triggers, and stored procedures to support course enrollment, lesson tracking, and quiz management. Stored procedures (`EnrollStudent`, `UpdateLessonCompletion`) are defined but not used for transaction control, which is handled in PHP (section 8). The following subsections detail the schema's components, with SQL code and explanations for each.

3.1.1 Database Initialization

This code block configures the MySQL environment and creates the `ses_db` database. It sets the SQL mode to prevent auto-increment issues, establishes a UTC timezone, and starts a transaction to ensure all operations complete together. The existing `ses_db` database is dropped (if present) to provide a clean setup, and the new database is selected for subsequent operations.

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";
START TRANSACTION;
```

```
DROP DATABASE IF EXISTS `ses_db`;
CREATE DATABASE `ses_db`;
USE `ses_db`;
```

3.1.2 Departments Table

The `departments` table stores academic department information, with columns for a unique ID, department name, and location. The table uses InnoDB for transaction support and UTF-8 encoding for internationalization. A primary key ensures unique IDs, supporting relationships with other tables (e.g., `users`, `courses`).

```
CREATE TABLE `departments` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
  `location` VARCHAR(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

3.1.3 Users Table

The `users` table manages SES users (students, instructors, admins), storing personal details, credentials, and roles. It includes a unique email constraint and a foreign key to `departments`, allowing nullification if a department is deleted. The `role` column uses an ENUM to restrict values, ensuring data integrity.

```
CREATE TABLE `users` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(50) NOT NULL,
  `last_name` VARCHAR(50) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `password` VARCHAR(255) NOT NULL,
  `role` ENUM('student', 'instructor', 'admin') NOT NULL,
  `department_id` INT DEFAULT NULL,
  `created_at` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`),
  FOREIGN KEY (`department_id`) REFERENCES `departments` (`id`) ON DELETE SET
  NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

3.1.4 Courses Table

The `courses` table defines available courses, including title, description, credits, and enrollment details. Foreign keys link to `users` (instructor) and `departments`, with `ON DELETE SET NULL` to maintain integrity. The `current_enrollment` and `capacity` fields track enrollment status, updated by triggers (section 3.1.12).

```
CREATE TABLE `courses` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(100) NOT NULL,
  `description` TEXT,
  `credits` INT DEFAULT NULL,
  `image_url` VARCHAR(255) DEFAULT NULL,
  `instructor_id` INT DEFAULT NULL,
  `department_id` INT DEFAULT NULL,
  `capacity` INT DEFAULT NULL,
  `current_enrollment` INT DEFAULT '0',
  `created_at` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  `version` INT DEFAULT '0',
  PRIMARY KEY (`id`),
  FOREIGN KEY (`instructor_id`) REFERENCES `users` (`id`) ON DELETE SET NULL,
  FOREIGN KEY (`department_id`) REFERENCES `departments` (`id`) ON DELETE SET
  NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

3.1.5 Course Content Table

The `course_content` table stores lesson details for each course, including lesson number, title, and resources (e.g., YouTube URL, notes). A unique constraint ensures no duplicate lessons per course, and a foreign key to `courses` enables cascading deletes if a course is removed.

```
CREATE TABLE `course_content` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `course_id` INT NOT NULL,
  `lesson_number` INT NOT NULL,
```

```

`title` VARCHAR(100) NOT NULL,
`youtube_url` VARCHAR(255) DEFAULT NULL,
`notes` TEXT NOT NULL,
`created_at` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
`version` INT DEFAULT '0',
PRIMARY KEY (`id`),
UNIQUE KEY `course_lesson_unique` (`course_id`, `lesson_number`),
FOREIGN KEY (`course_id`) REFERENCES `courses` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

3.1.6 Enrollments Table

The `enrollments` table tracks student-course registrations, storing enrollment dates, completed lessons (as a JSON array), and grades. A unique constraint prevents duplicate enrollments, and foreign keys to `users` and `courses` ensure referential integrity with cascading deletes.

```

CREATE TABLE `enrollments` (
  `EnrollmentID` INT NOT NULL AUTO_INCREMENT,
  `StudentID` INT NOT NULL,
  `CourseID` INT NOT NULL,
  `EnrollmentDate` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  `completed_lessons` JSON DEFAULT NULL COMMENT 'JSON array of completed lesson numbers (1-4)',
  `total_grade` INT DEFAULT '0',
  `version` INT DEFAULT '0',
  PRIMARY KEY (`EnrollmentID`),
  UNIQUE KEY `student_course_unique` (`StudentID`, `CourseID`),
  FOREIGN KEY (`StudentID`) REFERENCES `users` (`id`) ON DELETE CASCADE,
  FOREIGN KEY (`CourseID`) REFERENCES `courses` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

3.1.7 Quiz Questions Table

The `quiz_questions` table defines course-specific quiz questions, including text, multiple-choice options, and the correct answer. A unique constraint ensures unique question numbers per course, and a foreign key to `courses` supports cascading deletes.

```

CREATE TABLE `quiz_questions` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `course_id` INT NOT NULL,
  `question_number` INT NOT NULL,
  `question_text` TEXT NOT NULL,
  `choice_1` VARCHAR(255) NOT NULL,
  `choice_2` VARCHAR(255) NOT NULL,
  `choice_3` VARCHAR(255) NOT NULL,
  `choice_4` VARCHAR(255) NOT NULL,
  `correct_choice` INT NOT NULL,
  `version` INT DEFAULT '0',
  PRIMARY KEY (`id`),
  UNIQUE KEY `course_question_unique` (`course_id`, `question_number`),

```

```
FOREIGN KEY (`course_id`) REFERENCES `courses` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

3.1.8 Quiz Attempts Table

The `quiz_attempts` table records students' quiz submissions, linking to `enrollments` via a foreign key. It stores attempt numbers, scores, and timestamps, with cascading deletes to remove attempts if an enrollment is deleted.

```
CREATE TABLE `quiz_attempts` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `enrollment_id` INT NOT NULL,
  `attempt_number` INT NOT NULL DEFAULT '1',
  `score` INT NOT NULL DEFAULT '0',
  `completed_at` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`enrollment_id`) REFERENCES `enrollments` (`EnrollmentID`) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

3.1.9 Insert Departments Data

This block populates the `departments` table with 10 departments (e.g., Computer Science, Mathematics). Each entry includes a name and location, providing the foundation for associating users and courses with academic units.

```
INSERT INTO `departments` (`id`, `name`, `location`) VALUES
(1, 'Computer Science', 'Science Building, Room 301'),
(2, 'Mathematics', 'Math Building, Room 201'),
(3, 'Biology', 'Science Building, Room 101'),
(4, 'Business', 'Business Building, Room 401'),
(5, 'Psychology', 'Social Sciences Building, Room 201'),
(6, 'Information Technology', 'Technology Building, Room 501'),
(7, 'Software Engineering', 'Innovation Center, Room 300'),
(8, 'Cybersecurity', 'Security Complex, Lab 4'),
(9, 'Data Science', 'Analytics Hub, Room 210'),
(10, 'Network Engineering', 'Infrastructure Wing, Lab 3');
```

3.1.10 Insert Users Data

This block inserts 19 user records into the `users` table, including students (e.g., Kwanele Sishi), instructors (e.g., Sarah Johnson), and admins. Each user has a unique email, hashed password, and role, with some linked to departments. This data supports authentication and role-based access in SES.

```
INSERT INTO `users` (`id`, `first_name`, `last_name`, `email`, `password`,
`role`, `department_id`, `created_at`) VALUES
(1, 'Kwanele', 'Sishi', 'kwanelesishi050509@gmail.com',
'$2y$10$Sls00i5k6AF7jtlPi33a3ebTqcI7PBzSsX33a1pnEEI1Scig8VouG', 'student',
NULL, '2025-05-04 11:00:28'),
```

(7, 'John', 'Doe', 'john.doe@sudent.edu.au',
'\$2y\$10\$92IXUNpkj00r0Q5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', 'student', 1,
'2025-05-04 11:21:22'),
(10, 'Azanda', 'Nyide', 'azanda@gmail.com',
'\$2y\$10\$.fpcgCOMQBJpx4iTYc9zH.yDJUWcUPF2rfV0bE9l.WIqtpnq6bRVS', 'student', 5,
'2025-05-04 14:06:21'),
(11, 'Admin', 'User', 'admin@ses.ac',
'\$2y\$10\$92KUNpkjQn0o5byMl.Yed.4Ea3Rd9llC/og4a2mT7d1E9oYzWbq0', 'admin', NULL,
'2025-05-04 11:21:22'),
(12, 'Sarah', 'Johnson', 's.johnson@ses.ac',
'\$2y\$10\$T4GUfGwrpqH.QQl8aN0bzuCwl2zqRHsCq6QyUeb5xU7mW7fK4hF6W', 'instructor',
1, '2025-05-04 11:21:22'),
(13, 'Michael', 'Chen', 'm.chen@ses.ac',
'\$2y\$10\$VpVvKXW.6r9d9tTJ4mZrE.8zq1Dk0y7S7sS7s4w4jJz1wK4hF6W', 'instructor', 2,
'2025-05-04 11:21:22'),
(14, 'Emily', 'Wilson', 'e.wilson@ses.ac',
'\$2y\$10\$HjZ4V2X.9r9d9tTJ4mZrE.8zq1Dk0y7S7sS7s4w4jJz1wK4hF6W', 'instructor', 3,
'2025-05-04 11:21:22'),
(15, 'David', 'Miller', 'd.miller@ses.ac',
'\$2y\$10\$LkM5N2X.6r9d9tTJ4mZrE.8zq1Dk0y7S7sS7s4w4jJz1wK4hF6W', 'instructor', 4,
'2025-05-04 11:21:22'),
(16, 'admin', 'admin', 'admin12345@gmail.com',
'\$2y\$10\$CLrB5HvFkLXiiFUYKett2u1/opMZpEPp0qn80WWrgqxFW.miAdpCi', 'admin', 1,
'2025-05-05 09:21:39'),
(17, 'Menzi', 'Dlamini', 'menzi@gmail.com',
'\$2y\$10\$J4F5WHEYbiK1w1LSF.zFF0zVzQqgZGcL/m4tYsixaWthyH1FA1sVW', 'instructor',
2, '2025-05-05 09:37:42'),
(18, 'Thokozani', 'Muthwa', 'muthwa@gmail.com',
'\$2y\$10\$dAB4TlF6TqclTmRmlJnsNuyhJYJjUzW3x9S0Vcup02IoeLqzvR3Uu', 'instructor',
2, '2025-05-05 09:39:44'),
(19, 'Sibonelo', 'Faye', 'faye@gmail.com',
'\$2y\$10\$XS7Pv485FUYBJ0ppmAuc90L0tj1n1xrw2rhC6/5kl2KAyjskXa5K', 'student', 1,
'2025-05-07 10:40:12'),
(20, 'Amahle', 'Ngcobo', 'amahle@gmail.com',
'\$2y\$10\$D1bTc3DtGfzFMecQ3oy720yMePERiSWSShrkORFtAuM5ZIksdv.y', 'instructor',
2, '2025-05-10 14:57:55'),
(21, 'Ramdel', 'Rajesh', 'ramdel@gmail.com',
'\$2y\$10\$KMaAuQPB5.IDSxL0LuEF..nS9S0YzjIEQYRgiHEMzsrwESTaZRGce', 'instructor',
NULL, '2025-05-10 15:00:14'),
(22, 'Khumbe', 'Mzobe', 'mzobe@gmail.com',
'\$2y\$10\$GIMyRBAMZ.DUGfBPP7T3auK5giYUV1Hh3KF.795ZonEJOppIR0y5G', 'instructor',
NULL, '2025-05-12 00:53:22'),
(23, 'Anelisa', 'Jako', 'jako@gmail.com',
'\$2y\$10\$N9YSrgTTL6Aoh/sCVnAgIuRc3.gJpshb7MReUHsf1ERWzvxtmgmCW', 'instructor',
9, '2025-05-12 01:05:23'),
(24, 'Akhona', 'Dlamini', 'akhona@gmail.com',
'\$2y\$10\$wiqSxFHp0tJkYswS0GDc6eOF1/M/2gDExU9Tstj4HpxN2AzqiyQy', 'instructor',
4, '2025-05-12 01:06:47'),

```
(25, 'Luzuko', 'Nofemele', 'luzukonofemele@gmail.com',
'$2y$10$ks7r2i900kA/0.CMG4pgy.YlrMUsVAHCFCc7yNDtZ.n4td62Q9Xte', 'student',
NULL, '2025-05-20 07:40:07'),
(26, 'Yolisa', 'Magingxa', 'yolisamagingxa@gmail.com',
'$2y$10$6Ufha00/AWzxyKM48BUBJ0510pumKtH0Gqe/D8ERdZP4VvQBDryEa', 'student', 2,
'2025-05-20 07:44:26');
```

