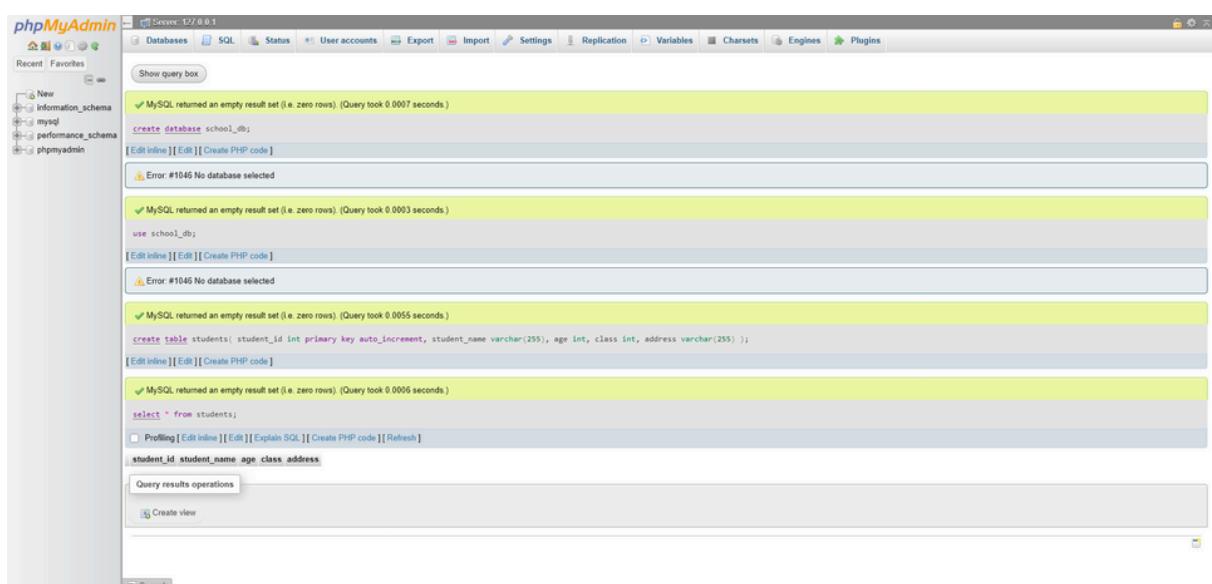


Module 4 – Introduction to DBMS (Lab Exercises)

1 Introduction to SQL

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

```
1 create database school_db;
2 use school_db;
3
4 create table students(
5     student_id int primary key auto_increment,
6     student_name varchar(255),
7     age int,
8     class int,
9     address varchar(255)
10);
11
12 select *
13 from students;
```



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

```

1 use school_db;
2
3 insert into students (student_name,age, class, address) values
4     ("student_1", 6, 1, "address of student_1"),
5     ("student_2", 11, 6, "address of student_2"),
6     ("student_3", 13, 8, "address of student_3"),
7     ("student_4", 16, 11, "address of student_4"),
8     ("student_5", 8, 3, "address of student_5");
9
10 select *
11 from students;
```

The screenshot shows the phpMyAdmin interface with the following details:

- Servers:** 1/2 0.0.1
- Databases:** Information_schema, mysql, performance_schema, school_db
- Query Results:**
 - MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)
 - use school_db;
 - Error: #1045 No database selected
 - 5 rows inserted. Inserted row id: 5 (Query took 0.0032 seconds.)
 - insert into students (student_name, age, class, address) values ('student_1', 6, 1, 'address of student_1'), ('student_2', 11, 6, 'address of student_2'), ('student_3', 13, 8, 'address of student_3'), ('student_4', 16, 11, 'address of student_4'), ('student_5', 8, 3, 'address of student_5');
 - Showing rows 0 - 4 (5 total). Query took 0.0003 seconds.
 - select * from students;
- Table Data:** A table named 'students' is displayed with the following data:

student_id	student_name	age	class	address
1	student_1	6	1	address of student_1
2	student_2	11	6	address of student_2
3	student_3	13	8	address of student_3
4	student_4	16	11	address of student_4
5	student_5	8	3	address of student_5

2 SQL Syntax

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

```

1 use school_db;
2
3 select student_name, age
4 from students;
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists databases: New, information_schema, mysql, performance_schema, and phpmyadmin. The 'school_db' database is selected. The main area displays the 'students' table with the following data:

	student_name	age
<input type="checkbox"/>	student_1	6
<input type="checkbox"/>	student_2	11
<input type="checkbox"/>	student_3	13
<input type="checkbox"/>	student_4	16
<input type="checkbox"/>	student_5	8

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

```

1 use school_db;
2
3 select *
4 from students
5 where age > 10;

```

The screenshot shows the results of the SQL query in the 'Query results operations' section. The 'students' table is displayed again, but only the rows for student_1, student_2, and student_3 are highlighted in yellow, while student_4 and student_5 are white. The highlighted rows correspond to the results of the query where age > 10.

	student_name	age
<input type="checkbox"/>	student_1	6
<input type="checkbox"/>	student_2	11
<input type="checkbox"/>	student_3	13
<input type="checkbox"/>	student_4	16
<input type="checkbox"/>	student_5	8

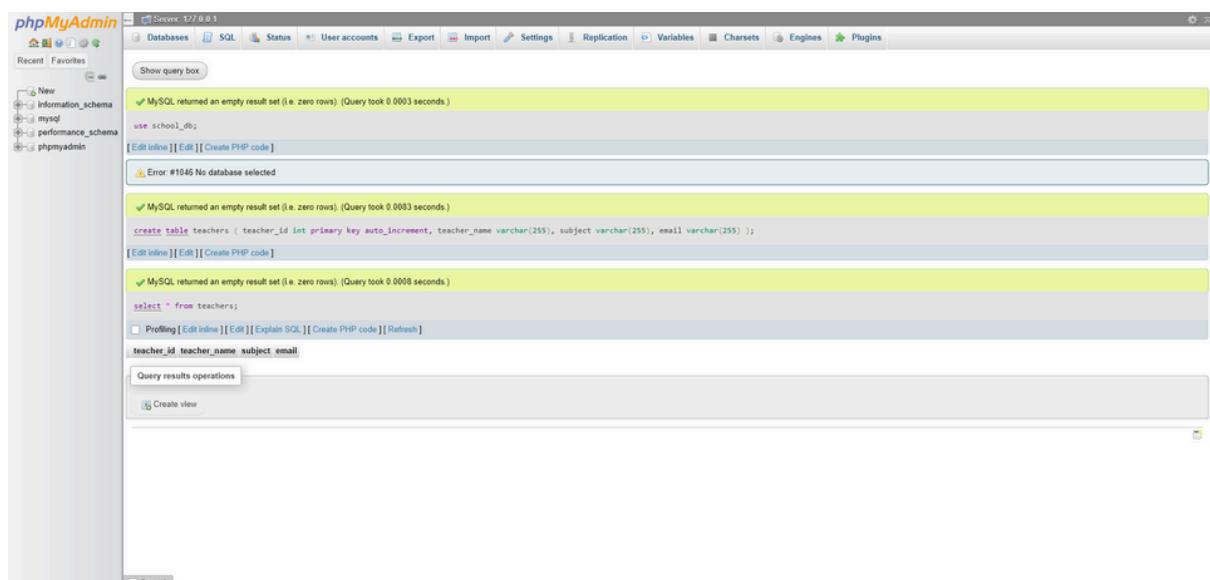
3 SQL Constraints

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

```

1 use school_db;
2
3 create table teachers (
4     teacher_id int primary key auto_increment,
5     teacher_name varchar(255),
6     subject varchar(255),
7     email varchar(255)
8 );
9
10 select * from teachers;
11

```

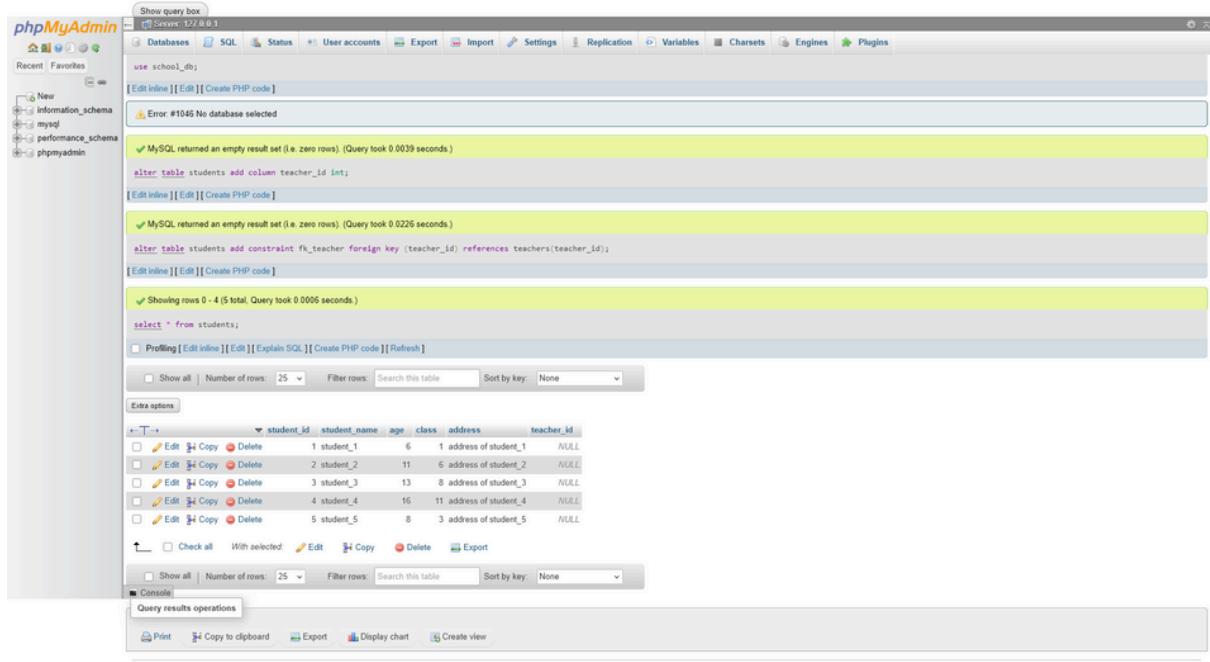


- 3.2 Implement a **FOREIGNKEY** constraint to relate the **teacher_id** from the **teachers** table with the **students** table.

