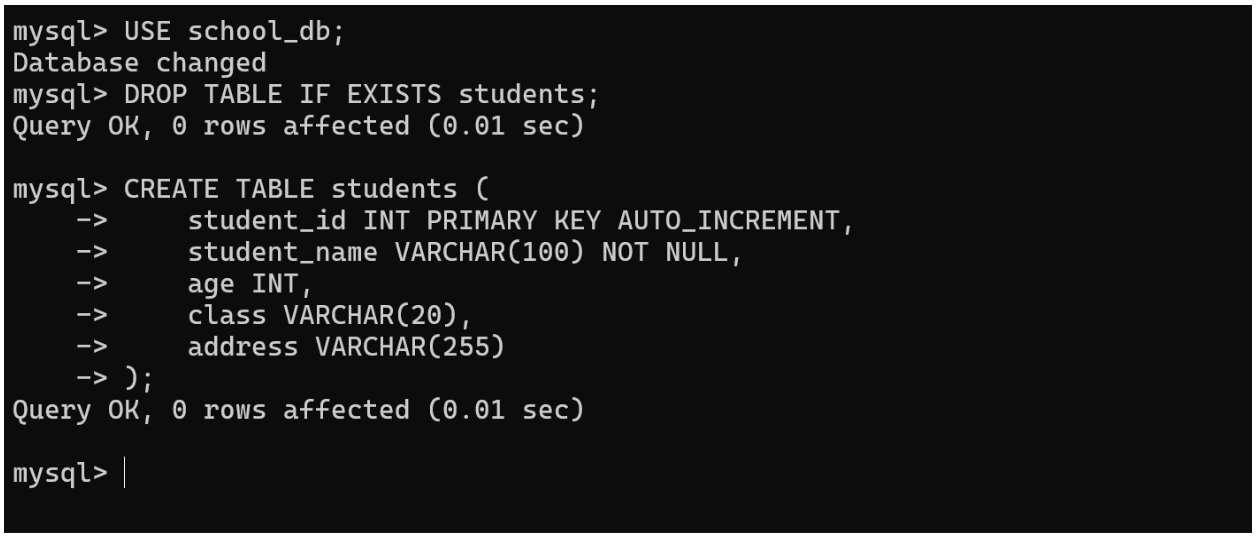
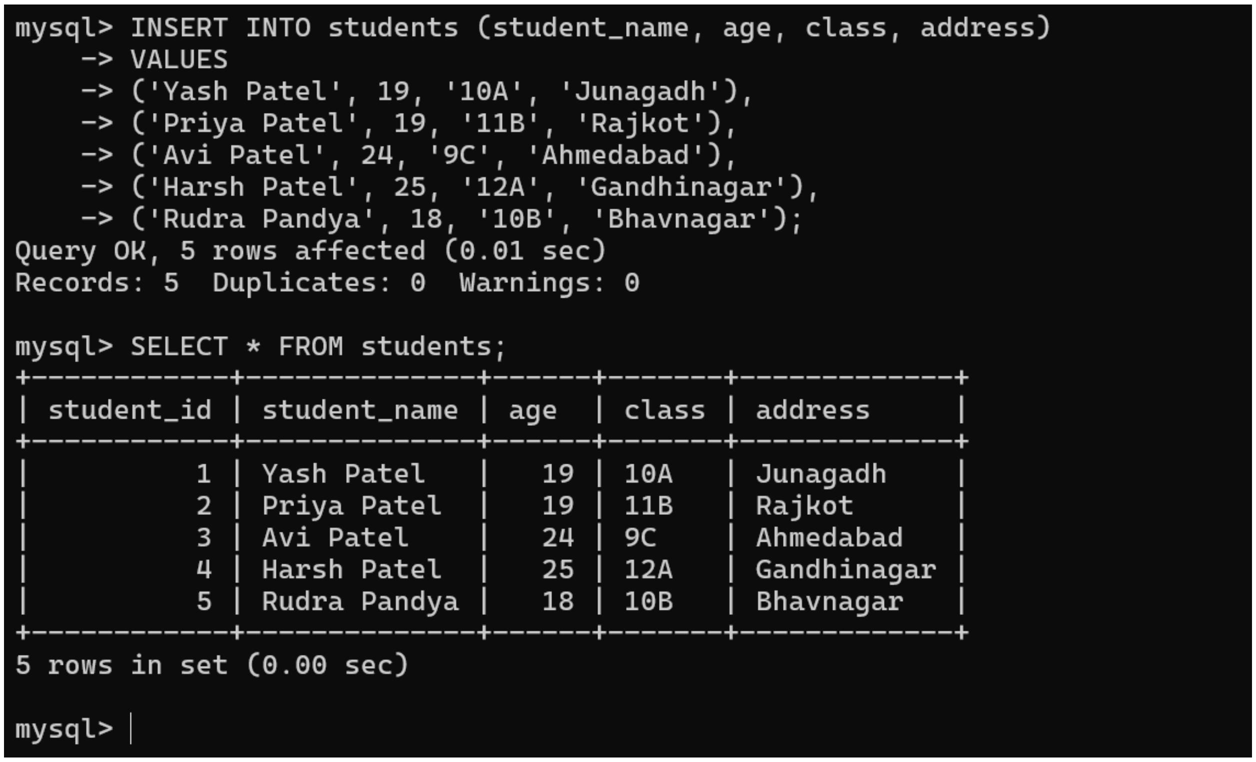
Module 4 Introduction to DBMS

1. Create a new database named school_db and a table called students with the following columns: student_id, student_name, age, class, and address.



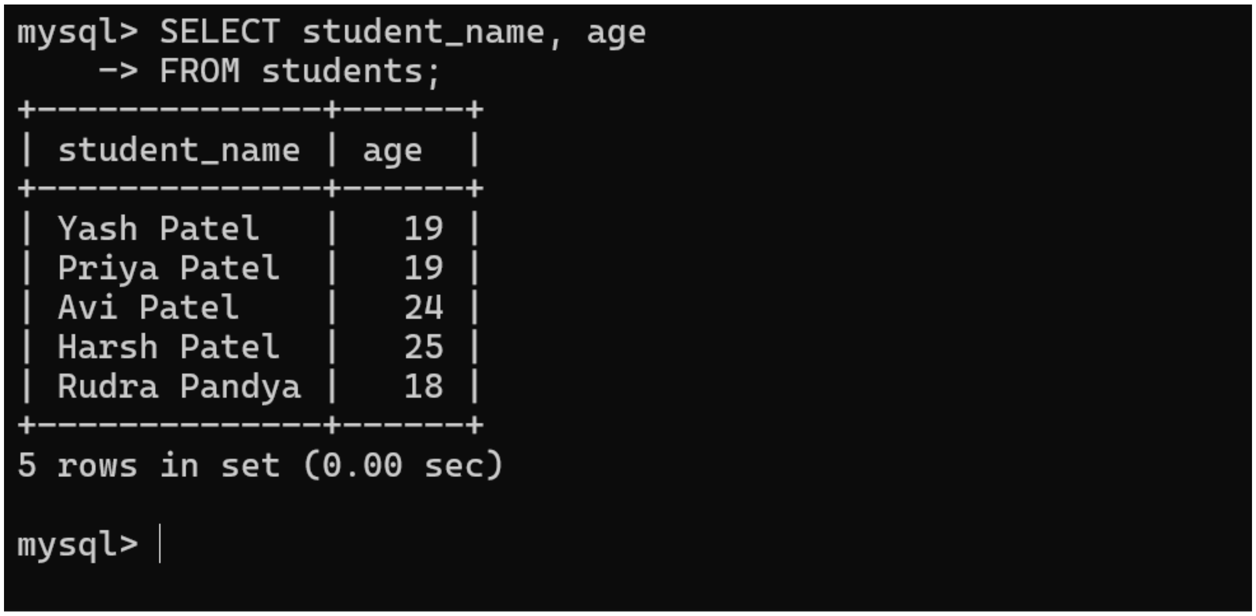
```
mysql> USE school_db;
Database changed
mysql> DROP TABLE IF EXISTS students;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE students (
    ->     student_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     student_name VARCHAR(100) NOT NULL,
    ->     age INT,
    ->     class VARCHAR(20),
    ->     address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
```

2. Insert five records into the students table and retrieve all records using the SELECT statement



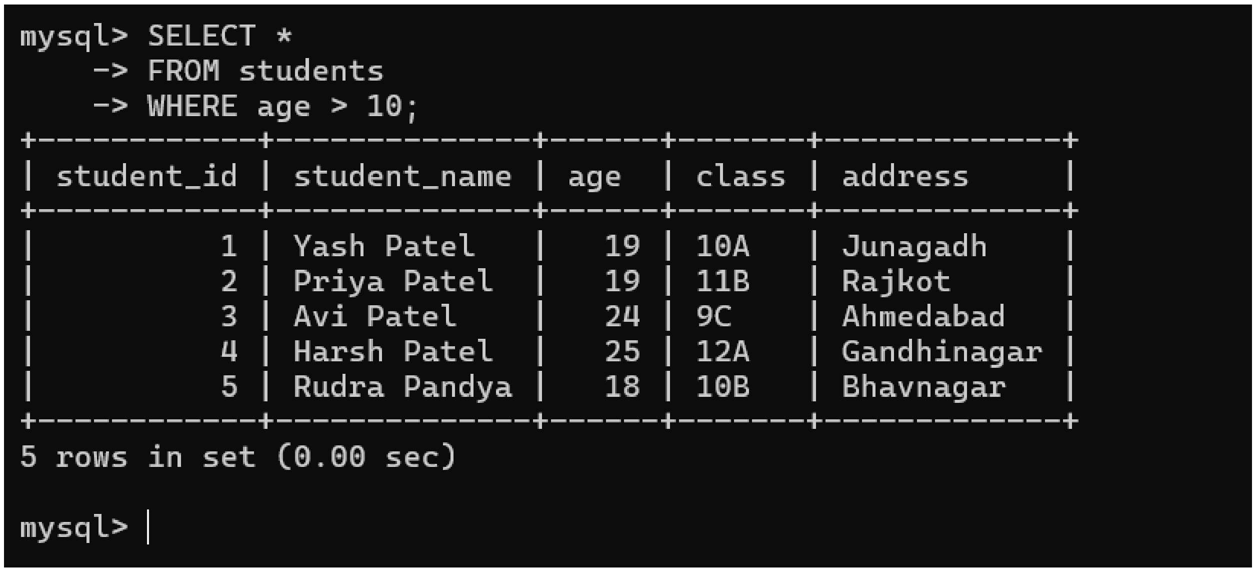
```
mysql> INSERT INTO students (student_name, age, class, address)
    -> VALUES
    -> ('Yash Patel', 19, '10A', 'Junagadh'),
    -> ('Priya Patel', 19, '11B', 'Rajkot'),
    -> ('Avi Patel', 24, '9C', 'Ahmedabad'),
    -> ('Harsh Patel', 25, '12A', 'Gandhinagar'),
    -> ('Rudra Pandya', 18, '10B', 'Bhavnagar');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM students;
+------------+--------------+------+-------+-------------+
| student_id | student_name | age  | class | address     |
+------------+--------------+------+-------+-------------+
|          1 | Yash Patel   |   19 | 10A   | Junagadh    |
|          2 | Priya Patel  |   19 | 11B   | Rajkot      |
|          3 | Avi Patel    |   24 | 9C    | Ahmedabad   |
|          4 | Harsh Patel  |   25 | 12A   | Gandhinagar |
|          5 | Rudra Pandya |   18 | 10B   | Bhavnagar   |
+------------+--------------+------+-------+-------------+
5 rows in set (0.00 sec)

mysql>
```

3. Write SQL queries to retrieve specific columns (student_name and age) from the students table.



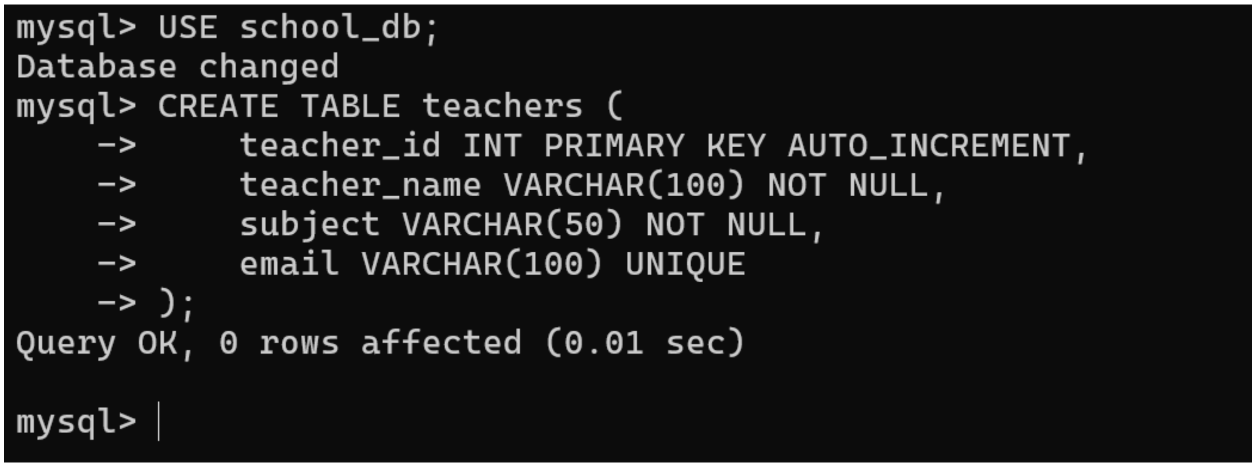
```
mysql> SELECT student_name, age
    -> FROM students;
+--------------+------+
| student_name | age  |
+--------------+------+
| Yash Patel   |   19 |
| Priya Patel  |   19 |
| Avi Patel    |   24 |
| Harsh Patel  |   25 |
| Rudra Pandya |   18 |
+--------------+------+
5 rows in set (0.00 sec)

mysql>
```

4. Write SQL queries to retrieve all students whose age is greater than 10.

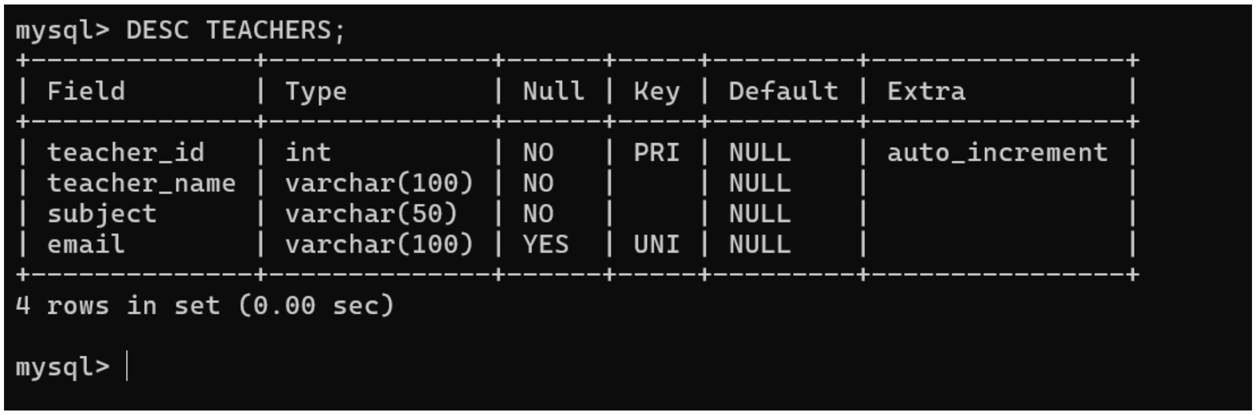


```
mysql> SELECT *
    -> FROM students
    -> WHERE age > 10;
+------------+--------------+------+-------+-------------+
| student_id | student_name | age  | class | address     |
+------------+--------------+------+-------+-------------+
|          1 | Yash Patel   |   19 | 10A   | Junagadh    |
|          2 | Priya Patel  |   19 | 11B   | Rajkot      |
|          3 | Avi Patel    |   24 | 9C    | Ahmedabad   |
|          4 | Harsh Patel  |   25 | 12A   | Gandhinagar |
|          5 | Rudra Pandya |   18 | 10B   | Bhavnagar   |
+------------+--------------+------+-------+-------------+
5 rows in set (0.00 sec)

mysql>
```

5. Create a table teachers with the following columns: teacher_id (Primary Key), teacher_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

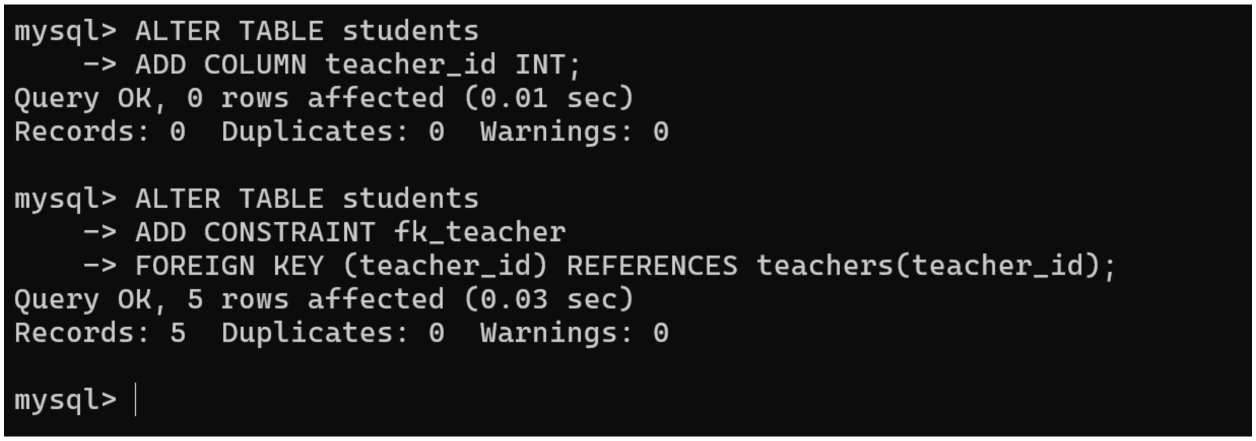


```
mysql> USE school_db;
Database changed
mysql> CREATE TABLE teachers (
    ->     teacher_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     teacher_name VARCHAR(100) NOT NULL,
    ->     subject VARCHAR(50) NOT NULL,
    ->     email VARCHAR(100) UNIQUE
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
```

