

DBMS Practical Implementation, Lab 8.

Task 1

1. Write a stored procedure to accept salary of the employee and display grade of employee accordingly.

If salary > 50000 then grade is 'A'

If salary between 30000 to 50000 the grade is 'B' and

If salary < 30000 then grade is 'C':

DELIMITER //

CREATE PROCEDURE sal_grades(IN salary int, OUT grade varchar(1))

BEGIN

IF salary >= 50000 THEN

set grade = 'A';

elseif salary >= 30000 then

set grade = 'B';

else

set grade = 'C';

end if;

end //

DELIMITER ;

CALL sal_grades(75000,@grade);

SELECT @grade;

CALL sal_grades(45000,@grade);

SELECT @grade;

CALL sal_grades(25000,@grade);

SELECT @grade;

phpMyAdmin interface showing the SQL query editor for the database dbmspracs. The query is a CREATE PROCEDURE statement:

```
1 DELIMITER //
2 CREATE PROCEDURE sal_grades(IN salary int, OUT grade varchar(1))
3 BEGIN
4     IF salary >= 50000 THEN
5         set grade = 'A';
6     elseif salary >= 30000 then
7         set grade = 'B';
8     else
9         set grade = 'C';
10    end if;
11 end //
12 DELIMITER ;
13
```

The query was executed successfully, returning an empty result set (0 rows). The status bar shows the query took 0.0068 seconds.

phpMyAdmin interface showing the results of the SQL query. The query was executed successfully, returning an empty result set (0 rows). The status bar shows the query took 0.0005 seconds.

The query results are displayed as follows:

```
CALL sal_grades(75000,@grade);
```

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds)

```
SELECT @grade
```

Options: @grade, A

Query results operations: Print, Copy to clipboard, Export, Display chart, Create view

phpMyAdmin interface showing a query result for the database dbmspracs. The query executed is `CALL sal_grades(45000, @grade)`, which returned an empty result set (0 rows). The second query executed is `SELECT @grade`, which returned 1 row with the value 'B'. The interface includes a sidebar with a database tree, a top navigation bar, and a bottom console area.

Database: dbmspracs

Query: `CALL sal_grades(45000, @grade)`

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

Query: `SELECT @grade`

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

Options: `@grade`

Query results operations: Print, Copy to clipboard, Export, Display chart, Create view

Bookmark this SQL query

Label: ☐ Let every user access this bookmark

Bookmark this SQL query

phpMyAdmin interface showing a query result for the database dbmspracs. The query executed is `CALL sal_grades(25000, @grade)`, which returned an empty result set (0 rows). The second query executed is `SELECT @grade`, which returned 1 row with the value 'C'. The interface includes a sidebar with a database tree, a top navigation bar, and a bottom console area.

Database: dbmspracs

Query: `CALL sal_grades(25000, @grade)`

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

Query: `SELECT @grade`

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

Options: `@grade`

Query results operations: Print, Copy to clipboard, Export, Display chart, Create view

Bookmark this SQL query

Label: ☐ Let every user access this bookmark

Bookmark this SQL query

2. To Create Table Employee: 2. Write a block to display sum of 1 to 10 numbers

```
DELIMITER //
```

```
CREATE procedure sum_of_10 ( OUT sum_int INT )
```

```
BEGIN
```

```
    DECLARE i INT;
```

```
    SET i = 1;
```

```
    SET sum_int = 0;
```

```
    label1: WHILE i < 11
```

```
    DO
```

```
        SET sum_int = sum_int + i;
```

```
        SET i = i + 1;
```

```
    END WHILE label1;
```

```
END; //
```

```
DELIMITER ;
```

```
CALL sum_of_10(@sumof10);
```

```
SELECT @sumof10;
```

phpMyAdmin interface showing the SQL query editor for database dbmspracs. The query is a stored procedure named sum_of_10.

```
1 DELIMITER //
2 CREATE procedure sum_of_10 ( OUT sum_int INT )
3 BEGIN
4   DECLARE i INT;
5   SET i = 1;
6   SET sum_int = 0;
7   label1: WHILE i < 11
8   DO
9     SET sum_int = sum_int + i;
10    SET i = i + 1;
11  END WHILE label1;
12 END; //
13 DELIMITER ;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0052 seconds)

CREATE procedure sum_of_10 (OUT sum_int INT) BEGIN DECLARE i INT; SET i = 1; SET sum_int = 0; label1: WHILE i < 11 DO SET sum_int = sum_int + i; SET i = i + 1; END WHILE label1; END;

phpMyAdmin interface showing the SQL query editor for database dbmspracs. The query is a stored procedure named sum_of_10.