```

1 use school_db;
2
3 alter table students
4 add column teacher_id int;
5
6 alter table students
7 add constraint fk_teacher
8 foreign key (teacher_id) references teachers(teacher_id);
9
10 select *
11 from students;

```



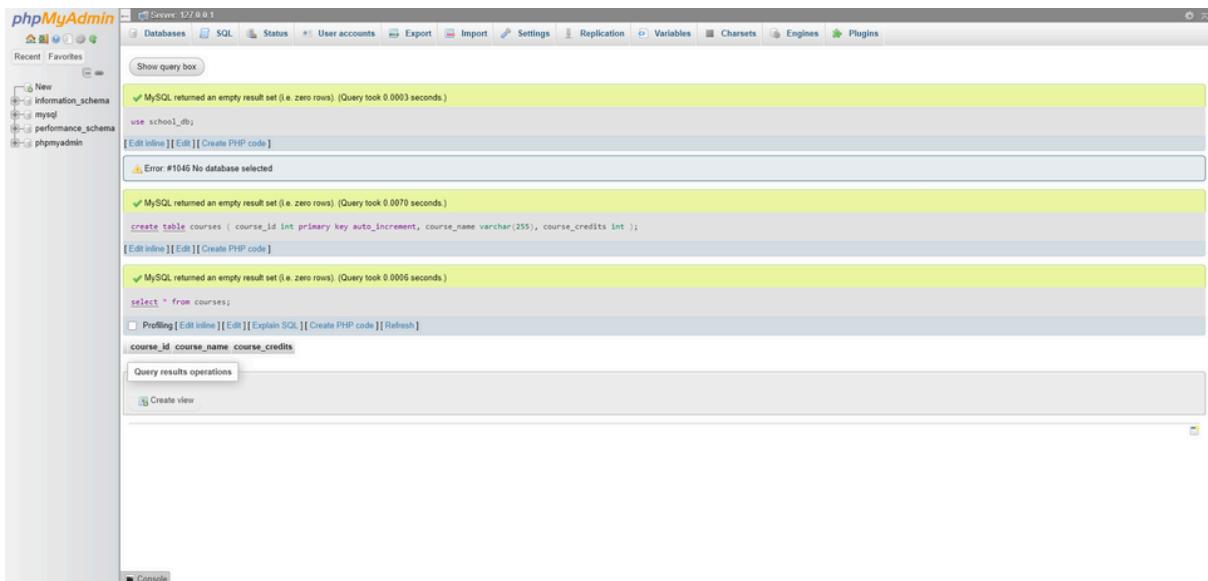
4 MainSQL Commands and Sub-commands (DDL)

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

```

1 use school_db;
2
3 create table courses (
4     course_id  int primary key auto_increment,
5     course_name  varchar(255),
6     course_credits  int
7 );
8
9 select *
10 from courses;

```



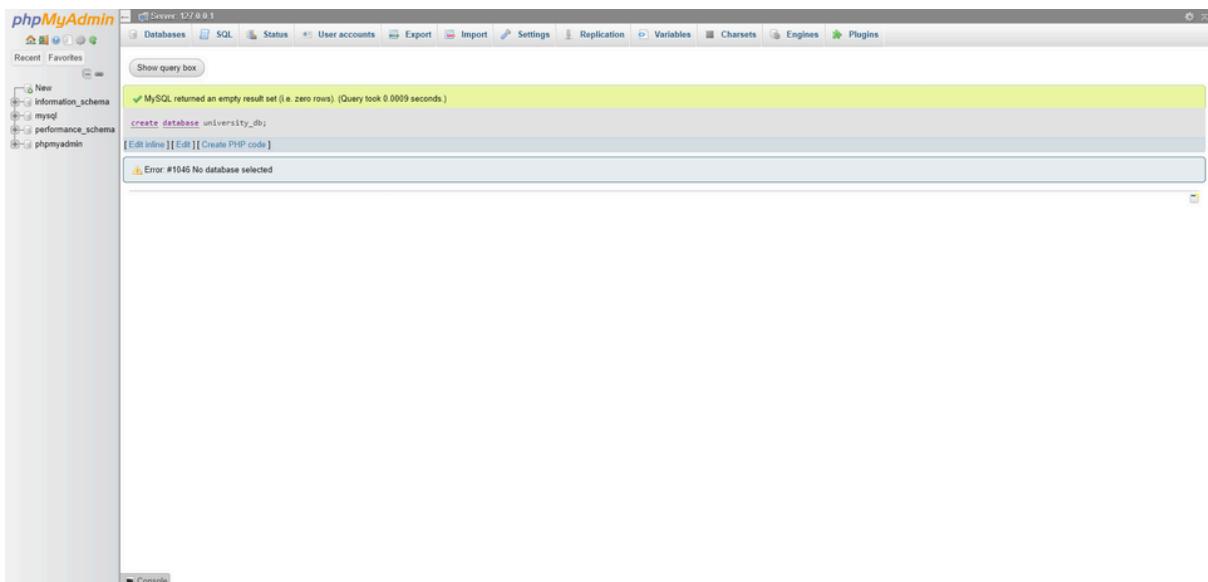
The screenshot shows the phpMyAdmin interface for MySQL version 127.0.0.1. In the left sidebar, databases like information_schema, mysql, performance_schema, and phpmyadmin are listed. The main query window contains the following SQL code:

```
use school_db;
create table courses (course_id int primary key auto_increment, course_name varchar(255), course_credits int);
select * from courses;
```

The results pane shows a green success message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)". Below the code, there are buttons for Edit inline, Edit, Create PHP code, Profiling, Explain SQL, and Refresh.

4.2 Use the CREATE command to create a database university_db.

```
1 create database university_db;
```



The screenshot shows the phpMyAdmin interface for MySQL version 127.0.0.1. In the left sidebar, databases like information_schema, mysql, performance_schema, and phpmyadmin are listed. The main query window contains the following SQL code:

```
create database university_db;
```

The results pane shows a green success message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0009 seconds.)". Below the code, there are buttons for Edit inline, Edit, Create PHP code, Profiling, Explain SQL, and Refresh.

5 ALTERCommand

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

```
1 use school_db;
2
3 alter table courses
4 add column course_duration int;
```

Python-Backend

```
5
6      *
7 from courses;
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'school1_db'. In the left sidebar, the 'Information_schema' and 'performance_schema' databases are visible. The main query window contains the following SQL code:

```
use school1_db;
alter table courses add column course_duration int;
```

The results pane shows two successful queries:

- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds)
- use school1_db;

Below the queries, there is an error message:

Error #1046 No database selected

At the bottom of the query window, the table definition is shown:

```
course_id course_name course_credits course_duration
```

5.2 Drop the `course_credits` column from the `courses` table.

```
1 use school_db;
2
3 alter table courses
4 drop column course_credits;
5
6 select *
7 from courses;
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'school1_db'. In the left sidebar, the 'Information_schema' and 'performance_schema' databases are visible. The main query window contains the following SQL code:

```
use school1_db;
alter table courses drop column course_credits;
```

The results pane shows two successful queries:

- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds)
- use school1_db;

Below the queries, there is an error message:

Error #1046 No database selected

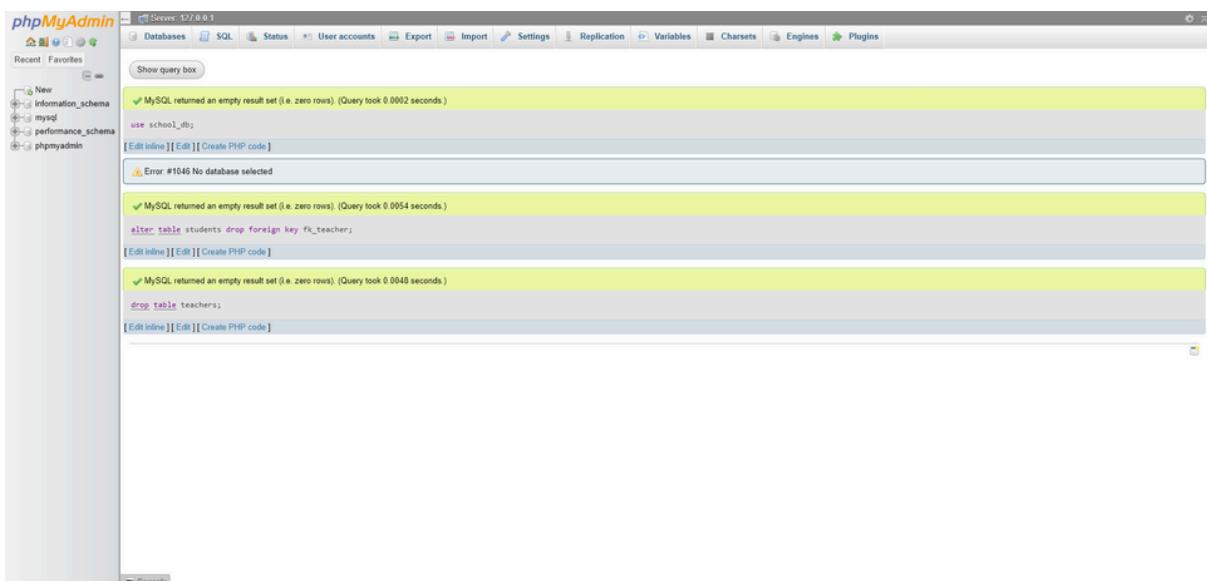
At the bottom of the query window, the table definition is shown:

```
course_id course_name course_duration
```

6 DROPCommand

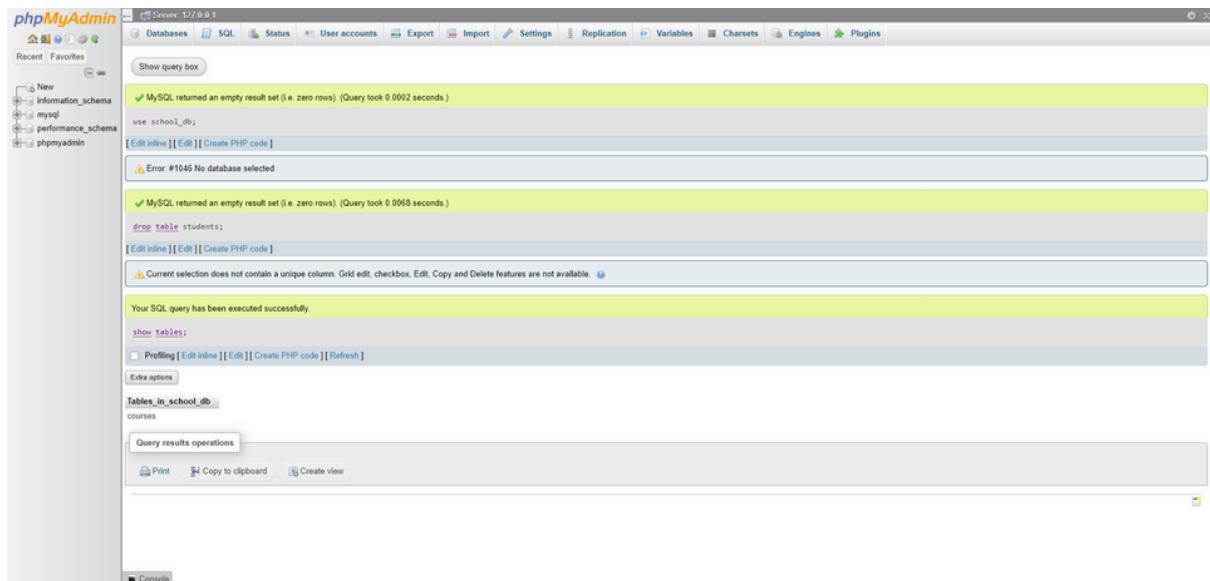
6.1 Drop the `teachers` table from the `school_db` database.

```
1 use school_db;
2
3 alter table students
4 drop foreign key fk_teacher;
5
6 drop table teachers;
```



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

```
1 use school_db;
2
3 drop table students;
4
5 tables;
```



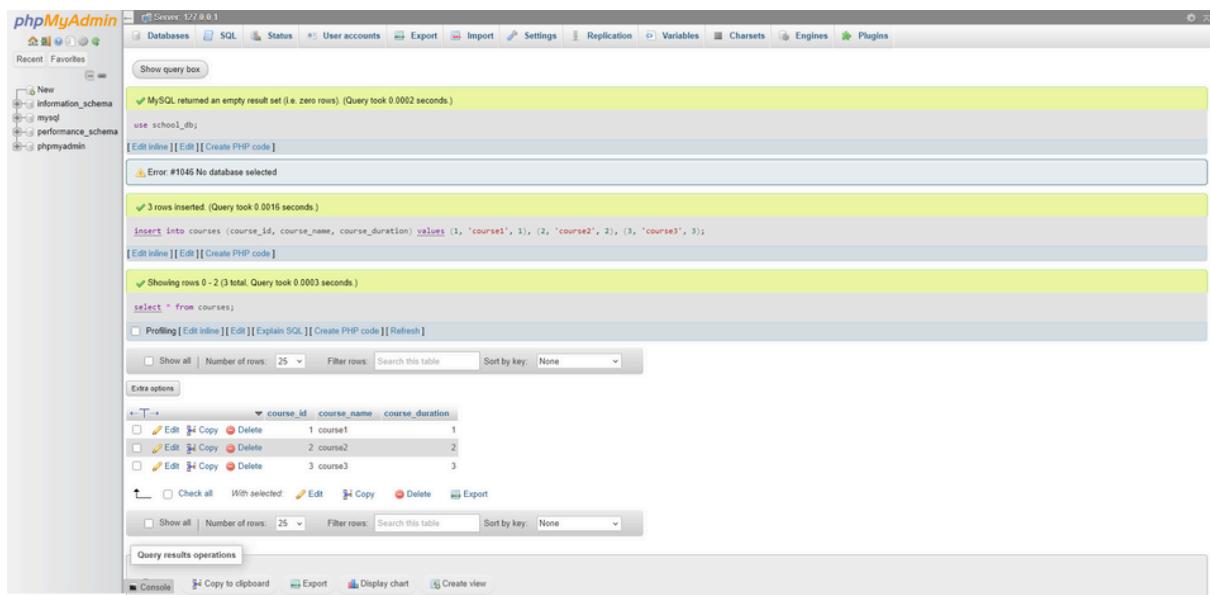
7 DataManipulationLanguage(DML)

7.1 Insert three records into the `courses` table using the **INSERT** command.