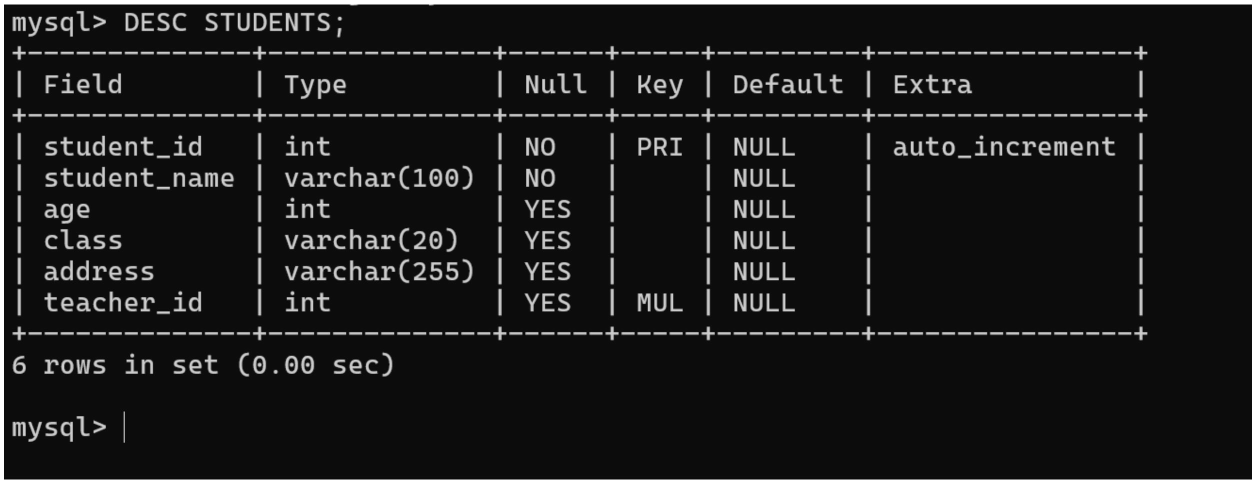


```
mysql> DESC TEACHERS;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| teacher_id   | int          | NO   | PRI | NULL    | auto_increment |
| teacher_name | varchar(100) | NO   |     | NULL    |                |
| subject      | varchar(50)  | NO   |     | NULL    |                |
| email        | varchar(100) | YES  | UNI | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

6. Implement a FOREIGN KEY constraint to relate the teacher_id from the teachers table with the students table.

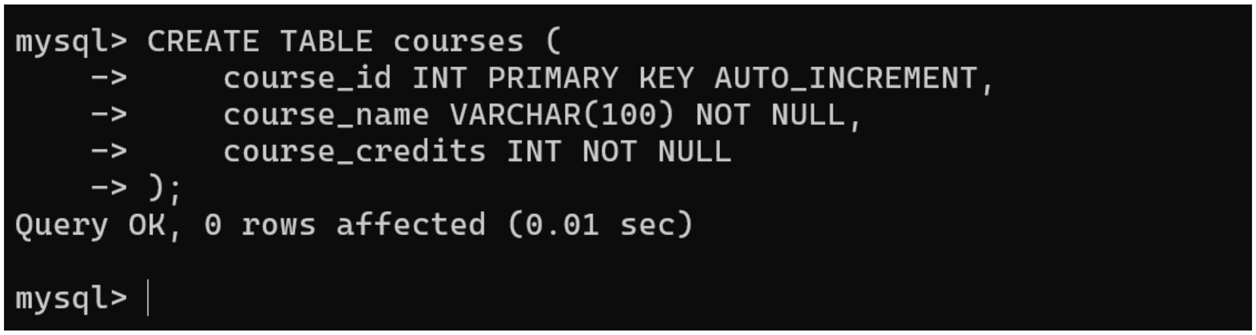


```
mysql> ALTER TABLE students
    -> ADD COLUMN teacher_id INT;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE students
    -> ADD CONSTRAINT fk_teacher
    -> FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id);
Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
```

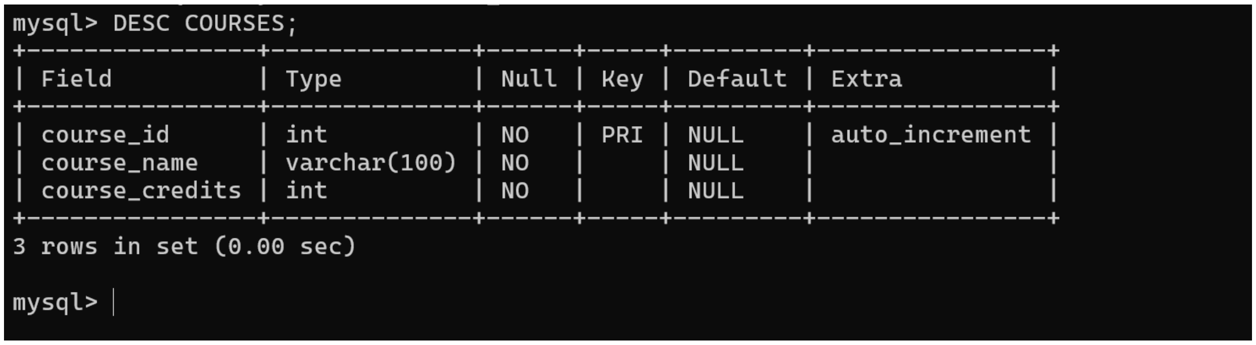


```
mysql> DESC STUDENTS;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| student_id   | int          | NO   | PRI | NULL    | auto_increment |
| student_name | varchar(100) | NO   |     | NULL    |                |
| age          | int          | YES  |     | NULL    |                |
| class        | varchar(20)  | YES  |     | NULL    |                |
| address      | varchar(255) | YES  |     | NULL    |                |
| teacher_id   | int          | YES  | MUL | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql>
```

7. Create a table courses with columns: course_id, course_name, and course_credits. Set the course_id as the primary key.

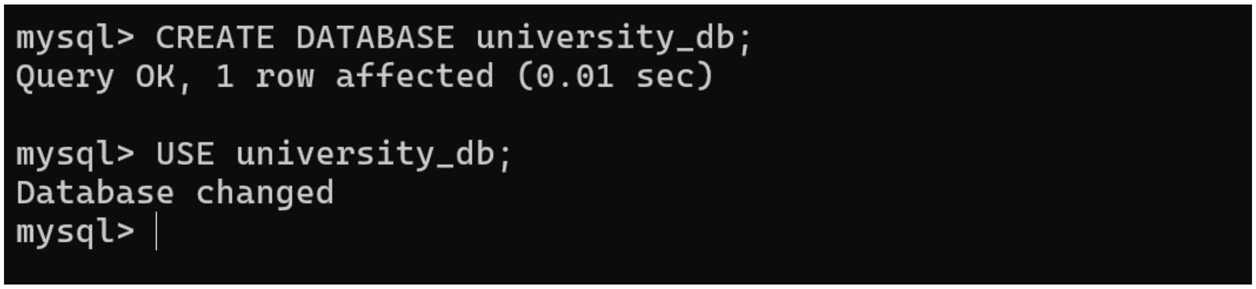


```
mysql> CREATE TABLE courses (
    ->     course_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     course_name VARCHAR(100) NOT NULL,
    ->     course_credits INT NOT NULL
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
```



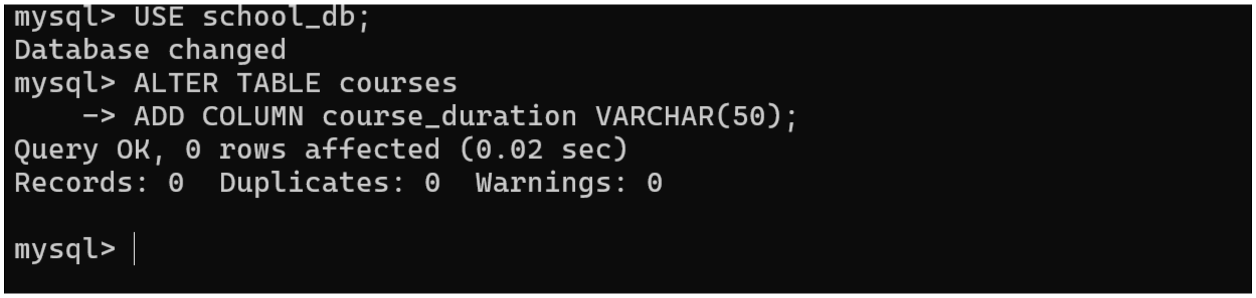
```
mysql> DESC COURSES;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| course_id      | int          | NO   | PRI | NULL    | auto_increment |
| course_name    | varchar(100) | NO   |     | NULL    |                |
| course_credits | int          | NO   |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql>
```

8. Use the CREATE command to create a database university_db.

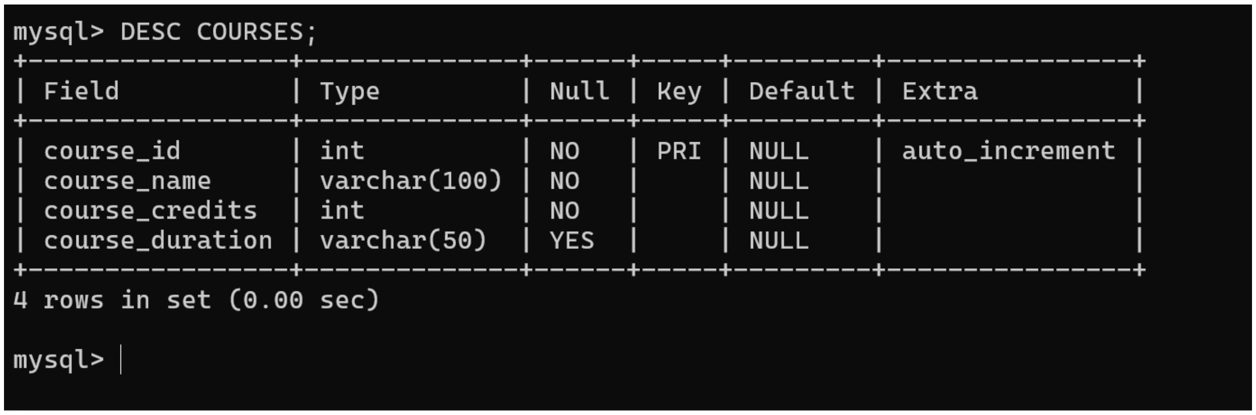


```
mysql> CREATE DATABASE university_db;
Query OK, 1 row affected (0.01 sec)

mysql> USE university_db;
Database changed
mysql>
```

9. Modify the courses table by adding a column course_duration using the ALTER command.

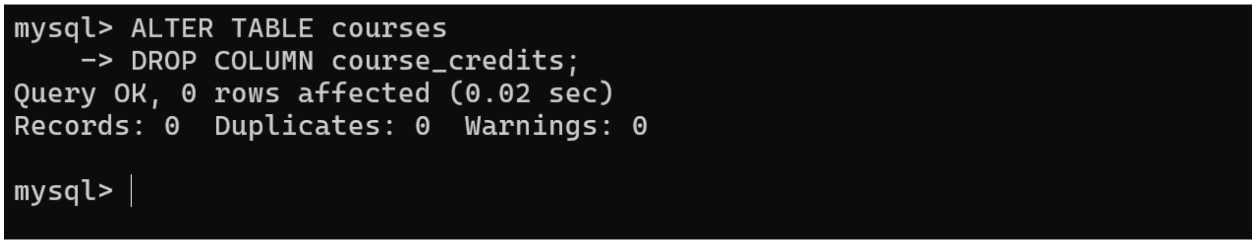


```
mysql> USE school_db;
Database changed
mysql> ALTER TABLE courses
    -> ADD COLUMN course_duration VARCHAR(50);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```



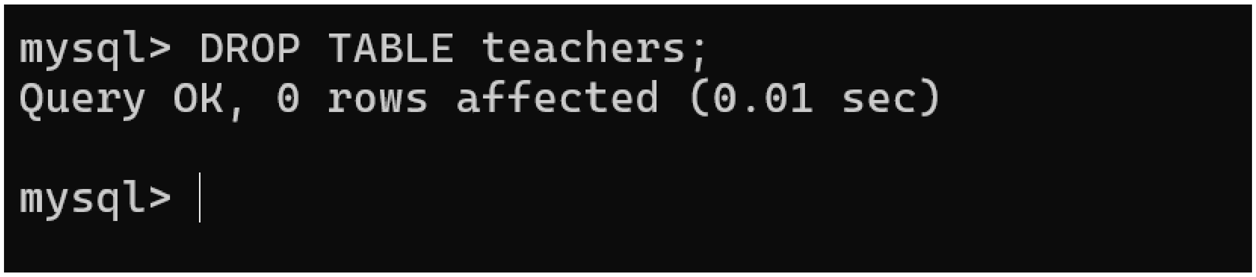
```
mysql> DESC COURSES;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| course_id       | int          | NO   | PRI | NULL    | auto_increment |
| course_name     | varchar(100) | NO   |     | NULL    |                |
| course_credits  | int          | NO   |     | NULL    |                |
| course_duration | varchar(50)  | YES  |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

10. Drop the course_credits column from the courses table.



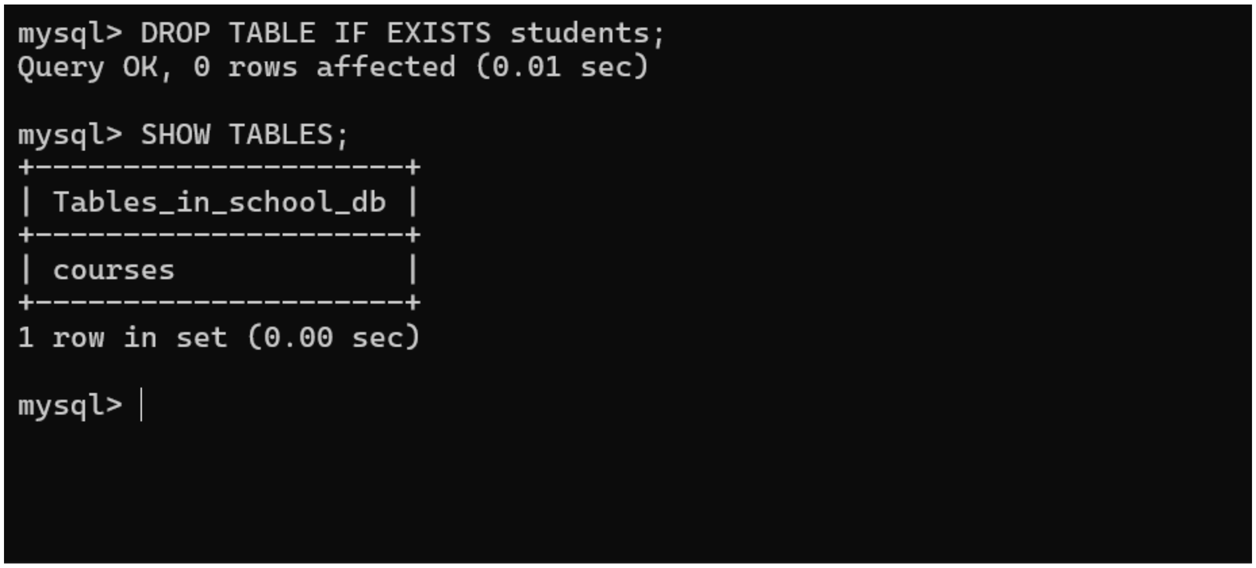
```
mysql> ALTER TABLE courses
    -> DROP COLUMN course_credits;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

11. Drop the teachers table from the school_db database.



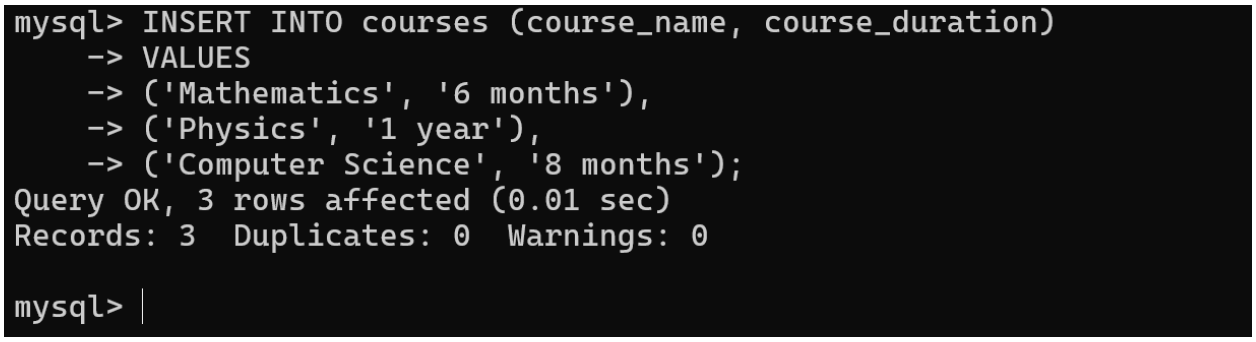
```
mysql> DROP TABLE teachers;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

12. Drop the students table from the school_db database and verify that the table has been removed.

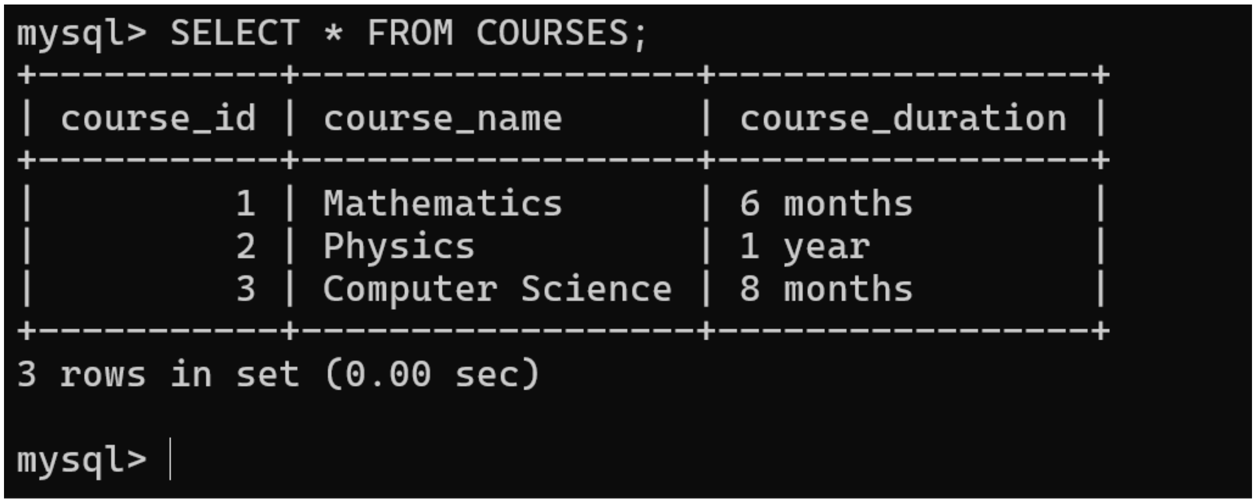


