# **REVIEW – 3**

**PROJECT TITLE:** LEANER’S GUIDE (learning management system)

**NAME**: Y. AKHILA TEJASWI – RA2211003011417

KHUSHI – RA2211003011460

**BATCH ID:** 20

**DATE:** 05-04-24

**Subject Title & Code:** Database Management System

(21CSC205P)

**CONSTRAINTS**

**Constraints for Learners Table**

ALTER TABLE Learners

ADD CONSTRAINT PK\_Learners PRIMARY KEY (learner\_id),

ADD CONSTRAINT FK\_Course\_Learner FOREIGN KEY (course\_id) REFERENCES Courses(course\_id);

**Constraints for Instructors Table**

ALTER TABLE Instructors

ADD CONSTRAINT PK\_Instructors PRIMARY KEY (instructor\_id);

**Constraints for Administrator Table**

ALTER TABLE Administrator

ADD CONSTRAINT PK\_Administrator PRIMARY KEY (admin\_id);

**Constraints for Courses Table**

ALTER TABLE Courses

ADD CONSTRAINT PK\_Courses PRIMARY KEY (course\_id),

ADD CONSTRAINT FK\_Instructor\_Courses FOREIGN KEY (instructor\_id) REFERENCES Instructors(instructor\_id);

**Constraints for Quiz Table**

ALTER TABLE Quiz

ADD CONSTRAINT PK\_Quiz PRIMARY KEY (quiz\_id),

ADD CONSTRAINT FK\_Course\_Quiz FOREIGN KEY (course\_id) REFERENCES Courses(course\_id);

**Constraints for Feedback Table**

ALTER TABLE Feedback

ADD CONSTRAINT PK\_Feedback PRIMARY KEY (feedback\_id),

ADD CONSTRAINT FK\_Learner\_Feedback FOREIGN KEY (learner\_id) REFERENCES Learners(learner\_id),

ADD CONSTRAINT FK\_Course\_Feedback FOREIGN KEY (course\_id) REFERENCES Courses(course\_id);

**Constraints** **for Certificate Table**

ALTER TABLE Certificate

ADD CONSTRAINT PK\_Certificate PRIMARY KEY (certificate\_id),

ADD CONSTRAINT FK\_Learner\_Certificate FOREIGN KEY (learner\_id) REFERENCES Learners(learner\_id),

ADD CONSTRAINT FK\_Course\_Certificate FOREIGN KEY (course\_id) REFERENCES Courses(course\_id);

**VIEW**

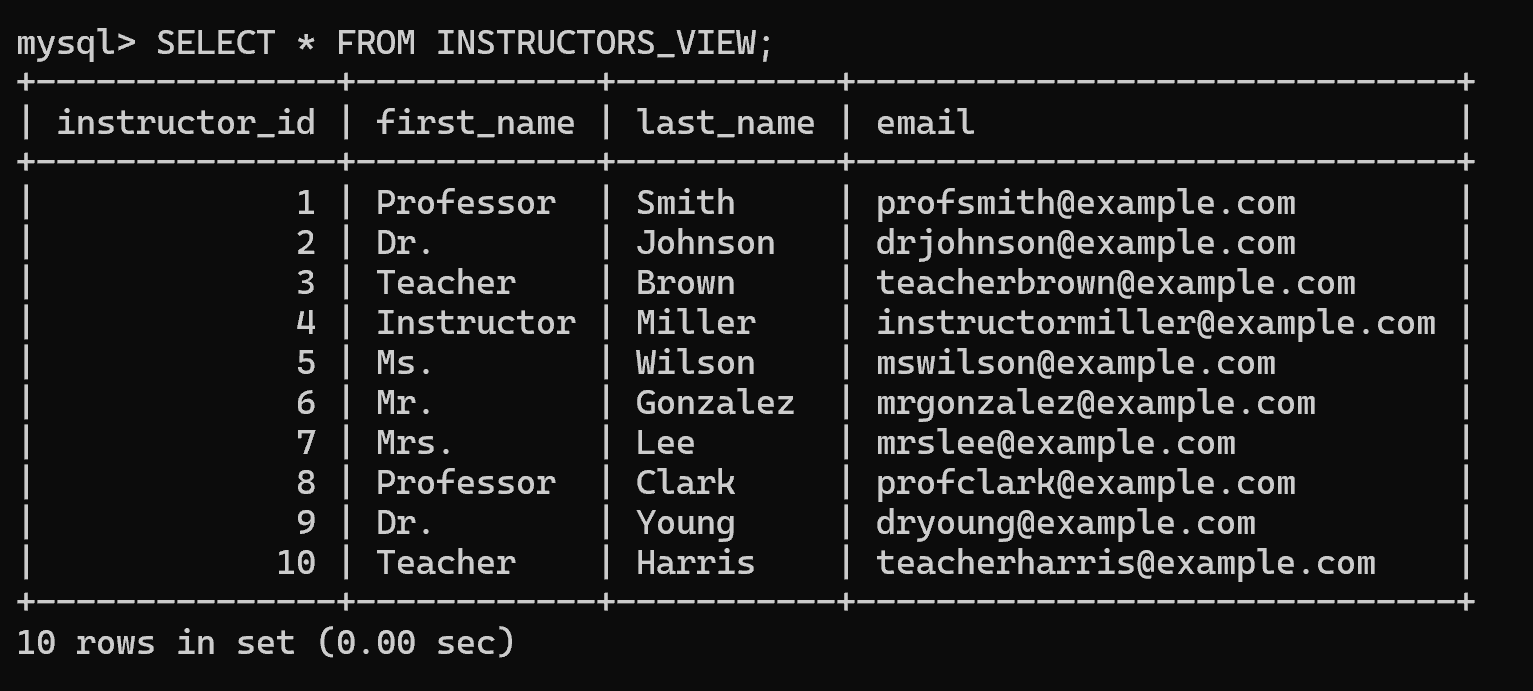
**1. Create a view for learners, instructors, administrator, courses, quiz, feedback and certificate.**