```

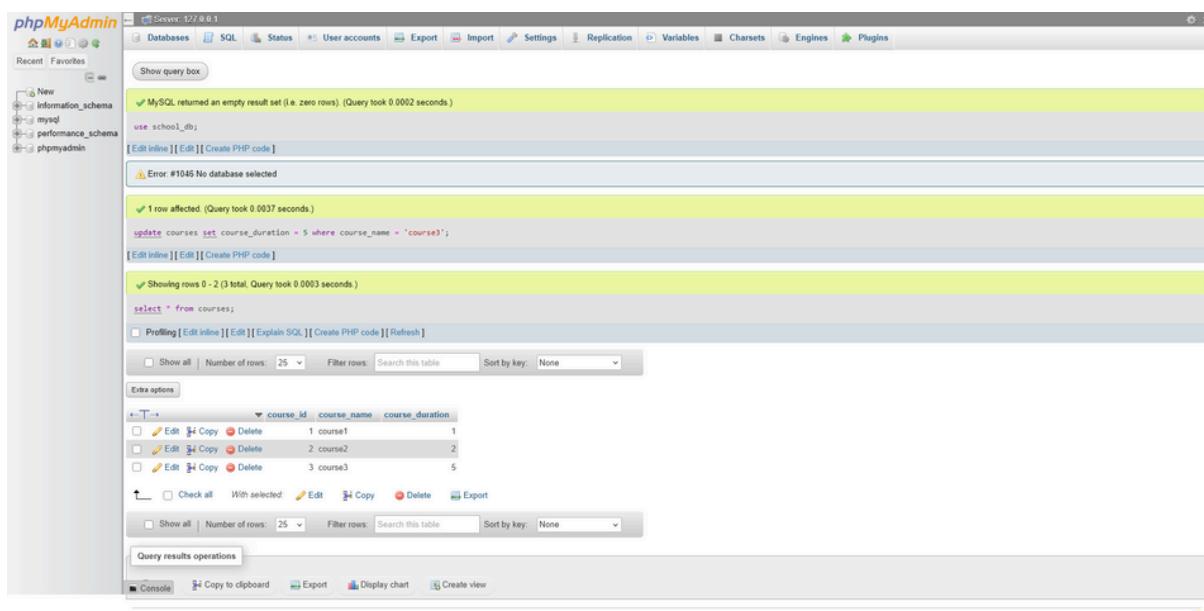
1 use school_db;
2
3 insert into courses (course_id, course_name, course_duration) values
4   (1, 'course1', 1),
5   (2, 'course2', 2),
6   (3, 'course3', 3);
7
8 select *
9 from courses;

```



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

```
1 use school_db;
2
3 update courses
4 set course_duration = 5
5 where course_name = 'course3';
6
7 select *
8 from courses;
```



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

```
1 use school_db;
2
3 delete from courses
4 where course_id = 1;
5
6 select *
7 from courses;
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'school_db'. The 'courses' table is selected, displaying three rows:

course_id	course_name	course_duration
2	course2	2
3	course3	5

8 DataQueryLanguage(DQL)

8.1 Retrieve all courses from the `courses` table using the `SELECT` statement.

```

1 use school_db;
2
3 select *
4 from courses;
```

The screenshot shows the phpMyAdmin interface after running the provided SQL query. The results pane indicates "Showing rows 0 - 1 (2 total. Query took 0.0002 seconds.)". The results table is empty.

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

```

1 use school_db;
2
3 select *
4 from courses
5 order by course_duration desc;
```

The screenshot shows the phpMyAdmin interface. In the left sidebar, the 'Databases' section lists 'Information_schema', 'mysql', and 'performance_schema'. The main area shows a query results table with the following data:

	course_id	course_name	course_duration
3	course3	5	
2	course2	2	

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

```

1 use school_db;
2
3 select *
4 from courses limit 2;
```

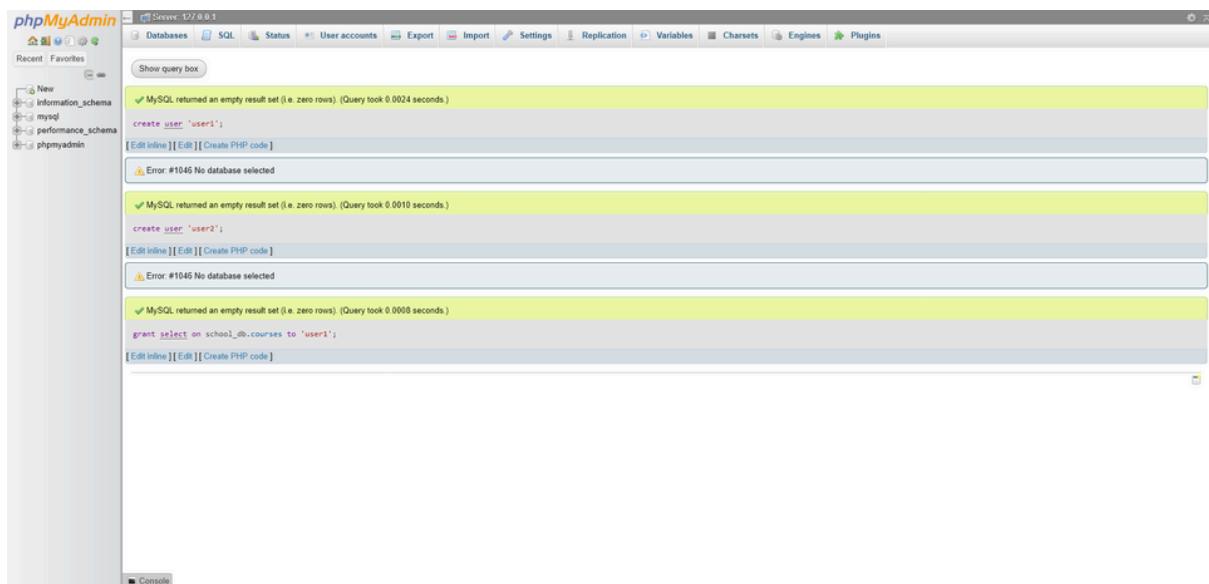
The screenshot shows the phpMyAdmin interface. The database structure is identical to the previous screenshot. The query results table now shows the following data:

	course_id	course_name	course_duration
2	course2	2	
3	course3	5	

9 DataControlLanguage(DCL)

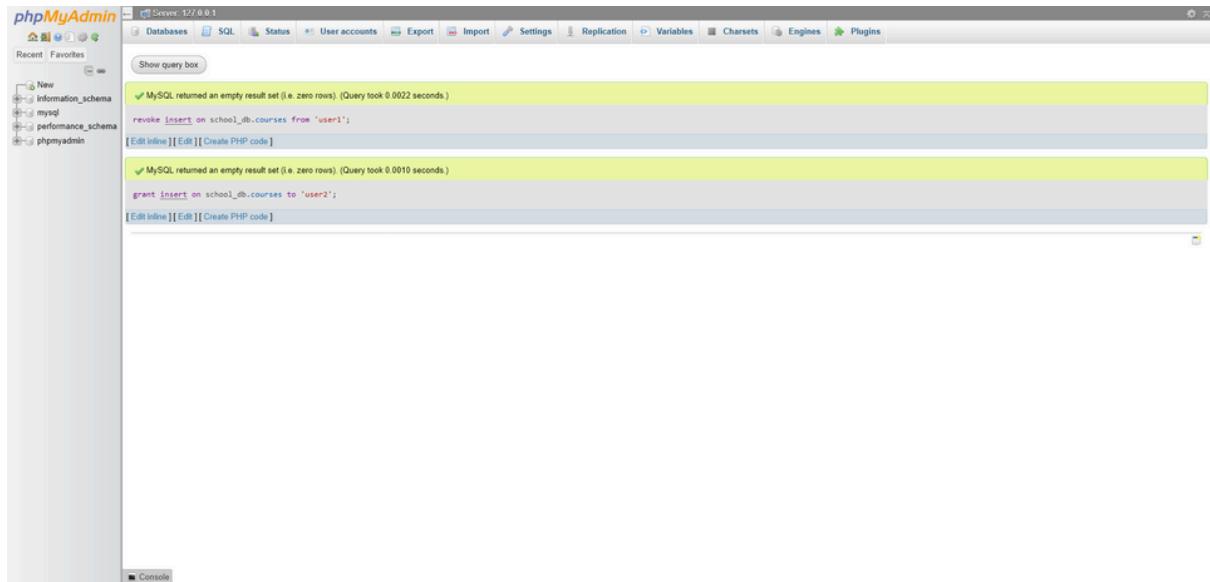
9.1 Create two new users `user1` and `user2` and grant `user1` permission to `SELECT` from the `courses` table.

```
1 create user 'user1';
2
3 create user 'user2';
4
5 grant select on school_db.courses
6 to 'user1';
```



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

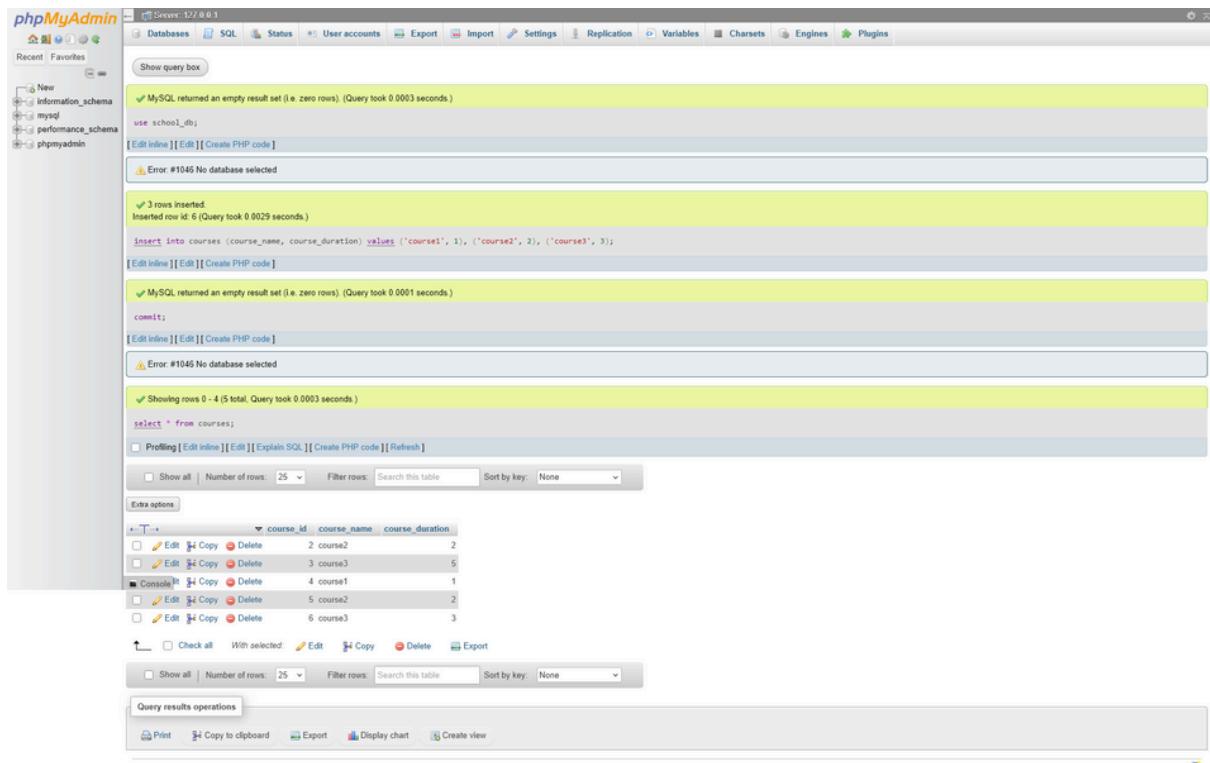
```
1 revoke insert on school_db.courses
2 from 'user1';
3
4 grant insert on school_db.courses
5 to 'user2';
```