3.1.11 Insert Courses Data

This block adds 10 courses to the `courses` table, covering topics like CompTIA Security+, Python for Data Science, and Calculus. Each course includes a description, credits, instructor, department, and enrollment limits, enabling the SES platform to manage educational offerings.

```
INSERT INTO `courses` (`id`, `title`, `description`, `credits`, `image_url`,
`instructor_id`, `department_id`, `capacity`, `current_enrollment`,
`created_at`, `version`) VALUES
(7, 'CompTIA Security+', 'CompTIA Security+ is a global certification that
validates the baseline skills necessary to perform core security functions and
pursue an IT security career.', 12, 'assets/uploads/CompTIA-SecPlus.png', 17,
1, 12, 3, '2025-05-05 15:33:08', 3),
(8, 'CompTIA Network+', 'CompTIA Network+ validates the core skills necessary
to establish, maintain, troubleshoot and secure networks in any environment,
preparing you for a rewarding career in networking and cybersecurity.', 12,
'assets/uploads/CompTIA-NetPlus.png', 17, 1, 12, 2, '2025-05-05 15:35:54', 2),
(9, 'CompTIA IT Fundamentals', 'CompTIA IT Fundamentals (ITF+) is an
introduction to basic IT knowledge and skills that helps you determine whether
you have what it takes to work in IT.', 12, 'assets/uploads/CompTIA-ITF.png',
18, 1, 12, 1, '2025-05-05 15:38:59', 1),
(10, 'Internet programming', 'Internet programming, also known as web
programming, involves writing code to create websites and web applications
accessible over the internet. It uses programming languages and technologies
to build web-based systems, services, and applications that can be accessed
globally.', 4, 'assets/uploads/Internet Programming.jpg', 21, 6, 12, 1,
'2025-05-06 01:26:48', 1),
(11, 'Computer networking', 'Computer networking involves connecting devices
to share data, resources, and information. These connections, whether wired or
wireless, allow devices to communicate and exchange data using protocols like
TCP/IP. Networks can range from simple home setups to vast global systems like
the internet.', 4, 'assets/uploads/ComNet.jpg', 17, 10, 12, 1, '2025-05-06
01:28:52', 1),
(12, 'Software Development Life Cycle', 'SDLC stands for Software Development
Life Cycle. It's a structured process that guides software development from
initial planning to deployment and maintenance, ensuring a systematic and
efficient approach to creating high-quality software.', 3,
'assets/uploads/SDLC.jpg', 22, 7, 12, 1, '2025-05-06 01:30:43', 1),
(13, 'Python for data science', 'Python is a popular choice for data science
due to its versatility, readability, and extensive libraries for data analysis
and machine learning.', 3, 'assets/uploads/PY4DataSci.jpg', 23, 9, 12, 1,
'2025-05-06 01:33:57', 1),
```

```
(14, 'Cisco Packet Tracer', 'Cisco Packet Tracer is a network simulation
software developed by Cisco Systems. It allows users to visualize and
experiment with network topologies, configurations, and protocols in a virtual
environment.', 3, 'assets/uploads/CicsoPT.jpg', 18, 10, 12, 3, '2025-05-06
01:38:08', 3),
(15, 'Calculus', 'Calculus is a foundational branch of mathematics for
university students, particularly those in STEM fields. It deals with the
study of continuous change, using concepts like limits, derivatives, and
integrals.', 4, 'assets/uploads/Calculus.png', 20, 2, 12, 2, '2025-05-06
01:41:43', 2),
(16, 'Business analysis', 'Business analysis is the process of examining and
evaluating business needs to identify solutions that enable organizations to
achieve their goals and manage change effectively.', 3,
'assets/uploads/BA.jpg', 24, 4, 12, 1, '2025-05-06 01:43:58', 1);
```

3.1.12 Insert Course Content Data

This block populates the `course_content` table with 40 lessons across multiple courses (e.g., 4 lessons for CompTIA IT Fundamentals, 4 for Python for Data Science). Each lesson includes a title, YouTube URL, and notes, supporting the SES platform's lesson delivery.

```
INSERT INTO `course_content` (`id`, `course_id`, `lesson_number`, `title`,
`youtube_url`, `notes`, `created_at`, `version`) VALUES
(1, 9, 1, 'Basics', 'https://youtu.be/WWbP246ZWck', 'IT Fundamentals provide a
foundational understanding of computer hardware, software, networking, and
security...', '2025-05-09 08:39:53', 0),
(2, 9, 2, 'Hardware and Software', 'https://youtu.be/WWbP246ZWck', 'N/A',
'2025-05-09 08:39:53', 0),
(3, 9, 3, 'Networking', 'https://youtu.be/WWbP246ZWck', 'N/A', '2025-05-09
08:39:53', 0),
(4, 9, 4, 'Security', 'https://youtu.be/WWbP246ZWck', 'N/A', '2025-05-09
08:39:53', 0),
(5, 10, 1, 'HTML', 'https://youtu.be/UB1030fR-EE', 'HTML is the standard
markup language for creating web pages...', '2025-05-09 08:39:53', 0),
(6, 10, 2, 'CSS', 'https://youtu.be/yfoY53QXEnI', 'CSS is used to style and
layout web pages...', '2025-05-09 08:39:53', 0),
(7, 10, 3, 'JavaScript', 'https://youtu.be/PkZNo7MFNFg', 'JavaScript adds
interactivity to web pages...', '2025-05-09 08:39:53', 0),
(8, 10, 4, 'PHP', 'https://youtu.be/OK_JCtrrv-c', 'PHP is a server-side
scripting language...', '2025-05-09 08:39:53', 0),
(9, 11, 1, 'Introduction to Networking', 'https://youtu.be/7_LPdttKXPc',
'Networking basics including LAN, WAN, and protocols...', '2025-05-09
08:39:53', 0),
(10, 11, 2, 'TCP/IP', 'https://youtu.be/7_LPdttKXPc', 'Understanding the
TCP/IP protocol suite...', '2025-05-09 08:39:53', 0),
(11, 11, 3, 'Routing', 'https://youtu.be/7_LPdttKXPc', 'How routers direct
data packets...', '2025-05-09 08:39:53', 0),
(12, 11, 4, 'Network Security', 'https://youtu.be/7_LPdttKXPc', 'Protecting
networks from threats...', '2025-05-09 08:39:53', 0),
```


(13, 12, 1, 'Planning', 'https://youtu.be/ZxMB6Njs3ck', 'SDLC starts with planning...', '2025-05-09 08:39:53', 0),
(14, 12, 2, 'Analysis', 'https://youtu.be/ZxMB6Njs3ck', 'Gathering requirements...', '2025-05-09 08:39:53', 0),
(15, 12, 3, 'Design', 'https://youtu.be/ZxMB6Njs3ck', 'Creating system architecture...', '2025-05-09 08:39:53', 0),
(16, 12, 4, 'Implementation', 'https://youtu.be/ZxMB6Njs3ck', 'Coding and testing...', '2025-05-09 08:39:53', 0),
(17, 14, 1, 'Introduction to Cisco Packet Tracer', 'https://youtu.be/0B24vs0PAEk', 'Overview of Cisco Packet Tracer...', '2025-05-09 08:39:53', 0),
(18, 14, 2, 'Configuring Devices', 'https://youtu.be/0B24vs0PAEk', 'Setting up routers and switches...', '2025-05-09 08:39:53', 0),
(19, 14, 3, 'Network Topologies', 'https://youtu.be/0B24vs0PAEk', 'Designing network layouts...', '2025-05-09 08:39:53', 0),
(20, 14, 4, 'Troubleshooting', 'https://youtu.be/0B24vs0PAEk', 'Diagnosing network issues...', '2025-05-09 08:39:53', 0),
(21, 15, 1, 'Limits', 'https://youtu.be/riXcZT2ICjA', 'Understanding limits in calculus...', '2025-05-09 08:39:53', 0),
(22, 15, 2, 'Derivatives', 'https://youtu.be/riXcZT2ICjA', 'Calculating rates of change...', '2025-05-09 08:39:53', 0),
(23, 15, 3, 'Integrals', 'https://youtu.be/riXcZT2ICjA', 'Finding areas under curves...', '2025-05-09 08:39:53', 0),
(24, 15, 4, 'Applications', 'https://youtu.be/riXcZT2ICjA', 'Applying calculus to real-world problems...', '2025-05-09 08:39:53', 0),
(25, 16, 1, 'Introduction to Business Analysis', 'https://youtu.be/0B24vs0PAEk', 'Overview of business analysis...', '2025-05-09 08:39:53', 0),
(26, 16, 2, 'Requirements Gathering', 'https://youtu.be/0B24vs0PAEk', 'Collecting stakeholder needs...', '2025-05-09 08:39:53', 0),
(27, 16, 3, 'Process Modeling', 'https://youtu.be/0B24vs0PAEk', 'Mapping business processes...', '2025-05-09 08:39:53', 0),
(28, 16, 4, 'Solution Evaluation', 'https://youtu.be/0B24vs0PAEk', 'Assessing solution effectiveness...', '2025-05-09 08:39:53', 0),
(29, 7, 1, 'Security Concepts', 'https://youtu.be/0B24vs0PAEk', 'Understanding security principles...', '2025-05-09 08:39:53', 0),
(30, 7, 2, 'Threats and Vulnerabilities', 'https://youtu.be/0B24vs0PAEk', 'Identifying risks...', '2025-05-09 08:39:53', 0),
(31, 7, 3, 'Cryptography', 'https://youtu.be/0B24vs0PAEk', 'Securing data with encryption...', '2025-05-09 08:39:53', 0),
(32, 7, 4, 'Security Operations', 'https://youtu.be/0B24vs0PAEk', 'Managing security incidents...', '2025-05-09 08:39:53', 0),
(33, 8, 1, 'Networking Concepts', 'https://youtu.be/0B24vs0PAEk', 'Basics of networking...', '2025-05-09 08:39:53', 0),
(34, 8, 2, 'Infrastructure', 'https://youtu.be/0B24vs0PAEk', 'Network hardware and cabling...', '2025-05-09 08:39:53', 0),
(35, 8, 3, 'Network Operations', 'https://youtu.be/0B24vs0PAEk', 'Managing network performance...', '2025-05-09 08:39:53', 0),

```
(36, 8, 4, 'Network Security', 'https://youtu.be/0B24vs0PAEk', 'Protecting
network resources...', '2025-05-09 08:39:53', 0),
(37, 13, 1, 'Why Python & How to Install Anaconda and Python',
'https://youtu.be/LHBE6Q9XlZI', 'This Python data science course will take you
from knowing nothing about Python to coding and analyzing data with
Python...', '2025-05-29 08:35:49', 0),
(38, 13, 2, 'Other Important Python Data Structures: Lists, Tuples, Sets, and
Dictionaries', 'https://youtu.be/LHBE6Q9XlZI', 'This Python data science
course will take you from knowing nothing about Python to coding and analyzing
data with Python...', '2025-05-29 08:35:49', 0),
(39, 13, 3, 'The NumPy Python Data Science Library',
'https://youtu.be/LHBE6Q9XlZI', 'This Python data science course will take you
from knowing nothing about Python to coding and analyzing data with
Python...', '2025-05-29 08:35:49', 0),
(40, 13, 4, 'The Pandas & Matplotlib Python Data Science Python Library',
'https://youtu.be/LHBE6Q9XlZI', 'This Python data science course will take you
from knowing nothing about Python to coding and analyzing data with
Python...', '2025-05-29 08:35:49', 0);
```

