**Test Plan**

1. Data integrity tests
   1. Validate the primary keys are unique
   2. Ensure foreign key constraints are enforced
   3. Check for null values in non-nullable fields
2. CRUD Operations
   1. Create: Test inserting new records into each table.
   2. Read: Execute SELECT queries to retrieve data and verify accuracy.
   3. Update: Modify existing records and ensure changes are reflected.
   4. Delete: Remove records and check that they are properly deleted.
3. Relationship Tests:
   1. Verify many-to-many relationships.
   2. Ensure one-to-one relationships.
4. Security Testing
   1. Access Control: Ensure that only authorized users can access or modify data.
5. Backup and Recovery Testing
   1. Verify that backup procedures work correctly.
   2. Test data restoration from backups to ensure data integrity.
6. **Data integrity tests**

**Testing environment: MySQL Workbench**

* 1. **Validate the primary keys are unique, test cases will only include the table with Primary Key is not auto increment**

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Primary Key** | **Test Require** |
| ResearchGroups | ResearchGroupID - Auto Increment | FALSE |
| Lecturers | LecturerID - Auto Increment | FALSE |
| Students | StudentID - Auto Increment | FALSE |
| NonAcademicStaff | Staff - Auto Increment | FALSE |
| Courses | CourseCode | TRUE |
| Departments | DepartmentName | TRUE |
| Programs | ProgramName | TRUE |
| ProgramEnrollmentDetails | N/A | FALSE |
| ResearchProjects | ProjectTitle | TRUE |
| Enrollments | EnrollmentID - Auto Increment | FALSE |
| Advisors | AdvisorID - Auto Increment | FALSE |
| TeachingAssignments | AssignmentID - Auto Increment | FALSE |
| ResearchSupervision | SupervisionID - Auto Increment | FALSE |
| DepartmentStaff | DepartmentStaffID - Auto Increment | FALSE |
| Committees | CommitteeID - Auto Increment | FALSE |

**Test case 1 – Courses**

INSERT INTO Courses (CourseCode, Name, Description, Department, Level, Credits, Prerequisites, Schedule, Materials, Lecturers, EnrolledStudents, ProgramManagerID) VALUES ('CS101', 'New Course', 'New Description', 'Computer Science', 1, 3, 'None', 'MWF 9-10', 'New Textbook', 'Emily White', 2, 2);

Result: **PASS**

Error Code: 1062. Duplicate entry 'CS101' for key 'courses.PRIMARY'

**Test case 2 – Departments**

INSERT INTO Departments (DepartmentName, Faculty, ResearchAreas, CourseOffered, StaffMembers) VALUES ('Computer Science', 'Engineering', 'AI, Cybersecurity', 'CS101, CS102', 'Emily White, Nathan Drake');

Result: **PASS**

Error Code: 1062. Duplicate entry 'Computer Science' for key 'departments.PRIMARY'

**Test case 3 – Programs**

INSERT INTO Programs (ProgramName, DegreeAwarded, Duration, CourseRequirements, ProgramManagerID) VALUES ('Computer Science', 'BSc', 4, 'CS101, CS102, CS201, CS202', 2);

Result: **PASS**

Error Code: 1062. Duplicate entry 'Computer Science' for key 'programs.PRIMARY'

**Test case 4 – ResearchProjects**

INSERT INTO ResearchProjects (ProjectTitle, PrincipalInvestigator, FundingSources, TeamMembers, Publications, Outcomes) VALUES (' AI for Healthcare', 'New Investigator', 'New Funding', 'New Team', 'New Publications', 'New Outcomes');

Result: **PASS**

Error Code: 1062. Duplicate entry ' AI for Healthcare' for key 'researchprojects.PRIMARY'

* 1. **Foreign key test to confirm that foreign key constraints are working correctly based on the table structure and dummy data.**

