DATABASE & TABLE CREATION

```
CREATE DATABASE Student_Management_DB


CREATE TABLE Faculty (
   FacultyID INT PRIMARY KEY AUTO_INCREMENT,
   FirstName VARCHAR(50) NOT NULL,
   LastName VARCHAR(50) NOT NULL,
   Department VARCHAR(100) NOT NULL,
   Status ENUM('Active', 'Inactive') DEFAULT 'Active'
);

CREATE TABLE Student (
   StudentID INT AUTO_INCREMENT PRIMARY KEY,
   StudentNumber VARCHAR(20) UNIQUE NOT NULL,
   FirstName VARCHAR(50) NOT NULL,
   LastName VARCHAR(50) NOT NULL,
   Gender ENUM('Male','Female') NOT NULL,
   Department VARCHAR(100) NOT NULL,
   YearLevel INT NOT NULL,
   AcademicStanding ENUM('Good Standing','Probation','Dismissed') DEFAULT 'Good
Standing',
   Status ENUM('Active','Inactive','Graduated') DEFAULT 'Active',
   CONSTRAINT chk_year_level CHECK (YearLevel BETWEEN 1 AND 5)
);


CREATE TABLE Course (
   CourseID INT PRIMARY KEY AUTO_INCREMENT,
   CourseCode VARCHAR(20) NOT NULL UNIQUE,
   CourseName VARCHAR(150) NOT NULL,
   Department VARCHAR(100) NOT NULL,
   FacultyID INT NOT NULL,
   Semester VARCHAR(50) NOT NULL,
   AcademicYear VARCHAR(9) NOT NULL,
   MaxCapacity INT NOT NULL DEFAULT 40,
   Status ENUM('Active', 'Inactive') DEFAULT 'Active',
   CONSTRAINT fk_course_faculty FOREIGN KEY (FacultyID)
      REFERENCES Faculty(FacultyID) ON DELETE RESTRICT
);

CREATE TABLE Enrollment (
   EnrollmentID INT PRIMARY KEY AUTO_INCREMENT,
   StudentID INT NOT NULL,
   CourseID INT NOT NULL,
   EnrollmentDate DATE NOT NULL,
   Status ENUM('Enrolled', 'Dropped', 'Completed') DEFAULT 'Enrolled',
   CONSTRAINT fk_enrollment_student FOREIGN KEY (StudentID)
      REFERENCES Student(StudentID) ON DELETE CASCADE,
   CONSTRAINT fk_enrollment_course FOREIGN KEY (CourseID)
      REFERENCES Course(CourseID) ON DELETE CASCADE,
   CONSTRAINT uq_enrollment UNIQUE (StudentID, CourseID)
);

CREATE TABLE Grades (
   GradeID INT AUTO_INCREMENT PRIMARY KEY,
   EnrollmentID INT UNIQUE NOT NULL,
   MidtermGrade DECIMAL(3,2),
   FinalGrade DECIMAL(3,2),
   CONSTRAINT fk_grade_enrollment
      FOREIGN KEY (EnrollmentID)
      REFERENCES Enrollment(EnrollmentID)
      ON DELETE CASCADE
);
```

—------------------ INSERTS —-------------------------

```
INSERT INTO Faculty (FirstName, LastName, Department)
VALUES
('Roxanne','Oliveros','IT'),
('Christine','Domat-ol','CS'),
('Jobel','Araw','IT'),
('Meagan','Enguerra','CS'),
('Jessica','Mananes','IT'),
('Andrea','Conception','IT');


INSERT INTO Student
(StudentNumber, FirstName, LastName, Gender, Department, YearLevel)
VALUES
```

```
('2023-001','Taylor','Swift','Female','BSIT',2),
('2023-002','Beyonce','Knowles','Female','BSIT',1),
('2023-003','Sabrina','Carpenter','Female','BSIT',3),
('2023-004','Ariana','Grande','Female','BSIT',4),
('2023-005','Billie','Eilish','Female','BSIT',1),
('2023-006','Olivia','Rodrigo','Female','BSIT',2),
('2023-007','Chappell','Roan','Female','BSIT',3),
('2023-008','Dua','Lipa','Female','BSIT',1),
('2023-009','Bruno','Mars','Male','BSIT',2),
('2023-010','Selena','Gomez','Female','BSIT',3);

INSERT INTO Course
(CourseCode, CourseName, Department, FacultyID, Semester, AcademicYear)
VALUES
('IT101','Intro to Programming','IT',1,'1st Semester','2023-2024'),
('CS101','Programming 1','CS',2,'1st Semester','2023-2024'),
('IT102','Database Administration ','IT',3,'1st Semester','2023-2024'),
('CS102','OOP','CS',4,'1st Semester','2023-2024'),
('IT103','Web Development','IT',5,'1st Semester','2023-2024'),
('CS201','Data Structure & Algorithm','CS',6,'2nd Semester','2023-2024'),
('IT201','Networking','IT',1,'2nd Semester','2023-2024'),
('CS202','Programming 2','CS',4,'2nd Semester','2023-2024'),
('IT202','System Analysis','IT',3,'2nd Semester','2023-2024'),
('CS203','Software Engineering','CS',5,'2nd Semester','2023-2024');
```