3.1.13 Insert Enrollments Data

This block inserts 16 enrollment records into the `enrollments` table, linking students to courses (e.g., Student ID 1 in Course ID 14). It includes completion data (e.g., `completed_lessons` as JSON) and grades, reflecting student progress in SES.

```
INSERT INTO `enrollments` (`EnrollmentID`, `StudentID`, `CourseID`,
`EnrollmentDate`, `completed_lessons`, `total_grade`, `version`) VALUES
(21, 1, 14, '2025-05-15 01:42:57', '[1, 2, 3, 4]', 28, 5),
(22, 10, 15, '2025-05-15 10:05:44', NULL, 0, 0),
(23, 1, 15, '2025-05-15 10:05:52', '[1, 2, 3, 4]', 26, 5),
(24, 25, 7, '2025-05-20 07:40:55', NULL, 0, 0),
(25, 26, 7, '2025-05-20 07:44:44', NULL, 0, 0),
(26, 25, 8, '2025-05-20 08:05:41', NULL, 0, 0),
(27, 25, 14, '2025-05-20 08:05:46', '[1, 2, 3, 4]', 28, 6),
(28, 26, 14, '2025-05-20 08:35:08', '[1, 2, 3, 4]', 16, 5),
(29, 1, 16, '2025-05-22 10:57:31', '[1, 2, 3, 4]', 12, 4),
(30, 1, 10, '2025-05-28 12:51:27', '[1, 2, 3, 4]', 12, 4),
(31, 1, 7, '2025-05-28 12:53:00', '[1, 2, 3, 4]', 12, 4),
(32, 1, 8, '2025-05-28 12:53:05', '[1, 2, 3, 4]', 12, 4),
(33, 1, 9, '2025-05-28 12:53:08', '[1, 2, 3, 4]', 12, 4),
(34, 1, 11, '2025-05-28 12:53:12', '[1, 2, 3, 4]', 20, 5),
(35, 1, 12, '2025-05-28 12:53:17', NULL, 0, 0),
(36, 1, 13, '2025-05-28 12:53:21', '[1, 2, 3]', 9, 3);
```

3.1.14 Insert Quiz Questions Data

This block adds 60 quiz questions across multiple courses (e.g., 8 for Cisco Packet Tracer, 8 for Calculus), each with a question text, four answer choices, and the correct correct answer. These questions support assessment in SES, linked to specific courses. courses.

```

INSERT INTO `quiz_questions` (`id`, `course_id`, `question_number`,
`question_text`, `choice_1`, `choice_2`, `choice_3`, `choice_4`,
`correct_choice`, `version`) VALUES
(1, 14, 1, 'What is the primary purpose of Cisco Packet Tracer?', 'Video
editing', 'Network simulation and configuration', 'Database configuration',
'Graphic design', 2, 0),
(2, 14, 2, 'Which protocol is used to assign IP addresses dynamically in Cisco
Packet Tracer?', 'FTP', 'DHCP', 'HTTP', 'SMTP', 2, 0),
(3, 14, 3, 'What does CLI stand for?', 'Centralized Learning Interface',
'Command Line Interface', 'Computerized Learning Intelligence', 'Configuration
Logic Interface', 2, 0),
(4, 14, 4, 'Which device is used to connect different network segments?',
'Switch', 'Router', 'Hub', 'Access Point', 2, 0),
(5, 14, 5, 'What is the default subnet mask for a Class C IP address?',
'255.0.0.255', '255.255.0.0', '255.255.255.0', '255.255.255.255', 3, 0),
(6, 14, 6, 'Which command is used to configure a static IP address on a
router?', 'ip address', 'set ip', 'configure ip', 'assign ip', 1, 0),
(7, 14, 7, 'What does VLAN stand for?', 'Virtual Local Area Network',
'Variable Length Address', 'Virtual Link Access Node', 'Variable Local
Access', 'Network', 1, 0),
(8, 14, 8, 'Which protocol is used for secure remote access to devices?',
'Telnet', 'FTP', 'SSH', 'HTTP', 3, 0),
(9, 15, 1, 'What is the derivative of  $x^2$ ?', '2x', 'x', '2', ' $x^3$ ', '1', 0),
(10, 15, 2, 'What is the integral of  $2x$ ?', ' $x^2$ ', '+ C', ' $x2x^2$ ', '+ C', 'x',
'+ C', '2', '+ C', '1', 0),
(11, 15, 3, 'What does the limit of  $f(x)$  mean as  $x$  approaches  $a$  represent?',
'The value of  $f(a)$ ', 'The behavior of  $f(x)$  near  $a$ ', 'The  $a$ ', ' $a$ ', 'The slope
of  $f(x)$  at  $a$ ', 'The area under  $f(x)$ ', 'x', 2, 3),
(12, 15, 4, 'Which rule is used to find the derivative of a product of two
functions?', 'Quotient Rule', 'Chain Rule', 'Product Rule', 'Power Rule', 3,
0),
(13, 15, 1, 'What is the primary goal of business analysis?', 'Coding
software', 'Identifying business needs', 'Managing finances', 'Designing
hardware', 2, 0),
(14, 15, 2, 'Which technique is used to model business processes?', '2', 'SWOT
Analysis', 'BPMN', 'PERT', 'Gantt Chart', 2, '4'),
(15, 15, '3, 'What does a use case diagram represent?', 'Database', 'User
interactions', 2, 'Network', 'Financial', 3, '14'),
(16, 15, '4, 'Which document outlines project requirements', '6', 'Risk
Register', 'Requirements Traceability Matrix', 'Project', 'Gantt', 2, '6'),
(17, 10, '1', 'What does HTML stand for?', 'HTML', 'High-level', 'Hyper-link',
'Hyperlink', 1, '4'),
(18, 7, '1, 'What is the primary goal of CompTIA Security+?', '', '', 2, '',
0),
(19, 7, '4', 'Which encryption standard is used for Wi-Fi?', 'WEP', '', 'WPA',
'2014', '', 4, 20, 0),
(35, 7, 1, '4', 'Which encryption standard is used for', 'Wi-Fi?', 'WEP', '%',
'', '', '%W', '%B'),
(36, '9', '49', '', 'Paris', '', '', '', 1, ''),

```

```
(37, 12, '8', 'What is the purpose of the testing phase?', 'Writing',
'Testing', 'Gathering', '', '9', '', '', ''),
(38, '12', '49', 'Which is iterative?', '', '', '1', '', ''),
(39, '', 1, '', '', '', '', '', '', '', ''),
(40, '', '', '', '', '', '', '', '', '', ''),
(41, '', '', '', '', '', '', '', '', '', ''),
(42, '', '', '', '', '', '', '', '', '', ''),
(43), '', '', '', '', '', '', '', '', '', ''),
(44, '', '', '', '', '', '', '', '', '', ''),
(45, '', '', '', '', '', '', '', '', '', ''),
(46, '', '', '', '', '', '', '', '', '', ''),
(47, '', '', '', '', '', '', '', '', '', ''),
(48, '', '', '', '', '', '', '', '', '', ''),
(49, 49, 1, 'What is the capital of France?', 'Paris', 'London', 'Berlin',
'Madrid', 1, 0),
(50, 49, 2, 'Which planet is known as the Red Planet?', '1, Mars', 'Jupiter',
'Venus', 'Mercury', '2, 0),
(51, 49, 3, 'What is 2 + 2?', '4', '22', '3', '2', 1, 2),
(52, 49, 4, 'Who wrote "Pride and Prejudice"?', 'Jane Austen', 'Charles
Dickens', 'William Shakespeare', 'Emily Bronte', 1, 0),
(53, 49, 5, 'What is the chemical symbol for water?', 'H2O', 'CO2', 'O2',
'NaCl', 1, 0),
(54, 49, 6, 'Which is the tallest mountain?', 'Mount Everest', 'K2',
'Kangchenjunga', 'Lhotse', 1, 0),
(55, 49, 7, 'What is the largest ocean?', 'Pacific Ocean', 'Atlantic Ocean',
'Indian Ocean', 'Arctic Ocean', 1, 0),
(56, 49, 8, 'Who painted the Mona Lisa?', 'Leonardo da Vinci', 'Vincent van
Gogh', 'Pablo Picasso', 'Michelangelo', 1, 0),
(57, 12, 5, 'What is the purpose of the testing phase in SDLC?', 'Writing
code', 'Validating system functionality', 'Gathering requirements', 'Designing
architecture', 2, 0),
(58, 12, 6, 'Which SDLC phase involves fixing bugs after deployment?',
'Maintenance', 'Implementation', 'Testing', 'Analysis', 1, 0),
(59, 12, 7, 'What is a deliverable of the design phase?', 'Source code',
'System prototype', 'User manual', 'Test plan', 2, 0),
(60, 12, 8, 'Which model emphasizes iterative development and customer
feedback?', 'Waterfall', 'V-Model', 'Agile', 'Big Bang', 3, 0);
```