```
1 CALL sum_of_10(@sumof10);
2 SELECT @sumof10;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0009 seconds)

CALL sum_of_10(@sumof10)

Showing rows 0 - 0 (1 total. Query took 0.0002 seconds.)

SELECT @sumof10

Options: @sumof10, 55

3. Write a block to display Fibonacci series upto 8th term (start with 0,1)

DELIMITER //

CREATE procedure fibbonacci_series (IN no_of_terms INT)

BEGIN

DECLARE i INT;

DECLARE f0,f1,f INT;

SET i = 2;

SET f0 = 0;

SET f1 = 1;

SELECT f0,f1;

SET f = f0+f1;

label1: WHILE i <= no_of_terms

DO

SELECT f;

SET i = i + 1;

SET f0 = f1;

SET f1 = f;

SET f = f0+f1;

END WHILE label1;

END; //

DELIMITER ;

SET @p0='8';

CALL `fibbonacci_series`(@p0);

phpMyAdmin interface showing the SQL query editor for the database dbmspracs. The query is a stored procedure named fibonacci_series.

```
1 DELIMITER //
2 CREATE procedure fibonacci_series ( IN no_of_terms INT )
3 BEGIN
4     DECLARE i INT;
5     DECLARE f0,f1,f INT;
6     SET i = 2;
7     SET f0 = 0;
8     SET f1 = 1;
9     SELECT f0,f1;
10    SET f = f0+f1;
11    label1: WHILE i <= no_of_terms