10 Transaction Control Language (TCL)

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

```
1 use school_db;
2
3 insert into courses (course_name, course_duration) values
4     ('course1', 1),
5     ('course2', 2),
6     ('course3', 3);
7
8 commit;
9
10 select * from courses;
```

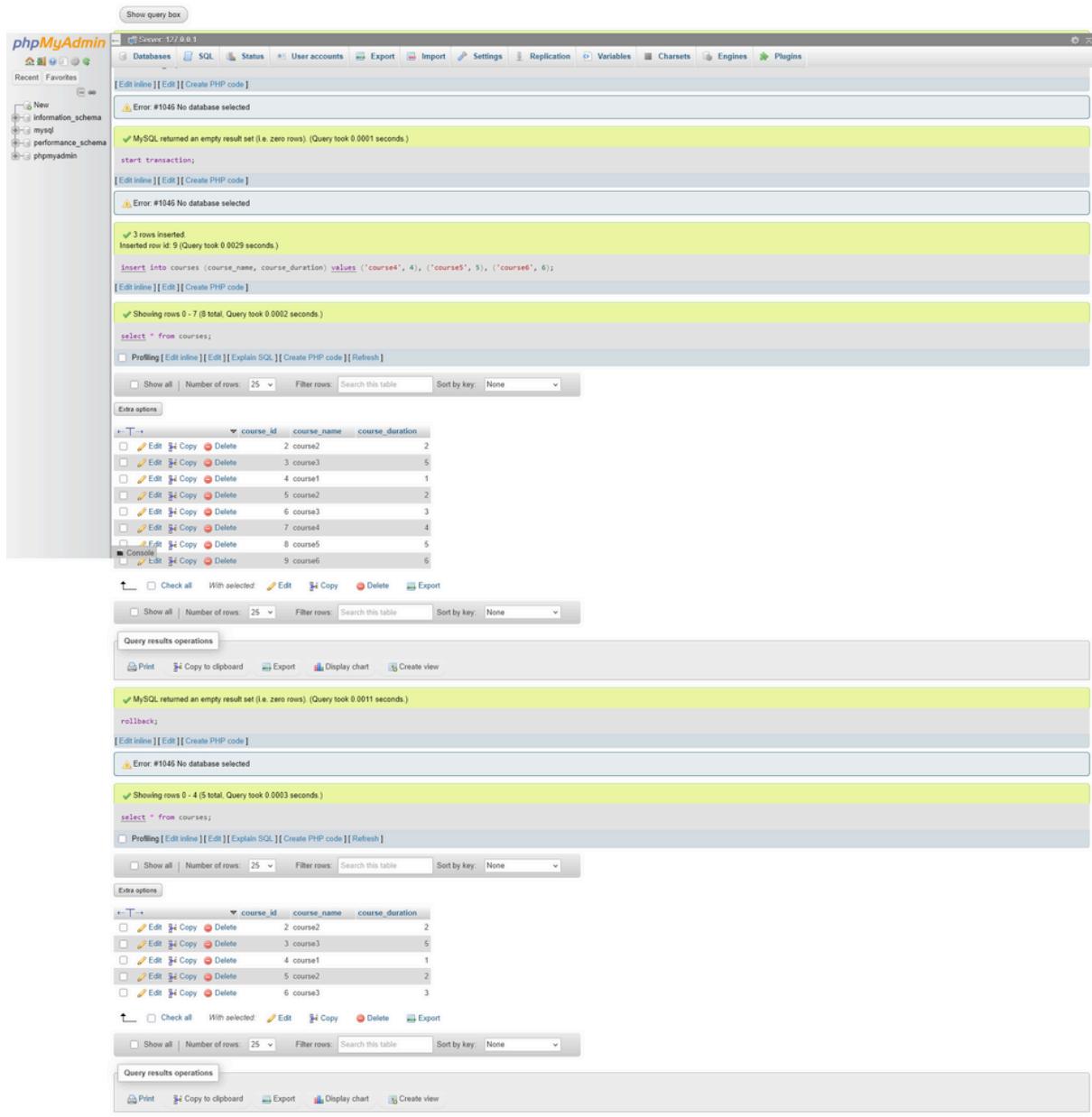


10.2 Insert additional rows, then use `ROLLBACK` to undo the last insert operation.

```

1 use school_db;
2
3 start transaction;
4
5 insert into courses (course_name, course_duration) values
6     ('course4', 4),
7     ('course5', 5),
8     ('course6', 6);
9
10 select * from courses;
11
12 rollback ;
13
14 select * from courses;

```



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

```

1 use school_db;
2
3 start transaction;
4
5 delete from courses
6 where course_id=5;
7
8 select * from courses;
9
10 savepoint sp1;

```

```
11  
12 delete from courses  
13 where course_id=6;  
14  
15 select * from courses;  
16  
17 rollback to savepoint spl;  
18  
19 select * from courses;  
20  
21 commit;
```

Python-Backend

The screenshot shows the MySQL Workbench interface with three vertically stacked sections of database operations:

- Top Section:** Shows a query result set from the 'courses' table. The table has columns: course_id, course_name, and course_duration. The data is:

course_id	course_name	course_duration
2	course2	2
3	course3	5
4	course1	1

With a total of 3 rows.
- Middle Section:** Shows a query result set from the 'courses' table. The table has columns: course_id, course_name, and course_duration. The data is:

course_id	course_name	course_duration
2	course2	2
3	course3	5
4	course1	1

With a total of 3 rows.
- Bottom Section:** Shows a query result set from the 'courses' table. The table has columns: course_id, course_name, and course_duration. The data is:

course_id	course_name	course_duration
2	course2	2
3	course3	5
4	course1	1
6	course3	3

With a total of 4 rows.

Each section includes a 'Query results operations' panel with options like Print, Copy to clipboard, Export, Display chart, and Create view.

The screenshot shows the phpMyAdmin interface with a single section of database operations:

- Section:** Shows a query result set from the 'courses' table. The table has columns: course_id, course_name, and course_duration. The data is:

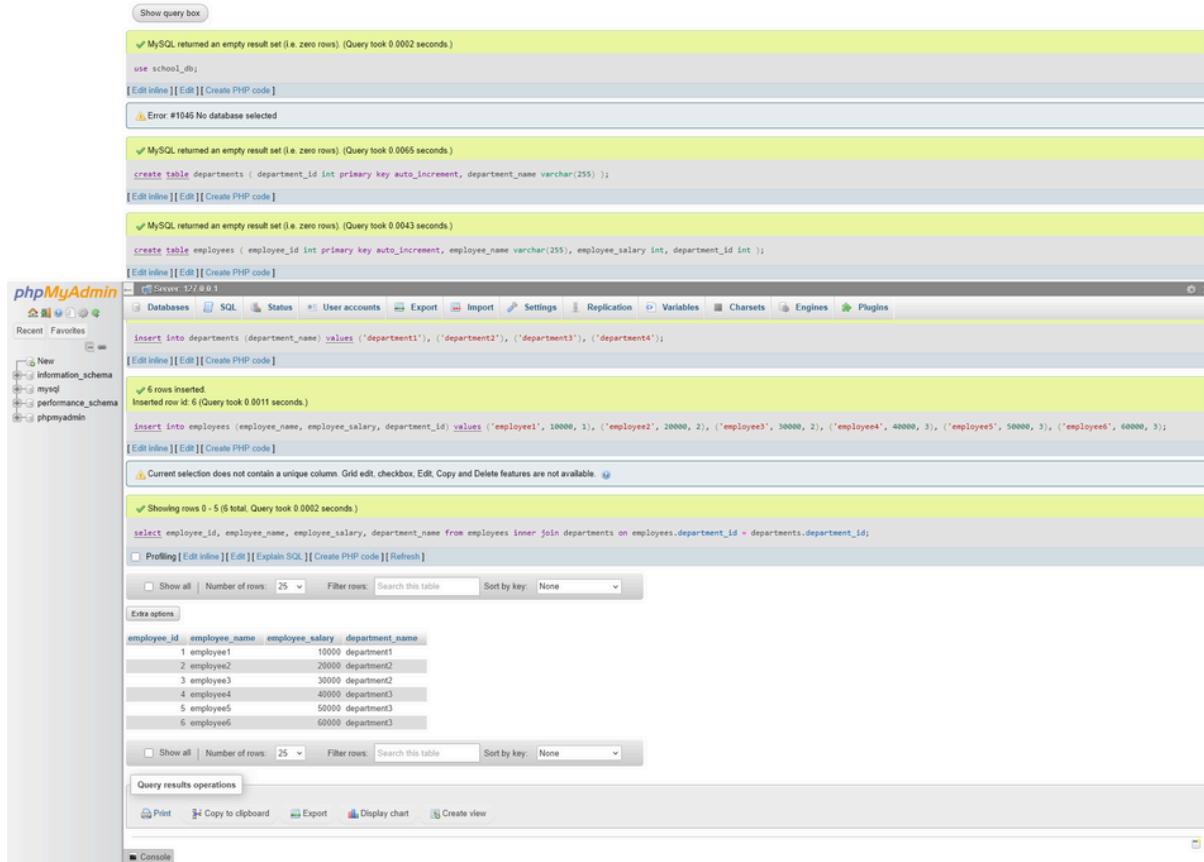
course_id	course_name	course_duration
2	course2	2
3	course3	5
4	course1	1
6	course3	3

With a total of 4 rows.
- Bottom Panel:** Includes a 'Console' tab.