```
mysql> DROP TABLE IF EXISTS students;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES;
+---------------------+
| Tables_in_school_db |
+---------------------+
| courses             |
+---------------------+
1 row in set (0.00 sec)

mysql>
```

13. Insert three records into the courses table using the INSERT command.

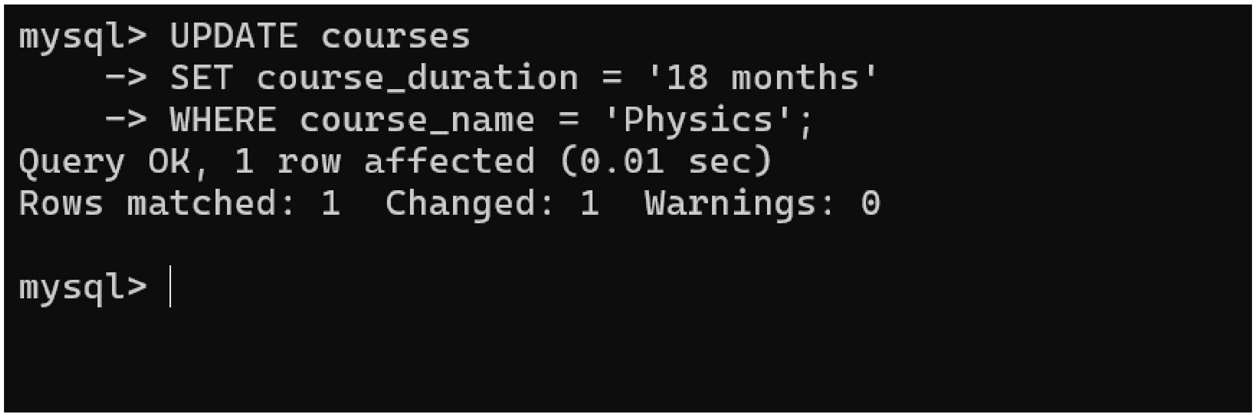


```
mysql> INSERT INTO courses (course_name, course_duration)
    -> VALUES
    -> ('Mathematics', '6 months'),
    -> ('Physics', '1 year'),
    -> ('Computer Science', '8 months');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```



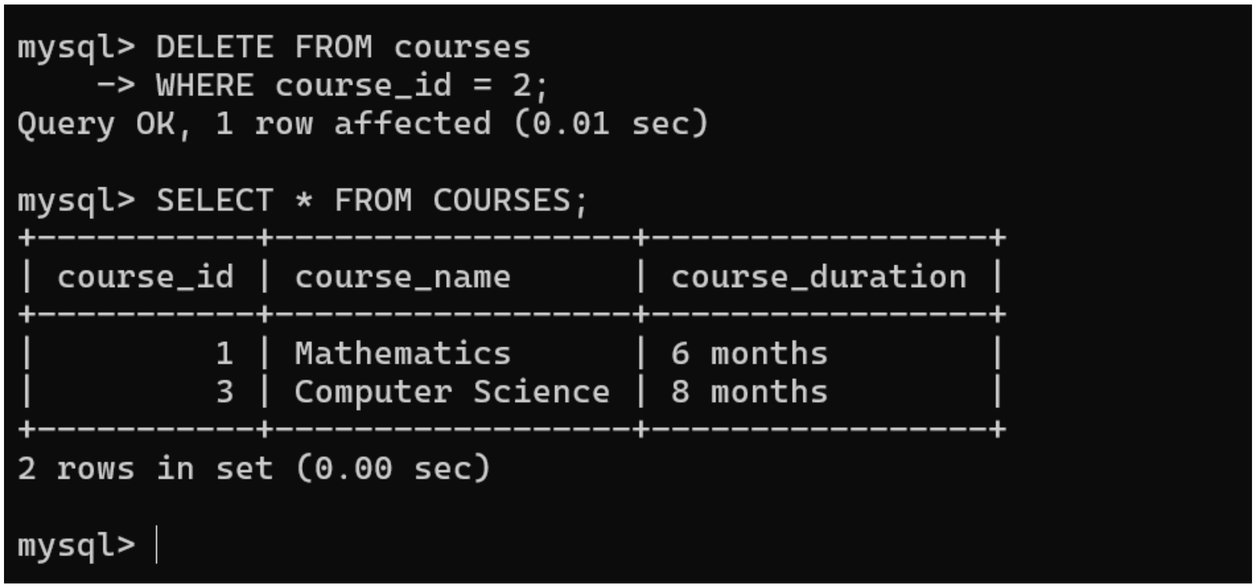
```
mysql> SELECT * FROM COURSES;
+-----------+------------------+-----------------+
| course_id | course_name      | course_duration |
+-----------+------------------+-----------------+
|         1 | Mathematics      | 6 months        |
|         2 | Physics          | 1 year          |
|         3 | Computer Science | 8 months        |
+-----------+------------------+-----------------+
3 rows in set (0.00 sec)

mysql>
```

14. Update the course duration of a specific course using the UPDATE command.



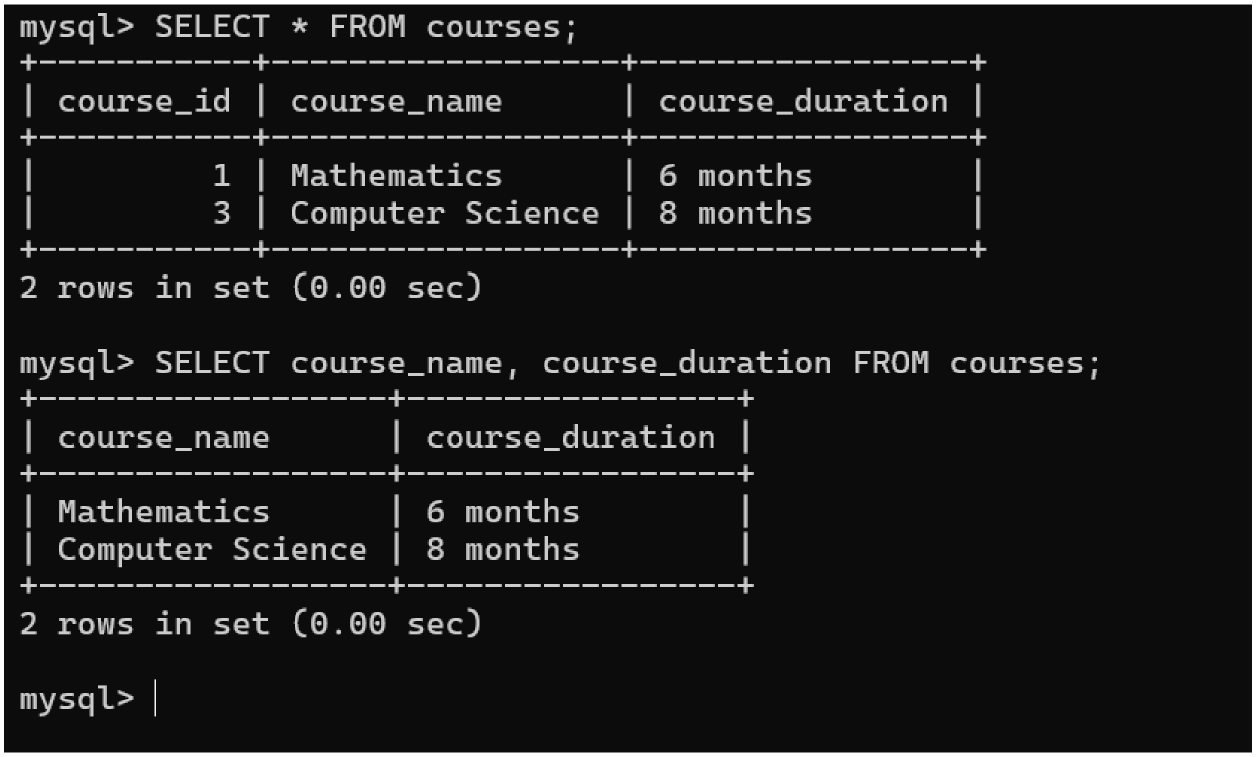
```
mysql> UPDATE courses
    -> SET course_duration = '18 months'
    -> WHERE course_name = 'Physics';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> |
```

15. Delete a course with a specific course_id from the courses table using the DELETE command.



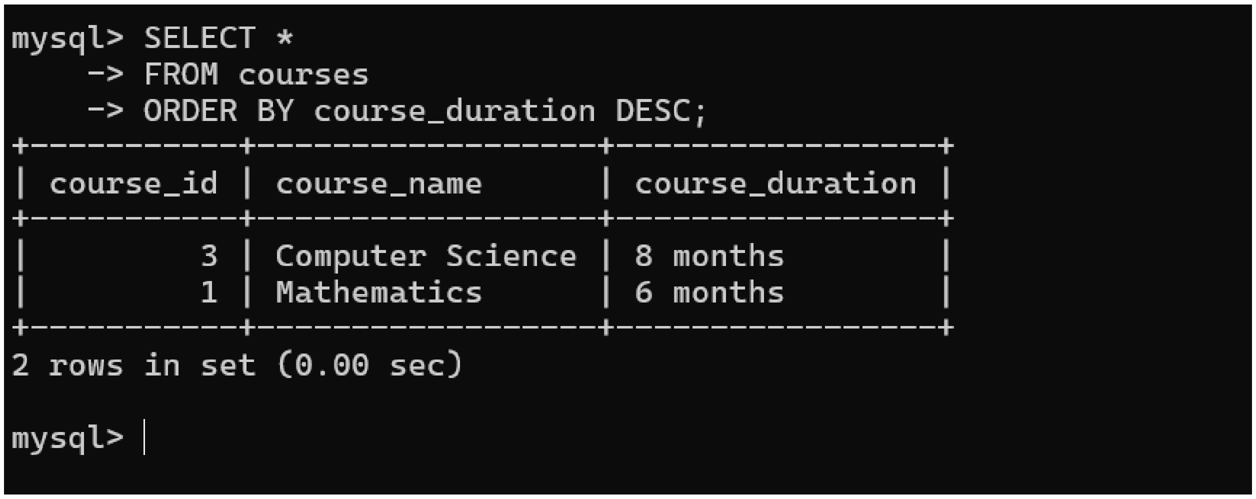
```
mysql> DELETE FROM courses
    -> WHERE course_id = 2;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM COURSES;
+-----------+------------------+-----------------+
| course_id | course_name      | course_duration |
+-----------+------------------+-----------------+
|         1 | Mathematics      | 6 months        |
|         3 | Computer Science | 8 months        |
+-----------+------------------+-----------------+
2 rows in set (0.00 sec)

mysql> |
```

16. Retrieve all courses from the courses table using the SELECT statement.



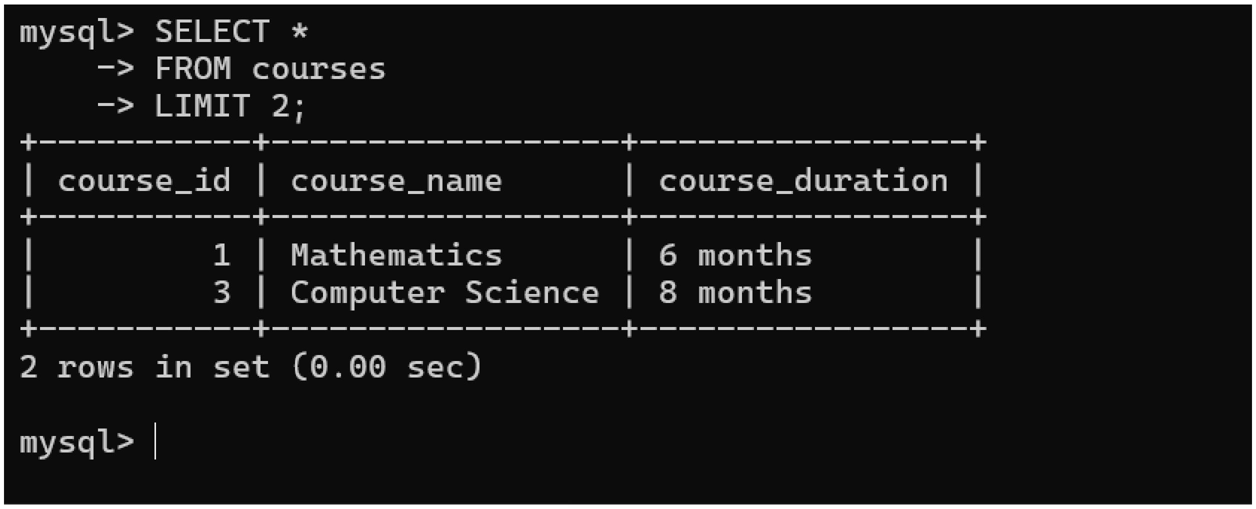
```
mysql> SELECT * FROM courses;
+-----------+------------------+-----------------+
| course_id | course_name      | course_duration |
+-----------+------------------+-----------------+
|         1 | Mathematics      | 6 months        |
|         3 | Computer Science | 8 months        |
+-----------+------------------+-----------------+
2 rows in set (0.00 sec)

mysql> SELECT course_name, course_duration FROM courses;
+------------------+-----------------+
| course_name      | course_duration |
+------------------+-----------------+
| Mathematics      | 6 months        |
| Computer Science | 8 months        |
+------------------+-----------------+
2 rows in set (0.00 sec)

mysql>
```

17. Sort the courses based on course_duration in descending order using ORDER BY.



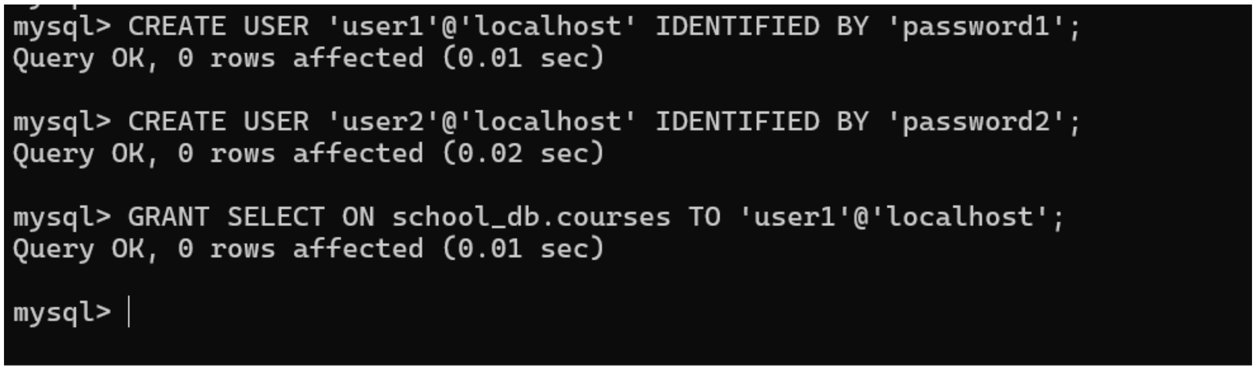
```
mysql> SELECT *
    -> FROM courses
    -> ORDER BY course_duration DESC;
+-----------+------------------+-----------------+
| course_id | course_name      | course_duration |
+-----------+------------------+-----------------+
|         3 | Computer Science | 8 months        |
|         1 | Mathematics      | 6 months        |
+-----------+------------------+-----------------+
2 rows in set (0.00 sec)

mysql>
```

18. Limit the results of the SELECT query to show only the top two courses using LIMIT.



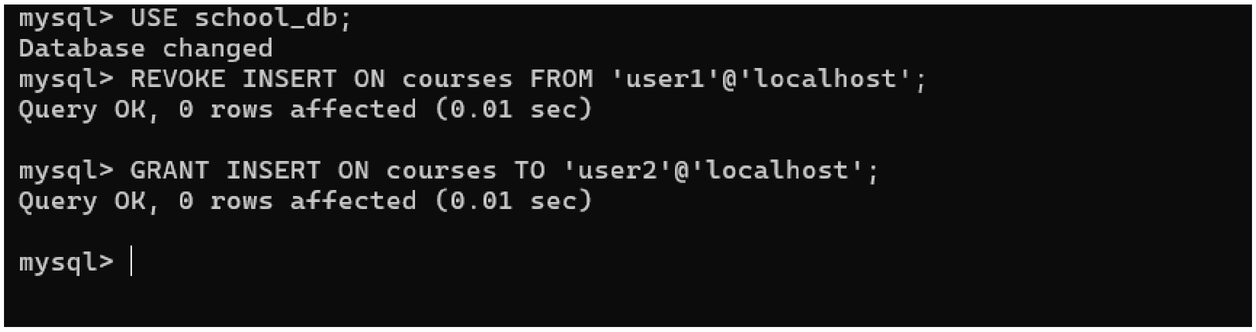
```
mysql> SELECT *
    -> FROM courses
    -> LIMIT 2;
+-----------+-------------------+-----------------+
| course_id | course_name       | course_duration |
+-----------+-------------------+-----------------+
|         1 | Mathematics       | 6 months        |
|         3 | Computer Science  | 8 months        |
+-----------+-------------------+-----------------+
2 rows in set (0.00 sec)

mysql>
```

19. Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table.