12    DO
13        SELECT f;
14        SET i = i + 1;
15        SET f0 = f1;
16        SET f1 = f;
17        SET f = f0+f1;
18    END WHILE label1;
19 END; //
20 DELIMITER ;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0081 seconds)

Console

phpMyAdmin interface showing the execution results of the stored procedure fibonacci_series. The query was executed successfully, affecting 1 row.

Execution results of routine 'fibonacci_series'

f0	f1
0	1
f	1
f	2
f	3
f	5
f	8
f	13
f	21

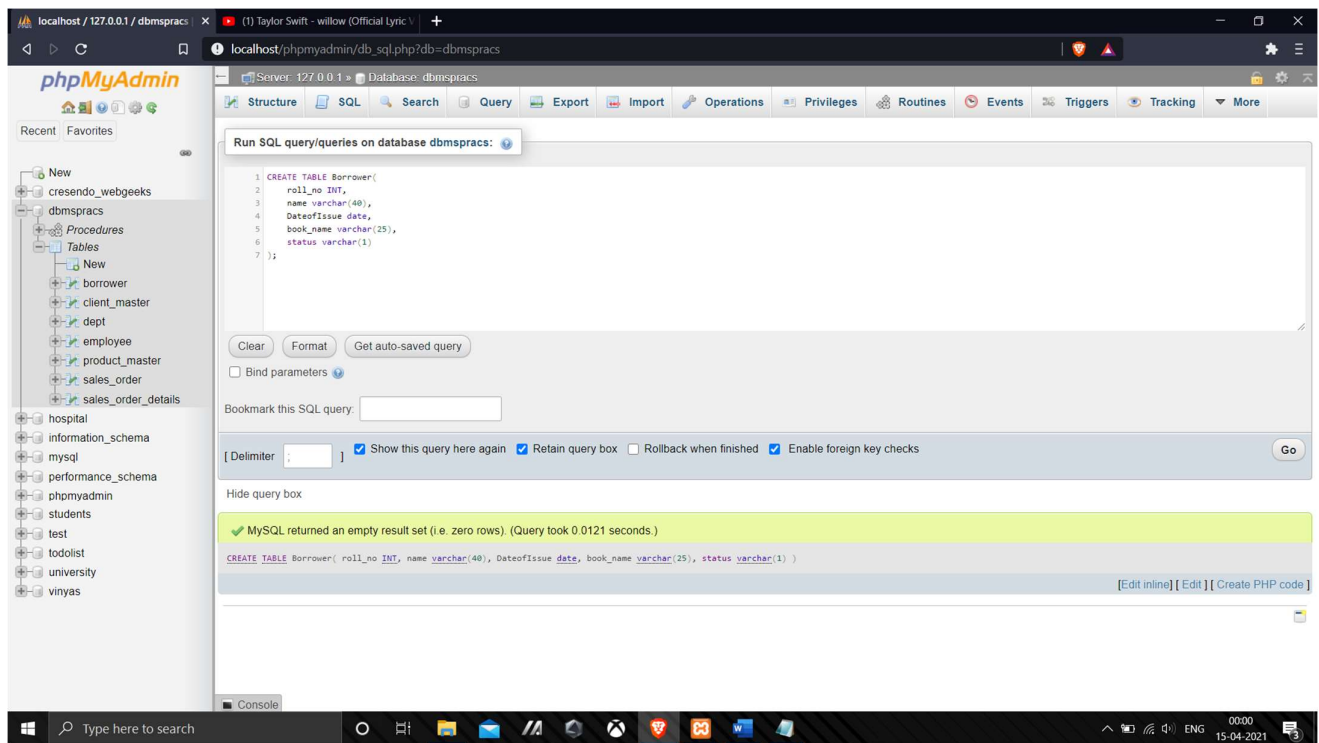
Routines

Name	Action	Type	Returns
fibonacci_series	Execute	PROCEDURE	

Task 2

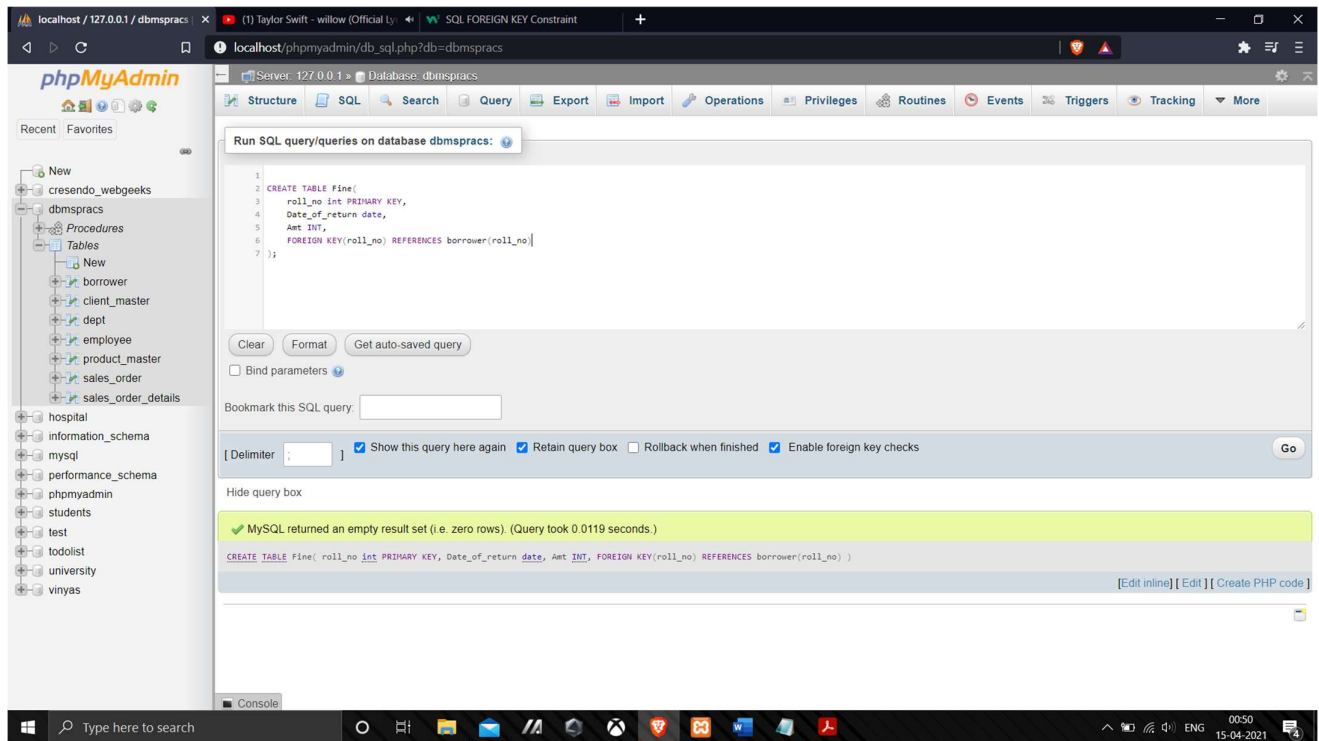
1. Creating Table Borrower:

```
CREATE TABLE Borrower(  
    roll_no INT PRIMARY KEY,  
    name varchar(40),  
    DateofIssue date,  
    book_name varchar(25),  
    status varchar(1)  
);
```



2. Creating Table fine:

```
CREATE TABLE Fine(  
    roll_no int PRIMARY KEY,  
    Date_of_return date,  
    Amt INT,  
    FOREIGN KEY(roll_no) REFERENCES borrower(roll_no)  
);
```



3. Inserting the necessary values:

```
INSERT INTO `borrower` (`roll_no`, `name`, `DateofIssue`, `book_name`,  
`status`) VALUES ('1', 'Amrita', '2021-04-01', 'Java', 'I');
```

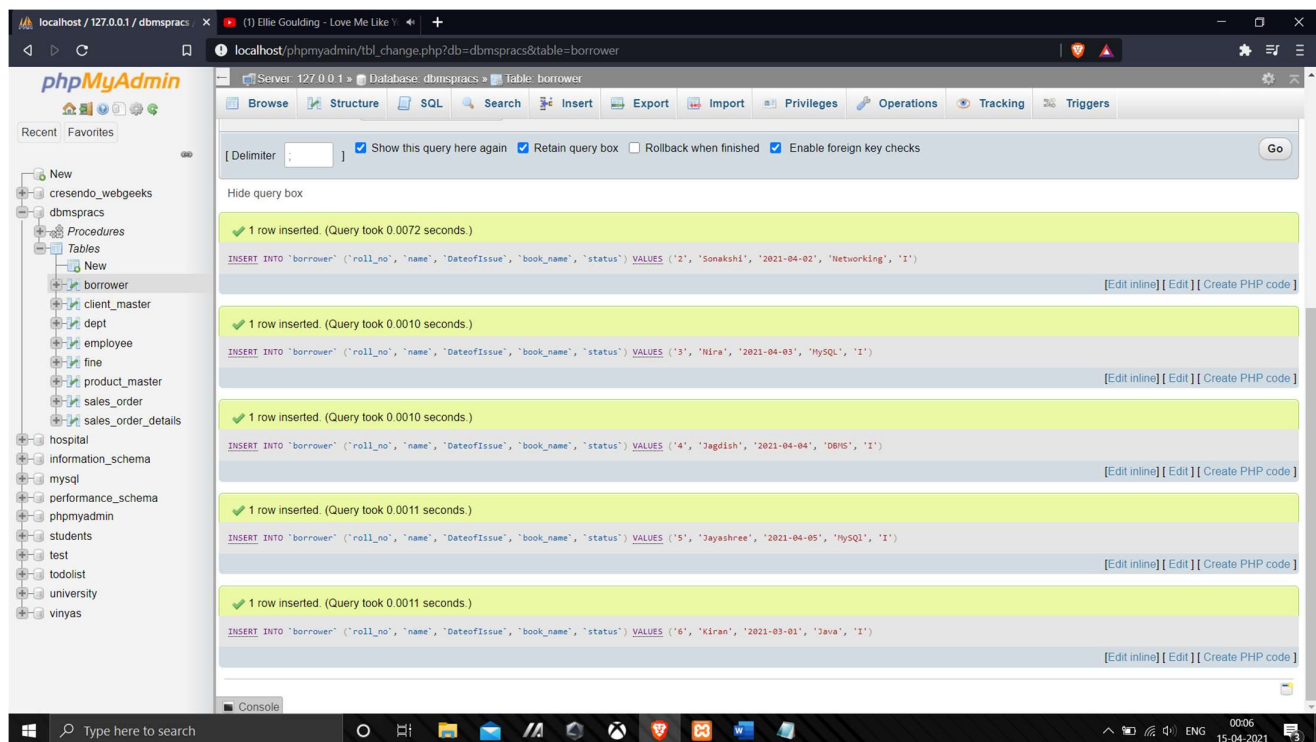
```
INSERT INTO `borrower` (`roll_no`, `name`, `DateofIssue`, `book_name`,  
`status`) VALUES ('2', 'Sonakshi', '2021-04-02', 'Networking', 'I');
```

```
INSERT INTO `borrower` (`roll_no`, `name`, `DateofIssue`, `book_name`,  
`status`) VALUES ('3', 'Nira', '2021-04-03', 'MySQL', 'I');
```

```
INSERT INTO `borrower` (`roll_no`, `name`, `DateofIssue`, `book_name`,  
`status`) VALUES ('4', 'Jagdish', '2021-04-04', 'DBMS', 'I');
```

```
INSERT INTO `borrower` (`roll_no`, `name`, `DateofIssue`, `book_name`,  
`status`) VALUES ('5', 'Jayashree', '2021-04-05', 'MySQL', 'I');
```