**View for Instructors table**

CREATE VIEW Instructors\_View AS

SELECT instructor\_id, first\_name, last\_name, email

FROM Instructors;



**View for Learners table**

CREATE VIEW Learners\_View AS

SELECT learner\_id, first\_name, last\_name, email, date\_of\_birth, enrollment\_date

FROM Learners;

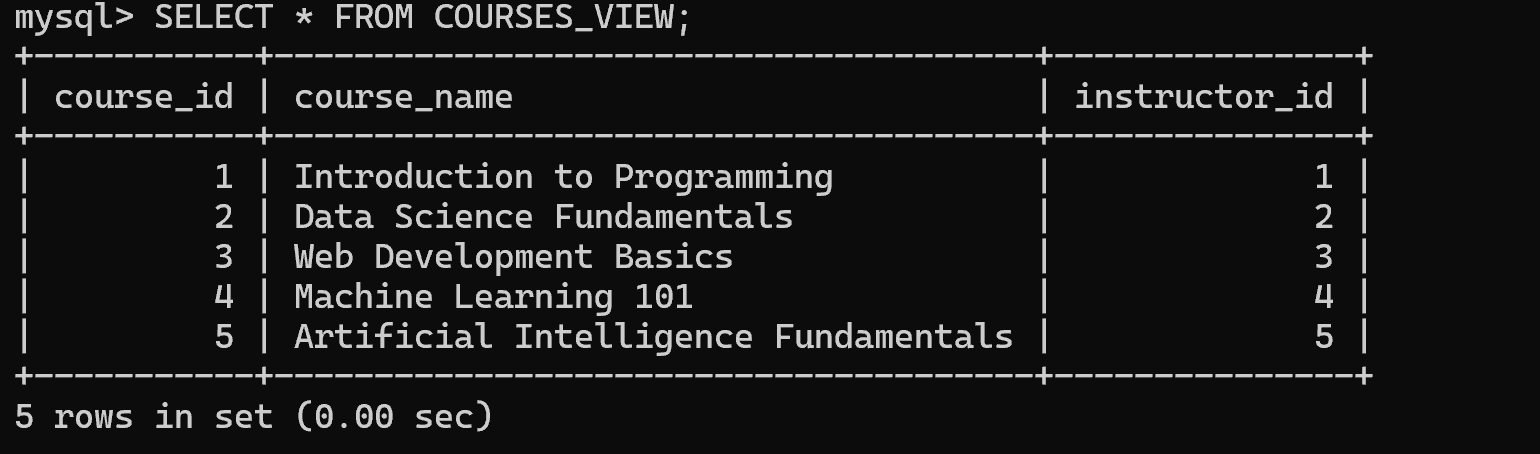


**View for Courses table**

CREATE VIEW Courses\_View AS

SELECT course\_id, course\_name, instructor\_id

FROM Courses;