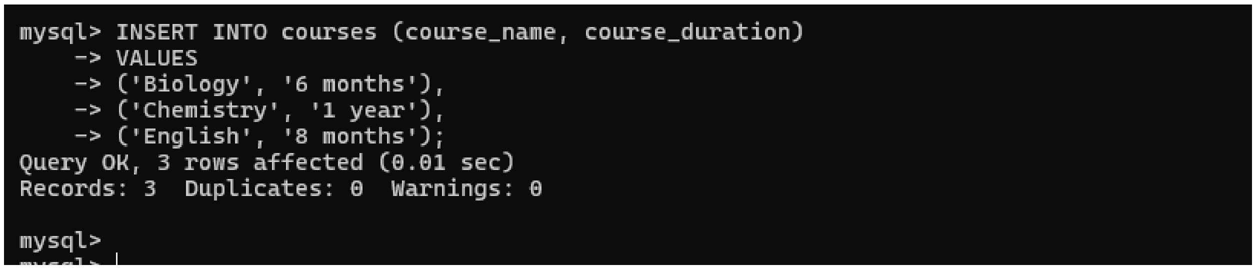
```
mysql> CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER 'user2'@'localhost' IDENTIFIED BY 'password2';
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT SELECT ON school_db.courses TO 'user1'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql>
```

20. Revoke the INSERT permission from user1 and give it to user2.

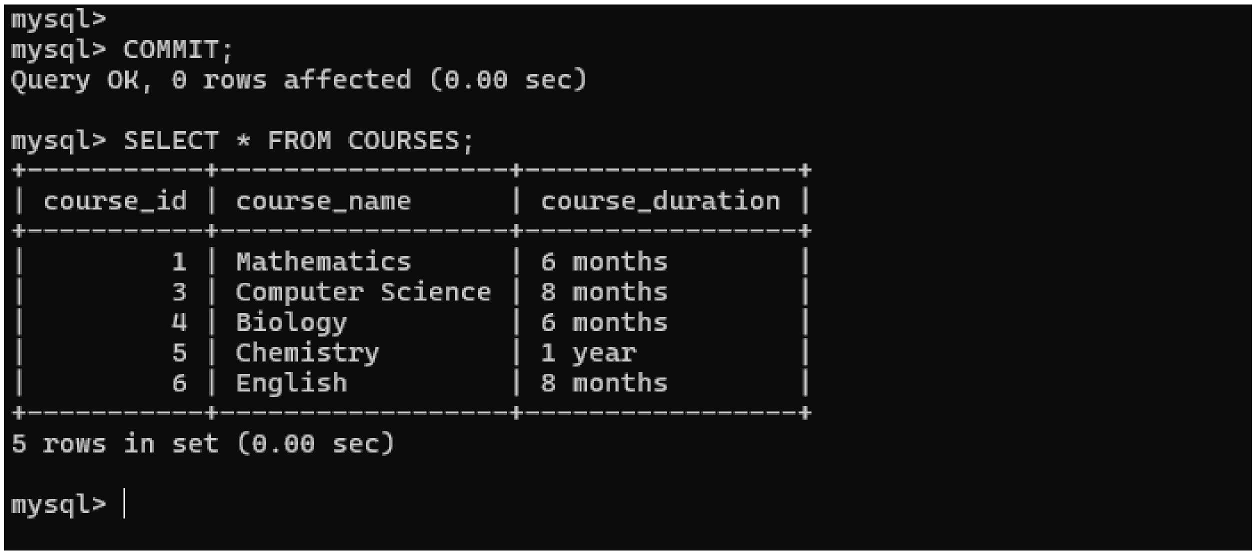


```
mysql> USE school_db;
Database changed
mysql> REVOKE INSERT ON courses FROM 'user1'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT INSERT ON courses TO 'user2'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql>
```

21. Insert a few rows into the courses table and use COMMIT to save the changes.

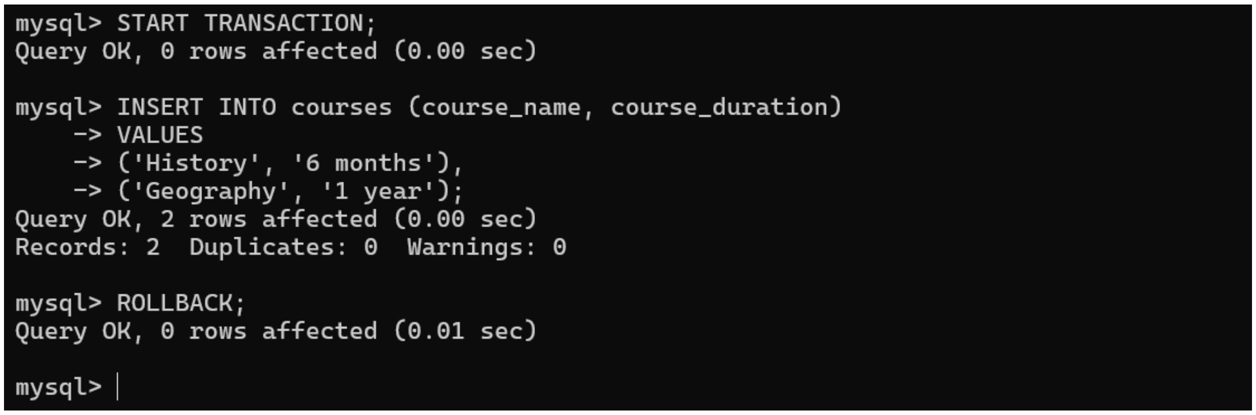


```
mysql> INSERT INTO courses (course_name, course_duration)
    -> VALUES
    -> ('Biology', '6 months'),
    -> ('Chemistry', '1 year'),
    -> ('English', '8 months');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

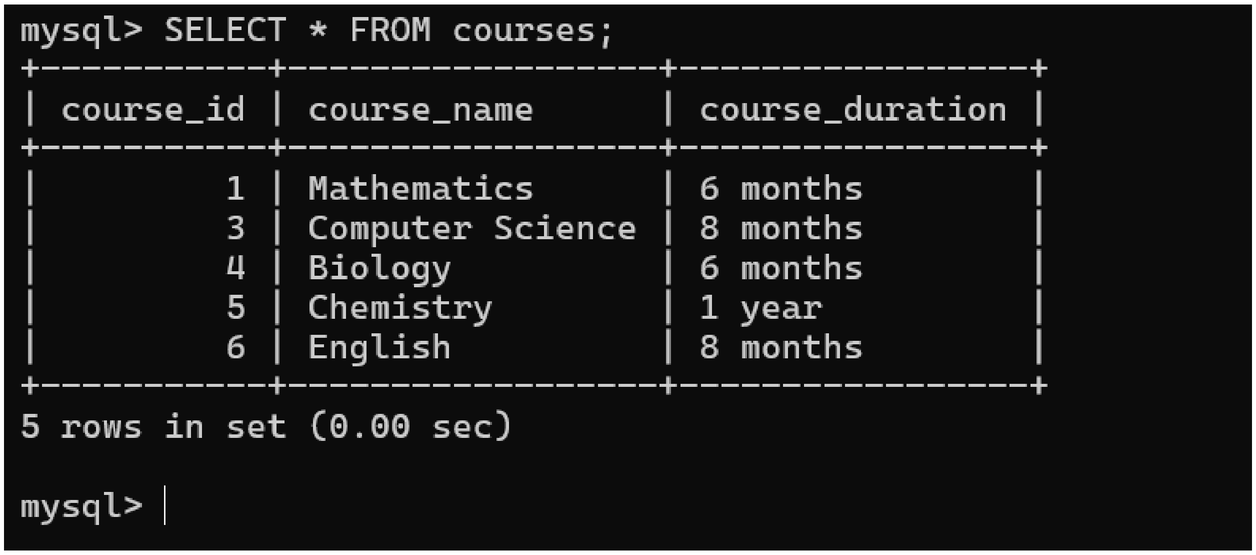


```
mysql>
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM COURSES;
+-----------+------------------+-----------------+
| course_id | course_name      | course_duration |
+-----------+------------------+-----------------+
|         1 | Mathematics      | 6 months        |
|         3 | Computer Science | 8 months        |
|         4 | Biology          | 6 months        |
|         5 | Chemistry        | 1 year          |
|         6 | English          | 8 months        |
+-----------+------------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

22. Insert additional rows, then use ROLLBACK to undo the last insert operation.

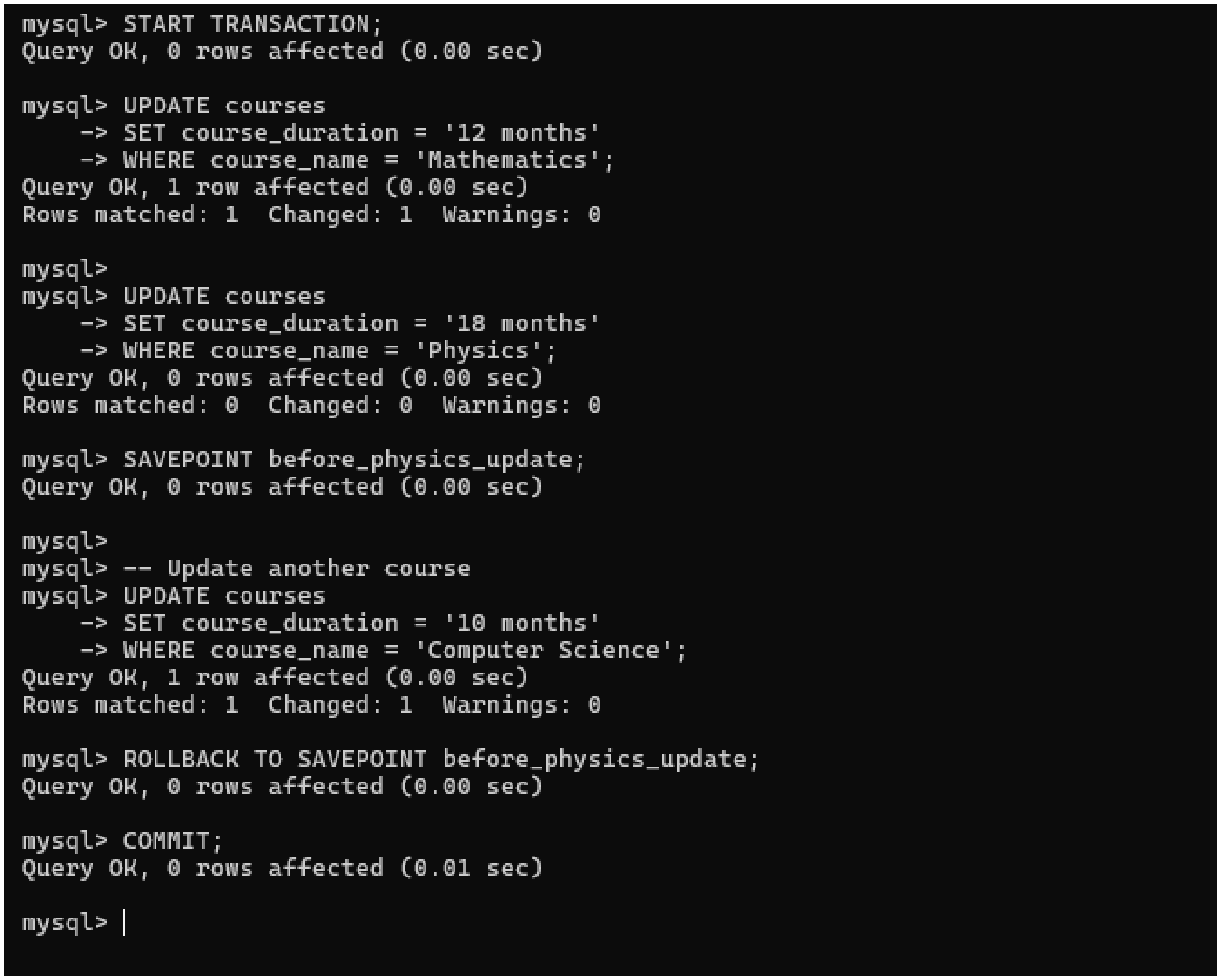


```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO courses (course_name, course_duration)
    -> VALUES
    -> ('History', '6 months'),
    -> ('Geography', '1 year');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> ROLLBACK;
Query OK, 0 rows affected (0.01 sec)

mysql>
```



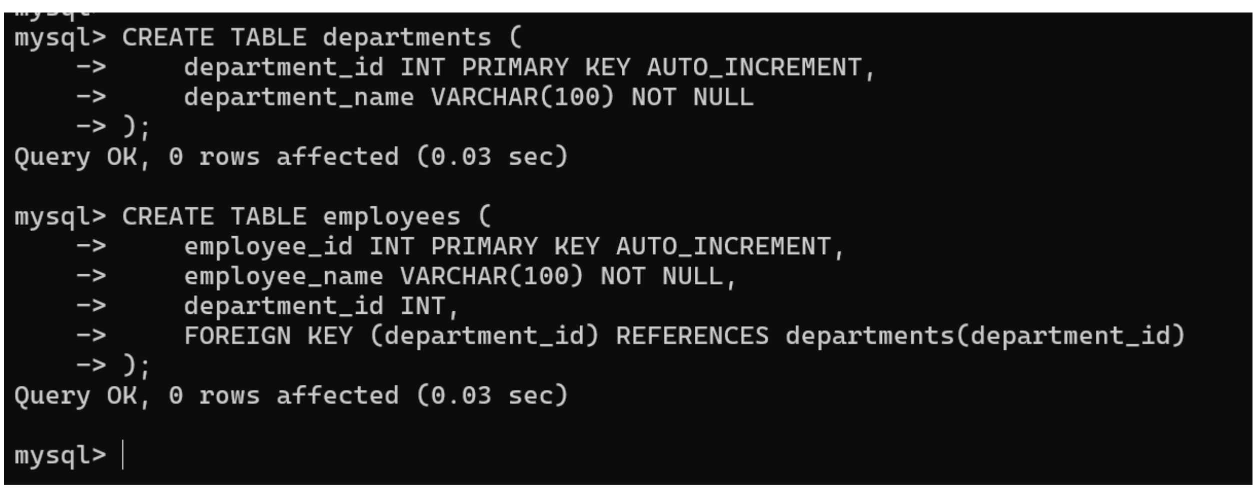
```
mysql> SELECT * FROM courses;
+-----------+------------------+-----------------+
| course_id | course_name      | course_duration |
+-----------+------------------+-----------------+
|         1 | Mathematics      | 6 months        |
|         3 | Computer Science | 8 months        |
|         4 | Biology          | 6 months        |
|         5 | Chemistry        | 1 year          |
|         6 | English          | 8 months        |
+-----------+------------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

23. Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

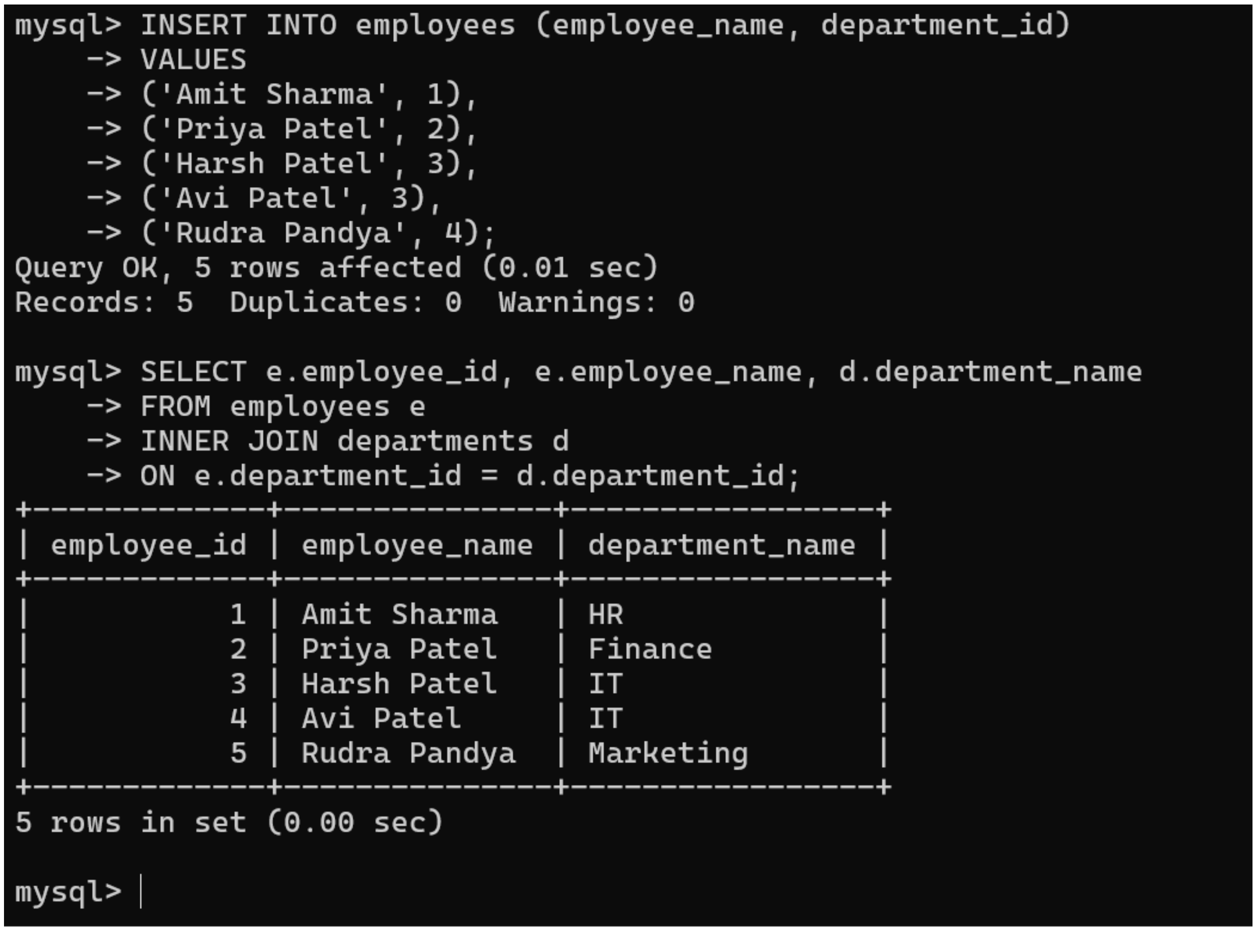


```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE courses
    -> SET course_duration = '12 months'
    -> WHERE course_name = 'Mathematics';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> UPDATE courses
    -> SET course_duration = '18 months'
    -> WHERE course_name = 'Physics';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> SAVEPOINT before_physics_update;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> -- Update another course
mysql> UPDATE courses
    -> SET course_duration = '10 months'
    -> WHERE course_name = 'Computer Science';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> ROLLBACK TO SAVEPOINT before_physics_update;
Query OK, 0 rows affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

24. Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

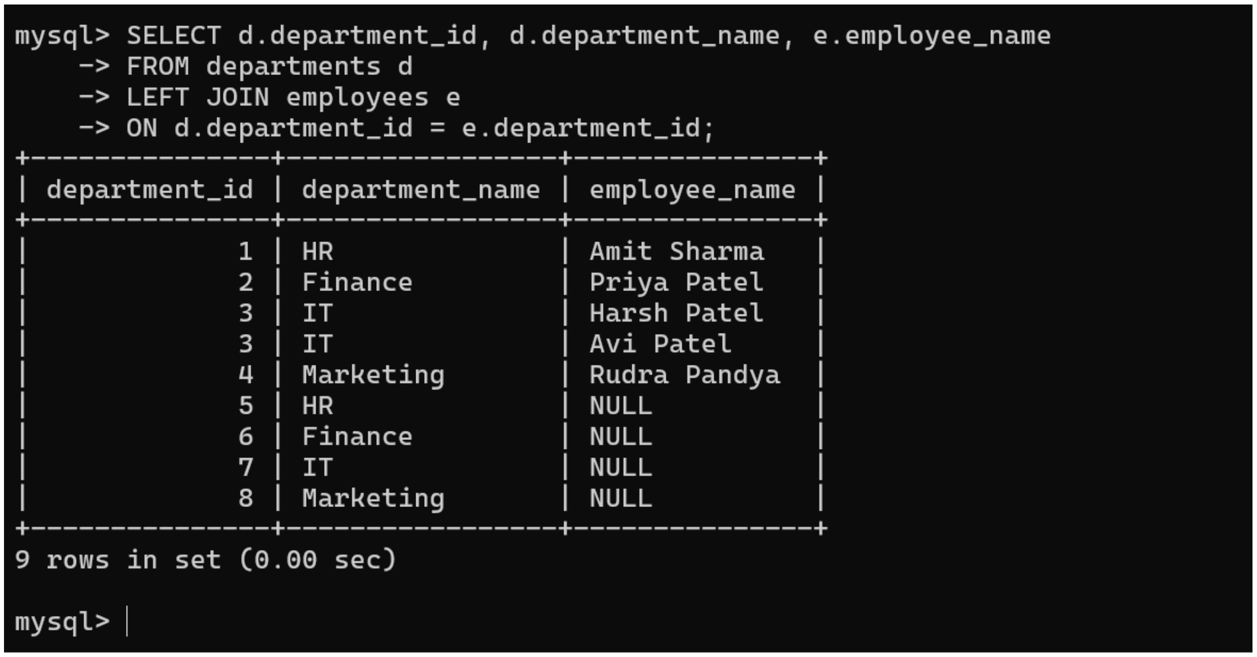


```
mysql> CREATE TABLE departments (
    ->     department_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     department_name VARCHAR(100) NOT NULL
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE employees (
    ->     employee_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     employee_name VARCHAR(100) NOT NULL,
    ->     department_id INT,
    ->     FOREIGN KEY (department_id) REFERENCES departments(department_id)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
```



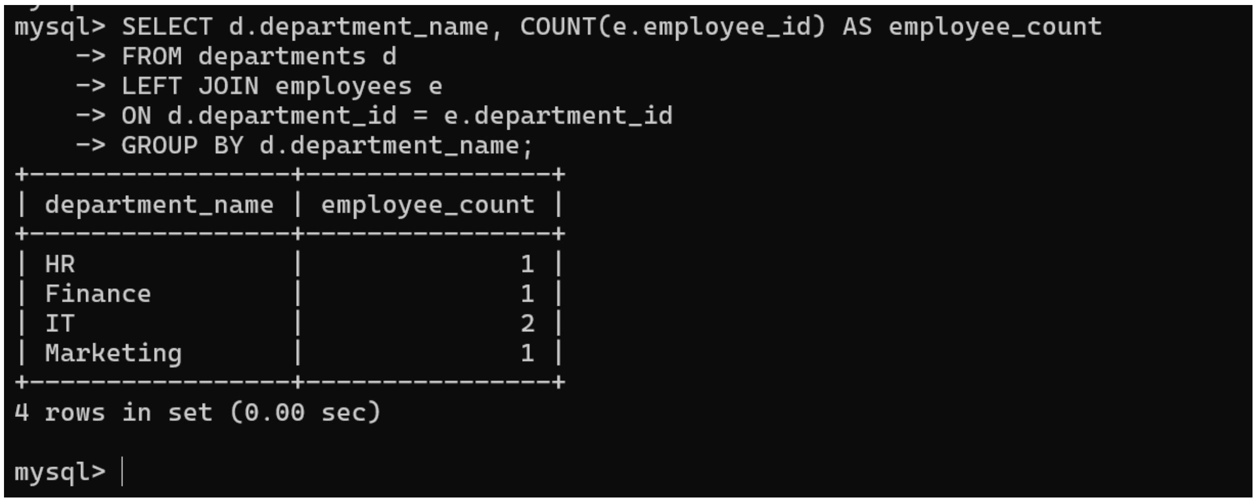
```
mysql> INSERT INTO employees (employee_name, department_id)
    -> VALUES
    -> ('Amit Sharma', 1),
    -> ('Priya Patel', 2),
    -> ('Harsh Patel', 3),
    -> ('Avi Patel', 3),
    -> ('Rudra Pandya', 4);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT e.employee_id, e.employee_name, d.department_name
    -> FROM employees e
    -> INNER JOIN departments d
    -> ON e.department_id = d.department_id;
+-------------+---------------+-----------------+
| employee_id | employee_name | department_name |
+-------------+---------------+-----------------+
|           1 | Amit Sharma   | HR              |
|           2 | Priya Patel   | Finance         |
|           3 | Harsh Patel   | IT              |
|           4 | Avi Patel     | IT              |
|           5 | Rudra Pandya  | Marketing       |
+-------------+---------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

25. Use a LEFT JOIN to show all departments, even those without employees.



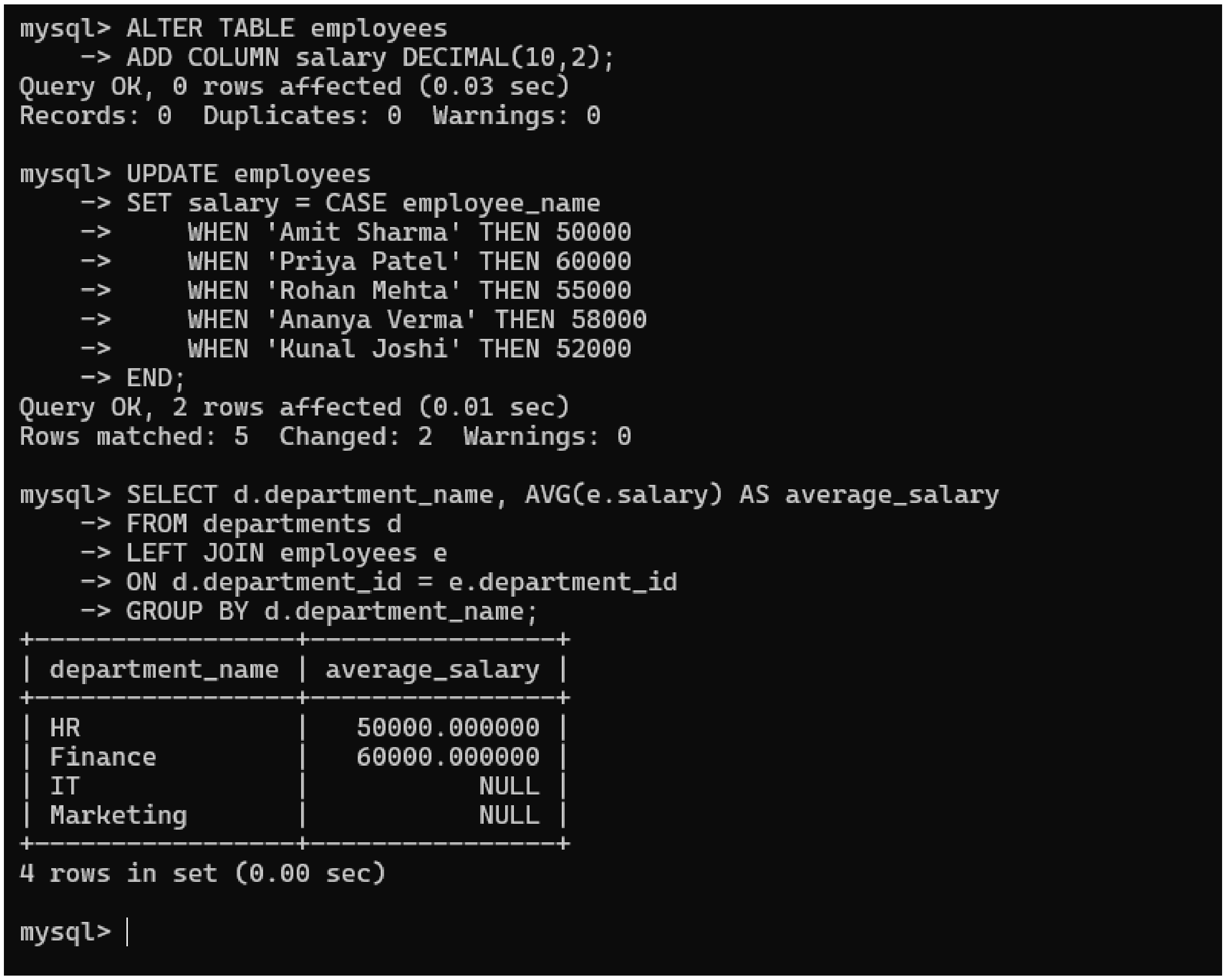
```
mysql> SELECT d.department_id, d.department_name, e.employee_name
    -> FROM departments d
    -> LEFT JOIN employees e
    -> ON d.department_id = e.department_id;
+---------------+-----------------+---------------+
| department_id | department_name | employee_name |
+---------------+-----------------+---------------+
|             1 | HR              | Amit Sharma   |
|             2 | Finance         | Priya Patel   |
|             3 | IT              | Harsh Patel   |
|             3 | IT              | Avi Patel     |
|             4 | Marketing       | Rudra Pandya  |
|             5 | HR              | NULL          |
|             6 | Finance         | NULL          |
|             7 | IT              | NULL          |
|             8 | Marketing       | NULL          |
+---------------+-----------------+---------------+
9 rows in set (0.00 sec)

mysql>
```

26. Group employees by department and count the number of employees in each department using GROUP BY.



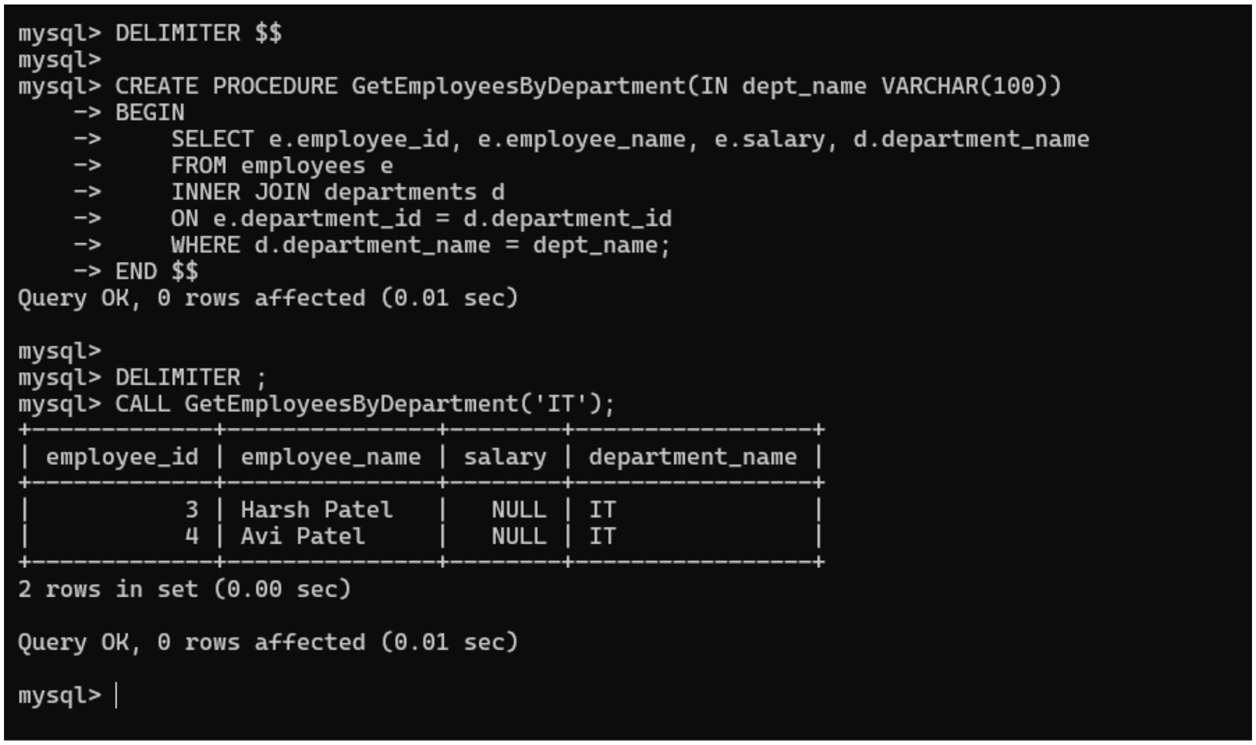
```
mysql> SELECT d.department_name, COUNT(e.employee_id) AS employee_count
    -> FROM departments d
    -> LEFT JOIN employees e
    -> ON d.department_id = e.department_id
    -> GROUP BY d.department_name;
+-----------------+----------------+
| department_name | employee_count |
+-----------------+----------------+
| HR              |              1 |
| Finance         |              1 |
| IT              |              2 |
| Marketing       |              1 |
+-----------------+----------------+
4 rows in set (0.00 sec)

mysql>
```

27. Use the AVG aggregate function to find the average salary of employees in each department.



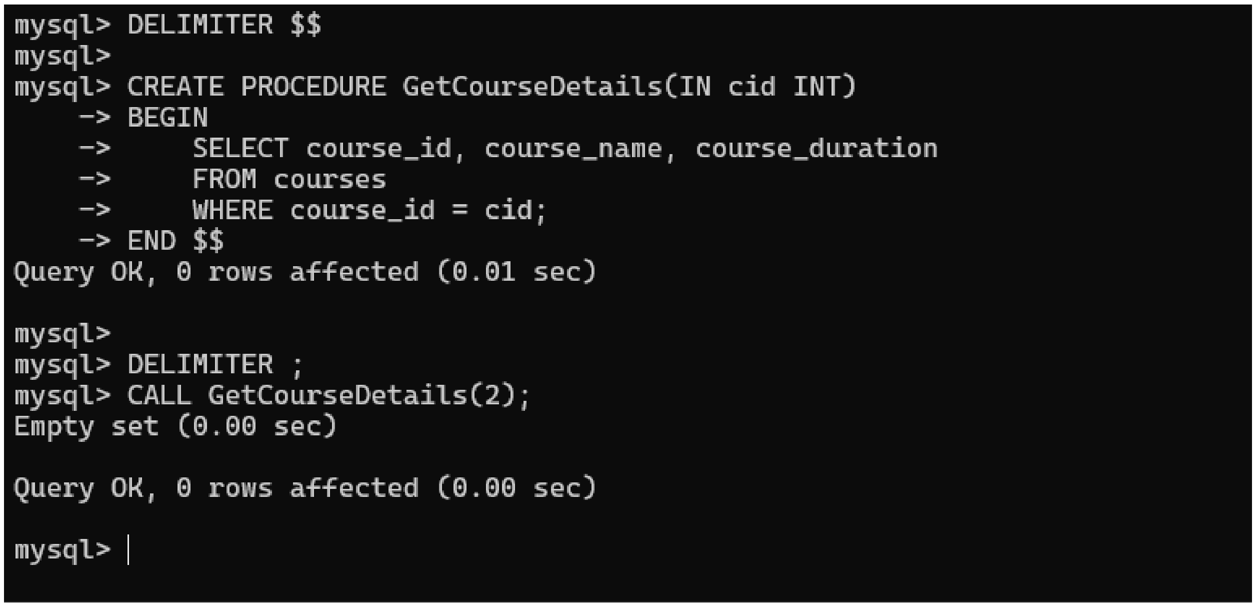
```
mysql> ALTER TABLE employees
    -> ADD COLUMN salary DECIMAL(10,2);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> UPDATE employees
    -> SET salary = CASE employee_name
    ->     WHEN 'Amit Sharma' THEN 50000
    ->     WHEN 'Priya Patel' THEN 60000
    ->     WHEN 'Rohan Mehta' THEN 55000
    ->     WHEN 'Ananya Verma' THEN 58000
    ->     WHEN 'Kunal Joshi' THEN 52000
    -> END;
Query OK, 2 rows affected (0.01 sec)
Rows matched: 5  Changed: 2  Warnings: 0

mysql> SELECT d.department_name, AVG(e.salary) AS average_salary
    -> FROM departments d
    -> LEFT JOIN employees e
    -> ON d.department_id = e.department_id
    -> GROUP BY d.department_name;
+-----------------+----------------+
| department_name | average_salary |
+-----------------+----------------+
| HR              |   50000.000000 |
| Finance         |   60000.000000 |
| IT              |           NULL |
| Marketing       |           NULL |
+-----------------+----------------+
4 rows in set (0.00 sec)