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Foreign Key** | **Test Require** |
| ResearchGroups | N/A | FALSE |
| Lecturers | ResearchGroupID | TRUE |
| Students | LecturerID | TRUE |
| NonAcademicStaff | N/A | FALSE |
| Courses | ProgramManagerID | TRUE |
| Departments | N/A | FALSE |
| Programs | StaffID | TRUE |
| ProgramEnrollmentDetails | ProgramName | TRUE |
| ResearchProjects | N/A | FALSE |
| Enrollments | StudentID, CourseCode | TRUE |
| Advisors | StudentID, LecturerID | TRUE |
| TeachingAssignments | LecturerID, CourseCode | TRUE |
| ResearchSupervision | LecturerID, ProjectTitle | TRUE |
| DepartmentStaff | DepartmentName, LecturerID | TRUE |
| Committees | LecturerID | TRUE |

**Test case 1 – Lecturers**

-- Attempt to insert a Lecturer with an invalid ResearchGroupID

INSERT INTO Lecturers (LecturerID, Name, ResearchGroupID) VALUES (999, 'Dr. Green', 999);

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`lecturers`, CONSTRAINT `lecturers\_ibfk\_1` FOREIGN KEY (`ResearchGroupID`) REFERENCES `researchgroups` (`ResearchGroupID`))

**Test case 2 – Students**

-- Attempt to insert a Student with an invalid FacultyAdvisorID

INSERT INTO Students (StudentID, Name, FacultyAdvisorID) VALUES (999, 'John Doe', 999);

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`students`, CONSTRAINT `students\_ibfk\_1` FOREIGN KEY (`FacultyAdvisorID`) REFERENCES `lecturers` (`LecturerID`))

**Test case 3 – Courses**

-- Attempt to insert a Course with an invalid ProgramManagerID

INSERT INTO Courses (CourseCode, Name, ProgramManagerID) VALUES ('CS999', 'Database Structures', 999);

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`courses`, CONSTRAINT `courses\_ibfk\_1` FOREIGN KEY (`ProgramManagerID`) REFERENCES `nonacademicstaff` (`StaffID`))

**Test case 4 – Programs**

-- Attempt to insert a Program with an invalid ProgramManagerID

INSERT INTO Programs ( ProgramName, ProgramManagerID) VALUES ('B.Sc. Computer Science', 999);

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`programs`, CONSTRAINT `programs\_ibfk\_1` FOREIGN KEY (`ProgramManagerID`) REFERENCES `nonacademicstaff` (`StaffID`))

**Test case 5 – ProgramEnrollmentDetails**

-- Attempt to insert ProgramEnrollmentDetails with an invalid ProgramName

INSERT INTO ProgramEnrollmentDetails (ProgramName, EnrollmentDetails) VALUES (999, "Testing Enrollemnt");

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`programenrollmentdetails`, CONSTRAINT `programenrollmentdetails\_ibfk\_1` FOREIGN KEY (`ProgramName`) REFERENCES `programs` (`ProgramName`))

**Test case 6 - Enrollments**

-- Attempt to insert an Enrollment with an invalid StudentID

INSERT INTO Enrollments (EnrollmentID, StudentID, CourseCode) VALUES (999, 999, 'CS101');

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`enrollments`, CONSTRAINT `enrollments\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `students` (`StudentID`))

**Test case 7 - Enrollments**

-- Attempt to insert an Enrollment with an invalid CourseCode

INSERT INTO Enrollments (EnrollmentID, StudentID, CourseCode) VALUES (999, 1, 'CS999');

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`enrollments`, CONSTRAINT `enrollments\_ibfk\_2` FOREIGN KEY (`CourseCode`) REFERENCES `courses` (`CourseCode`))

**Test case 10 – Advisor**

-- Attempt to insert an Advisor with an invalid StudentID

INSERT INTO Advisors (AdvisorID, StudentID) VALUES (999, 999);

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`advisors`, CONSTRAINT `advisors\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `students` (`StudentID`))