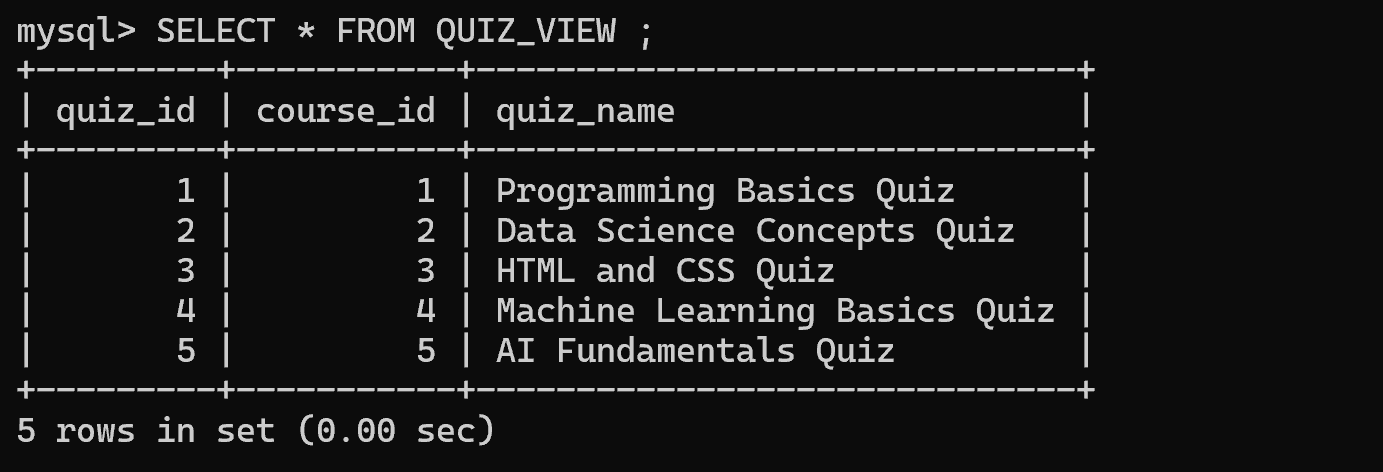


**View for Quiz table**

CREATE VIEW Quiz\_View AS

SELECT quiz\_id, course\_id, quiz\_name

FROM Quiz;

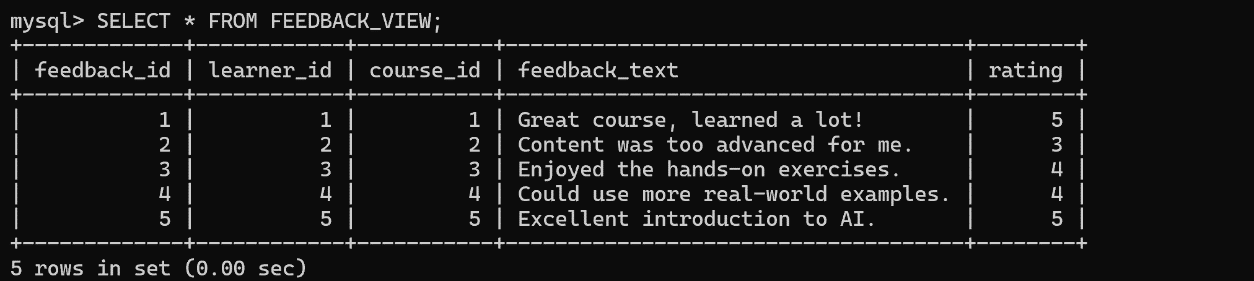


**View for Feedback table**

CREATE VIEW Feedback\_View AS

SELECT feedback\_id, learner\_id, course\_id, feedback\_text, rating

FROM Feedback;