11 SQL Joins

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

```
1 use school_db;
2
3 create table departments (
4     department_id int primary key auto_increment,
5     department_name varchar(255)
6 );
7
8 create table employees (
9     employee_id int primary key auto_increment,
10    employee_name varchar(255),
11    employee_salary int,
12    department_id int
13 );
14
15 insert into departments (department_name) values
16 ('department1'),
17 ('department2'),
18 ('department3'),
19 ('department4');
20
21 insert into employees (employee_name, employee_salary, department_id)
22     ) values
23 ('employee1', 10000, 1),
24 ('employee2', 20000, 2),
25 ('employee3', 30000, 2),
26 ('employee4', 40000, 3),
27 ('employee5', 50000, 3),
28 ('employee6', 60000, 3);
29
30 select employee_id, employee_name, employee_salary, department_name
31 from employees
32 inner join departments
33 on employees.department_id = departments.department_id;
```

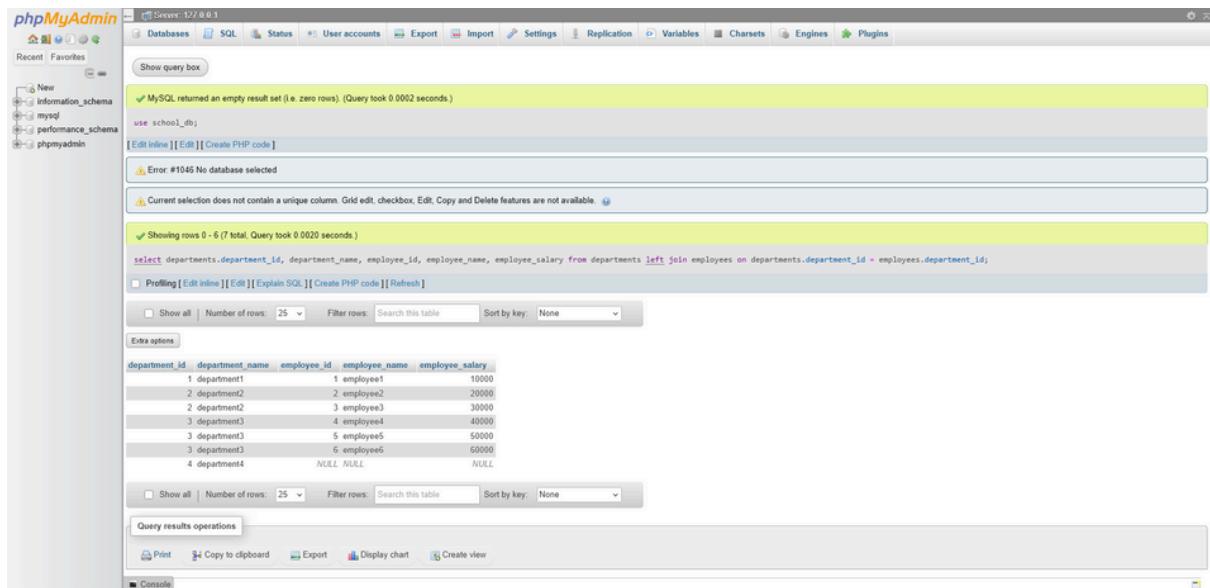


11.2 Use a LEFT JOIN to show all departments, even those without employees.

```

1 use school_db;
2
3 select departments.department_id, department_name, employee_id,
4       employee_name, employee_salary
5   from departments
6   m  join employees
7      leftdepartments.department_id employees.department_id;

```



The screenshot shows the phpMyAdmin interface with the following details:

- Servers:** 127.0.0.1
- Databases:** Databases, SQL, Status, User accounts, Export, Import, Settings, Replication, Variables,Charsets, Engines, Plugins
- Recent:** New, Information_schema, mysql, performance_schema, phpmyadmin
- Query Box:**
 - MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)
 - use school_db;
 - Error: #1045 No database selected
 - Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
 - Showing rows 0 - 6 (7 total). Query took 0.0020 seconds.
- Table:** departments, employees

department_id	department_name	employee_id	employee_name	employee_salary
1	department1	1	employee1	10000
2	department2	2	employee2	20000
2	department2	3	employee3	30000
3	department3	4	employee4	40000
3	department3	5	employee5	50000
3	department3	6	employee6	60000
4	department4	NULL	NULL	NULL
- Operations:** Print, Copy to clipboard, Export, Display chart, Create view
- Console:** (empty)

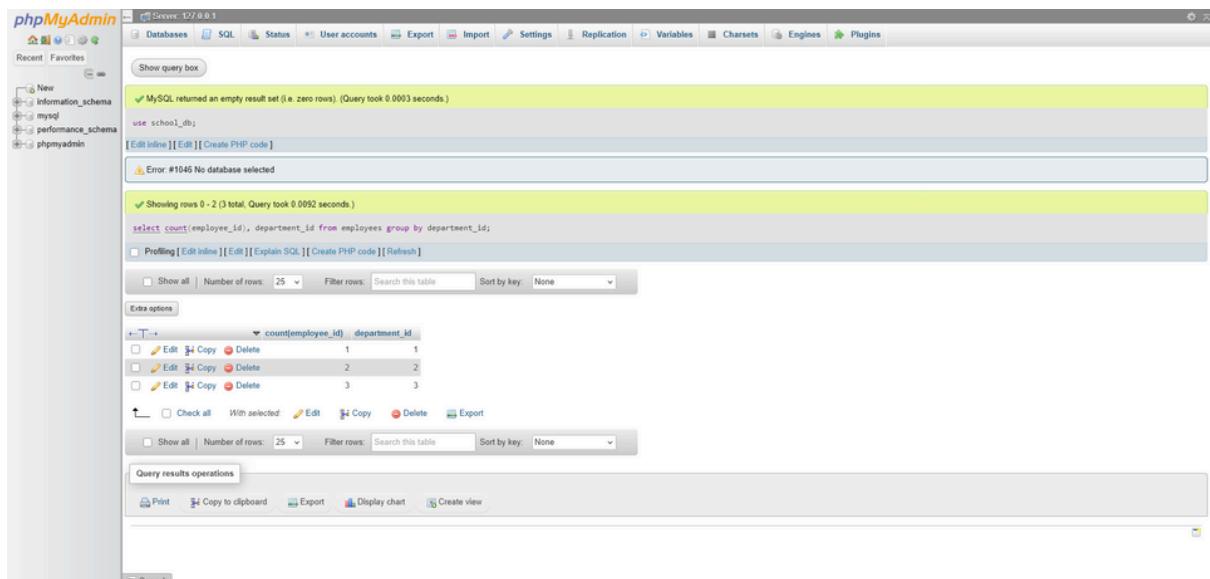
12 SQL Group By

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

```

1 use school_db;
2
3 select count(employee_id), department_id
4 from employees
5 group by department_id;

```



The screenshot shows the phpMyAdmin interface with the following details:

- Servers:** 127.0.0.1
- Databases:** Databases, SQL, Status, User accounts, Export, Import, Settings, Replication, Variables,Charsets, Engines, Plugins
- Recent:** New, Information_schema, mysql, performance_schema, phpmyadmin
- Query Box:**
 - MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
 - use school_db;
 - Error: #1046 No database selected
 - Showing rows 0 - 2 (3 total). Query took 0.0092 seconds.
 - select count(employee_id), department_id from employees group by department_id;
- Table:** employees

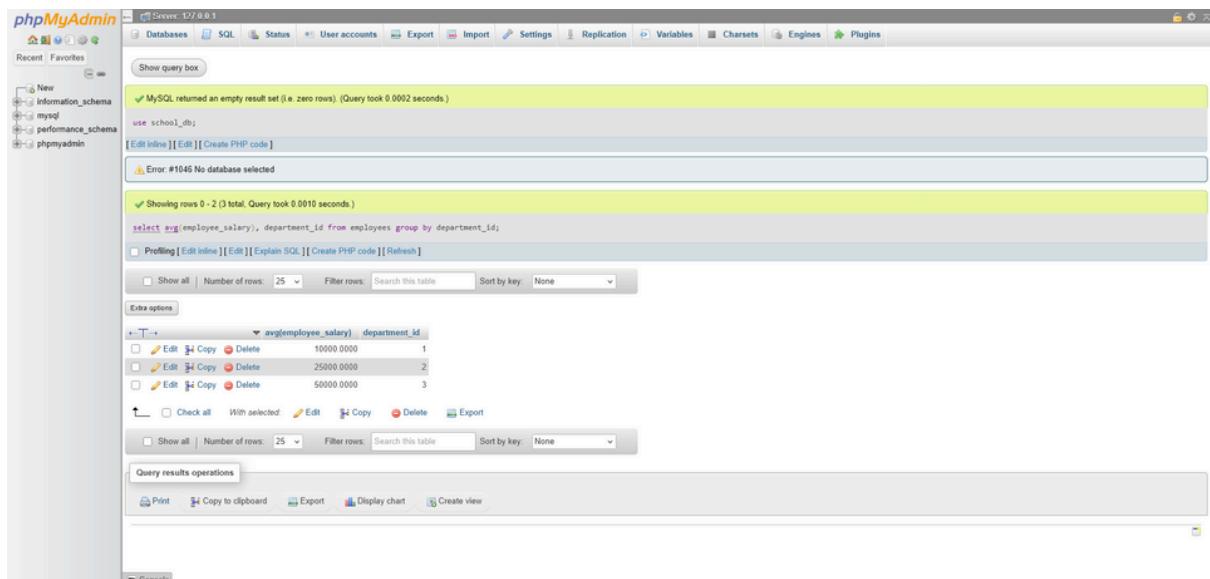
	count(employee_id)	department_id
1	1	1
2	2	2
3	3	3
- Operations:** Print, Copy to clipboard, Export, Display chart, Create view
- Console:** (empty)

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

```

1 use school_db;
2
3 select avg(employee_salary), department_id
4 from employees
5 group by department_id;

```



13 SQL Stored Procedure

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

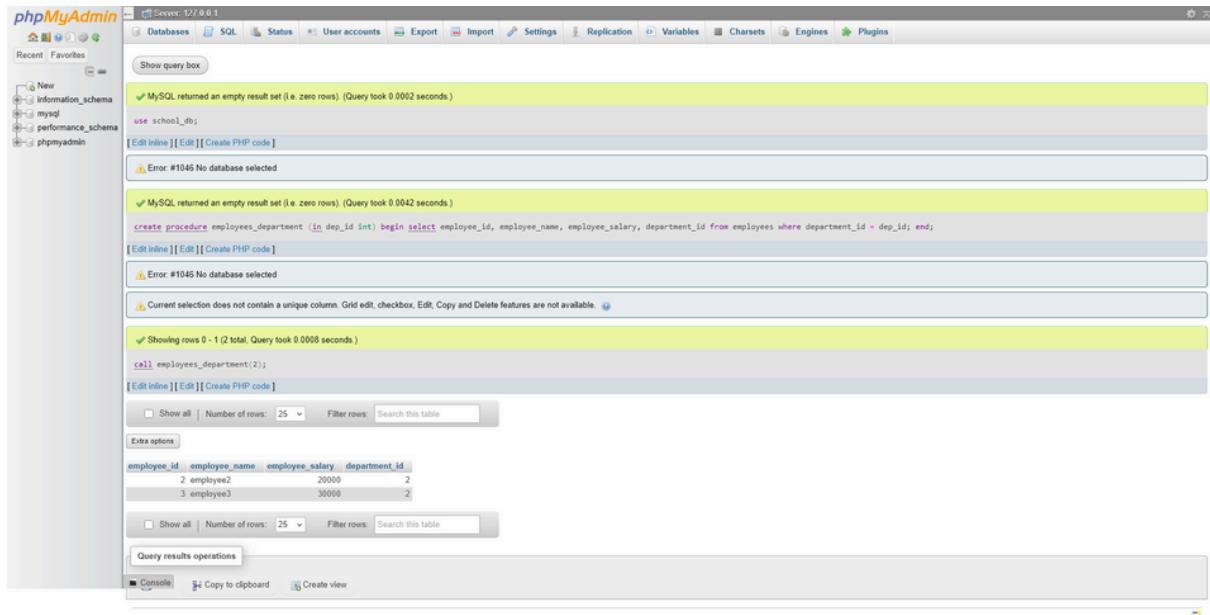
```

1 use school_db;
2
3 delimiter $$

4
5 create procedure employees_department(in dep_id int)
6 begin
7     select employee_id, employee_name, employee_salary,
8           department_id
9     from employees
10    where department_id = dep_id;
11 end $$

12 delimiter ;
13
14
15 call employees_department(2);