**Test case 11 – TeachingAssignment**

-- Attempt to insert a TeachingAssignment with an invalid LecturerID

INSERT INTO TeachingAssignments (AssignmentID, LecturerID, CourseCode) VALUES (111, 999, 'CS101');

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`teachingassignments`, CONSTRAINT `teachingassignments\_ibfk\_1` FOREIGN KEY (`LecturerID`) REFERENCES `lecturers` (`LecturerID`))

**Test case 12 – TeachingAssignment**

-- Attempt to insert a TeachingAssignment with an invalid CourseCode

INSERT INTO TeachingAssignments (AssignmentID, LecturerID, CourseCode) VALUES (111, 1, 'CS999');

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`teachingassignments`, CONSTRAINT `teachingassignments\_ibfk\_1` FOREIGN KEY (`LecturerID`) REFERENCES `lecturers` (`LecturerID`))

**Test case 13 - ResearchSupervision**

-- Attempt to insert ResearchSupervision with an invalid LecturerID

INSERT INTO ResearchSupervision (SupervisionID, LecturerID, ProjectTitle) VALUES (999, 999, "AI & Healthcare");

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`researchsupervision`, CONSTRAINT `researchsupervision\_ibfk\_1` FOREIGN KEY (`LecturerID`) REFERENCES `lecturers` (`LecturerID`))

**Test case 14 - ResearchSupervision**

-- Attempt to insert ResearchSupervision with an invalid ProjectTitle

INSERT INTO ResearchSupervision (SupervisionID, LecturerID, ProjectTitle) VALUES (999, 1, "AI & Robot");

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`researchsupervision`, CONSTRAINT `researchsupervision\_ibfk\_2` FOREIGN KEY (`ProjectTitle`) REFERENCES `researchprojects` (`ProjectTitle`))

**Test case 15 - DepartmentStaff**

-- Attempt to insert DepartmentStaff with an invalid DepartmentName

INSERT INTO DepartmentStaff (DepartmentStaffID, DepartmentName, LecturerID) VALUES (999, 'Unknown Department', 999);

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`departmentstaff`, CONSTRAINT `departmentstaff\_ibfk\_1` FOREIGN KEY (`DepartmentName`) REFERENCES `departments` (`DepartmentName`))

**Test case 16 - DepartmentStaff**

-- Attempt to insert DepartmentStaff with an invalid LecturerID

INSERT INTO DepartmentStaff (DepartmentStaffID, DepartmentName, LecturerID) VALUES (999, 'Biology', 999);

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`departmentstaff`, CONSTRAINT `departmentstaff\_ibfk\_2` FOREIGN KEY (`LecturerID`) REFERENCES `lecturers` (`LecturerID`))

**Test case 17 - Committee**

-- Attempt to insert a Committee with an invalid ChairpersonID

INSERT INTO Committees (CommitteeID, CommitteeName, ChairpersonID) VALUES (999, 'Curriculum Committee', 999);

Result: **PASS**

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`abcu`.`committees`, CONSTRAINT `committees\_ibfk\_1` FOREIGN KEY (`ChairpersonID`) REFERENCES `lecturers` (`LecturerID`))

* 1. **Non-nullable test to ensure the field should not be null value**

|  |  |  |
| --- | --- | --- |
| **Table Name** | **NOT NULL** | **Test Require** |
| ResearchGroups | N/A | FALSE |
| Lecturers | N/A | FALSE |
| Students | N/A | FALSE |
| NonAcademicStaff | N/A | FALSE |
| Courses | N/A | FALSE |
| Departments | N/A | FALSE |
| Programs | N/A | FALSE |
| ProgramEnrollmentDetails | N/A | FALSE |
| ResearchProjects | N/A | FALSE |
| Enrollments | N/A | FALSE |
| Advisors | N/A | FALSE |
| TeachingAssignments | N/A | FALSE |
| ResearchSupervision | N/A | FALSE |
| DepartmentStaff | N/A | FALSE |
| Committees | CommitteeName | TRUE |