—------------------ CRUD —-------------------------

-- CREATE

```
INSERT INTO Enrollment (StudentID, CourseID, EnrollmentDate)
VALUES
(1, 1, '2023-08-15'),
(1, 2, '2023-08-15'),
(2, 1, '2023-08-16'),
(2, 3, '2023-08-16'),
(3, 2, '2023-08-17'),
(3, 4, '2023-08-17'),
(4, 3, '2023-08-18'),
(5, 5, '2023-08-18'),
(6, 6, '2023-08-19'),
(7, 7, '2023-08-19'),
(8, 8, '2023-08-20'),
(9, 9, '2023-08-20'),
(10, 10, '2023-08-21');

INSERT INTO Grades (EnrollmentID, MidtermGrade, FinalGrade)
VALUES
(1, 1.50, 1.25),
(2, 2.00, 1.75),
(3, 2.50, 2.25),
(4, 1.75, 1.50),
(5, 1.25, 1.00),
(6, 2.00, 1.75),
(7, 3.00, 2.75),
(8, 1.50, 1.25),
(9, 2.25, 2.00),
(10, 1.75, 1.50),
(11, 2.00, 1.75),
(12, 3.00, 2.75),
(13, 1.25, 1.00);
```

-- READ

```
SELECT StudentNumber, FirstName, LastName, YearLevel
FROM Student
WHERE Status = 'Active';


SELECT s.StudentNumber, s.FirstName, s.LastName
FROM Student s
JOIN Enrollment e ON s.StudentID = e.StudentID
WHERE e.CourseID = 1;


SELECT * FROM Student
WHERE LastName = 'Swift';
```

-- UPDATE

```
UPDATE Student
```

```
SET LastName = 'Knowles-Carter'
WHERE StudentNo = '2023-002';

UPDATE Enrollment
SET Status = 'Completed'
WHERE EnrollmentID = 1;
```

-- DELETE

```
DELETE FROM Student
WHERE StudentNo = '2023-014';

DELETE FROM Student
WHERE Status = 'Inactive';
```

—------------------ REPORTS —-------------------------

Students per Course :

```
SELECT
   c.CourseCode,
   c.CourseName,
   COUNT(e.StudentID) AS TotalStudents
FROM Course c
LEFT JOIN Enrollment e
   ON c.CourseID = e.CourseID
WHERE e.Status = 'Enrolled'
GROUP BY c.CourseID, c.CourseCode, c.CourseName
ORDER BY TotalStudents DESC;
```

Grades Summary:

[pag by course ang summarize]

```
SELECT
   c.CourseCode,
   c.CourseTitle,
   ROUND(AVG(g.Grade), 2) AS AverageGrade,
   MIN(g.Grade) AS LowestGrade,
   MAX(g.Grade) AS HighestGrade
```

```
FROM Course c
JOIN Enrollment e
   ON c.CourseID = e.CourseID
JOIN Grades g
   ON e.EnrollmentID = g.EnrollmentID
GROUP BY c.CourseCode, c.CourseTitle
HAVING AVG(g.Grade) <= 3.00
ORDER BY AverageGrade;
```

[pagstudent avg yung isusummarize]

```
SELECT
   s.StudentNumber,
   CONCAT(s.FirstName,' ', s.LastName) AS StudentName,
   c.CourseCode,
   c.CourseName,
   g.MidtermGrade,
   g.FinalGrade,
   ROUND((g.MidtermGrade + g.FinalGrade)/2,2) AS AverageGrade
FROM Student s
JOIN Enrollment e ON s.StudentID = e.StudentID
JOIN Course c ON e.CourseID = c.CourseID
JOIN Grades g ON e.EnrollmentID = g.EnrollmentID
ORDER BY c.CourseCode, s.LastName;
```

—---------- VIEW AND SP —-------------

student performance view:

```
CREATE VIEW vw_student_performance AS
SELECT
   s.StudentNumber,
   CONCAT(s.FirstName, ' ', s.LastName) AS StudentName,
   c.CourseCode,
   c.CourseName,
   g.MidtermGrade,
   g.FinalGrade,
   e.Status AS EnrollmentStatus
FROM Student s
JOIN Enrollment e
```

```
   ON s.StudentID = e.StudentID
JOIN Course c
   ON e.CourseID = c.CourseID
LEFT JOIN Grades g
   ON e.EnrollmentID = g.EnrollmentID;
```