```



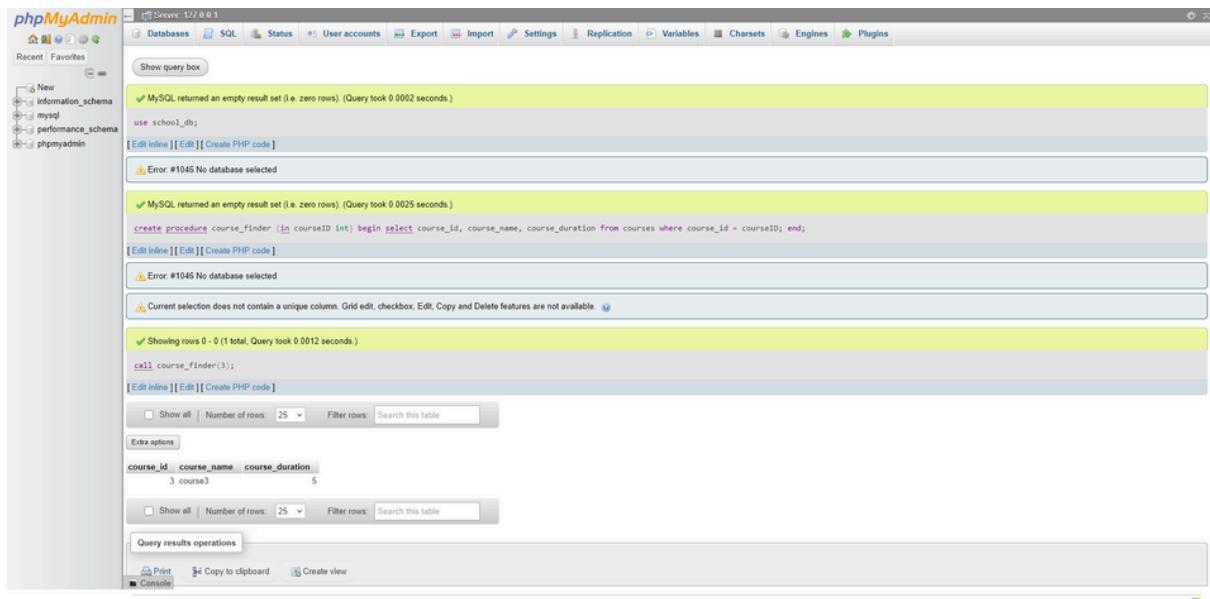
13.2 Write a stored procedure that accepts
and returns the course details.

`course_id` as input

```

1 use school_db;
2
3 delimiter $$ 
4
5 create procedure course_finder (in courseID int)
6 begin
7     select course_id, course_name, course_duration
8     from courses
9     where course_id = courseID;
10 end $$ 
11
12 delimiter ;
13
14
15 call course_finder(3);

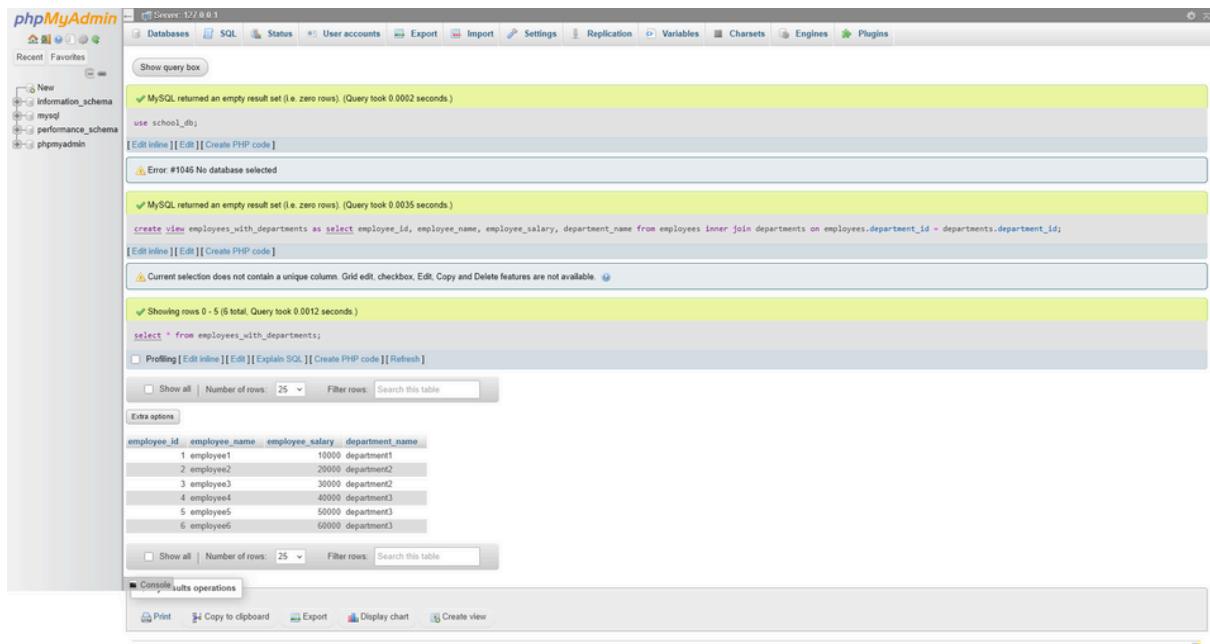
```



14 SQL View

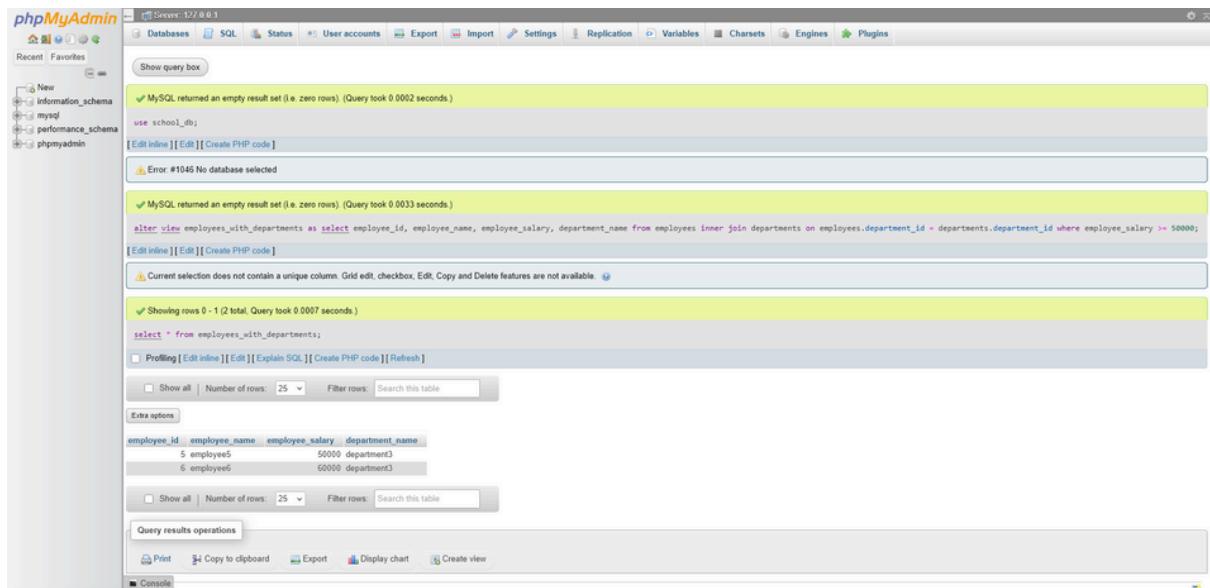
14.1 Create a view to show all employees along with their department names.

```
1 use school_db;
2
3 create view employees_with_departments
4 select employee_id, employee_name, employee_salary, department_name
5 from employees
6 inner join departments
7 on employees.department_id = departments.department_id;
8
9
10 select * from employees_with_departments;
```



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

```
1 use school_db;
2
3 alter view employees_with_departments
4 select employee_id, employee_name, employee_salary, department_name
5 from employees
6 inner join departments
7 on employees.department_id = departments.department_id
8 where employee_salary >= 50000;
9
10
11 select * from employees_with_departments;
```



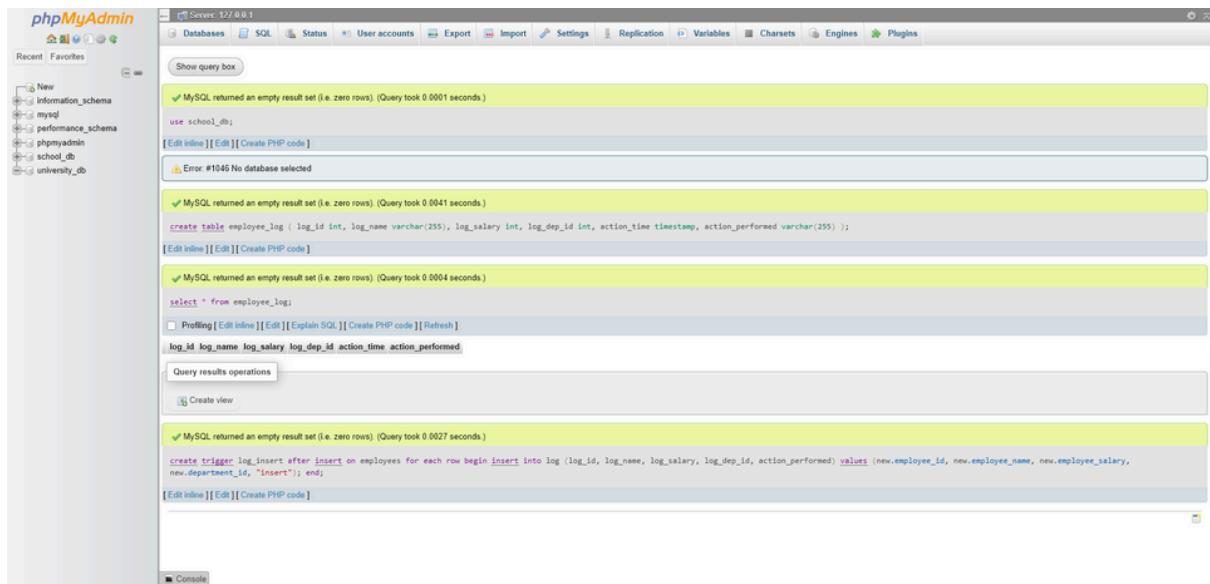
15 SQL Triggers

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

```

1 use school_db;
2
3 create table employee_log (
4     log_id int,
5     log_name varchar(255),
6     log_dep_id int,
7     action_time int,
8     action_performed timestamp,
9     d varchar(255)
10);
11
12 select * from employee_log;
13
14 delimiter $$ 
15 create trigger log_insert
16 after insert on employees
17 for each row
18 begin
19     insert into log (log_id, log_name, log_salary, log_dep_id,
20                     action_performed)
21     values (new.employee_id, new.employee_name, new.employee_salary,
22             new.department_id, "insert");
23 end $$ 
24 delimiter ;

```



The screenshot shows the phpMyAdmin interface for a MySQL database named 'school_db'. In the left sidebar, the 'Databases' section lists 'Information_schema', 'mysql', 'performance_schema', 'phpmyadmin', 'school_db', and 'university_db'. The main query window displays the following SQL code:

```

use school_db;

create table employee_log (
    log_id int,
    log_name varchar(255),
    log_salary int,
    log_dep_id int,
    action_time timestamp,
    action_performed varchar(255));

```

Below this, another query is shown:

```

select * from employee_log;