mysql> |
```

28. Write a stored procedure to retrieve all employees from the employees table based on department.



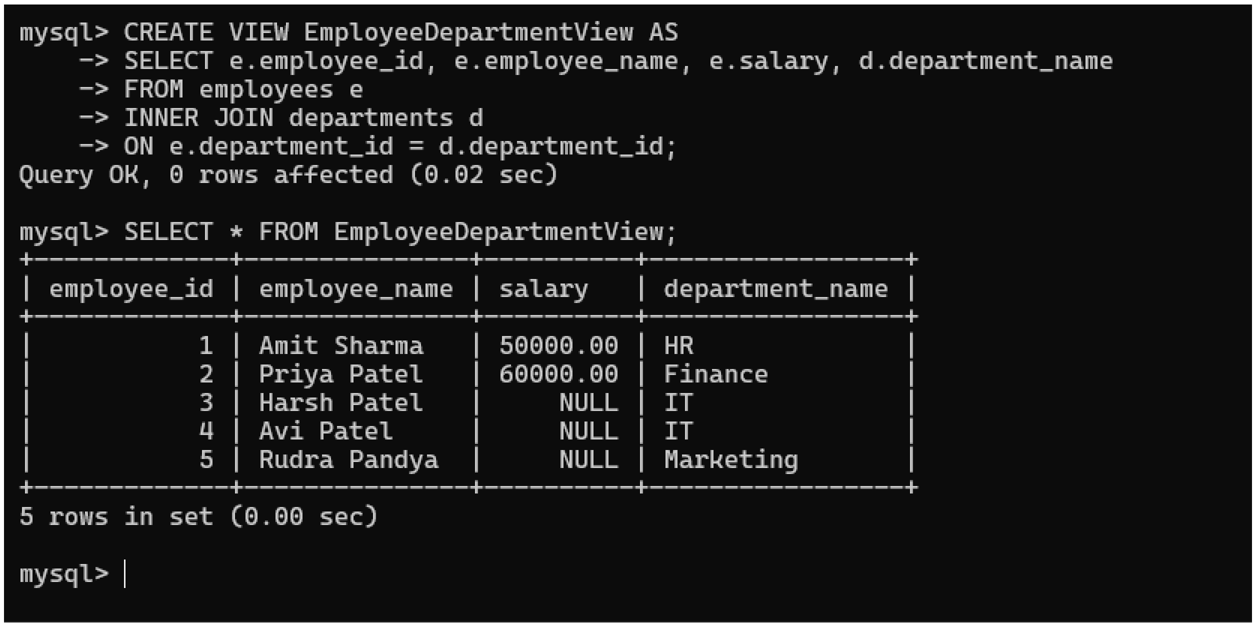
```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE GetEmployeesByDepartment(IN dept_name VARCHAR(100))
    -> BEGIN
    ->     SELECT e.employee_id, e.employee_name, e.salary, d.department_name
    ->     FROM employees e
    ->     INNER JOIN departments d
    ->     ON e.department_id = d.department_id
    ->     WHERE d.department_name = dept_name;
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetEmployeesByDepartment('IT');
+-------------+---------------+--------+-----------------+
| employee_id | employee_name | salary | department_name |
+-------------+---------------+--------+-----------------+
|           3 | Harsh Patel   |   NULL | IT              |
|           4 | Avi Patel     |   NULL | IT              |
+-------------+---------------+--------+-----------------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> |
```

29. Write a stored procedure that accepts course_id as input and returns the course details



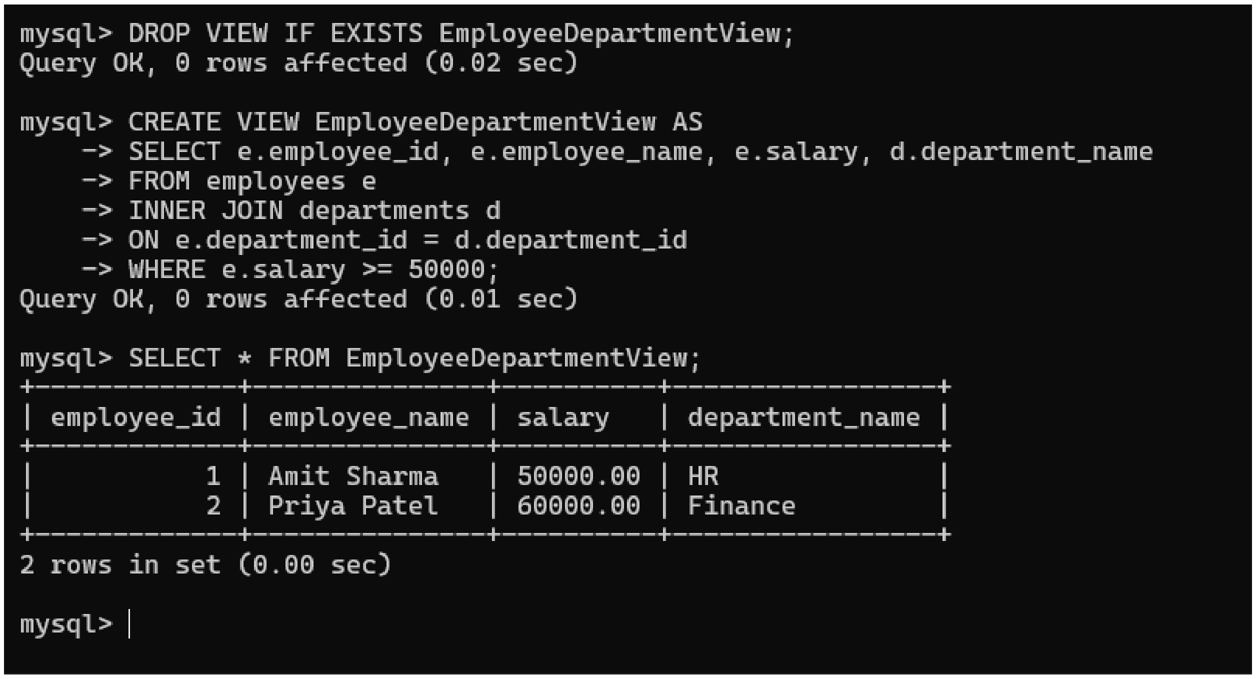
```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE GetCourseDetails(IN cid INT)
    -> BEGIN
    ->     SELECT course_id, course_name, course_duration
    ->     FROM courses
    ->     WHERE course_id = cid;
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetCourseDetails(2);
Empty set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> |
```

30. Create a view to show all employees along with their department names.



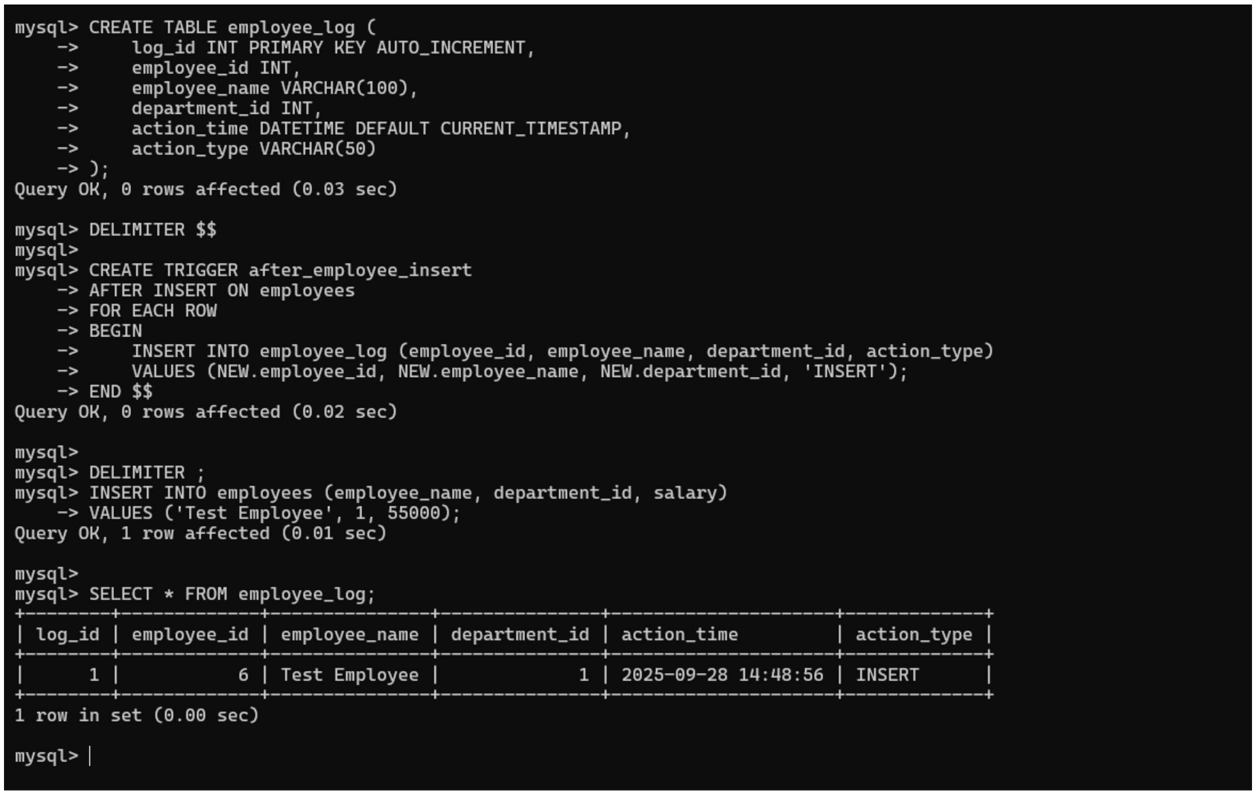
```
mysql> CREATE VIEW EmployeeDepartmentView AS
    -> SELECT e.employee_id, e.employee_name, e.salary, d.department_name
    -> FROM employees e
    -> INNER JOIN departments d
    -> ON e.department_id = d.department_id;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM EmployeeDepartmentView;
+-------------+---------------+----------+-----------------+
| employee_id | employee_name | salary   | department_name |
+-------------+---------------+----------+-----------------+
|           1 | Amit Sharma   | 50000.00 | HR              |
|           2 | Priya Patel   | 60000.00 | Finance         |
|           3 | Harsh Patel   |     NULL | IT              |
|           4 | Avi Patel     |     NULL | IT              |
|           5 | Rudra Pandya  |     NULL | Marketing       |
+-------------+---------------+----------+-----------------+
5 rows in set (0.00 sec)

mysql> |
```

31. Modify the view to exclude employees whose salaries are below $50,000.



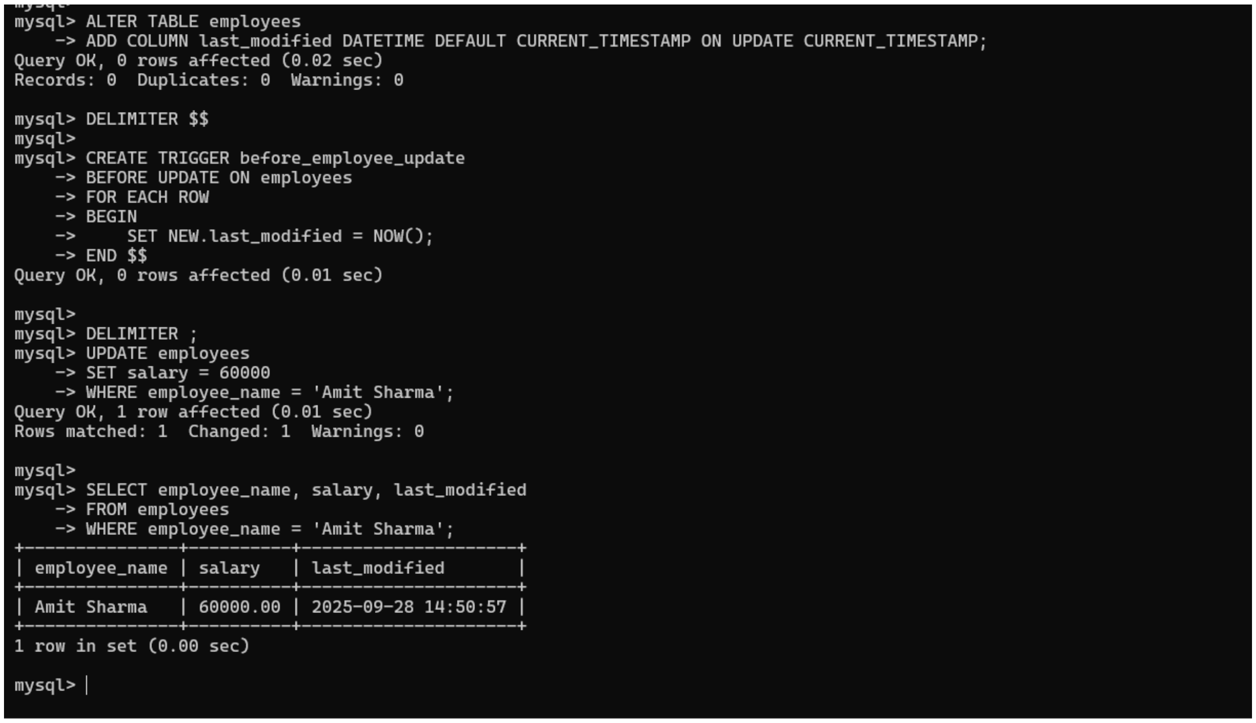
```
mysql> DROP VIEW IF EXISTS EmployeeDepartmentView;
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE VIEW EmployeeDepartmentView AS
    -> SELECT e.employee_id, e.employee_name, e.salary, d.department_name
    -> FROM employees e
    -> INNER JOIN departments d
    -> ON e.department_id = d.department_id
    -> WHERE e.salary >= 50000;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM EmployeeDepartmentView;
+-------------+---------------+----------+-----------------+
| employee_id | employee_name | salary   | department_name |
+-------------+---------------+----------+-----------------+
|           1 | Amit Sharma   | 50000.00 | HR              |
|           2 | Priya Patel   | 60000.00 | Finance         |
+-------------+---------------+----------+-----------------+
2 rows in set (0.00 sec)

mysql> |
```

32. Create a trigger to automatically log changes to the employees table when a new employee is added.



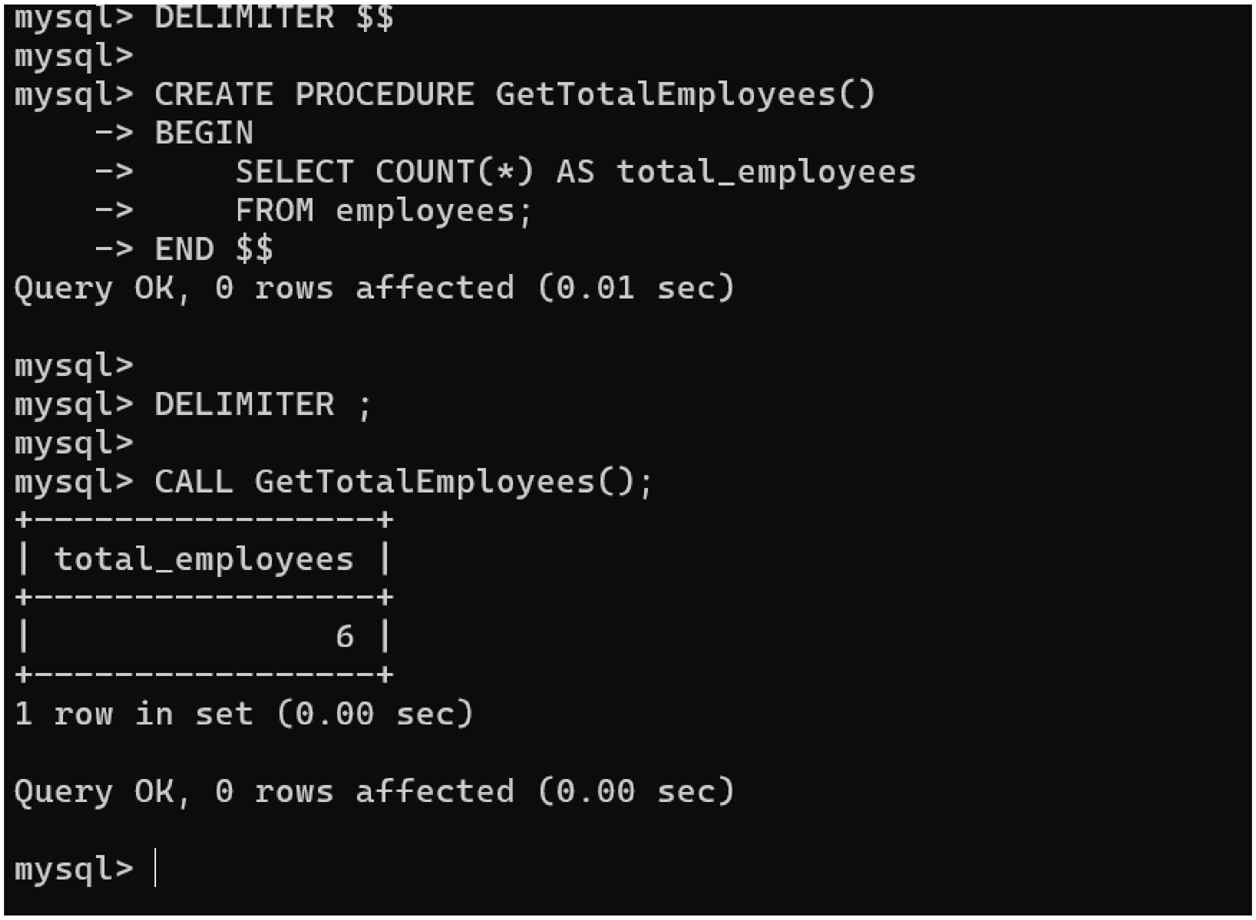
```
mysql> CREATE TABLE employee_log (
    ->     log_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     employee_id INT,
    ->     employee_name VARCHAR(100),
    ->     department_id INT,
    ->     action_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     action_type VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER after_employee_insert
    -> AFTER INSERT ON employees
    -> FOR EACH ROW
    -> BEGIN
    ->     INSERT INTO employee_log (employee_id, employee_name, department_id, action_type)
    ->     VALUES (NEW.employee_id, NEW.employee_name, NEW.department_id, 'INSERT');
    -> END $$
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> INSERT INTO employees (employee_name, department_id, salary)
    -> VALUES ('Test Employee', 1, 55000);
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> SELECT * FROM employee_log;
+--------+-------------+---------------+---------------+---------------------+-------------+
| log_id | employee_id | employee_name | department_id | action_time         | action_type |
+--------+-------------+---------------+---------------+---------------------+-------------+
|      1 |           6 | Test Employee |             1 | 2025-09-28 14:48:56 | INSERT      |
+--------+-------------+---------------+---------------+---------------------+-------------+
1 row in set (0.00 sec)

mysql> |
```

33. Create a trigger to update the last_modified timestamp whenever an employee record is updated.

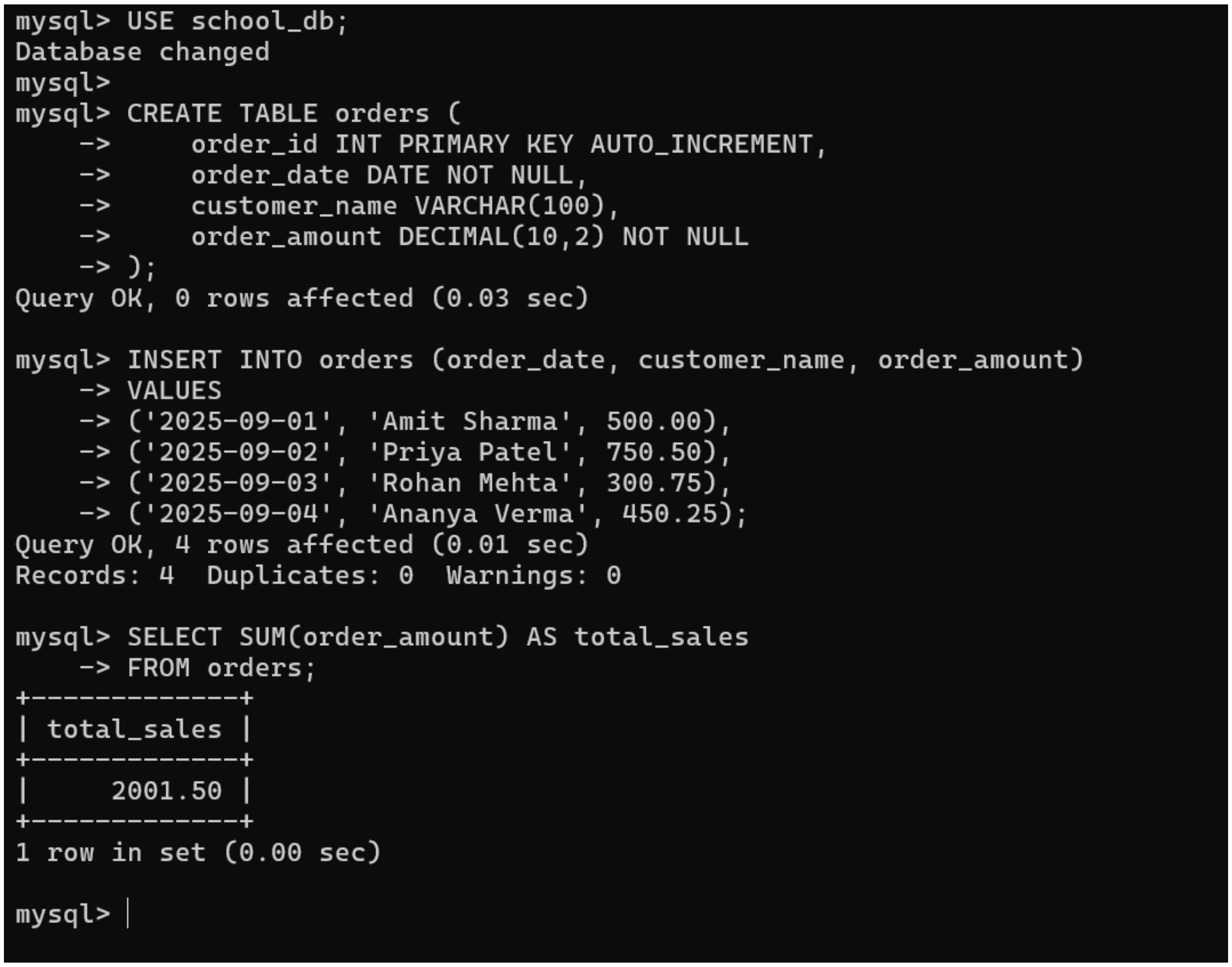