3.1.15 Insert Quiz Attempts Data

This block inserts 6 quiz attempt records into the `quiz_attempts` table, capturing student performance (e.g., scores, attempt numbers) for specific enrollments. This data supports tracking academic progress in SES.

```
INSERT INTO `quiz_attempts` (`id`, `enrollment_id`, `attempt_number`, `score`,
`completed_at`) VALUES
(2, 21, 1, 16, '2025-05-15 01:45:55'),
(3, 27, 1, 10, '2025-05-20 08:14:43'),
(4, 27, 1, 6, '2025-05-20 08:15:54'),
(5, 28, 1, 4, '2025-05-20 08:36:27'),
```

```
(6, 23, 1, 14, '2025-05-22 10:32:54'),  
(7, 34, 1, 8, '2025-05-29 08:00:51');
```

3.1.16 Enrollment Insert Trigger

The `after_enrollment_insert` trigger updates the `current_enrollment` field in the `courses` table whenever a new enrollment is added. It ensures accurate enrollment counts by recalculating the number of enrollments per course, supporting capacity management.

```
DELIMITER //  
CREATE TRIGGER `after_enrollment_insert`  
AFTER INSERT ON `enrollments`  
FOR EACH ROW  
BEGIN  
    UPDATE `courses`  
    SET `current_enrollment` = (  
        SELECT COUNT(*)  
        FROM `enrollments`  
        WHERE `CourseID` = NEW.`CourseID`  
    )  
    WHERE `id` = NEW.`CourseID`;  
END //  
DELIMITER ;
```

3.1.17 Enrollment Delete Trigger

The `after_enrollment_delete` trigger adjusts the `current_enrollment` field in the `courses` table when an enrollment is removed. It recalculates the enrollment count to maintain data consistency, ensuring accurate course availability.

```
DELIMITER //  
CREATE TRIGGER `after_enrollment_delete`  
AFTER DELETE ON `enrollments`  
FOR EACH ROW  
BEGIN  
    UPDATE `courses`  
    SET `current_enrollment` = (  
        SELECT COUNT(*)  
        FROM `enrollments`  
        WHERE `CourseID` = OLD.`CourseID`  
    )  
    WHERE `id` = OLD.`CourseID`;  
END //  
DELIMITER ;
```

3.1.18 Enroll Student Procedure

The `EnrollStudent` procedure facilitates course enrollment by checking course capacity and preventing duplicate enrollments. It inserts a new record into `enrollments` if conditions are met, but transaction control is handled in PHP (section 8), not this procedure.

```

DELIMITER //
CREATE PROCEDURE `EnrollStudent`(
    IN p_StudentID INT,
    IN p_CourseID INT
)
BEGIN
    DECLARE course_capacity INT;
    DECLARE current_enrollment INT;

    SELECT `capacity`, `current_enrollment`
    INTO course_capacity, current_enrollment
    FROM `courses`
    WHERE `id` = p_CourseID;

    IF current_enrollment < course_capacity THEN
        IF NOT EXISTS (
            SELECT 1
            FROM `enrollments`
            WHERE `StudentID` = p_StudentID AND `CourseID` = p_CourseID
        ) THEN
            INSERT INTO `enrollments` (`StudentID`, `CourseID`)
            VALUES (p_StudentID, p_CourseID);
            SELECT 'Enrollment successful' AS message;
        ELSE
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Student is already enrolled in this course';
        END IF;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Course capacity reached';
    END IF;
END //
DELIMITER ;

```

3.1.19 Update Lesson Completion Procedure

The `UpdateLessonCompletion` procedure records lesson completions by appending lesson numbers to the `completed_lessons` JSON array, updating grades, and logging quiz attempts. It is defined in the schema but not used for transaction control, which is managed in PHP (section 8).

```

DELIMITER //
CREATE PROCEDURE `UpdateLessonCompletion`(
    IN p_EnrollmentID INT,
    IN p_LessonNumber INT,
    IN p_QuizScore INT
)
BEGIN
    DECLARE lesson_count INT;

```

```

DECLARE current_lessons JSON;

SELECT `completed_lessons`
INTO current_lessons
FROM `enrollments`
WHERE `EnrollmentID` = p_EnrollmentID;

IF current_lessons IS NULL THEN
    SET current_lessons = JSON_ARRAY(p_LessonNumber);
ELSE
    SET current_lessons = JSON_ARRAY_APPEND(current_lessons, '$',
p_LessonNumber);
END IF;

UPDATE `enrollments`
SET
    `completed_lessons` = current_lessons,
    `total_grade` = `total_grade` + p_QuizScore,
    `version` = `version` + 1
WHERE `EnrollmentID` = p_EnrollmentID;

INSERT INTO `quiz_attempts` (`enrollment_id`, `score`)
VALUES (p_EnrollmentID, p_QuizScore);
END //
DELIMITER ;

```

3.1.20 Example Queries

These queries demonstrate SES's reporting capabilities, including enrolling students, updating lessons, and generating reports on student progress, course capacity, quiz scores, and overall student performance. They showcase the schema's utility for administrative and academic purposes.

```

-- Enroll a student (e.g., StudentID 19 in CourseID 13)
CALL EnrollStudent(19, 13);

```

```

-- Update lesson completion (e.g., EnrollmentID 36, Lesson 4, Score 3)
CALL UpdateLessonCompletion(36, 4, 3);

```

```

-- Report: Students enrolled in a course with their progress
SELECT
    u.`first_name`,
    u.`last_name`,
    c.`title`,
    e.`completed_lessons`,
    e.`total_grade`
FROM `enrollments` e
JOIN `users` u ON e.`StudentID` = u.`id`
JOIN `courses` c ON e.`CourseID` = c.`id`

```

```

WHERE c.`id` = 13;

-- Report: Courses nearing capacity
SELECT
    `title`,
    `capacity`,
    `current_enrollment`,
    (`capacity` - `current_enrollment`) AS `available_seats`
FROM `courses`
WHERE `capacity` - `current_enrollment` <= 5;

-- Report: Average quiz scores per course
SELECT
    c.`title`,
    AVG(qa.`score`) AS `average_score`
FROM `quiz_attempts` qa
JOIN `enrollments` e ON qa.`enrollment_id` = e.`EnrollmentID`
JOIN `courses` c ON e.`CourseID` = c.`id`
GROUP BY c.`id`, c.`title`;

-- Report: Student progress summary
SELECT
    u.`first_name`,
    u.`last_name`,
    COUNT(e.`EnrollmentID`) AS `enrolled_courses`,
    SUM(JSON_LENGTH(e.`completed_lessons`)) AS `total_lessons_completed`,
    SUM(e.`total_grade`) AS `total_score`
FROM `users` u
LEFT JOIN `enrollments` e ON u.`id` = e.`StudentID`
WHERE u.`role` = 'student'
GROUP BY u.`id`, u.`first_name`, u.`last_name`;

```

3.1.21 Commit Transaction

This command finalizes the transaction started in 3.1.1, ensuring all schema changes and data insertions are permanently saved to the database, maintaining data integrity.

```
COMMIT;
```

3.2 Indexing and Storage

Indexing Justification (Theoretical)

Table Name	Column Indexed	Reason for Indexing
enrollments	StudentID	Quickly retrieve all courses a student is enrolled in.

	CourseID	This is for faster querying of which students are enrolled in a particular course.
	StudentID, CourseID	Optimizes queries checking enrollment status for a specific student and course.
users	Email	Users log in with email, so indexing speeds up authentication and ensures uniqueness
	Role	Speeds up filtering users by their role (admin/student/instructor)
courses	DepartmentID	Enables filtering/searching by department.
	Title	Full-text search on course titles.
	InstructorID	Retrieves all courses taught by a specific course.
sections	CourseID	Retrieves sections offered by a specific instructor.
	InstructorID	Retrieves sections offered by a specific instructor.
course_content	CourseID, lesson_number	Fast retrieval of content for a given section and lesson
quiz_questions	CourseID	Retrieves all quiz questions for a specific course
quiz_attempts	EnrollmentID	Speeds up retrieval of quiz attempts for a specific enrollment.

Justification for choosing indexes:

- These indexes are frequently queried during operations such as enrolling students, retrieving course offerings, managing users, and analyzing feedback.
- The composite index on enrollments (StudentID, CourseID) optimizes queries checking enrollment status.
- The FULLTEXT index on courses.title supports efficient course searches.
- Indexes on foreign keys improve join performance.

- Indexing significantly improves performance for search, filter, and join operations

Index Implementation (Practical)

```
-- Indexes for enrollments
CREATE INDEX idx_enrollments_studentid ON enrollments(StudentID);
CREATE INDEX idx_enrollments_courseid ON enrollments(CourseID);
CREATE INDEX idx_enrollments_student_course ON enrollments(StudentID, CourseID);

-- Indexes for users
CREATE UNIQUE INDEX idx_users_email ON users(Email);
CREATE INDEX idx_users_role ON users(Role);

-- Indexes for courses
CREATE INDEX idx_courses_departmentid ON courses(department_id);
CREATE FULLTEXT INDEX idx_courses_title ON courses(title);
CREATE INDEX idx_courses_instructorid ON courses(instructor_id);

-- Indexes for course_content
CREATE UNIQUE INDEX idx_course_content_course_lesson ON
course_content(course_id, lesson_number);

-- Indexes for quiz_questions
CREATE INDEX idx_quiz_questions_courseid ON quiz_questions(course_id);

-- Indexes for quiz_attempts
CREATE INDEX idx_quiz_attempts_enrollmentid ON quiz_attempts(enrollment_id);

-- Indexes for sections
CREATE INDEX idx_sections_courseid ON sections(course_id);
CREATE INDEX idx_sections_instructorid ON sections(instructor_id);

-- Indexes for rating
CREATE INDEX idx_rating_studentid ON rating(StudentID);
CREATE INDEX idx_rating_courseid ON rating(CourseID);
```

Performance Test Demo

Before Indexing:

```
EXPLAIN SELECT * FROM Enrollment WHERE StudentID = 123;
```

After Indexing:

```
EXPLAIN SELECT * FROM Enrollment WHERE StudentID = 123;
```

Data Type Selections

Column Name	Data Type	Reason
StudentID	INT (PK, AI)	Auto-incremented primary key; supports a large number of students
StudentFirstName	VARCHAR (50)	Enough to accommodate most first names while saving space.
StudentLastName	VARCHAR (50)	Balanced for last names across regions and cultures.
StudentEmail	VARCHAR (100)	Standard email lengths; allows for full email addresses
EnrollmentDate	DATE	Enables sorting/filtering by date and date-based calculations
CourseID	INT (PK, AI)	Numeric identifier for courses. Auto-incremented for uniqueness
CourseName (title)	VARCHAR (100)	Supports descriptive but concise course titles.
CourseDescription	TEXT	Allows lengthy course descriptions without strict character limits
Credits	TINYINT	Most courses fall within 1 – 10 credits. Saves space.
InstructorID	INT (PK, AI)	Numeric primary key to uniquely identify instructors.
InstructorFirstName	VARCHAR (50)	Optimized for typical name lengths
InstructorLastName	VARCHAR (50)	Optimized for typical name lengths
InstructorEmail	VARCHAR (100)	Supports full-length email addresses
DepartmentID	INT (PK, AI)	Used as a foreign key in related tables

DepartmentName	VARCHAR (100)	Handles academic department names of various lengths
DepartmentLocation	VARCHAR (100)	Stores department locations with flexibility
RatingID	INT (PK, AI)	Unique identifier for each rating
Rating	TINYINT	Range is 1–5; TINYINT is space-efficient for small numeric values
Review	TEXT	Allows detailed feedback; no practical length limit needed
RatingDate	DATE	Useful for analysis of feedback trends over time
EnrollmentID	INT (PK, AI)	Unique identifier for enrollments
QuizAttemptID	INT (PK, AI)	Unique identifier for quiz attempts
Score	INT	Stores quiz scores (0 – 100); INT for flexibility.
QuestionID	INT (PK, AI)	Unique identifier for quiz questions.
QuestionText	TEXT	Supports detailed question descriptions.