```

Finally, a trigger is created:

```

create trigger log_insert after insert on employees for each row begin insert into log (log_id, log_name, log_salary, log_dep_id, action_performed) values (new.employee_id, new.employee_name, new.employee_salary, new.department_id, "insert"); end;

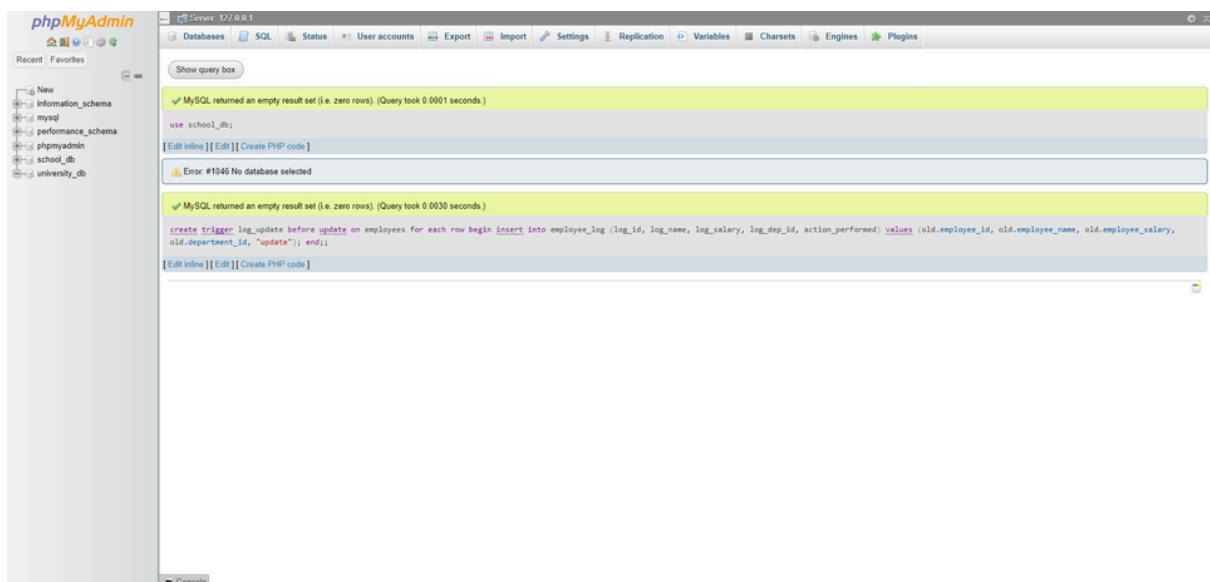
```

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

```

1 use school_db;
2
3 delimiter $$ 
4 create trigger log_update
5 before update on employees
6 for each row
7 begin
8     insert into employee_log(log_id, log_name, log_salary,
9         log_dep_id, action_performed)
10    values (old.employee_id, old.employee_name, old.employee_salary,
11            old.department_id, "update");
12 end; $$ 
13 delimiter ;

```



The screenshot shows the phpMyAdmin interface for a MySQL database named 'school_db'. The left sidebar shows the same database list as the previous screenshot. The main query window displays the following SQL code:

```

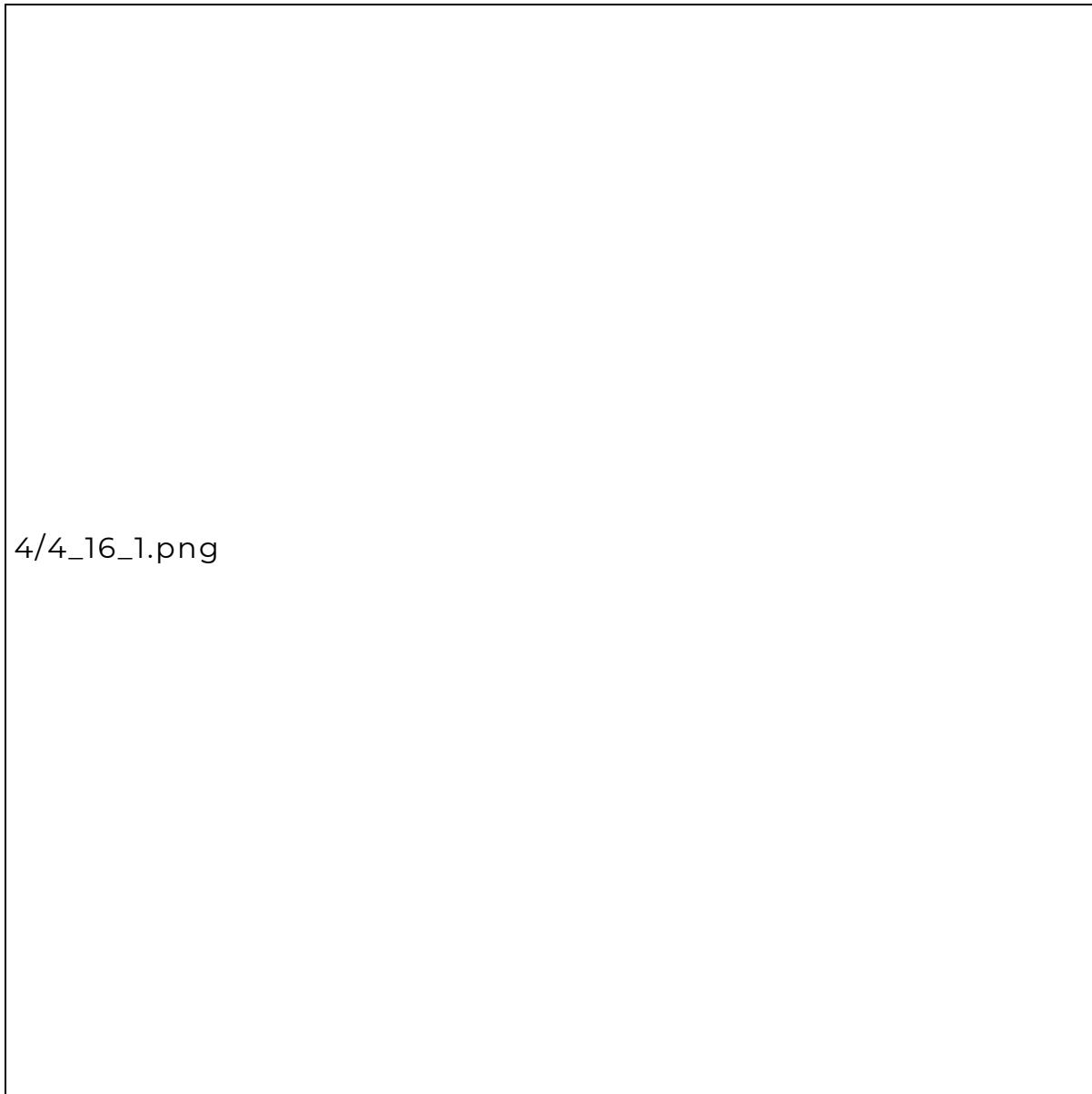
use school_db;

create trigger log_update before update on employees for each row begin insert into employee_log (log_id, log_name, log_salary, log_dep_id, action_performed) values (old.employee_id, old.employee_name, old.employee_salary, old.department_id, "update"); end;

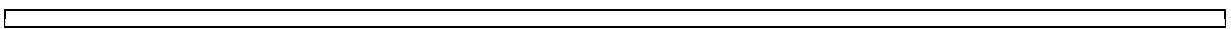
```