Stored procedure

```
DELIMITER $$

CREATE PROCEDURE ComputeGPAAndStanding (
   IN p_StudentID INT
)
BEGIN
   DECLARE v_GPA DECIMAL(4,2);

   -- Compute GPA (remove status restriction)
   SELECT AVG(g.FinalGrade)
   INTO v_GPA
   FROM Grades g
   JOIN Enrollment e
      ON g.EnrollmentID = e.EnrollmentID
   WHERE e.StudentID = p_StudentID
     AND g.FinalGrade IS NOT NULL;

   -- If no grades yet
   IF v_GPA IS NULL THEN
      UPDATE Student
      SET AcademicStanding = 'Good Standing'
      WHERE StudentID = p_StudentID;
   ELSE
      UPDATE Student
      SET AcademicStanding =
         CASE
            WHEN v_GPA <= 2.50 THEN 'Good Standing'
            WHEN v_GPA > 2.50 AND v_GPA <= 3.00 THEN 'Probation'
            ELSE 'Dismissed'
         END
      WHERE StudentID = p_StudentID;
   END IF;

   -- Show result
   SELECT
```

```
        StudentID,
        v_GPA AS GPA,
        AcademicStanding
    FROM Student
    WHERE StudentID = p_StudentID;

END $$

DELIMITER ;
```

TRIGGER

Prevent UPDATE of Grades After Completion

```
DELIMITER $$


CREATE TRIGGER trg_security_lock_grade_update
BEFORE UPDATE ON Grades
FOR EACH ROW
BEGIN
        DECLARE v_status VARCHAR(20);

        SELECT Status
        INTO v_status
        FROM Enrollment
        WHERE EnrollmentID = OLD.EnrollmentID;

        IF v_status = 'Completed' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT =
      'SECURITY VIOLATION: Grades cannot be modified after enrollment is
completed.';
        END IF;
END$$
```

```
DELIMITER ;
```

PREVENT DELETE OF GRADES AFTER COMPLETION

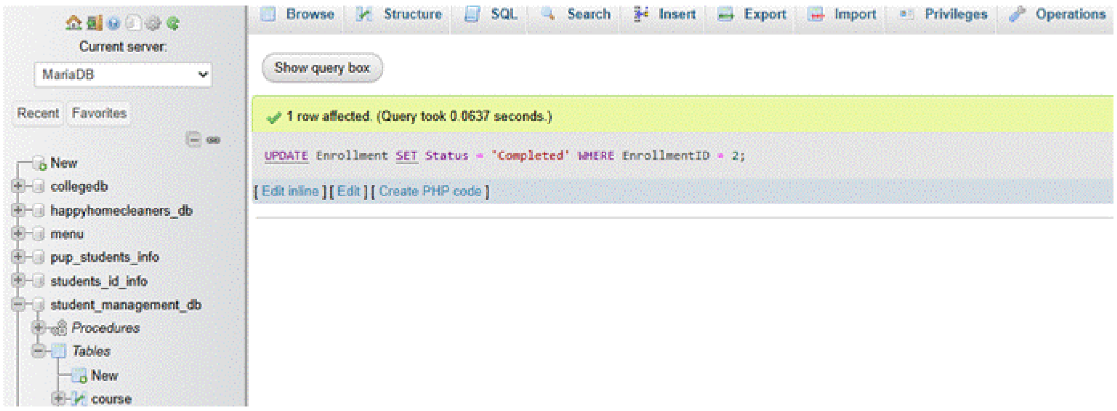
```
DELIMITER $$

CREATE TRIGGER trg_security_lock_grade_delete
BEFORE DELETE ON Grades
FOR EACH ROW
BEGIN
    DECLARE v_status VARCHAR(20);

    SELECT Status
    INTO v_status
    FROM Enrollment
    WHERE EnrollmentID = OLD.EnrollmentID;

    IF v_status = 'Completed' THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT =
  'SECURITY VIOLATION: Grades cannot be deleted after enrollment is completed.';
```

```
        END IF;

END$$


DELIMITER ;
```

TO TEST

MARKS ENROLLMENT COMPLETED

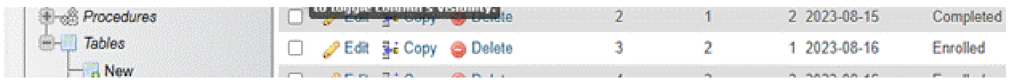
```
UPDATE Enrollment

SET Status = 'Completed'

WHERE EnrollmentID = 2;
```



1 row affected. (Query took 0.0637 seconds.)

```
UPDATE Enrollment SET Status = 'Completed' WHERE EnrollmentID = 2;
```

RESULT



```
2    1    2  2023-08-15    Completed
3    2    1  2023-08-16    Enrolled
```