```
mysql> ALTER TABLE employees
    -> ADD COLUMN last_modified DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER before_employee_update
    -> BEFORE UPDATE ON employees
    -> FOR EACH ROW
    -> BEGIN
    ->     SET NEW.last_modified = NOW();
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> UPDATE employees
    -> SET salary = 60000
    -> WHERE employee_name = 'Amit Sharma';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> SELECT employee_name, salary, last_modified
    -> FROM employees
    -> WHERE employee_name = 'Amit Sharma';
+---------------+----------+---------------------+
| employee_name | salary   | last_modified       |
+---------------+----------+---------------------+
| Amit Sharma   | 60000.00 | 2025-09-28 14:50:57 |
+---------------+----------+---------------------+
1 row in set (0.00 sec)

mysql> |
```

34. Write a PL/SQL block to print the total number of employees from the employees table.

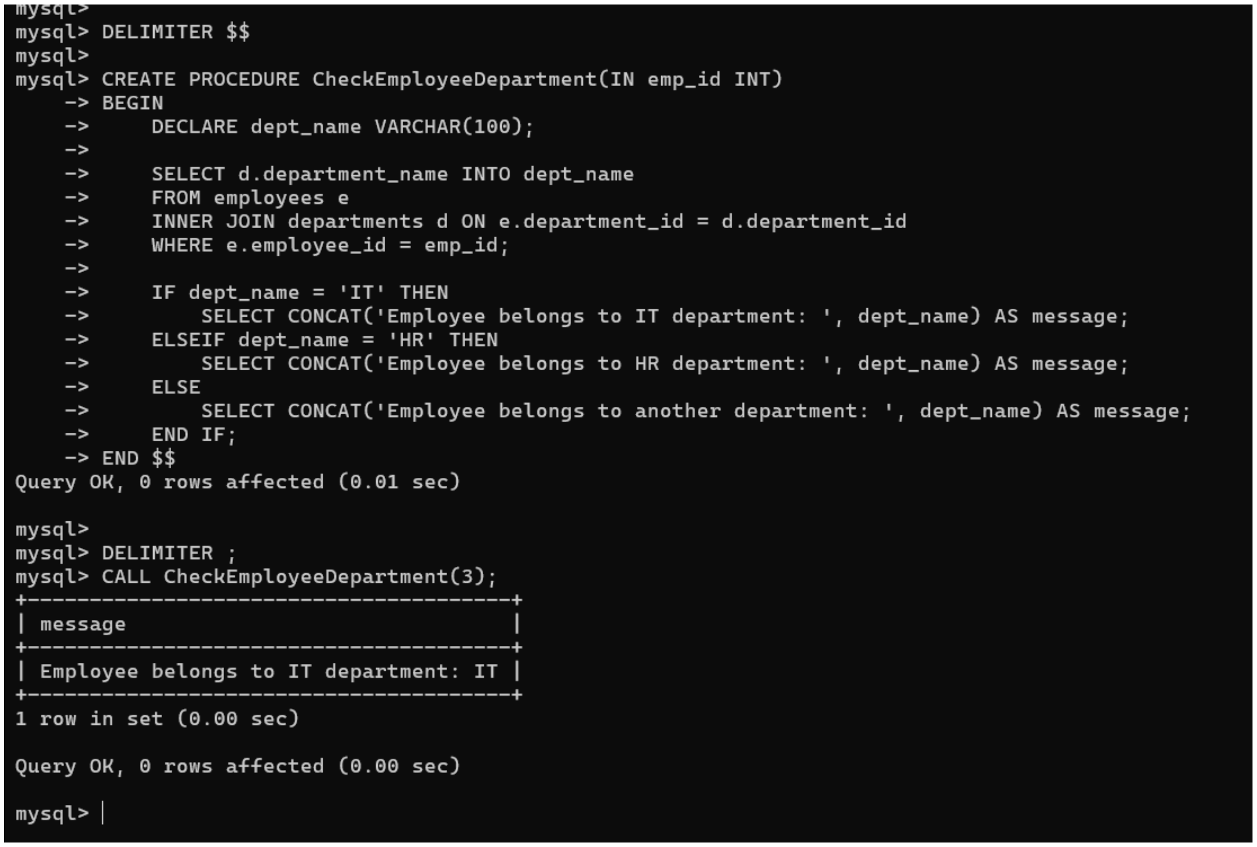

```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE GetTotalEmployees()
    -> BEGIN
    ->     SELECT COUNT(*) AS total_employees
    ->     FROM employees;
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> CALL GetTotalEmployees();
+-----------------+
| total_employees |
+-----------------+
|               6 |
+-----------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

35. Create a PL/SQL block that calculates the total sales from an orders table.



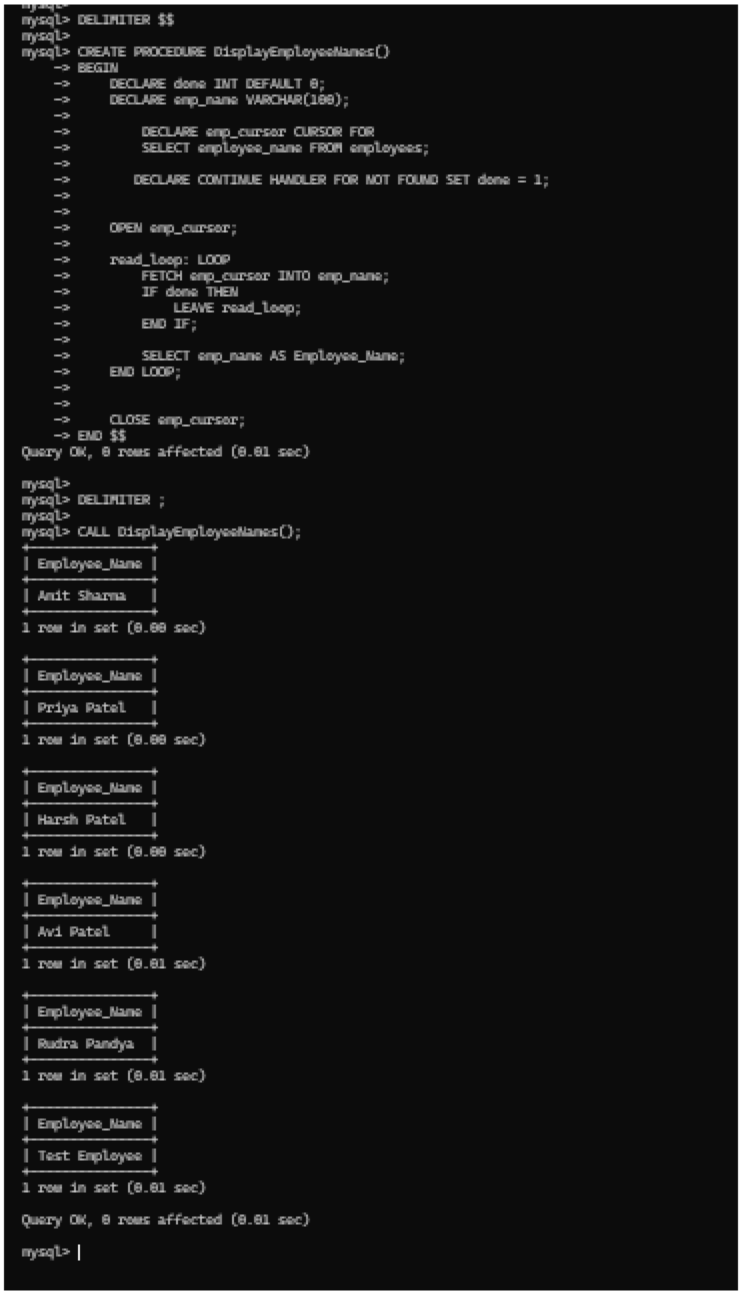
```
mysql> USE school_db;
Database changed
mysql>
mysql> CREATE TABLE orders (
    ->     order_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     order_date DATE NOT NULL,
    ->     customer_name VARCHAR(100),
    ->     order_amount DECIMAL(10,2) NOT NULL
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO orders (order_date, customer_name, order_amount)
    -> VALUES
    -> ('2025-09-01', 'Amit Sharma', 500.00),
    -> ('2025-09-02', 'Priya Patel', 750.50),
    -> ('2025-09-03', 'Rohan Mehta', 300.75),
    -> ('2025-09-04', 'Ananya Verma', 450.25);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT SUM(order_amount) AS total_sales
    -> FROM orders;
+-------------+
| total_sales |
+-------------+
|     2001.50 |
+-------------+
1 row in set (0.00 sec)

mysql> 
```

36. Write a PL/SQL block using an IF-THEN condition to check the department of an employee.



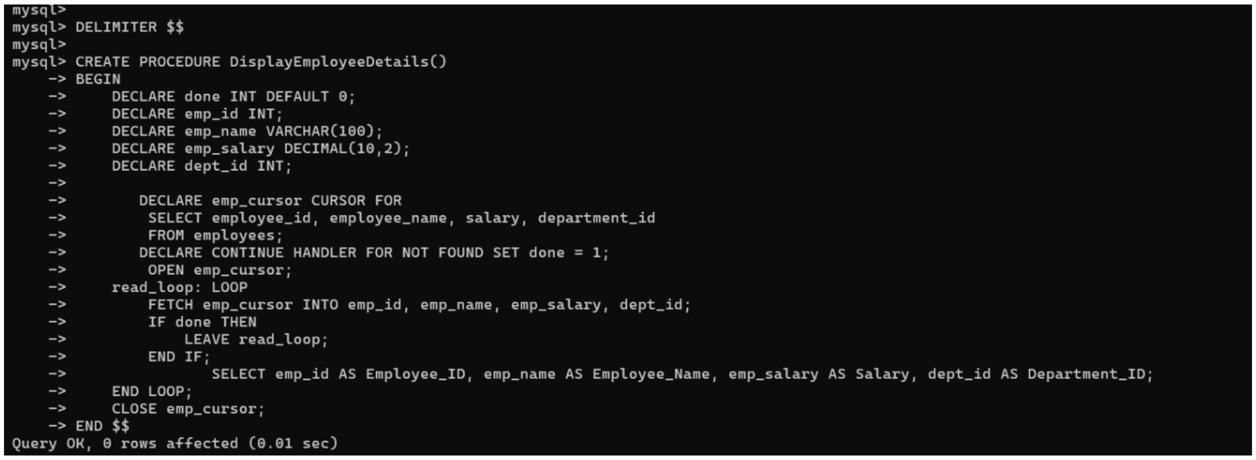
```
mysql>
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE CheckEmployeeDepartment(IN emp_id INT)
    -> BEGIN
    ->     DECLARE dept_name VARCHAR(100);
    ->
    ->     SELECT d.department_name INTO dept_name
    ->     FROM employees e
    ->     INNER JOIN departments d ON e.department_id = d.department_id
    ->     WHERE e.employee_id = emp_id;
    ->
    ->     IF dept_name = 'IT' THEN
    ->         SELECT CONCAT('Employee belongs to IT department: ', dept_name) AS message;
    ->     ELSEIF dept_name = 'HR' THEN
    ->         SELECT CONCAT('Employee belongs to HR department: ', dept_name) AS message;
    ->     ELSE
    ->         SELECT CONCAT('Employee belongs to another department: ', dept_name) AS message;
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL CheckEmployeeDepartment(3);
+----------------------------------------+
| message                                |
+----------------------------------------+
| Employee belongs to IT department: IT |
+----------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> |
```

37. Use a FOR LOOP to iterate through employee records and display their names.



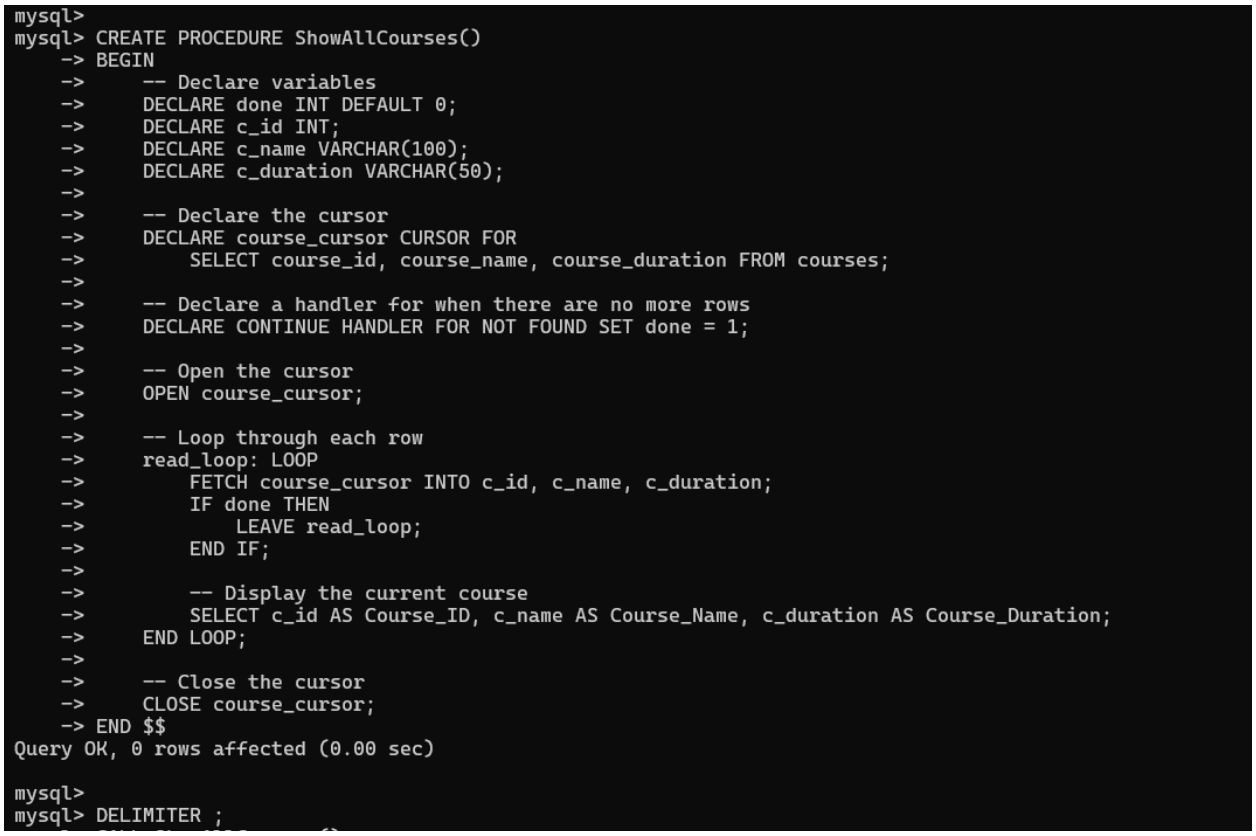
```
mysql>
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE DisplayEmployeeNames()
    -> BEGIN
    ->     DECLARE done INT DEFAULT 0;
    ->     DECLARE emp_name VARCHAR(100);
    ->
    ->         DECLARE emp_cursor CURSOR FOR
    ->         SELECT employee_name FROM employees;
    ->
    ->        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->
    ->
    ->     OPEN emp_cursor;
    ->
    ->     read_loop: LOOP
    ->         FETCH emp_cursor INTO emp_name;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->
    ->         SELECT emp_name AS Employee_Name;
    ->     END LOOP;
    ->
    ->
    ->     CLOSE emp_cursor;
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> CALL DisplayEmployeeNames();
+---------------+
| Employee_Name |
+---------------+
| Amit Sharma   |
+---------------+
1 row in set (0.00 sec)

+---------------+
| Employee_Name |
+---------------+
| Priya Patel   |
+---------------+
1 row in set (0.00 sec)

+---------------+
| Employee_Name |
+---------------+
| Harsh Patel   |
+---------------+
1 row in set (0.00 sec)

+---------------+
| Employee_Name |
+---------------+
| Avi Patel     |
+---------------+
1 row in set (0.01 sec)

+---------------+
| Employee_Name |
+---------------+
| Rudra Pandya  |
+---------------+
1 row in set (0.01 sec)

+---------------+
| Employee_Name |
+---------------+
| Test Employee |
+---------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> |
```