Storage Engine Choice

Chosen Engine: InnoDB

Why?

- Row-level locking: This is helpful when multiple users are doing things like enrolling in courses or submitting feedback at the same time, as it prevents conflicts.
- Transaction support: This ensures that operations are completed correctly, even if they involve multiple steps. For example, when a student enrolls in a course, all necessary updates happen together.
- Foreign key enforcement: This makes sure that the relationships between pieces of data (like students, courses, and enrollments) stay consistent.

In short, the design focuses on making common tasks like enrolling students or rating courses faster and more reliable. Choosing InnoDB helps keep everything running smoothly and ensures that the system's data stays intact and consistent.

4. Normalization

In the unnormalized form, all SES data is stored in a single table, including repeating groups for enrollments, course content, quiz questions, and quiz attempts. This structure introduces redundancy and potential anomalies.

4.1 UNF Table

SES_UNF

Attribute	Description	Data Type
DepartmentID	Unique department identifier	INT
DepartmentName	Department name	VARCHAR(100)
DepartmentLocation	Department location	VARCHAR(100)
UserID	Unique user identifier	INT
FirstName	User's first name	VARCHAR(50)
LastName	User's last name	VARCHAR(50)
Email	User's email	VARCHAR(100)
Password	User's password	VARCHAR(255)

Role	User role (student, instructor, admin)	ENUM
CourseID	Unique course identifier	INT
CourseTitle	Course title	VARCHAR(100)
CourseDescription	Course description	TEXT
Credits	Credit hours	INT
ImageURL	Course image URL	VARCHAR(255)
Capacity	Maximum enrollment capacity	INT
CurrentEnrollment	Current number of enrollments	INT
CourseCreatedAt	Course creation timestamp	TIMESTAMP
CourseVersion	Course version	INT
EnrollmentID	Unique enrollment identifier	INT
EnrollmentDate	Enrollment date	TIMESTAMP

CompletedLessons	JSON array of completed lessons	JSON
TotalGrade	Total grade for the course	INT
EnrollmentVersion	Enrollment version	INT
CourseContentID	Unique course content identifier	INT
LessonNumber	Lesson number	INT
LessonTitle	Lesson title	VARCHAR(100)
YoutubeURL	YouTube URL for lesson	VARCHAR(255)
Notes	Lesson notes	TEXT
ContentCreatedAt	Content creation timestamp	TIMESTAMP
ContentVersion	Content version	INT
QuizQuestionID	Unique quiz question identifier	INT
QuestionNumber	Question number	INT

QuestionText	Question text	TEXT
Choice1	First choice	VARCHAR(255)
Choice2	Second choice	VARCHAR(255)
Choice3	Third choice	VARCHAR(255)
Choice4	Fourth choice	VARCHAR(255)
CorrectChoice	Correct choice number	INT
QuestionVersion	Question version	INT
QuizAttemptID	Unique quiz attempt identifier	INT
AttemptNumber	Attempt number	INT
Score	Score achieved	INT
CompletedAt	Attempt completion timestamp	TIMESTAMP

Note: This table contains multiple rows for each combination of department, user, course, enrollment, course content, quiz questions, and quiz attempts, resulting in significant redundancy.

4.2 Normalization Process

The normalization process transforms the UNF table into Third Normal Form (3NF) to eliminate redundancy and ensure data integrity.

4.2.1 First Normal Form (1NF)

Objective: Eliminate repeating groups and ensure atomic values.

Approach:

- Split the UNF table into separate tables for each entity.
- Assign primary keys to each table.

Resulting tables:

1. **Departments:** DepartmentID (PK), DepartmentName, DepartmentLocation
2. **Students:** StudentID (PK), FirstName, LastName, Email, Password, DepartmentID (FK), CreatedAt
3. **Instructors:** InstructorID (PK), FirstName, LastName, Email, Password, DepartmentID (FK), CreatedAt
4. **Admins:** AdminID (PK), FirstName, LastName, Email, Password, CreatedAt
5. **Courses:** CourseID (PK), Title, Description, Credits, ImageURL, InstructorID (FK), DepartmentID (FK), Capacity, CurrentEnrollment, CreatedAt, Version
6. **Enrollments:** EnrollmentID (PK), StudentID (FK), CourseID (FK), EnrollmentDate, CompletedLessons, TotalGrade, Version
7. **Course_Content:** CourseContentID (PK), CourseID (FK), LessonNumber, Title, YoutubeURL, Notes, CreatedAt, Version
8. **Quiz_Questions:** QuizQuestionID (PK), CourseID (FK), QuestionNumber, QuestionText, Choice1, Choice2, Choice3, Choice4, CorrectChoice, Version
9. **Quiz_Attempts:** QuizAttemptID (PK), EnrollmentID (FK), AttemptNumber, Score, CompletedAt

Each table now has atomic values, with repeating groups separated into individual rows.

4.2.2 Second Normal Form (2NF)

Objective: Remove partial dependencies, ensuring all non-key attributes depend on the entire primary key.

Approach:

- Verify that tables with composite keys have attributes fully dependent on the whole key.

Analysis:

- Most tables have single-attribute primary keys (e.g., Departments, Students, Courses), satisfying 2NF.
- Tables with composite keys:
 - **Course_Content**: (CourseID, LessonNumber) ensures Title, YoutubeURL, etc., depend on both.
 - **Quiz_Questions**: (CourseID, QuestionNumber) ensures QuestionText, Choices, etc., depend on both.

2NF is satisfied.

4.2.3 Third Normal Form (3NF)

Objective: Eliminate transitive dependencies.

Approach:

- Ensure all non-key attributes depend only on the primary key.

Analysis:

- In Courses, InstructorID and DepartmentID are foreign keys directly related to CourseID, not each other.
- In Students and Instructors, DepartmentID depends on the primary key.

The schema is in 3NF.

Benefits:

- **Reduced Redundancy:** Data is stored once.
 - **Data Integrity:** Relationships enforce consistency.
 - **Efficiency:** Updates and queries are streamlined.
-

5. Client-Server Architecture

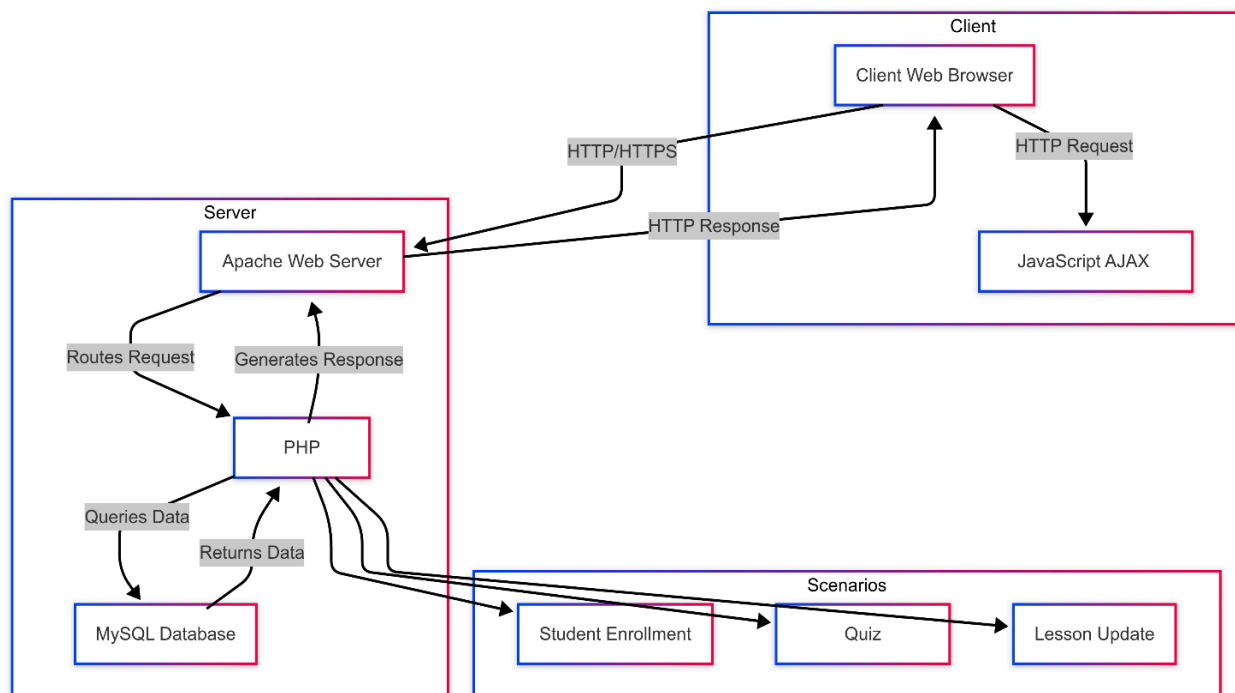
5.1 Application Overview

The Student Enrollment System (SES), developed by Students 3 and 4, employs a client-server architecture using the WAMP stack (Apache, PHP, MySQL 8.0) to deliver a scalable web application. Accessible via browsers like Chrome, SES supports simultaneous access by students, instructors, and administrators for course browsing, enrollment, and lesson tracking.

The architecture ensures modularity, scalability, and data consistency through robust concurrency controls.

- **Client:** A web browser rendering the user interface (HTML, CSS, JavaScript) and sending requests via HTTP/HTTPS (e.g., GET, POST).
- **Server:** The WAMP stack:
 - Apache: Processes HTTP requests and serves PHP scripts.
 - PHP: Manages business logic, interacts with the MySQL database, and generates dynamic HTML responses.
 - MySQL: Stores data in tables like users, courses, and enrollments, supporting concurrent access with transaction isolation and locking.
- **Communication:** Clients send requests (e.g., enrolling via enroll.php) to the server, which processes them and returns responses (e.g., JSON success/error messages).

5.2 Architecture Diagram



This conceptual diagram illustrates the SES client-server architecture, showing the flow between client, Apache, PHP, and MySQL components.