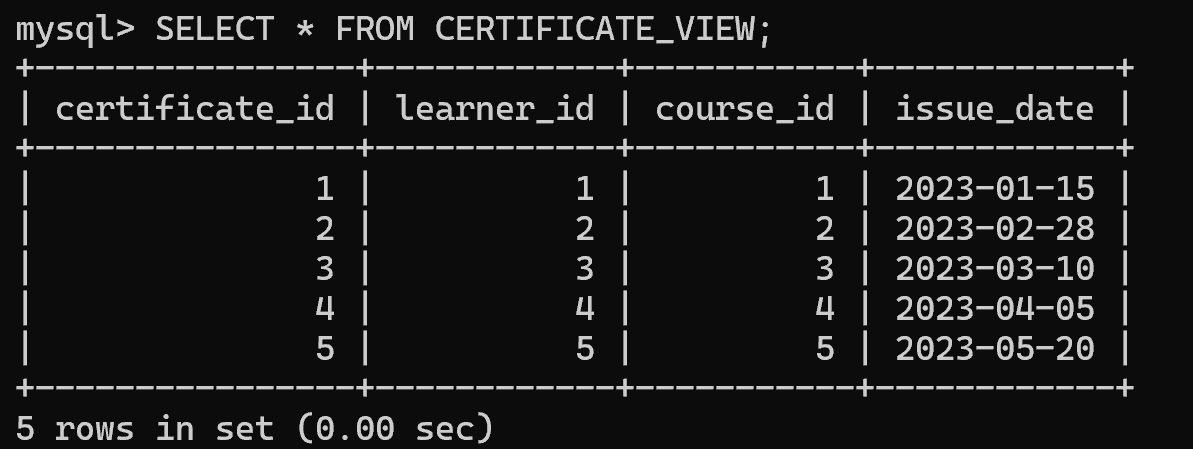


**View for Certificate table**

CREATE VIEW Certificate\_View AS

SELECT certificate\_id, learner\_id, course\_id, issue\_date

FROM Certificate;



**View for Administrator table**

CREATE VIEW Administrator\_View AS

SELECT admin\_id, first\_name, last\_name, email

FROM Administrator;

**SETS AND JOINS**

-- **Query to find learners who provided feedback, along with their courses and instructors**

**-- Selecting learners who provided feedback along with their feedback detail**s

SELECT

L.first\_name AS learner\_first\_name,

L.last\_name AS learner\_last\_name,

F.feedback\_text,

F.rating,

NULL AS course\_name, -- Placeholder for course name

NULL AS instructor\_first\_name, -- Placeholder for instructor first name

NULL AS instructor\_last\_name -- Placeholder for instructor last name

FROM

Learners L

LEFT JOIN

Feedback F ON L.learner\_id = F.learner\_id

UNION

course\_id | course\_name | instructor\_id | first\_name | last\_name | email

----------|--------------------------|---------------|------------|--------------------|-------------------------

1 | Introduction to Programming | 1 | Michael | Johnson | michael.johnson@example.com

2 | Web Development Fundamentals | 2 | Emily | Brown | emily.brown@example.com

-- **Selecting learners along with their enrolled courses and instructors**

SELECT

L.first\_name AS learner\_first\_name,

L.last\_name AS learner\_last\_name,

NULL AS feedback\_text, -- Placeholder for feedback text

NULL AS rating, -- Placeholder for rating

C.course\_name,

I.first\_name AS instructor\_first\_name,

I.last\_name AS instructor\_last\_name

FROM

Learners L

LEFT JOIN

Courses C ON L.course\_id = C.course\_id

LEFT JOIN

Instructors I ON C.instructor\_id = I.instructor\_id;

**Triggers**

**Trigger for Learners Table:**

Let's say you want to automatically insert a record into the Certificate table whenever a new learner is inserted into the Learners table.

sql CREATE TRIGGER insert\_certificate AFTER INSERT ON Learners

FOR EACH ROW

BEGIN

INSERT INTO Certificate (learner\_id, course\_id, issue\_date)

VALUES (NEW.learner\_id, 1, NOW()); -- Assuming course\_id 1 is a default course for certification

END;

**Trigger for Courses Table:**

Let's say you want to automatically insert a record into the Quiz table whenever a new course is inserted into the Courses table.

sql

CREATE TRIGGER insert\_quiz AFTER INSERT ON Courses

FOR EACH ROW

BEGIN

INSERT INTO Quiz (course\_id, quiz\_name)

VALUES (NEW.course\_id, CONCAT('Quiz for ', NEW.course\_name));

END;

**Trigger for Feedback Table:**

Let's say you want to log feedback actions in a separate table whenever a new feedback is inserted into the Feedback table.

sql

CREATE TRIGGER log\_feedback\_action AFTER INSERT ON Feedback

FOR EACH ROW

BEGIN

INSERT INTO Feedback\_Log (action\_type, learner\_id, course\_id, feedback\_text, rating, action\_timestamp)

VALUES ('INSERT', NEW.learner\_id, NEW.course\_id, NEW.feedback\_text, NEW.rating, NOW());

END;

CREATE TRIGGER prevent\_delete\_certificate BEFORE DELETE ON Certificate

FOR EACH ROW

BEGIN

-- Check if the certificate is associated with a learner

IF OLD.learner\_id IS NOT NULL THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Cannot delete certificates associated with learners.';

END IF;

END;

CREATE TRIGGER log\_administrator\_updates AFTER UPDATE ON Administrator

FOR EACH ROW

BEGIN

INSERT INTO Administrator\_Log (admin\_id, updated\_column, old\_value, new\_value, update\_timestamp)