38. Write a PL/SQL block using an explicit cursor to retrieve and display employee details.

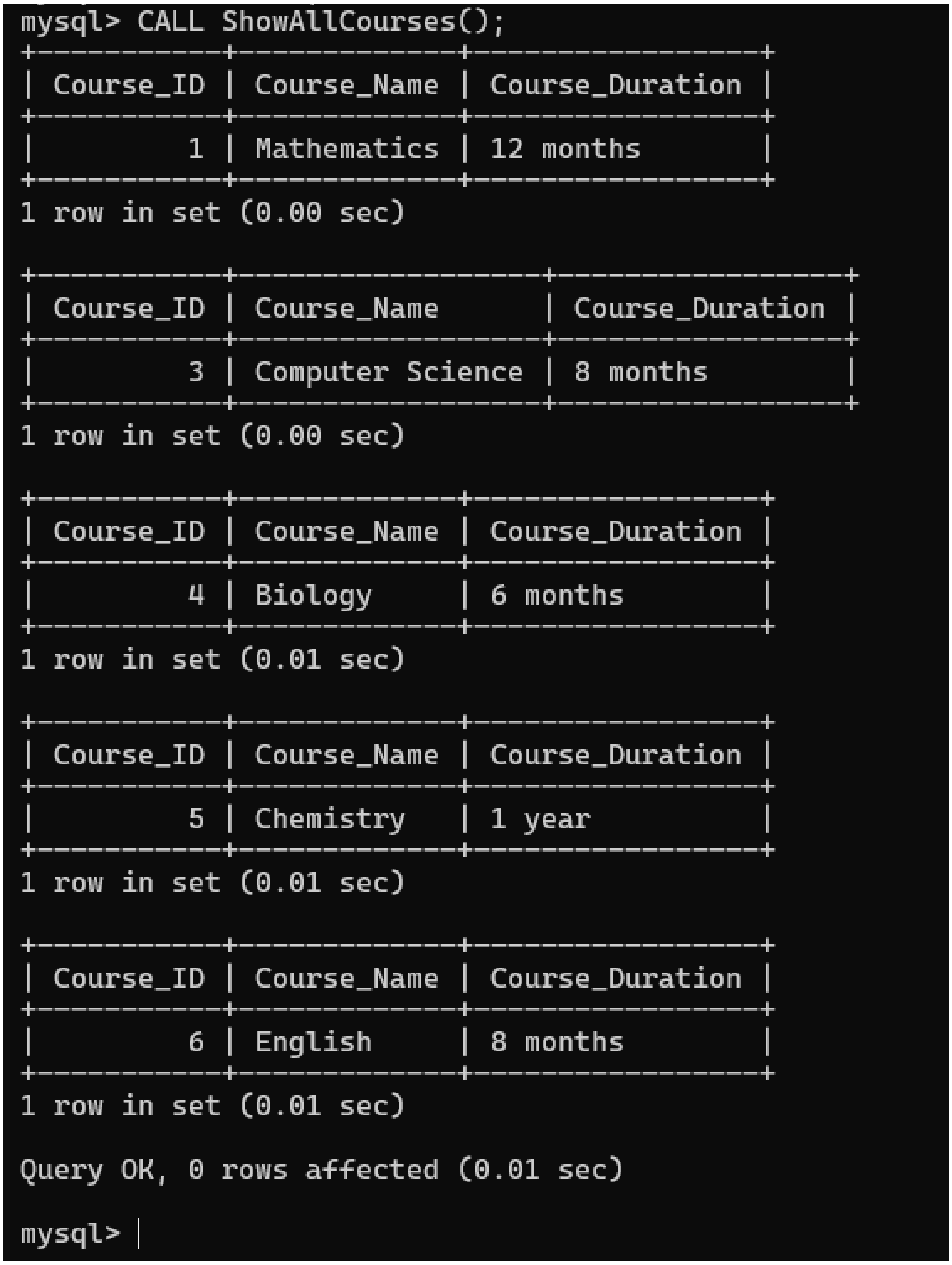


```
mysql>
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE DisplayEmployeeDetails()
    -> BEGIN
    ->     DECLARE done INT DEFAULT 0;
    ->     DECLARE emp_id INT;
    ->     DECLARE emp_name VARCHAR(100);
    ->     DECLARE emp_salary DECIMAL(10,2);
    ->     DECLARE dept_id INT;
    ->
    ->        DECLARE emp_cursor CURSOR FOR
    ->         SELECT employee_id, employee_name, salary, department_id
    ->         FROM employees;
    ->        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->         OPEN emp_cursor;
    ->     read_loop: LOOP
    ->         FETCH emp_cursor INTO emp_id, emp_name, emp_salary, dept_id;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->                SELECT emp_id AS Employee_ID, emp_name AS Employee_Name, emp_salary AS Salary, dept_id AS Department_ID;
    ->     END LOOP;
    ->     CLOSE emp_cursor;
    -> END $$
Query OK, 0 rows affected (0.01 sec)
```

39. Create a cursor to retrieve all courses and display them one by one.



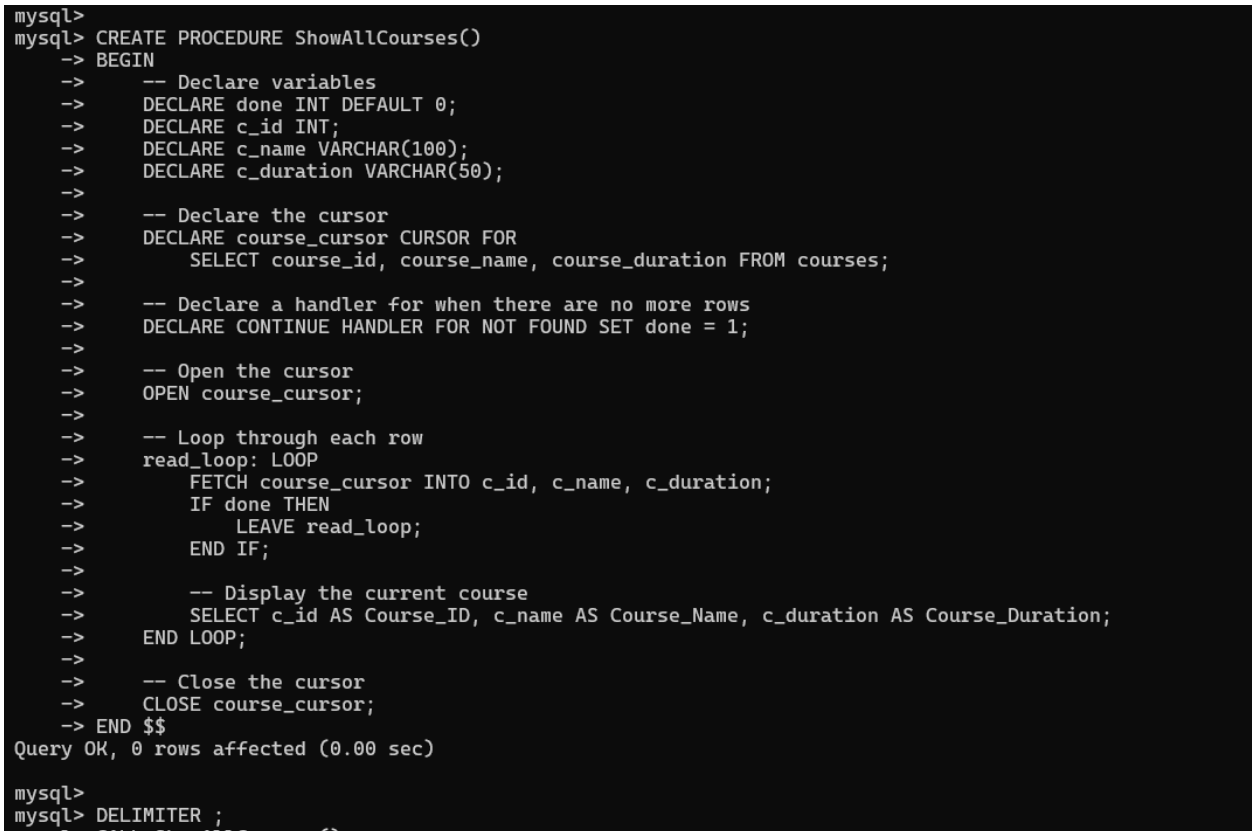
```
mysql>
mysql> CREATE PROCEDURE ShowAllCourses()
    -> BEGIN
    ->     -- Declare variables
    ->     DECLARE done INT DEFAULT 0;
    ->     DECLARE c_id INT;
    ->     DECLARE c_name VARCHAR(100);
    ->     DECLARE c_duration VARCHAR(50);
    ->
    ->     -- Declare the cursor
    ->     DECLARE course_cursor CURSOR FOR
    ->         SELECT course_id, course_name, course_duration FROM courses;
    ->
    ->     -- Declare a handler for when there are no more rows
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->
    ->     -- Open the cursor
    ->     OPEN course_cursor;
    ->
    ->     -- Loop through each row
    ->     read_loop: LOOP
    ->         FETCH course_cursor INTO c_id, c_name, c_duration;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->
    ->         -- Display the current course
    ->         SELECT c_id AS Course_ID, c_name AS Course_Name, c_duration AS Course_Duration;
    ->     END LOOP;
    ->
    ->     -- Close the cursor
    ->     CLOSE course_cursor;
    -> END $$
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> DELIMITER ;
```



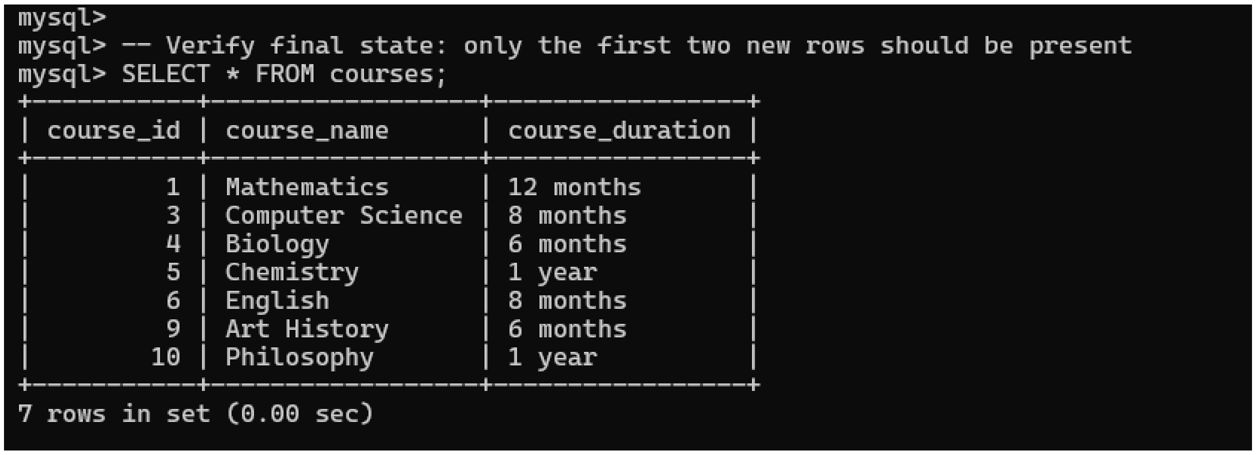
```
mysql> CALL ShowAllCourses();
+-----------+-------------+-----------------+
| Course_ID | Course_Name | Course_Duration |
+-----------+-------------+-----------------+
|         1 | Mathematics | 12 months       |
+-----------+-------------+-----------------+
1 row in set (0.00 sec)

+-----------+------------------+-----------------+
| Course_ID | Course_Name      | Course_Duration |
+-----------+------------------+-----------------+
|         3 | Computer Science | 8 months        |
+-----------+------------------+-----------------+
1 row in set (0.00 sec)

+-----------+-------------+-----------------+
| Course_ID | Course_Name | Course_Duration |
+-----------+-------------+-----------------+
|         4 | Biology     | 6 months        |
+-----------+-------------+-----------------+
1 row in set (0.01 sec)

+-----------+-------------+-----------------+
| Course_ID | Course_Name | Course_Duration |
+-----------+-------------+-----------------+
|         5 | Chemistry   | 1 year          |
+-----------+-------------+-----------------+
1 row in set (0.01 sec)

+-----------+-------------+-----------------+
| Course_ID | Course_Name | Course_Duration |
+-----------+-------------+-----------------+
|         6 | English     | 8 months        |
+-----------+-------------+-----------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> 
```

40. Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.

Ans. =



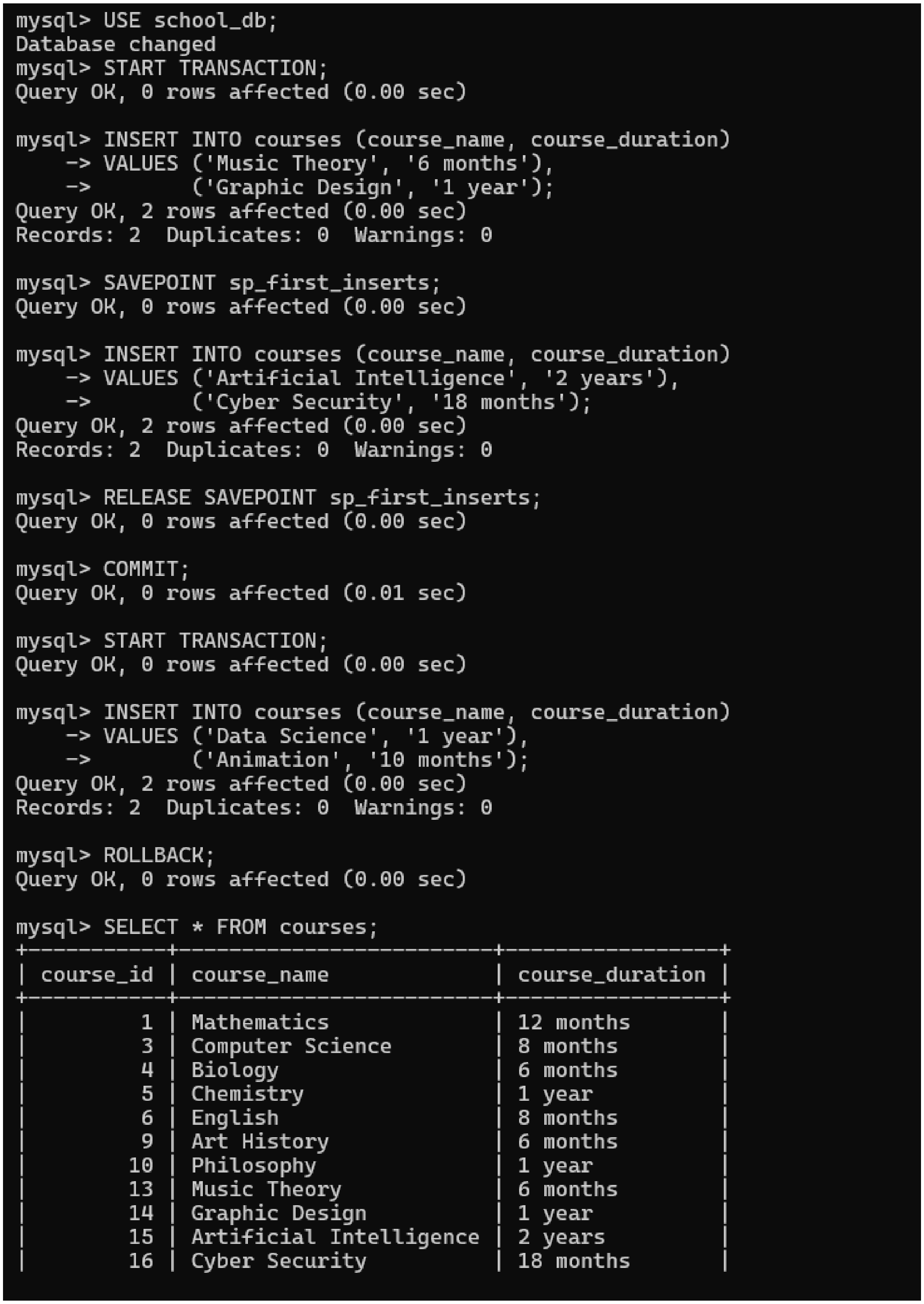
```
mysql>
mysql> CREATE PROCEDURE ShowAllCourses()
    -> BEGIN
    ->     -- Declare variables
    ->     DECLARE done INT DEFAULT 0;
    ->     DECLARE c_id INT;
    ->     DECLARE c_name VARCHAR(100);
    ->     DECLARE c_duration VARCHAR(50);
    ->
    ->     -- Declare the cursor
    ->     DECLARE course_cursor CURSOR FOR
    ->         SELECT course_id, course_name, course_duration FROM courses;
    ->
    ->     -- Declare a handler for when there are no more rows
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->
    ->     -- Open the cursor
    ->     OPEN course_cursor;
    ->
    ->     -- Loop through each row
    ->     read_loop: LOOP
    ->         FETCH course_cursor INTO c_id, c_name, c_duration;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->
    ->         -- Display the current course
    ->         SELECT c_id AS Course_ID, c_name AS Course_Name, c_duration AS Course_Duration;
    ->     END LOOP;
    ->
    ->     -- Close the cursor
    ->     CLOSE course_cursor;
    -> END $$
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> DELIMITER ;
```



```
mysql>
mysql> -- Verify final state: only the first two new rows should be present
mysql> SELECT * FROM courses;
+-----------+-------------------+-----------------+
| course_id | course_name       | course_duration |
+-----------+-------------------+-----------------+
|         1 | Mathematics       | 12 months       |
|         3 | Computer Science  | 8 months        |
|         4 | Biology           | 6 months        |
|         5 | Chemistry         | 1 year          |
|         6 | English           | 8 months        |
|         9 | Art History       | 6 months        |
|        10 | Philosophy        | 1 year          |
+-----------+-------------------+-----------------+
7 rows in set (0.00 sec)
```

41. Commit part of a transaction after using a savepoint and then rollback the remaining changes

Ans. =



```
mysql> USE school_db;
Database changed
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO courses (course_name, course_duration)
    -> VALUES ('Music Theory', '6 months'),
    ->        ('Graphic Design', '1 year');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SAVEPOINT sp_first_inserts;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO courses (course_name, course_duration)
    -> VALUES ('Artificial Intelligence', '2 years'),
    ->        ('Cyber Security', '18 months');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> RELEASE SAVEPOINT sp_first_inserts;
Query OK, 0 rows affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO courses (course_name, course_duration)
    -> VALUES ('Data Science', '1 year'),
    ->        ('Animation', '10 months');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM courses;
+-----------+-------------------------+-----------------+
| course_id | course_name             | course_duration |
+-----------+-------------------------+-----------------+
|         1 | Mathematics             | 12 months       |
|         3 | Computer Science        | 8 months        |
|         4 | Biology                 | 6 months        |
|         5 | Chemistry               | 1 year          |
|         6 | English                 | 8 months        |
|         9 | Art History             | 6 months        |
|        10 | Philosophy              | 1 year          |
|        13 | Music Theory            | 6 months        |
|        14 | Graphic Design          | 1 year          |
|        15 | Artificial Intelligence | 2 years         |
|        16 | Cyber Security          | 18 months       |
```