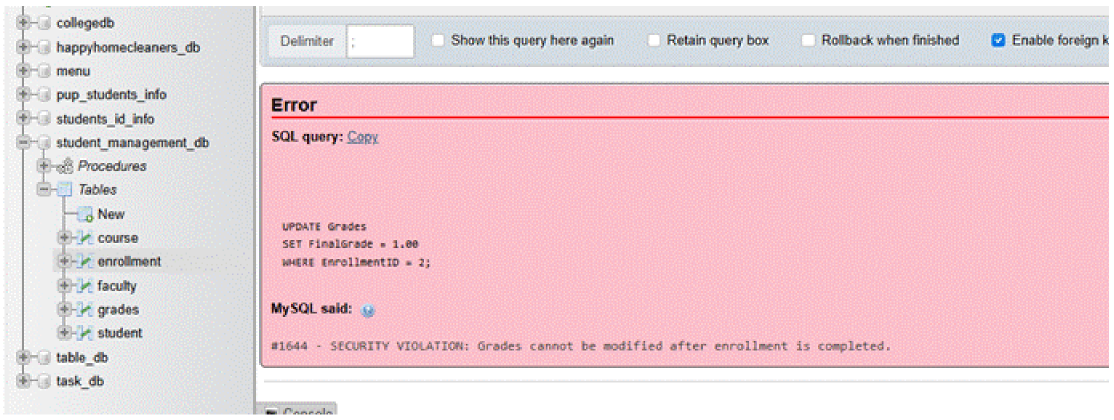
ATTEMPT TO MODIFY GRADE

UPDATE Grades

SET FinalGrade = 1.00

WHERE EnrollmentID = 2;

RESULT



Error

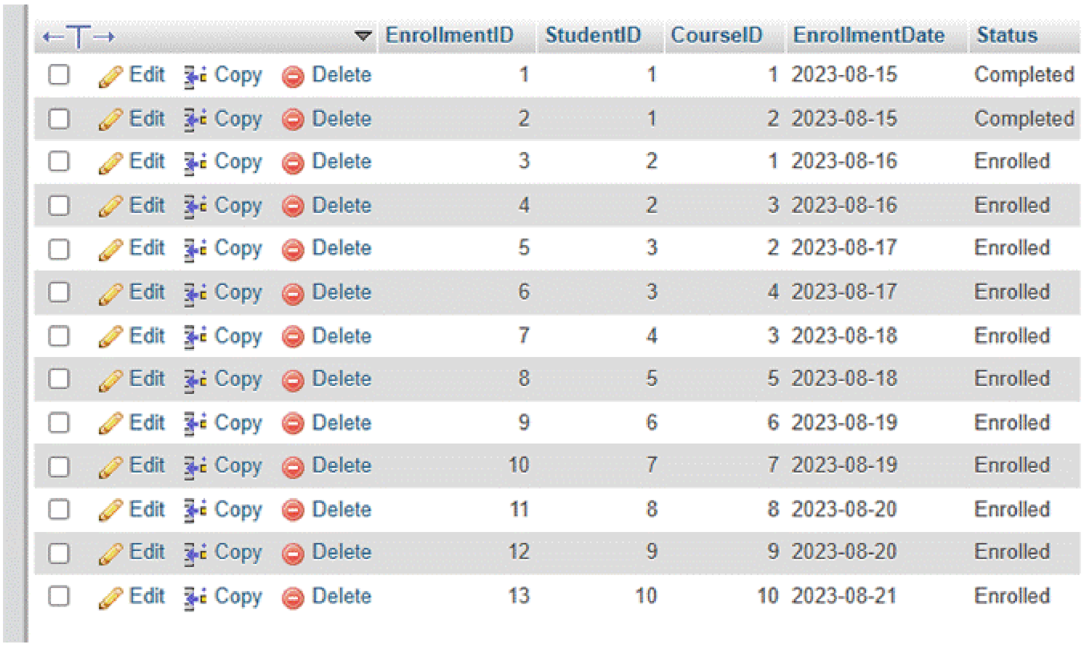
SQL query: Copy

```
UPDATE Grades
SET FinalGrade = 1.00
WHERE EnrollmentID = 2;
```

MySQL said:

```
#1644 - SECURITY VIOLATION: Grades cannot be modified after enrollment is completed.
```