VALUES (NEW.admin\_id, 'ALL', CONCAT('Old values: ', OLD.first\_name, ', ', OLD.last\_name, ', ', OLD.email), CONCAT('New values: ', NEW.first\_name, ', ', NEW.last\_name, ', ', NEW.email), NOW());

END;

**CURSOR**

**Cursor for Learners table**

DROP PROCEDURE IF EXISTS fetch\_learners;

DELIMITER //

CREATE PROCEDURE fetch\_learners()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE learner\_id INT;

DECLARE first\_name VARCHAR(50);

DECLARE last\_name VARCHAR(50);

DECLARE email VARCHAR(100);

DECLARE date\_of\_birth DATE;

DECLARE enrollment\_date TIMESTAMP;

DECLARE cur CURSOR FOR SELECT \* FROM Learners;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO learner\_id, first\_name, last\_name, email, date\_of\_birth, enrollment\_date;

IF done THEN

LEAVE read\_loop;

END IF;

-- Process fetched data as needed

-- For example, you can print or manipulate the fetched data

-- Here, we'll just print the fetched data

SELECT learner\_id, first\_name, last\_name, email, date\_of\_birth, enrollment\_date;

END LOOP;

CLOSE cur;

END //

DELIMITER ;

**Cursor for Instructors table**

DROP PROCEDURE IF EXISTS fetch\_instructors;

DELIMITER //

CREATE PROCEDURE fetch\_instructors()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE instructor\_id INT;

DECLARE first\_name VARCHAR(50);

DECLARE last\_name VARCHAR(50);

DECLARE email VARCHAR(100);

DECLARE cur CURSOR FOR SELECT \* FROM Instructors;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO instructor\_id, first\_name, last\_name, email;

IF done THEN

LEAVE read\_loop;

END IF;

-- Process fetched data as needed

-- For example, you can print or manipulate the fetched data

-- Here, we'll just print the fetched data

SELECT instructor\_id, first\_name, last\_name, email;

END LOOP;

CLOSE cur;

END //

DELIMITER ;

**Cursor for Administrator table**

DROP PROCEDURE IF EXISTS fetch\_administrators;

DELIMITER //

CREATE PROCEDURE fetch\_administrators()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE admin\_id INT;

DECLARE first\_name VARCHAR(50);

DECLARE last\_name VARCHAR(50);

DECLARE email VARCHAR(100);

DECLARE cur CURSOR FOR SELECT \* FROM Administrator;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO admin\_id, first\_name, last\_name, email;

IF done THEN

LEAVE read\_loop;

END IF;

-- Process fetched data as needed

-- For example, you can print or manipulate the fetched data

-- Here, we'll just print the fetched data

SELECT admin\_id, first\_name, last\_name, email;

END LOOP;

CLOSE cur;

END //

DELIMITER ;

**Cursor for Courses table**

DROP PROCEDURE IF EXISTS fetch\_courses;

DELIMITER //

CREATE PROCEDURE fetch\_courses()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE course\_id INT;

DECLARE course\_name VARCHAR(100);

DECLARE instructor\_id INT;

DECLARE cur CURSOR FOR SELECT \* FROM Courses;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO course\_id, course\_name, instructor\_id;

IF done THEN

LEAVE read\_loop;

END IF;

-- Process fetched data as needed

-- For example, you can print or manipulate the fetched data

-- Here, we'll just print the fetched data

SELECT course\_id, course\_name, instructor\_id;

END LOOP;

CLOSE cur;

END //

DELIMITER ;

**Cursor for Quiz table**

DROP PROCEDURE IF EXISTS fetch\_quiz;

DELIMITER //

CREATE PROCEDURE fetch\_quiz()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE quiz\_id INT;

DECLARE course\_id INT;

DECLARE quiz\_name VARCHAR(100);

DECLARE cur CURSOR FOR SELECT \* FROM Quiz;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO quiz\_id, course\_id, quiz\_name;

IF done THEN

LEAVE read\_loop;

END IF;