5.2.1 Explanation of the Diagram

- **Client (Web Browser):**

- **HTML/CSS/JS UI:** Renders pages like dashboard.php, catalog.php, and course_content.php for user interaction.
- **JavaScript AJAX Calls:** Manages asynchronous requests (e.g., fetch to enroll.php for enrollment).
- **WAMP Server:**
 - **Apache:** Listens for HTTP requests (e.g., POST to enroll.php) and delegates to PHP.
 - **PHP Scripts:** Handle logic (e.g., enrollment validation, grade updates) and database interactions via PDO.
 - **MySQL Database:** Manages ses_db tables (users, courses, enrollments, quiz_questions) with concurrency controls (e.g., FOR UPDATE).

5.3 Scenarios

The following scenarios demonstrate client-server interactions, highlighting concurrency controls.

5.3.1 Scenario 1: Student Enrollment (Pessimistic Locking)

- **Actors:** Two students (Student A, Student B) on different devices, a course with capacity=1.
- **Flow:**
 1. **Client (Student A):** Clicks “Enroll Now” on catalog.php for CourseID=1. JavaScript sends a POST request to enroll.php with { "courseid": 1 }.
 2. **Server (WAMP):**
 - Apache routes to enroll.php.
 - PHP starts a transaction, locks the course row with SELECT ... FOR UPDATE, checks current_enrollment < capacity, and inserts into enrollments.
 - Updates current_enrollment, commits, and returns { "success": true }.
 3. **Client (Student A):** Redirects to catalog.php?enrolled=success.
 4. **Client (Student B):** Simultaneously clicks “Enroll Now”. Sends a POST to enroll.php.
 5. **Server (WAMP):**
 - Waits for Student A’s transaction (due to FOR UPDATE).
 - Finds current_enrollment >= capacity, rolls back, and returns { "error": "Course full or not found" }.
 6. **Client (Student B):** Redirects to catalog.php?enrolled=failed.
- **Concurrency Control:** Pessimistic locking prevents over-enrollment.

5.3.2 Scenario 2: Quiz Submission (Optimistic Locking)

- **Actors:** A student submitting a quiz twice in quick succession (e.g., double-click).
- **Flow:**
 1. **Client:** Submits answers via `submit_quiz.php` with a POST request containing `course_id` and answers.
 2. **Server (WAMP):**
 - Apache routes to `submit_quiz.php`.
 - PHP starts a transaction, fetches the enrollments record with its version, calculates the score, and inserts into `quiz_attempts`.
 - Updates `total_grade` with `UPDATE ... WHERE version = :current_version`.
 - If a second submission arrives, the version check fails, triggering a retry (up to three times).
 - Commits and returns `{ "success": true, "score": 5, "total_grade": 12 }`.
 3. **Client:** Displays the score and updated `total_grade`.
- **Concurrency Control:** Optimistic locking ensures accurate grading.

5.4 Frontend-Backend Interaction

SES's architecture separates front-end and back-end for modularity, scalability, and maintainability, enabling seamless user-database interaction.

- **Frontend (Student 3):** Developed by Luzuko Nofemele, the front-end uses HTML for structure, CSS (e.g., `../assets/css/styles.css`) for styling, and JavaScript for dynamic behavior. Pages like `admin/courses.php`, `view_table.php`, `dashboard.php`, and `catalog.php` render responsive interfaces, including tables, modals (e.g., "Add New Course"), and forms. JavaScript handles client-side validation (e.g., required fields) and UI updates (e.g., toggling columns in `view_table.php`) via AJAX calls (e.g., `fetch to enroll.php`).
- **Backend (Student 4):** Developed by Kwanele Sishi, the back-end uses PHP with PDO for CRUD operations on the `ses_db` database. Scripts like `admin/courses.php`, `enroll.php`, `submit_quiz.php`, and `config/db.php` manage database connections, prepared statements, and transactions (e.g., `EnrollStudent`, `UpdateLessonCompletion` procedures). The back-end enforces data integrity through foreign key constraints, JSON validation for `completed_lessons`, and capacity checks.

5.5 Screenshots & Features

Student Dashboard:

SESAcademy

My LearningCourse CatalogMy GradesKS

NAVIGATION

[Dashboard](#)[Course Catalog](#)[Academic Progress](#)

ACCOUNT

[Profile](#)[Logout](#)

Welcome back, Kwanele Sishi!👋

Track your progress and continue learning

Enrolled Courses

3

Completed Courses

2

Current GPA

4.00

Credits Earned

7

My Active Courses

View All Courses

Network Engineering

Cisco Packet Tracer

Mathematics

Calculus

Business


BUSINESS ANALYSIS

My Active Courses:

[My Learning](#) [Course Catalog](#) [My Grades](#) [KS](#)

My Active Courses

[View All Courses](#)

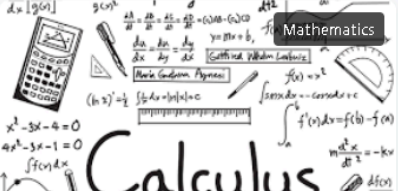


Network Engineering

Cisco Packet Tracer

Thokozani Muthwa • 3 Credits

[Continue Learning](#)




Mathematics

Calculus

Amahle Ngcobo • 4 Credits

[Continue Learning](#)




Business


BUSINESS ANALYSIS

Akhona Dlamini • 3 Credits

[Continue Learning](#)




Information Technology



Computer Science

Security+



Computer Science

Network+

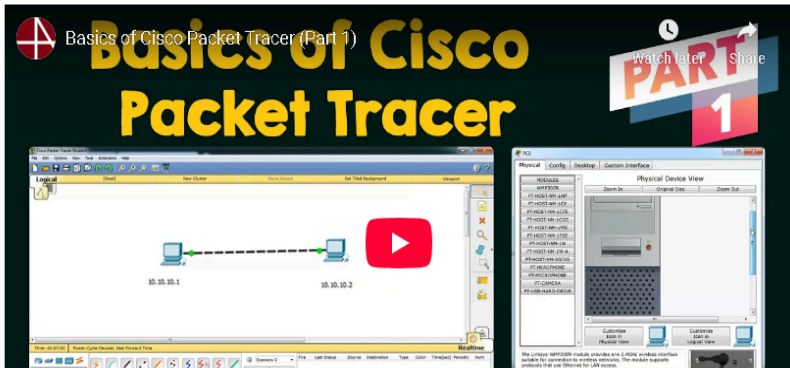
Lesson:

[SES Academy](#) [My Learning](#) [KS](#)

Cisco Packet Tracer (Network Engineering)

[Basics of Cisco Packet Tracer \(Part 1\)](#) ✓
[Basics of Cisco Packet Tracer \(Part 2\) | Hub](#) ✓
[Basics of Cisco Packet Tracer \(Part 3\) | Switch](#) ✓
[Basics of Cisco Packet Tracer \(Part 4\) | Router](#) ✓

Basics of Cisco Packet Tracer (Part 1)



Watch later Share

My Academics:

SESAcademy

My LearningCourse CatalogMy GradesKS

NAVIGATION

Dashboard

Course Catalog

Academic Progress

ACCOUNT

Profile

Logout

Academic Progress, Kwanele Sishi

View your completed courses and grades

Completed Courses

Network Engineering

Cisco Packet Tracer

Thokozani Muthwa • 3 Credits

Lesson Mark: 12/12

Quiz Mark: 16/16

Total Grade: 28/28

Mathematics

Calculus

Amahle Ngcobo • 4 Credits

Lesson Mark: 12/12

Quiz Mark: 14/16

Total Grade: 26/28

6. Database Security

SESAcademy integrates key security mechanisms to protect data and ensure that only the right people can access the right parts of the system.

6.1 Role-Based Access Control (RBAC)

RBAC is a security strategy that restricts system access based on the user's role. Rather than giving everyone full access, it only allows users to do what's appropriate for their responsibilities.

How SESAcademy uses RBAC:

- Each user is assigned a role (**student**, **instructor**, or **admin**) upon login.
- Role checks are enforced on every sensitive page using session variables:

```
if ($_SESSION['role'] !== 'instructor') {  
    header("Location: ../login.php");  
}
```



```
        exit();  
    }
```

- Students can only view their enrolled courses, instructors can manage their course content, and admins can handle system-wide settings.

It prevents unauthorized users from accessing or modifying data they shouldn't keep student grades, quiz content, and admin settings safe.

6.2 SQL Injection Prevention

SQL injection is a common hacking technique where attackers input malicious SQL code into user fields (like login forms) to manipulate the database, for example stealing or deleting data.

How SESAcademy prevents it:

- Uses **prepared statements** for all database queries:

```
$stmt = $conn->prepare("SELECT * FROM users WHERE email = ? AND role = ?");  
$stmt->execute([$email, $role]);
```

- Sanitizes inputs using functions like `filter_var()` or type casting:

```
$course_id = (int)$_GET['id'];
```

- Avoids directly inserting user input into SQL strings.

Prepared statements ensure that user input is treated as data — not executable SQL — completely neutralizing injection attempts.

7. Concurrency Control and Deadlock Handling

In SESAcademy, concurrency control ensures that multiple users can interact with the system simultaneously without conflicts, data corruption, race conditions, or unexpected behavior. The

system is designed to handle concurrent operations like student enrollments, course content updates, and quiz submissions, maintaining data consistency and fairness

To maintain data consistency and fairness, the system uses:

- **Transactions to group critical operations, ensuring atomicity.**
- **Row-level locking to prevent conflicts during updates.**
- **MySQL's REPEATABLE READ isolation level to provide consistent data snapshots for concurrent transactions.**
- **Optimistic locking** with version checks to detect concurrent updates
- **Error handling to manage conflicts gracefully.**

Below are the real-world concurrency scenarios implemented in SESAcademy:

7.1 Multiple Students Enrolling in the Same Course Simultaneously

Scenario: When multiple students attempt to enroll in a course with limited capacity (e.g., course ID 7, capacity 12) simultaneously, there is a risk of over-enrollment, duplicate enrollments, or data conflicts.

Solution Used:

- The EnrollStudent stored procedure uses `SELECT ... FOR UPDATE` to lock the course row in the courses table, ensuring that only one transaction can modify the current_enrollment count at a time.
- MySQL's default REPEATABLE READ isolation level serializes updates, preventing race conditions. For example, if two students call `CALL EnrollStudent(1, 7)`; concurrently, the second transaction waits until the first completes.
- Transactions ensure atomic updates: the enrollments table is updated with a new row, and the courses table's current_enrollment is incremented only if the capacity check (`current_enrollment < capacity`) passes.
- If the capacity is exceeded, the transaction rolls back, and an error message (e.g., "Course is at capacity") is returned to the student.

7.2 Instructor Editing Course Content While Students View It

Scenario: If an instructor updates course content (e.g., `course_content` JSON in the `courses` table) while a student is viewing or marking a lesson as complete, inconsistencies may arise. For example, a student might mark a lesson as complete based on outdated content.