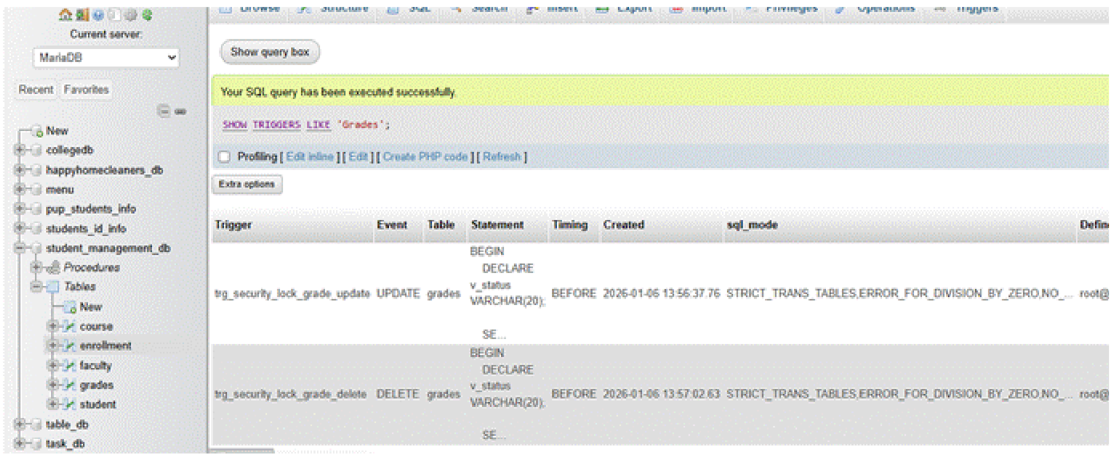
STATUS:



	EnrollmentID	StudentID	CourseID	EnrollmentDate	Status
Edit Copy Delete	1	1	1	2023-08-15	Completed
Edit Copy Delete	2	1	2	2023-08-15	Completed
Edit Copy Delete	3	2	1	2023-08-16	Enrolled
Edit Copy Delete	4	2	3	2023-08-16	Enrolled
Edit Copy Delete	5	3	2	2023-08-17	Enrolled
Edit Copy Delete	6	3	4	2023-08-17	Enrolled
Edit Copy Delete	7	4	3	2023-08-18	Enrolled
Edit Copy Delete	8	5	5	2023-08-18	Enrolled
Edit Copy Delete	9	6	6	2023-08-19	Enrolled
Edit Copy Delete	10	7	7	2023-08-19	Enrolled
Edit Copy Delete	11	8	8	2023-08-20	Enrolled
Edit Copy Delete	12	9	9	2023-08-20	Enrolled
Edit Copy Delete	13	10	10	2023-08-21	Enrolled

VERIFY IF THE TRIGGER EXIST

SHOW TRIGGERS LIKE 'Grades';



Your SQL query has been executed successfully.

```
SHOW TRIGGERS LIKE 'Grades';
```

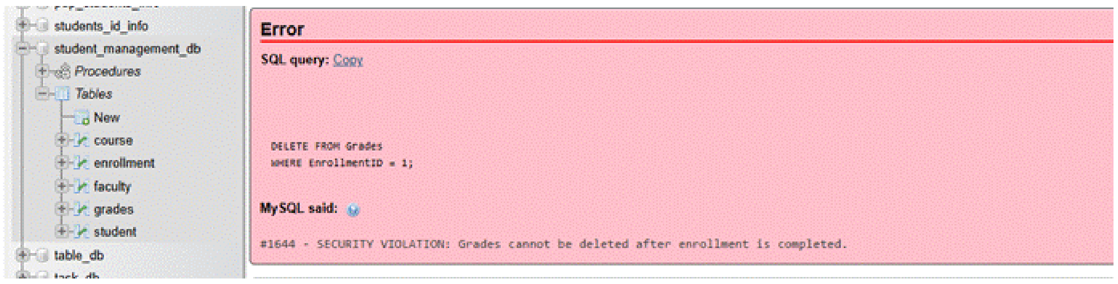
Trigger	Event	Table	Statement	Timing	Created	sql_mode	Defin
trg_security_lock_grade_update	UPDATE	grades	BEGIN DECLARE v_status VARCHAR(20); SE...	BEFORE	2026-01-06 13:56:37.76	STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_...	root@
trg_security_lock_grade_delete	DELETE	grades	BEGIN DECLARE v_status VARCHAR(20); SE...	BEFORE	2026-01-06 13:57:02.63	STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_...	root@

TRIGGER FOR DELETE (should be blocked)

DELETE FROM Grades

WHERE EnrollmentID = 1;

RESULT:



Error

SQL query: Copy

```
DELETE FROM Grades
WHERE EnrollmentID = 1;
```

MySQL said:

```
#1644 - SECURITY VIOLATION: Grades cannot be deleted after enrollment is completed.
```

To prove the trigger only blocks completed enrollments:

step 1 change status back

UPDATE Enrollment

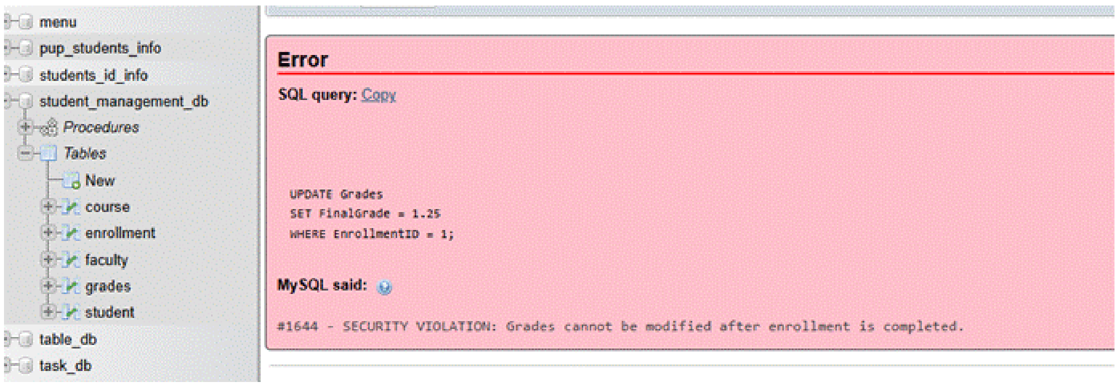
SET Status = 'Ongoing'

WHERE EnrollmentID = 2;

step 2 updating grades again

```
UPDATE Grades

SET FinalGrade = 1.25

WHERE EnrollmentID = 1;
```

Completed status result:



Error

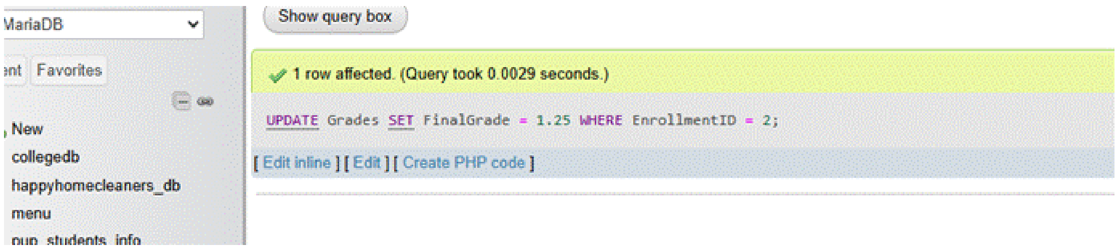
SQL query: Copy

```
UPDATE Grades
SET FinalGrade = 1.25
WHERE EnrollmentID = 1;
```

MySQL said:

```
#1644 - SECURITY VIOLATION: Grades cannot be modified after enrollment is completed.
```

Enrolled status result:



✓ 1 row affected. (Query took 0.0029 seconds.)

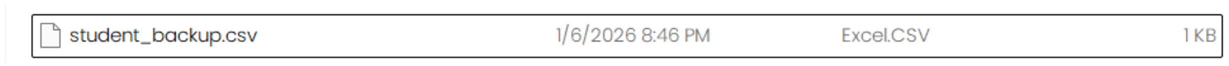
```
UPDATE Grades SET FinalGrade = 1.25 WHERE EnrollmentID = 2;
```

[Edit inline] [Edit] [Create PHP code]

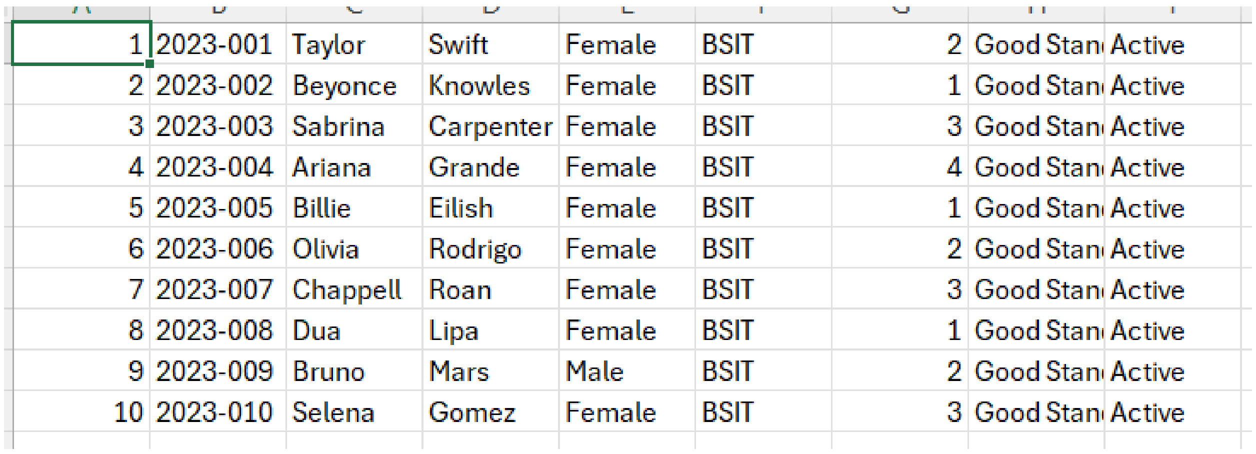
BACKUP

Student Manual Backup

```
SELECT *
FROM Student
INTO OUTFILE 'C:/wamp64/tmp/student_backup.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```