-- Process fetched data as needed

-- For example, you can print or manipulate the fetched data

-- Here, we'll just print the fetched data

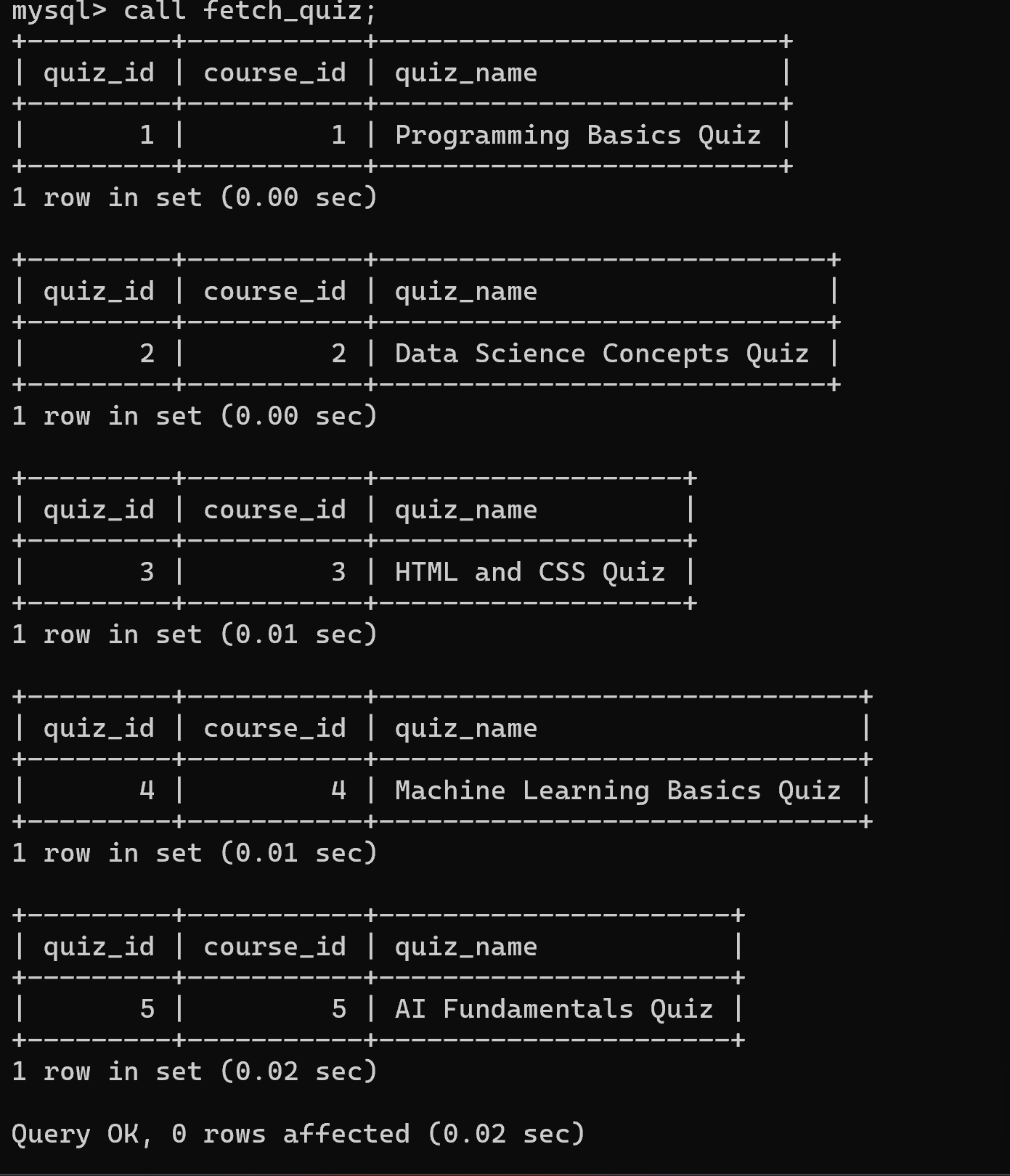
SELECT quiz\_id, course\_id, quiz\_name;

END LOOP;

CLOSE cur;

END //

DELIMITER ;



**Cursor for Feedback table**

DROP PROCEDURE IF EXISTS fetch\_feedback;

DELIMITER //

CREATE PROCEDURE fetch\_feedback()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE feedback\_id INT;

DECLARE learner\_id INT;

DECLARE course\_id INT;

DECLARE feedback\_text TEXT;

DECLARE rating INT;

DECLARE cur CURSOR FOR SELECT \* FROM Feedback;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO feedback\_id, learner\_id, course\_id, feedback\_text, rating;

IF done THEN

LEAVE read\_loop;

END IF;