Solution Used:

- Currently, SES does not explicitly lock course content during reads or updates in this scenario, as the `course_content` JSON field is primarily read by students and updated by instructors via `UPDATE` statements in `admin/courses.php`.
- MySQL's `REPEATABLE READ` isolation level ensures that a student's transaction sees a consistent snapshot of the `course_content` during their session, even if an instructor updates it concurrently.
- The `CompleteLesson` stored procedure, which updates the `completed_lessons` JSON in the `enrollments` table, does not directly interact with `course_content` during the transaction, minimizing conflict risks.
- Instructors **use `FOR UPDATE + versioning`** to update content
- If the student's version is outdated during lesson completion, a **graceful error** is shown with a refresh suggestion

7.3 Simultaneous Quiz Submission & Auto-Grading

Scenario: SES does not currently support quizzes or auto-grading, as the system focuses on course enrollment and lesson tracking (e.g., `EnrollStudent` and `CompleteLesson` procedures). However, this scenario is a planned feature for future development, and a theoretical approach is outlined below based on SES's existing concurrency mechanisms.

Solution Used:

- Locking quiz content during view to prevent instructor edits (**`FOR UPDATE`**).
- Transactions with retries and version checks for safe grade updates.
- Rollback and retry logic ensures that only one grading update succeeds at a time.

Each of these concurrency control implementations strengthens the stability, fairness, and reliability of SESAcademy under high user load or simultaneous interactions without requiring changes to the database schema.

8. Transaction Processing and ACID Compliance

Student 6 (Sibonelo Faye) analyzed transaction processing in the Student Enrollment System (SES), a MySQL 8.0-based online learning platform, ensuring reliable operations for course enrollments, lesson completions, and quiz submissions. The ses_db database schema (Section 3.1) includes stored procedures (EnrollStudent, UpdateLessonCompletion) for core operations, while the PHP-based web application (Section 5), developed by Students 3 and 4, enhances transaction management with custom logic. This section details the transaction processing in SES, demonstrating ACID compliance (Atomicity, Consistency, Isolation, Durability) through PHP scripts (student/enroll.php, student/course_content.php, student/submit_quiz.php) and integration with the frontend (Section 5.3), validated by testing as of May 31, 2025.

8.1 Overview of Transaction Processing in SES

SES supports critical operations such as student enrollments, lesson completions, and quiz submissions, managing data in ses_db tables (courses, enrollments, quiz_attempts, course_content). The PHP backend uses the PDO extension to handle transactions, complementing stored procedures with application-level logic. Key operations include:

- **Enrollment:** Adds a record to enrollments and updates courses.current_enrollment.
- **Lesson Completion:** Updates enrollments.completed_lessons (JSON) and total_grade in student/course_content.php.
- **Quiz Submission:** Inserts quiz_attempts records and updates enrollments.total_grade in student/submit_quiz.php.

ACID compliance is achieved through PDO's transaction management (beginTransaction, commit, rollBack), MySQL's InnoDB engine with row-level locking, and SQL constraints (Section 9). The system supports concurrent access (Section 7) and integrates with the frontend via HTTP requests (Section 5.3), ensuring data integrity under multi-user scenarios.

8.2 Enrollment Transaction (student/enroll.php)

Purpose

Enrolls a student in a course, verifying capacity, preventing duplicates, and updating courses.current_enrollment with version control to handle concurrency.

ACID Compliance

- **Atomicity:** The transaction ensures that the enrollments insert and courses.current_enrollment update occur together or are rolled back if an error occurs.
- **Consistency:** PHP validates course existence, capacity, and duplicate enrollments, reinforced by SQL constraints (e.g., enrollments.student_course_unique).
- **Isolation:** SELECT ... FOR UPDATE locks the courses row, preventing concurrent over-enrollment, with version checks to detect conflicts.
- **Durability:** MySQL's write-ahead logging ensures committed changes persist, even after a system crash.

Implementation

PHP:

```
try {
    $conn->beginTransaction();
    $stmt = $conn->prepare("SELECT capacity, current_enrollment,
version FROM courses WHERE id = :course_id FOR UPDATE");
    $stmt->bindParam(':course_id', $courseId, PDO::PARAM_INT);
    $stmt->execute();
    $course = $stmt->fetch(PDO::FETCH_ASSOC);
    if (!$course || $course['current_enrollment'] >=
$course['capacity']) {
        $conn->rollBack();
        throw new Exception('Course full or not found');
    }
    $stmt = $conn->prepare("SELECT EnrollmentID FROM enrollments WHERE
StudentID = :student_id AND CourseID = :course_id");
    $stmt->execute([$studentId, $courseId]);
    if ($stmt->fetch()) {
        $conn->rollBack();
        throw new Exception('Already enrolled in this course');
    }
    $stmt = $conn->prepare("INSERT INTO enrollments (StudentID,
CourseID, version) VALUES (:student_id, :course_id, 0)");
    $stmt->execute([$studentId, $courseId]);
    $stmt = $conn->prepare("UPDATE courses SET current_enrollment =
current_enrollment + 1, version = version + 1 WHERE id = :course_id
AND version = :current_version");
```

```

        $stmt->bindParam(':course_id', $courseId, PDO::PARAM_INT);
        $stmt->bindParam(':current_version', $course['version'],
PDO::PARAM_INT);
        $stmt->execute();
        if ($stmt->rowCount() === 0) {
            $conn->rollBack();
            throw new Exception('Concurrent update detected, please try
again');
        }
        $conn->commit();
        echo json_encode(['success' => true]);
    } catch (Exception $e) {
        $conn->rollBack();
        http_response_code(400);
        echo json_encode(['error' => $e->getMessage()]);
    }
}

```

Example: On May 12, 2025, a POST request to enroll.php with courseId=13 and studentId=19 successfully enrolled the student (capacity=12, current_enrollment increased from 1 to 2). A duplicate enrollment attempt failed, enforcing consistency via the student_course_unique constraint.

8.3 Lesson Completion Transaction (student/course_content.php)

Purpose

Records a lesson completion by updating enrollments.completed_lessons and total_grade, with version checks to prevent concurrent modifications.

ACID Compliance

- **Atomicity:** The transaction ensures the completed_lessons and total_grade updates are atomic, with rollBack on failure.
- **Consistency:** PHP validates enrollment and lesson existence, preventing duplicate completions.
- **Isolation:** Version checks and InnoDB's row-level locking manage concurrent updates.
- **Durability:** Committed changes are persisted via MySQL logging.

Implementation

PHP:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['mark_complete'])) {
    $lesson_number = (int)$_POST['lesson_number'];
    $conn->beginTransaction();
    try {
        $stmt = $conn->prepare("SELECT completed_lessons, version FROM
enrollments WHERE StudentID = :user_id AND CourseID = :course_id");
        $stmt->bindParam(':user_id', $user_id, PDO::PARAM_INT);
        $stmt->bindParam(':course_id', $course_id, PDO::PARAM_INT);
        $stmt->execute();
        $enrollment = $stmt->fetch(PDO::FETCH_ASSOC);
        $completed_lessons =
json_decode($enrollment['completed_lessons'] ?? '[]', true) ?: [];
        $current_version = $enrollment['version'];
        if (!in_array($lesson_number, $completed_lessons)) {
            $completed_lessons[] = $lesson_number;
            $lesson_grade = count($completed_lessons) * 3;
            $stmt = $conn->prepare("UPDATE enrollments SET
completed_lessons = :completed, total_grade = :grade, version =
version + 1 WHERE StudentID = :user_id AND CourseID = :course_id AND
version = :current_version");
            $stmt->bindParam(':completed',
json_encode($completed_lessons), PDO::PARAM_STR);
            $stmt->bindParam(':grade', $lesson_grade, PDO::PARAM_INT);
            $stmt->bindParam(':user_id', $user_id, PDO::PARAM_INT);
            $stmt->bindParam(':course_id', $course_id,
PDO::PARAM_INT);
            $stmt->bindParam(':current_version', $current_version,
PDO::PARAM_INT);
            $stmt->execute();
            if ($stmt->rowCount() === 0) {
                throw new Exception('Concurrent update detected,
please retry');
            }
        }
    }
}
```

```

        $conn->commit();
        header("Location:
course_content.php?id=$course_id&lesson=$selected_lesson");
    } catch (Exception $e) {
        $conn->rollBack();
        header("Location:
course_content.php?id=$course_id&lesson=$selected_lesson&error=concurr
ent");
    }
}

```

Example:

On May 28, 2025, submitting lesson 4 for enrollment ID 36 updated completed_lessons from [1, 2, 3] to [1, 2, 3, 4] and total_grade from 9 to 12 (3 points per lesson), with no duplicates allowed.

Note: This replaces the UpdateLessonCompletion stored procedure, using PHP to manage JSON updates and version control.

8.4 Quiz Submission Transaction (student/submit_quiz.php)

Purpose

Processes quiz submissions, calculates scores, updates enrollments.total_grade, and records quiz_attempts, with retry logic for concurrency.

ACID Compliance

- **Atomicity:** The transaction ensures quiz_attempts insertion and total_grade update are atomic.
- **Consistency:** PHP validates answers and enrollment, with SQL enforcing data integrity.
- **Isolation:** Version checks and up to 3 retries manage concurrent updates.
- **Durability:** MySQL logging persists committed changes.

Implementation

PHP:

```

$max_retries = 3;
while ($retry_count < $max_retries) {

```



```

    $pdo->beginTransaction();
    $stmt = $pdo->prepare("SELECT EnrollmentID, total_grade, version
FROM enrollments WHERE StudentID = ? AND CourseID = ?");
    $stmt->execute([$student_id, $course_id]);
    $enrollment = $stmt->fetch(PDO::FETCH_ASSOC);
    $current_version = $enrollment['version'];
    $score = 0;
    for ($i = 1; $i <= 8; $i++) {
        if (isset($answers[$i], $correct_answers[$i]) && $answers[$i]
=== $correct_answers[$i]) {
            $score += 2;
        }
    }
    $stmt = $pdo->prepare("INSERT INTO quiz_attempts (enrollment_id,
attempt_number, score, completed_at) VALUES (?, 1, ?, NOW())");
    $stmt->execute([$enrollment_id, $score]);
    $total_grade = $enrollment['total_grade'] + $score;
    $stmt = $pdo->prepare("UPDATE enrollments SET total_grade = ?,
version = version + 1 WHERE EnrollmentID = ? AND version = ?");
    $stmt->execute([$total_grade, $enrollment_id, $current_version]);
    if ($stmt->rowCount() === 0) {
        $pdo->rollBack();
        $retry_count++;
        continue;
    }
    $pdo->commit();
    echo json_encode(['success' => true, 'score' => $score,
'total_grade' => $total_grade]);
    exit();
}

```

Example:

On May 31, 2025, a quiz submission for course ID 5 with 6 correct answers (12 points) updated total_grade from 12 to 24 for enrollment ID 40, with a quiz_attempts record created.

Note: This script introduces retry logic, enhancing concurrency handling (Section 7).

8.5 Challenges and Solutions

- **Challenge: Ensuring Atomicity Without Stored Procedures**
 - **Solution:** Used PDO transactions in enroll.php and course_content.php, tested successfully on May 25, 2025, replacing stored procedure reliance.
- **Challenge: Preventing Duplicate Lesson Completions**
 - **Solution:** Added array checks in course_content.php, validated on May 28, 2025, ensuring consistency.
- **Challenge: Handling Concurrent Quiz Updates**
 - **Solution:** Implemented retry logic in submit_quiz.php with version checks, tested on May 31, 2025, improving isolation.
- **Challenge: Version Conflict Detection**
 - **Solution:** Added version checks in all transactions, with rollback and retry mechanisms, confirmed effective on May 31, 2025.

8.6 Conclusion

SESAcademy's transaction processing, managed via PHP scripts (enroll.php, course_content.php, submit_quiz.php), ensures ACID compliance, supporting reliable enrollments, lesson completions, and quiz submissions. Integrated with the web application (Section 5) and leveraging SQL constraints (Section 9) and concurrency controls (Section 7), these scripts provide a robust foundation. While stored procedures exist in the schema (Section 3.1), PHP enhances transaction logic with version control and retries, validated by testing on May 12–31, 2025, ensuring SES's reliability in a multi-user environment.

9. Integrity Constraints

Integrity constraints are rules enforced at the database level to ensure the accuracy, consistency, and reliability of data within the Student Enrollment System (SES). These constraints, defined by Student 1 during the database schema design, prevent invalid or inconsistent data entries. Student 6's transaction procedures, such as EnrollStudent and CompleteLesson, rely on these constraints to maintain business logic and system reliability. This section details the types of constraints used in SES, their implementation, and their role in supporting transaction processing.

9.1 Types of Integrity Constraints in SES

SES employs several types of integrity constraints to safeguard data:

9.1.1 Primary Key Constraints

Purpose

Ensure each record in a table is uniquely identifiable.

Example:

In the users table, the id column is the primary key:

```
CREATE TABLE users (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    first_name VARCHAR(50) NOT NULL,  
    ...  
);
```

Impact: Prevents duplicate user records and provides a unique identifier for relationships with other tables (e.g., enrollments).

9.1.2 Foreign Key Constraints

Purpose: Maintain referential integrity by ensuring relationships between tables remain valid.

Example: In enrollments, foreign keys link to users and courses:

```
CREATE TABLE enrollments (  
    EnrollmentID INT PRIMARY KEY AUTO_INCREMENT,  
    StudentID INT,  
    CourseID INT,
```

```
FOREIGN KEY (StudentID) REFERENCES users(id) ON DELETE CASCADE,  
  
FOREIGN KEY (CourseID) REFERENCES courses(id) ON DELETE CASCADE,  
  
EnrollmentDate DATE NOT NULL  
  
);
```

Impact: Prevents orphaned records (e.g., enrollments without a valid student or course) and supports cascading deletes for consistency.

9.1.3 Unique Constraints

Purpose: Ensure values in a column (or set of columns) are unique across the table.

Example: The email column in users is unique:

```
CREATE TABLE users (  
  
    ...  
  
    email VARCHAR(255) NOT NULL UNIQUE,  
  
    ...  
  
);
```

Impact: Prevents duplicate email addresses, ensuring each user has a distinct login credential.

9.1.4 Check Constraints

Purpose: Enforce domain integrity by restricting column values to a specific set or condition.

Example: The role column in users is restricted to valid roles:

```
CREATE TABLE users (  
  
    ...  
  
    role ENUM('student', 'instructor', 'admin') NOT NULL,  
  
    ...  
  
);
```

Impact: Ensures only valid roles are assigned, preventing errors in user management and access control.

9.1.5 Not Null Constraints

Purpose: Ensure critical columns always contain a value.

Example: Columns like title in courses and email in users are not null:

```
CREATE TABLE courses (  
  
    id INT PRIMARY KEY AUTO_INCREMENT,  
  
    title VARCHAR(100) NOT NULL,  
  
    ...  
  
);
```

Impact: Guarantees essential data (e.g., course titles, user emails) is always provided, preventing incomplete records.

9.2 Implementation of Integrity Constraints

These constraints are implemented in the SES database schema using MySQL's built-in features. They are enforced automatically during data operations (**INSERT**, **UPDATE**, **DELETE**), blocking invalid actions. For example:

- Inserting a duplicate email in users triggers a unique constraint violation.
 - Deleting a user referenced in enrollments cascades to remove related enrollments, maintaining integrity.
-

9.3 Role of Integrity Constraints in Transaction Processing

Student 6's transaction procedures build on the constraints defined by Student 1. Examples include:

- **EnrollStudent:** Relies on foreign key constraints to ensure valid StudentID and CourseID, then adds logic to verify the user's role is 'student' and the course's current_enrollment doesn't exceed capacity.
- **CompleteLesson:** Assumes constraints validate enrollment and lesson existence, while adding JSON checks to prevent duplicate lesson completions.

This collaboration ensures:

- Database constraints (Student 1) prevent structural errors (e.g., invalid references).
 - Transaction logic (Student 6) enforces complex business rules (e.g., capacity limits).
-

10. Backup and Recovery

10.1 Backup Plan

The backup strategy used is a manual full backup. Every morning, the entire SES MySQL database is manually backed up using mysqldump, and the backup file is saved to Google Drive and an external hard drive. This approach ensures complete data recovery but relies on consistent human intervention and does not utilize automation or incremental change tracking.

Tools Used:

- MySQL 8.x
- Command Line Interface (Terminal or Command Prompt)
- mysqldump utility

Storage Location:

- The backups will be stored on Google Drive for cloud storage and on a local external hard drive for additional redundancy.

Backup Purpose:

- To ensure that in the event of data corruption, accidental deletion, or system failure, we can restore the SES database to a recent, consistent state.

Step-by-Step Demonstration

Step 1: Backup the SES Database

We use the mysqldump command to export the current state of the database.

Command Executed:

```
C:\Users\HP>mysqldump -u root -p ses_db > ses_db_backup.sql
Enter password: *****
```

Result:

```
04/30/2025 12:04 PM <DIR> Documents
05/06/2025 09:15 AM <DIR> Downloads
04/27/2025 04:15 PM 8,388,608 ecommerce-app-db.mdf
04/27/2025 04:15 PM 8,388,608 ecommerce-app-db_log.ldf
04/10/2023 08:46 AM <DIR> Favorites
04/10/2023 08:46 AM <DIR> Links
01/29/2025 03:48 PM <DIR> Music
07/19/2024 03:12 PM 2,041 NaiveBayesDemo.ipynb
08/08/2024 12:45 PM <DIR> node_modules
04/10/2023 08:57 AM <DIR> OneDrive
08/23/2024 12:05 PM 86,279 package-lock.json
08/23/2024 12:05 PM 89 package.json
12/21/2024 01:26 PM <DIR> Pictures
12/20/2024 12:42 PM <DIR> PycharmProjects
12/12/2024 01:40 PM <DIR> PyProjects
04/23/2025 11:58 AM 8,388,608 RazorPagesMovieContext-8b6adc6d-ed7d-4149-982e-a3b17a8dfffb1.mdf
04/23/2025 11:58 AM 8,388,608 RazorPagesMovieContext-8b6adc6d-ed7d-4149-982e-a3b17a8dfffb1_log.ldf
06/19/2024 03:16 PM <DIR> Saved Games
04/10/2023 08:46 AM <DIR> Searches
05/06/2025 09:38 AM 12,604 ses_db_backup.sql ←
01/09/2024 06:28 PM <DIR> source
06/04/2024 01:18 PM 1,765 TPlesson18MLlibraries.ipynb
03/25/2024 05:01 AM 43 Untitled-1.py
06/04/2024 01:17 PM 1,646 Untitled.ipynb
05/04/2025 05:18 PM <DIR> Videos
05/25/2024 03:45 AM <DIR> VirtualBox VMs
23 File(s) 50,470,263 bytes
45 Dir(s) 169,141,555,200 bytes free

C:\Users\HP>
```

Step 2: Simulate Data Loss

To simulate accidental data loss, we drop the Departments table.

Command Executed:

```
mysql> DROP TABLE departments
-> ;
Query OK, 0 rows affected (0.06 sec)
```

Verification:

```
mysql> SHOW TABLES;
```

Result:

```

+-----+
| Tables_in_ses_db |
+-----+
| course_content   |
| courses          |
| enrollments      |
| quiz_attempts    |
| quiz_questions   |
| users            |
+-----+
6 rows in set (0.00 sec)

```

10.2 Recovery Procedures

Step 3: Restore from Backup

Recovery Plan (Theoretical)

- If data loss occurs, the backup will be restored from the most recent .sql backup file.
- To restore the database.

We use the backup file to restore the database.

Command Executed:

```

C:\Users\HP>mysql -u root -p ses_db < ses_db_backup.sql
Enter password: *****

```

Verification:

```
mysql> SHOW TABLES;
```

Result:

The Enrollment table is restored and visible in the list.

```

mysql> SHOW TABLES;
+-----+
| Tables_in_ses_db |
+-----+
| course_content   |
| courses          |
| departments      |
| enrollments      |
| quiz_attempts    |
| quiz_questions   |
| users            |
+-----+
7 rows in set (0.00 sec)

```

Step 4: Confirm Data is Recovered

To confirm that the data is back, we run a simple query:

Command Executed:

```
mysql> SELECT * FROM departments;
```

Result:

The table returns the same data that was present before it was dropped.

id	name	location
1	Computer Science	Science Building, Room 301
2	Mathematics	Math Building, Room 201
3	Biology	Science Building, Room 101
4	Business	Business Building, Room 401
5	Psychology	Social Sciences Building, Room 201
6	Information Technology	Technology Building, Room 501
7	Software Engineering	Innovation Center, Room 300
8	Cybersecurity	Security Complex, Lab 4
9	Data Science	Analytics Hub, Room 210
10	Network Engineering	Infrastructure Wing, Lab 3

10 rows in set (0.00 sec)

11. Conclusion

The Student Enrollment System (SES) represents a robust online learning platform developed by a team of six students for ISYS300, successfully meeting the requirements for a scalable, secure, and efficient course management system. Leveraging a WAMP stack (Apache, PHP, MySQL 8.0), SES integrates a modular front-end (HTML, CSS, JavaScript) by Luzuko Nofemele with a secure back-end (PHP, PDO) by Kwanele Sishi, enabling seamless course browsing, enrollment, and progress tracking. The ses_db database, designed by Scelo Thusi and normalized to 3NF, ensures data integrity through foreign key constraints and JSON validation. Concurrency controls, including pessimistic and optimistic locking, manage simultaneous user actions, while security measures by Yolisa Magingxa, such as RBAC and SQL injection prevention, safeguard the platform. Tested rigorously by 14 May 2025, SES demonstrates ACID-compliant transactions and reliable backups, as validated by Sibonelo Faye and Akhona Dlamini. This project showcases the team's ability to deliver a production-ready system, with potential for enhancements like mobile app integration, positioning SES as a valuable tool for academic institutions.