```
student_backup.csv    1/6/2026 8:46 PM    Excel.CSV    1 KB
```

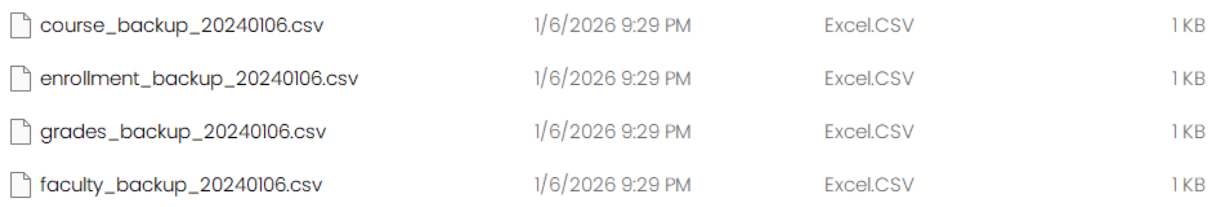


1	2023-001	Taylor	Swift	Female	BSIT	2	Good Stan	Active
2	2023-002	Beyonce	Knowles	Female	BSIT	1	Good Stan	Active
3	2023-003	Sabrina	Carpenter	Female	BSIT	3	Good Stan	Active
4	2023-004	Ariana	Grande	Female	BSIT	4	Good Stan	Active
5	2023-005	Billie	Eilish	Female	BSIT	1	Good Stan	Active
6	2023-006	Olivia	Rodrigo	Female	BSIT	2	Good Stan	Active
7	2023-007	Chappell	Roan	Female	BSIT	3	Good Stan	Active
8	2023-008	Dua	Lipa	Female	BSIT	1	Good Stan	Active
9	2023-009	Bruno	Mars	Male	BSIT	2	Good Stan	Active
10	2023-010	Selena	Gomez	Female	BSIT	3	Good Stan	Active

More backup samples

```
SELECT *
FROM Course
INTO OUTFILE 'C:/wamp64/tmp/course_backup_20240106.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT *
FROM Enrollment
INTO OUTFILE 'C:/wamp64/tmp/enrollment_backup_20240106.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT *
FROM Grades
```

```
INTO OUTFILE 'C:/wamp64/tmp/grades_backup_20240106.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT *
FROM Faculty
INTO OUTFILE 'C:/wamp64/tmp/faculty_backup_20240106.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```



```
course_backup_20240106.csv       1/6/2026 9:29 PM   Excel.CSV   1 KB
enrollment_backup_20240106.csv   1/6/2026 9:29 PM   Excel.CSV   1 KB
grades_backup_20240106.csv       1/6/2026 9:29 PM   Excel.CSV   1 KB
faculty_backup_20240106.csv      1/6/2026 9:29 PM   Excel.CSV   1 KB
```

RESTORE

Ps. delete all the table 1st before restoring

Faculty

```
LOAD DATA INFILE 'C:/wamp64/tmp/faculty_backup_20240106.csv'
INTO TABLE Faculty
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Student

```
LOAD DATA INFILE 'C:/wamp64/tmp/student_backup.csv'
INTO TABLE Student
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

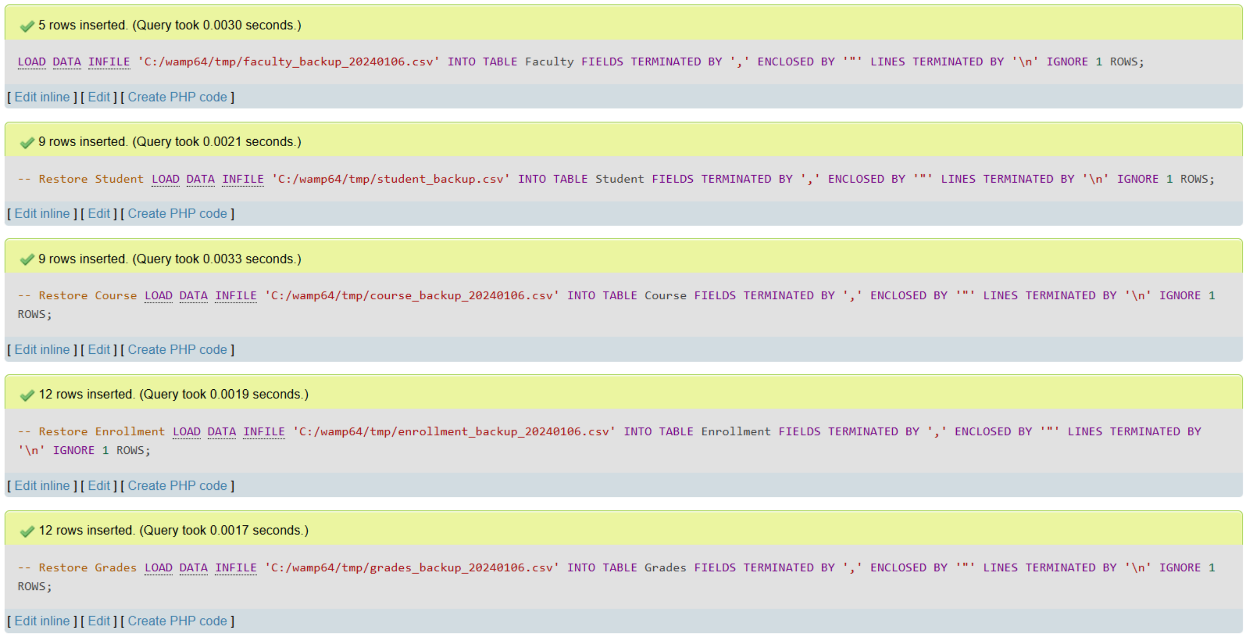
Course

```
LOAD DATA INFILE 'C:/wamp64/tmp/course_backup_20240106.csv'
INTO TABLE Course
FIELDS TERMINATED BY ','
```

ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

Enrollment
LOAD DATA INFILE 'C:/wamp64/tmp/enrollment_backup_20240106.csv'
INTO TABLE Enrollment
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

Grades
LOAD DATA INFILE 'C:/wamp64/tmp/grades_backup_20240106.csv'
INTO TABLE Grades
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;



```
✓ 5 rows inserted. (Query took 0.0030 seconds.)
LOAD DATA INFILE 'C:/wamp64/tmp/faculty_backup_20240106.csv' INTO TABLE Faculty FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n' IGNORE 1 ROWS;

✓ 9 rows inserted. (Query took 0.0021 seconds.)
-- Restore Student LOAD DATA INFILE 'C:/wamp64/tmp/student_backup.csv' INTO TABLE Student FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n' IGNORE 1 ROWS;

✓ 9 rows inserted. (Query took 0.0033 seconds.)
-- Restore Course LOAD DATA INFILE 'C:/wamp64/tmp/course_backup_20240106.csv' INTO TABLE Course FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n' IGNORE 1
ROWS;

✓ 12 rows inserted. (Query took 0.0019 seconds.)
-- Restore Enrollment LOAD DATA INFILE 'C:/wamp64/tmp/enrollment_backup_20240106.csv' INTO TABLE Enrollment FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY
'\n' IGNORE 1 ROWS;

✓ 12 rows inserted. (Query took 0.0017 seconds.)
-- Restore Grades LOAD DATA INFILE 'C:/wamp64/tmp/grades_backup_20240106.csv' INTO TABLE Grades FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n' IGNORE 1
ROWS;
```

SECURITY SETUP

ADMIN - Grants all permission

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin123';
GRANT ALL PRIVILEGES ON Student_Management_DB.* TO 'admin'@'localhost';
FLUSH PRIVILEGES;
```

REGISTRAR - Grants specific functions to students and enrollments, selection on course, faculty, grades and view student performance, and execute procedure on ComputeGPAAndStanding

```
CREATE USER 'registrar'@'localhost' IDENTIFIED BY 'registrar123!';
GRANT SELECT, INSERT, UPDATE ON Student_Management_DB.Student TO
'registrar'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON Student_Management_DB.Enrollment TO
'registrar'@'localhost';
GRANT SELECT ON Student_Management_DB.Course TO 'registrar'@'localhost';
GRANT SELECT ON Student_Management_DB.Faculty TO 'registrar'@'localhost';
GRANT SELECT ON Student_Management_DB.Grades TO 'registrar'@'localhost';
GRANT SELECT ON Student_Management_DB.vw_student_performance TO
'registrar'@'localhost';
GRANT EXECUTE ON PROCEDURE Student_Management_DB.ComputeGPAAndStanding
TO 'registrar'@'localhost';
```

FACULTY - Grants selection on student, course and enrollment, specific funtions on grades, and view student performance (View courses, students, manage grades)

```
CREATE USER 'faculty'@'localhost' IDENTIFIED BY 'faculty123!';
GRANT SELECT ON Student_Management_DB.Student TO 'faculty'@'localhost';
GRANT SELECT ON Student_Management_DB.Course TO 'faculty'@'localhost';
GRANT SELECT ON Student_Management_DB.Enrollment TO 'faculty'@'localhost';
GRANT SELECT, INSERT, UPDATE ON Student_Management_DB.Grades TO
'faculty'@'localhost';
GRANT SELECT ON Student_Management_DB.vw_student_performance TO
'faculty'@'localhost';
```

STUDENT - Read-only access to own records

```
CREATE USER 'student'@'localhost' IDENTIFIED BY 'student123!';
GRANT SELECT ON Student_Management_DB.vw_student_performance TO
'student'@'localhost';
```

*ps. dont forget to execute ‘***FLUSH PRIVILEGES;***’ to apply changes*