**Test Case 1 – Committees**

-- Attempt to insert a Committee with a NULL CommitteeName

INSERT INTO Committees (CommitteeID, CommitteeName, ChairpersonID) VALUES (999, NULL, 1);

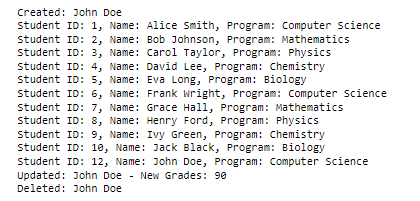
Result: **PASS**

Error Code: 1048. Column 'CommitteeName' cannot be null

1. **CRUD Operations**
   1. Create: Tests that a new item can be created and has a valid ID.
   2. Read: Tests that an item can be retrieved and its attributes are correct.
   3. Update: Tests that an existing item's attributes can be updated.
   4. Delete: Tests that an item can be deleted and is no longer retrievable.

We have created a python program “CRUD Test.py” to perform the CRUD test for our database. The program logic flow begins with initializing the database connection using SQLAlchemy(python library) create\_engine function, which sets up the connection pool. Next, a session is created to manage interactions with the database. The program then defines the data models using the declarative\_base() to establish the schema. Once the models are defined, the program enters a loop where it processes incoming requests, retrieving or updating data as needed. Each request is handled within a session context to ensure proper resource management, and once the operations are complete, the session is committed or rolled back based on success or failure. Finally, the program gracefully closes the session, releasing any resources back to the connection pool, ensuring optimal performance and stability.

Output:



Created: John Doe

Student ID: 1, Name: Alice Smith, Program: Computer Science

Student ID: 2, Name: Bob Johnson, Program: Mathematics

Student ID: 3, Name: Carol Taylor, Program: Physics

Student ID: 4, Name: David Lee, Program: Chemistry

Student ID: 5, Name: Eva Long, Program: Biology

Student ID: 6, Name: Frank Wright, Program: Computer Science

Student ID: 7, Name: Grace Hall, Program: Mathematics

Student ID: 8, Name: Henry Ford, Program: Physics

Student ID: 9, Name: Ivy Green, Program: Chemistry

Student ID: 10, Name: Jack Black, Program: Biology

Student ID: 12, Name: John Doe, Program: Computer Science

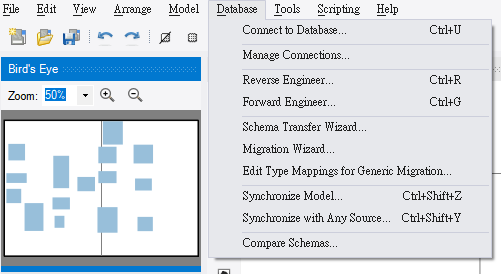
Updated: John Doe - New Grades: 90

Deleted: John Doe

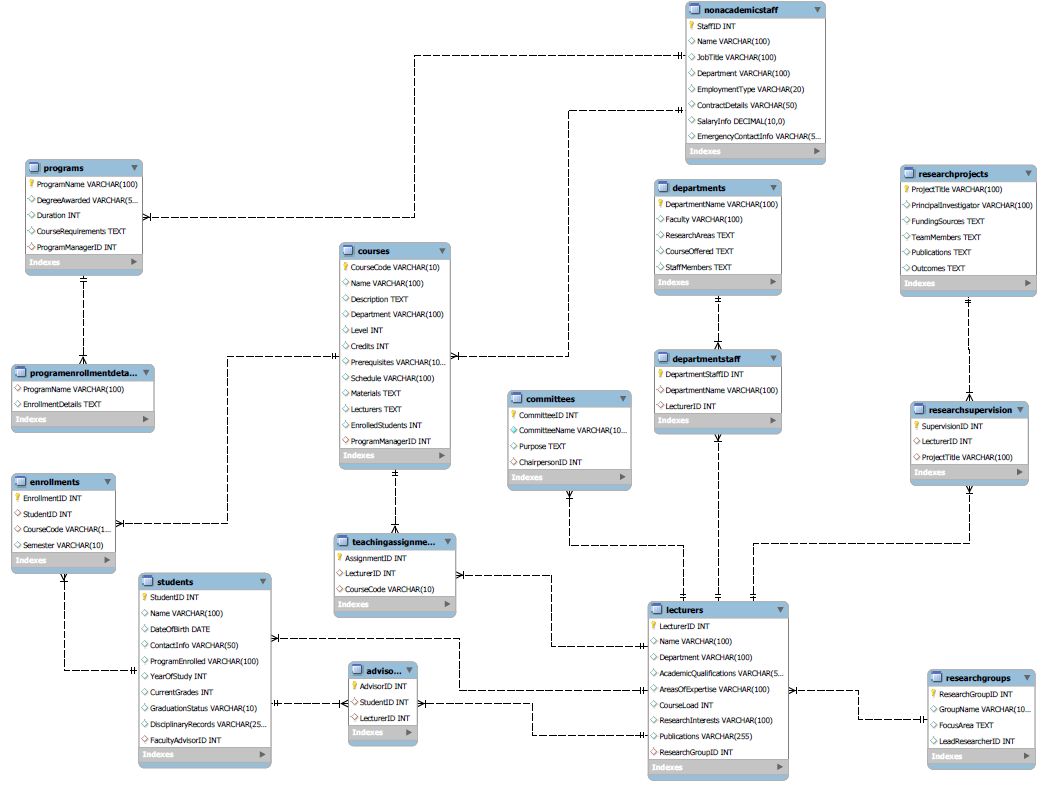
Result: **PASS**

1. **Relationship test**

By using the MySQL workbench build-in function, “Reverse Engineer” function, we could easy to get the Enhanced Entity Relationship (EER) diagram, and verify the relationship between table by study the table connection in the chart



Please refer to Reverse Engineer - EER Diagram.pdf

****

1. **Security Testing**

Access Control test: Ensure that only authorized users can access or modify data.

We have created the AccessControl Test.py to made sure different user have different database access right control

In the test case, we have created four users included

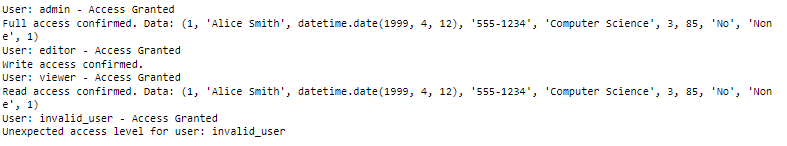
Admin – Full access right

Editor – Read, Write access right

Viewer – Only read access right

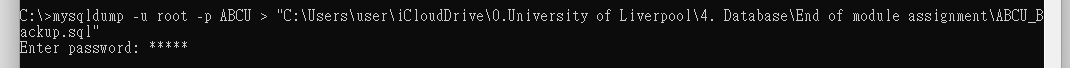
Invalid\_user – None access right

Output:



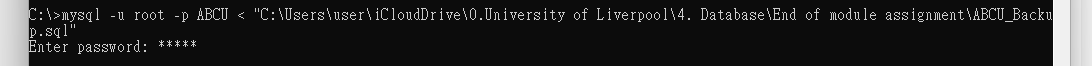
1. **Backup and Recovery Testing**

The backup process was using mysqldump command, which successful created a database backup. (ABCU\_backup.sql)



mysqldump -u root -p ABCU > "C:\Users\user\iCloudDrive\0.University of Liverpool\4. Database\End of module assignment\ABCU\_Backup.sql"

Recovery Test was using mysql command to restore the backup file to the ABCU database.



C:\>mysql -u root -p ABCU < "C:\Users\user\iCloudDrive\0.University of Liverpool\4. Database\End of module assignment\ABCU\_Backup.sql"