16 Introduction to PL/SQL

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



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



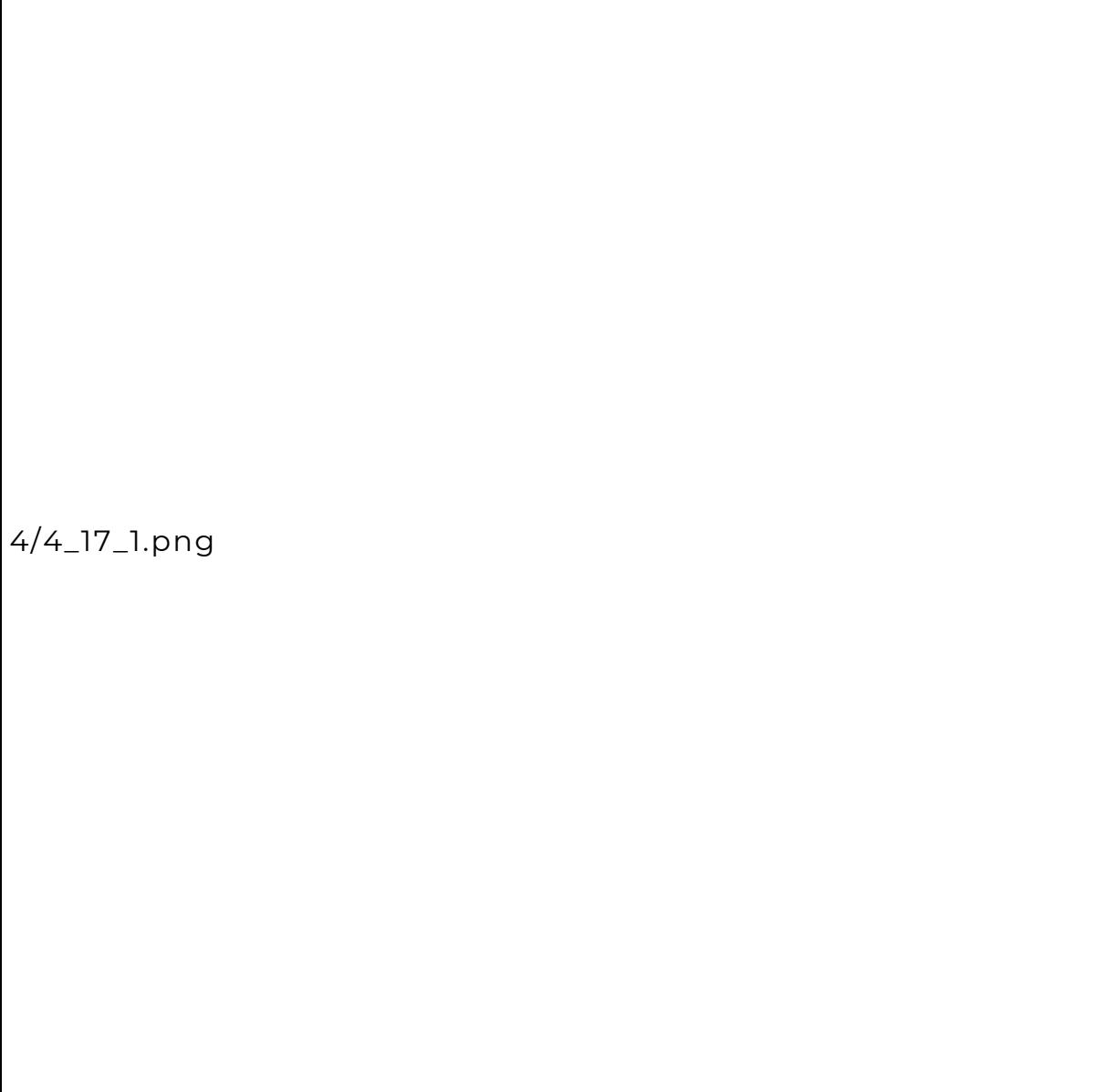
4/4_16_2.png

17 PL/SQL Control Structures

17.1 Write a PL/SQL block using an

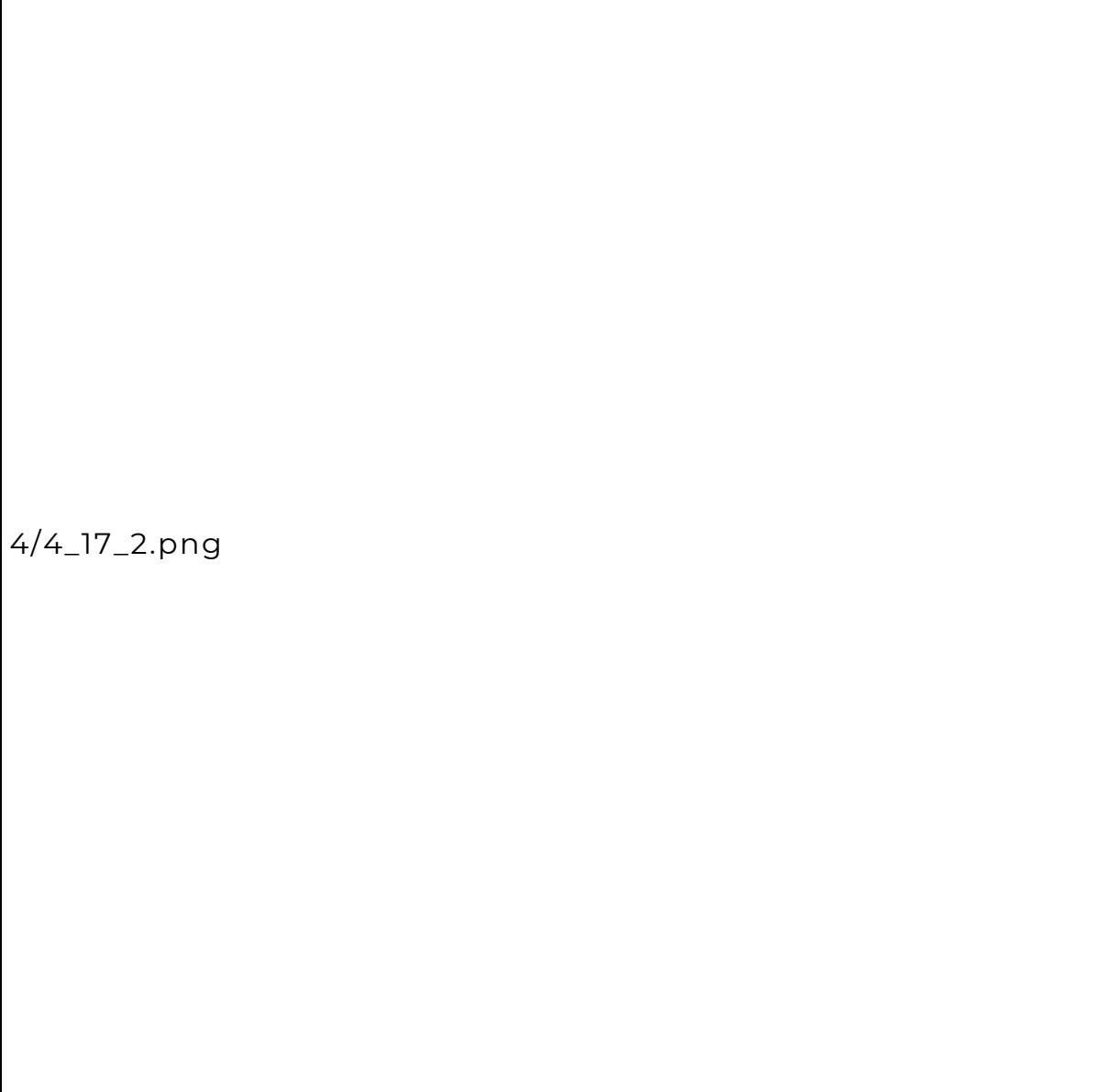
IF-THEN condition to

check the department of an employee.



4/4_17_1.png

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



4/4_17_2.png

18 SQL Cursors

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

4/4_18_1.png

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

4/4_18_2.png

19 Rollback and Commit Savepoint

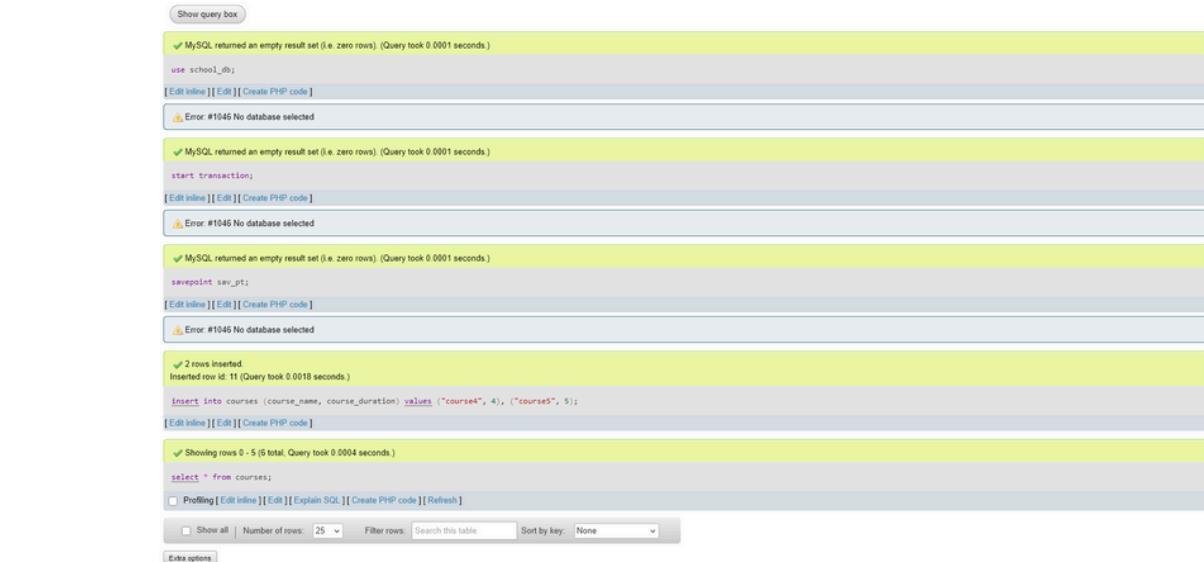
- 19.1 Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint?

```
1 use school_db;
2
3 start transaction;
4
5 savepoint sav_pt;
6
7 insert into courses (course_name, course_duration) values
8     ("course4", 4),
9     ("course5", 5);
```

```

11 select * from courses;
12
13 rollback to savepoint sav_pt;
14
15 select * from courses;
16
17 commit;

```



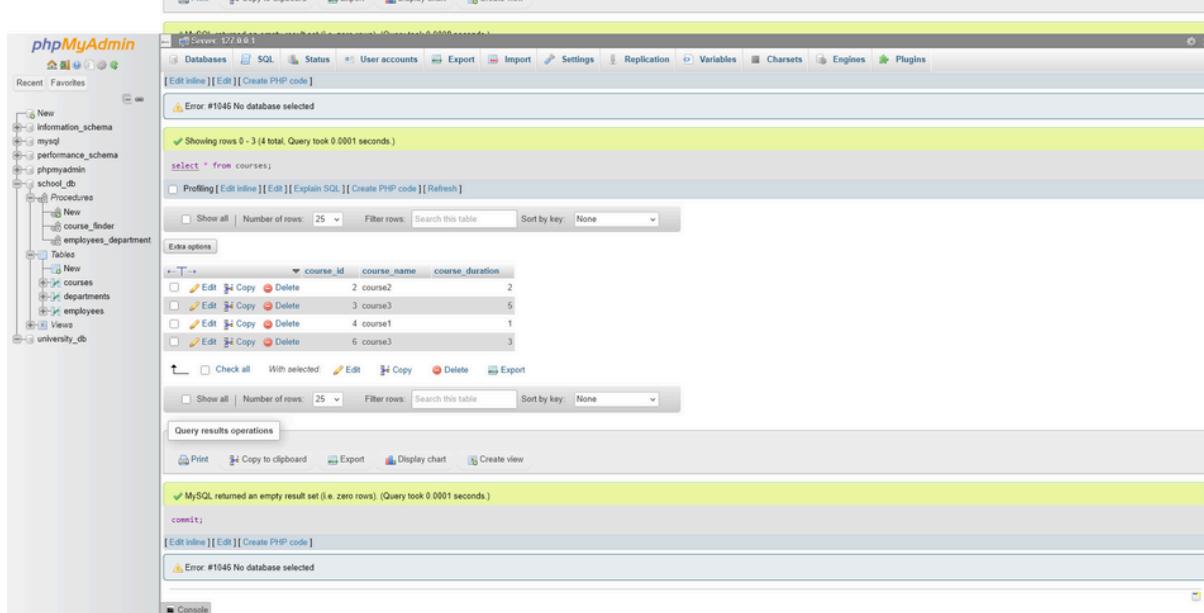
The screenshot shows the MySQL Workbench interface. At the top, there is a transaction log window with the following entries:

- Line 1: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)
- Line 2: use school_db;
- Line 3: Error #1046 No database selected
- Line 4: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)
- Line 5: start transaction;
- Line 6: Error #1046 No database selected
- Line 7: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)
- Line 8: savepoint sav_pt;
- Line 9: Error #1046 No database selected
- Line 10: 2 rows inserted.
- Line 11: Inserted row id: 11 (Query took 0.0018 seconds.)
- Line 12: insert into courses (course_name, course_duration) values ('course4', 4), ('course5', 5);
- Line 13: Error #1046 No database selected
- Line 14: Showing rows 0 - 5 (5 total). Query took 0.0004 seconds.
- Line 15: select * from courses;
- Line 16: Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Line 17: Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None
- Line 18: Extra options

Below the transaction log is a table named 'courses' with the following data:

	Edit	Copy	Delete	course_id	course_name	course_duration
<input type="checkbox"/>				2	course2	2
<input type="checkbox"/>				3	course3	5
<input type="checkbox"/>				4	course1	1
<input type="checkbox"/>				6	course3	3
<input type="checkbox"/>				10	course4	4

At the bottom of the table view are buttons: Show all, Number of rows: 25, Filter rows, Search this table, Sort by key: None, and Extra options.



The screenshot shows the phpMyAdmin interface. On the left, the database structure is displayed under the 'school_db' database:

- New
- Information_schema
- mysql
- performance_schema
- phpmyadmin
- school_db
 - Procedures
 - Tables
 - New
 - courses
 - departments
 - employees
 - employees_department
 - Views
- university_db

The main area shows a query results table with the following data:

	Edit	Copy	Delete	course_id	course_name	course_duration
<input type="checkbox"/>				2	course2	2
<input type="checkbox"/>				3	course3	5
<input type="checkbox"/>				4	course1	1
<input type="checkbox"/>				6	course3	3

At the bottom of the table view are buttons: Show all, Number of rows: 25, Filter rows, Search this table, Sort by key: None, and Extra options.

Below the table is a 'Query results operations' section with buttons: Print, Copy to clipboard, Export, Display chart, Create view.

The transaction log at the bottom of the page shows:

- Showing rows 0 - 3 (4 total). Query took 0.0001 seconds.
- select * from courses;
- Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None
- Extra options
- Error #1046 No database selected
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)
- commit;
- Error #1046 No database selected

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

```
1 use school_db;
2
3 start transaction;
4
5 update courses
6 set course_name = "course4"
7 where course_id = 6;
8
9 select * from courses;
10
11 savepoint sav_pt2;
12
13 update courses
14 set course_duration = 4
15 where course_id = 6;
16
17 select * from courses;
18
19 rollback to sav_pt2;
20
21 select * from courses;
22
23 commit
```

Python-Backend

MySQL returned an empty result set (i.e. zero rows) (Query took 0.0001 seconds)

```
use school_db;
```

Error #1046 No database selected

MySQL returned an empty result set (i.e. zero rows) (Query took 0.0001 seconds)

```
start transaction;
```

Error #1046 No database selected

1 row affected (Query took 0.0021 seconds)

```
update courses set course_name = "course4" where course_id = 6;
```

Showing rows 0 - 3 (4 total. Query took 0.0001 seconds.)

```
select * from courses;
```

Profiling | Edit inline | Edit | Explain SQL | Create PHP code | Refresh

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	Edit	Copy	Delete	2	course2	2	
<input type="checkbox"/>				3	course3	5	
<input type="checkbox"/>				4	course1	1	
					course_id	course_name	course_duration

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

MySQL returned an empty result set (i.e. zero rows) (Query took 0.0001 seconds)

```
savepoint xav_pt2;
```

Error #1046 No database selected

1 row affected (Query took 0.0002 seconds)

```
update courses set course_duration = 4 where course_id = 6;
```

Showing rows 0 - 3 (4 total. Query took 0.0001 seconds.)

```
select * from courses;
```

Profiling | Edit inline | Edit | Explain SQL | Create PHP code | Refresh

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	Edit	Copy	Delete	2	course2	2	
<input type="checkbox"/>				3	course3	5	
<input type="checkbox"/>				4	course1	1	
					course_id	course_name	course_duration

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

MySQL returned an empty result set (i.e. zero rows) (Query took 0.0001 seconds)

```
commit;
```

Error #1046 No database selected

Console