-- Process fetched data as needed

-- For example, you can print or manipulate the fetched data

-- Here, we'll just print the fetched data

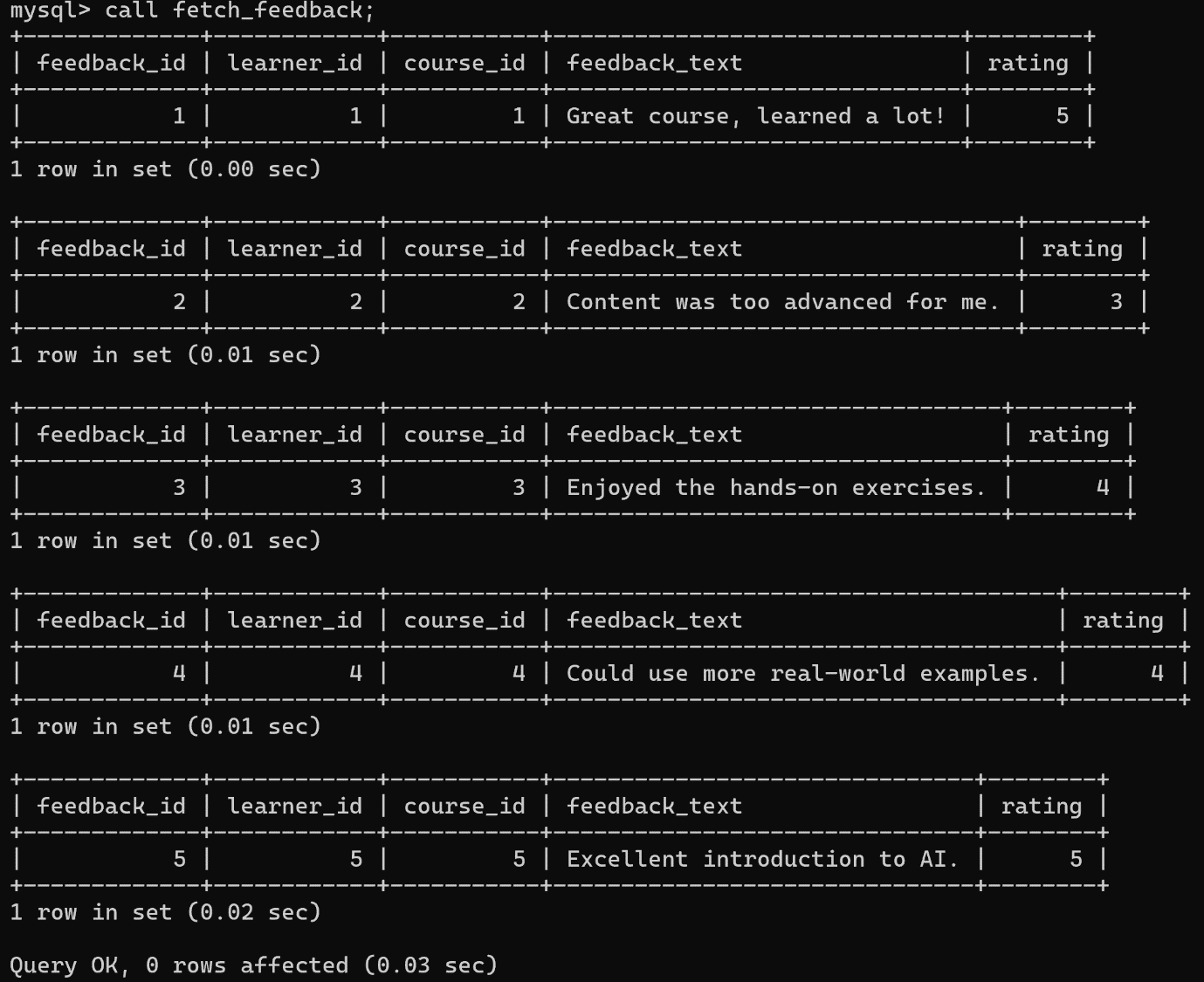
SELECT feedback\_id, learner\_id, course\_id, feedback\_text, rating;

END LOOP;

CLOSE cur;

END //

DELIMITER ;



**Cursor for Certificate table**

DROP PROCEDURE IF EXISTS fetch\_certificate;

DELIMITER //

CREATE PROCEDURE fetch\_certificate()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE certificate\_id INT;

DECLARE learner\_id INT;

DECLARE course\_id INT;

DECLARE issue\_date DATE;

DECLARE cur CURSOR FOR SELECT \* FROM Certificate;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO certificate\_id, learner\_id, course\_id, issue\_date;

IF done THEN

LEAVE read\_loop;

END IF;

-- Process fetched data as needed

-- For example, you can print or manipulate the fetched data

-- Here, we'll just print the fetched data

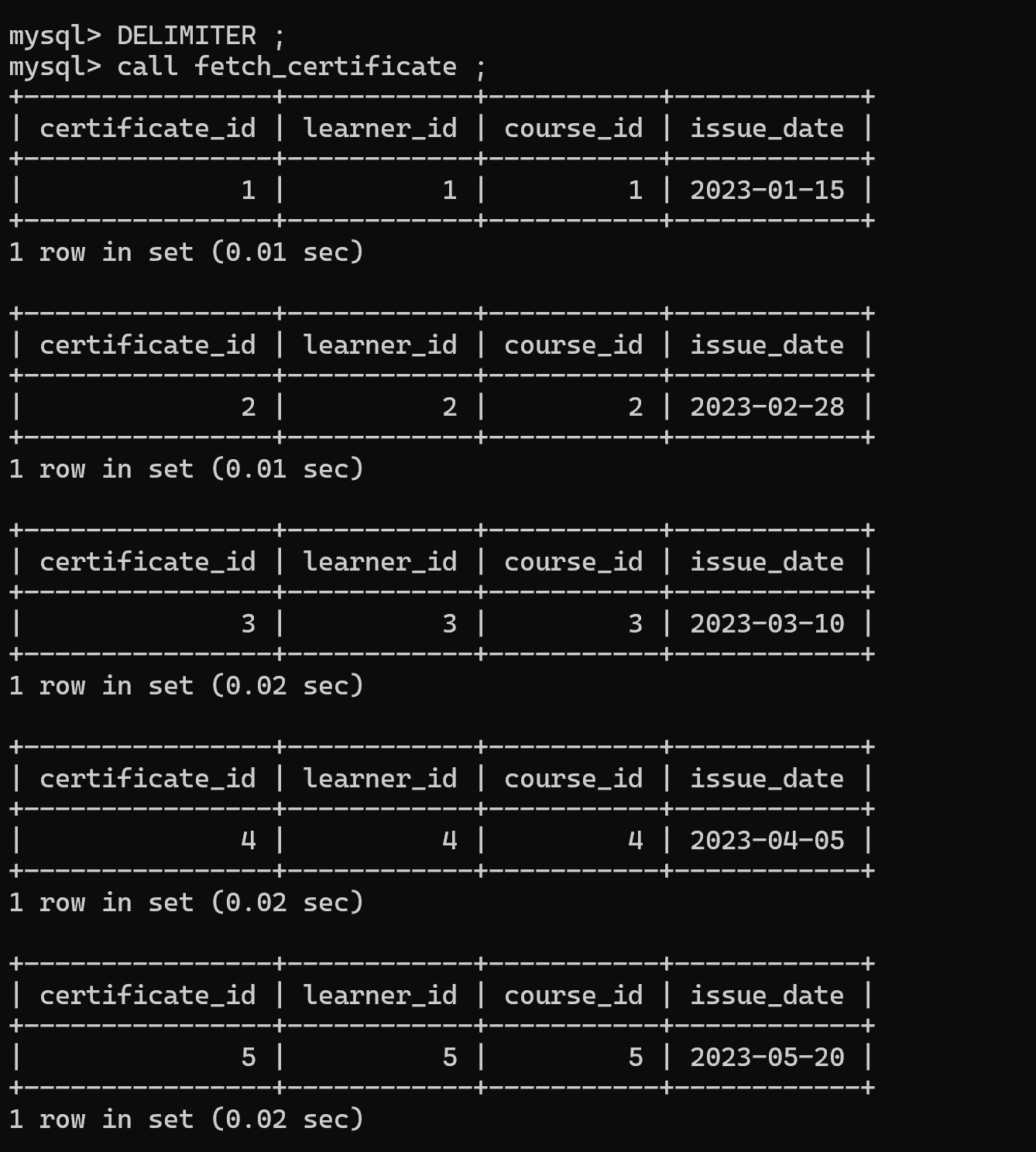
SELECT certificate\_id, learner\_id, course\_id, issue\_date;

END LOOP;

CLOSE cur;

END //

DELIMITER ;



DELIMITER //

CREATE TRIGGER UpdateEnrollmentDate

BEFORE INSERT ON Learners

FOR EACH ROW

BEGIN

SET NEW.enrollment\_date = CURRENT\_TIMESTAMP;

END;

//

DELIMITER ;

INSERT INTO Learners (first\_name, last\_name, email, password, date\_of\_birth)

VALUES ('Sarah', 'Johnson', 'sarah.johnson@example.com', 'sarah123', '1998-07-10');

SELECT \* FROM Learners;

UPDATE Learners

SET password = 'newpassword'

WHERE learner\_id = 1;

DELETE FROM Learners

WHERE learner\_id = 2;