```
INSERT INTO `borrower` (`roll_no`, `name`, `DateofIssue`, `book_name`,  
`status`) VALUES ('6', 'Kiran', '2021-03-01', 'Java', 'I');
```



4. Creating the Procedure:

DELIMITER //

CREATE procedure fine_student (IN stud_roll_no INT,IN name_of_book
varchar(25))

BEGIN

DECLARE diff INT;

DECLARE fine_amt INT;

DECLARE date_of_issue date;

SELECT borrower.DateofIssue INTO date_of_issue FROM borrower WHERE
borrower.roll_no = stud_roll_no;

SET diff = DATEDIFF(CURRENT_DATE,date_of_issue);

SET fine_amt = 0;

IF diff >= 30

THEN

SET fine_amt = fine_amt + (diff-30)*50;

SET fine_amt = fine_amt + (diff-15)*5;

ELSEIF diff >= 15

THEN

SET fine_amt = fine_amt + (diff-15)*5;

END IF;

UPDATE borrower SET borrower.status = "R" WHERE borrower.roll_no =
stud_roll_no;

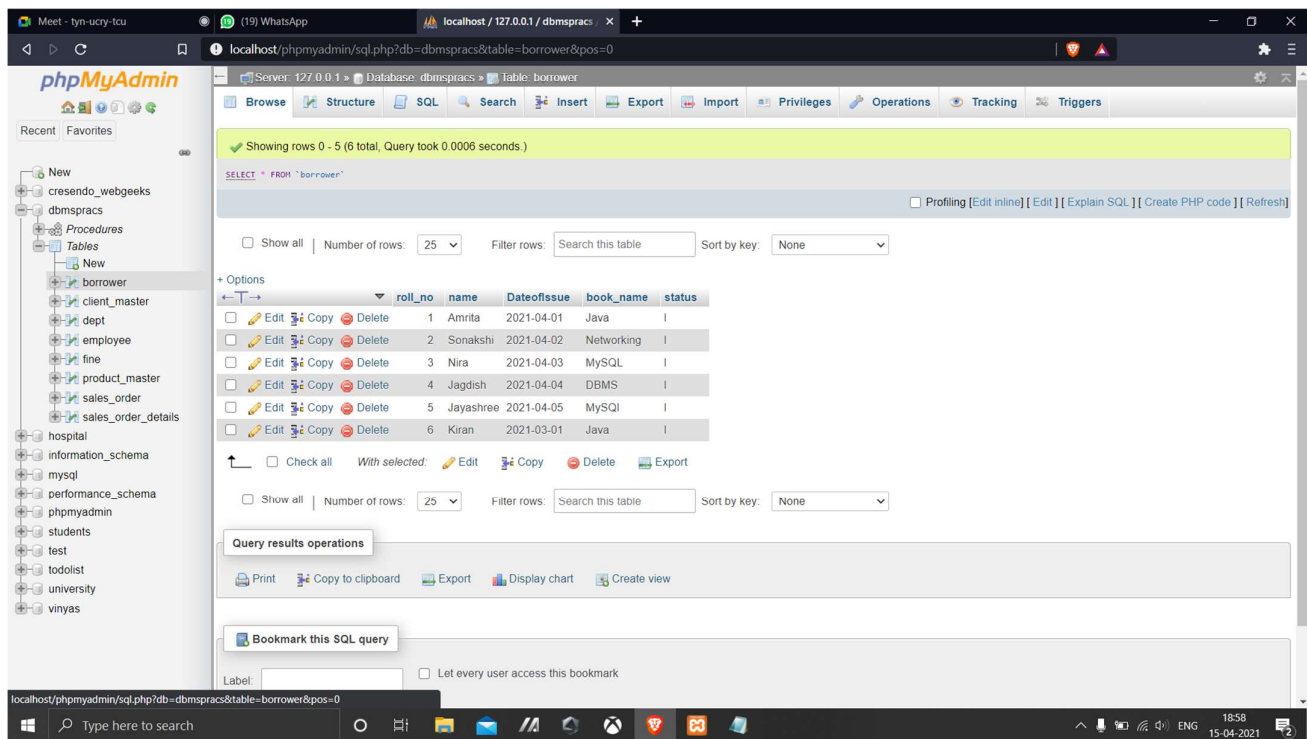
INSERT INTO fine VALUES(stud_roll_no,CURRENT_DATE,fine_amt);

END; //

DELIMITER ;

The screenshot shows the phpMyAdmin web interface. On the left, the database structure is visible, including tables like 'borrower', 'client_master', 'dept', 'employee', 'fine', 'product_master', 'sales_order', and 'sales_order_details'. The main panel displays the SQL query that was executed, which is the same procedure creation code shown in the text above. Below the query, there are buttons for 'Clear', 'Format', and 'Get auto-saved query'. There is also a checkbox for 'Bind parameters' and a text input for 'Bookmark this SQL query:'. At the bottom, a green message box states: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0117 seconds)'. The console at the very bottom shows the full SQL statement that was executed.

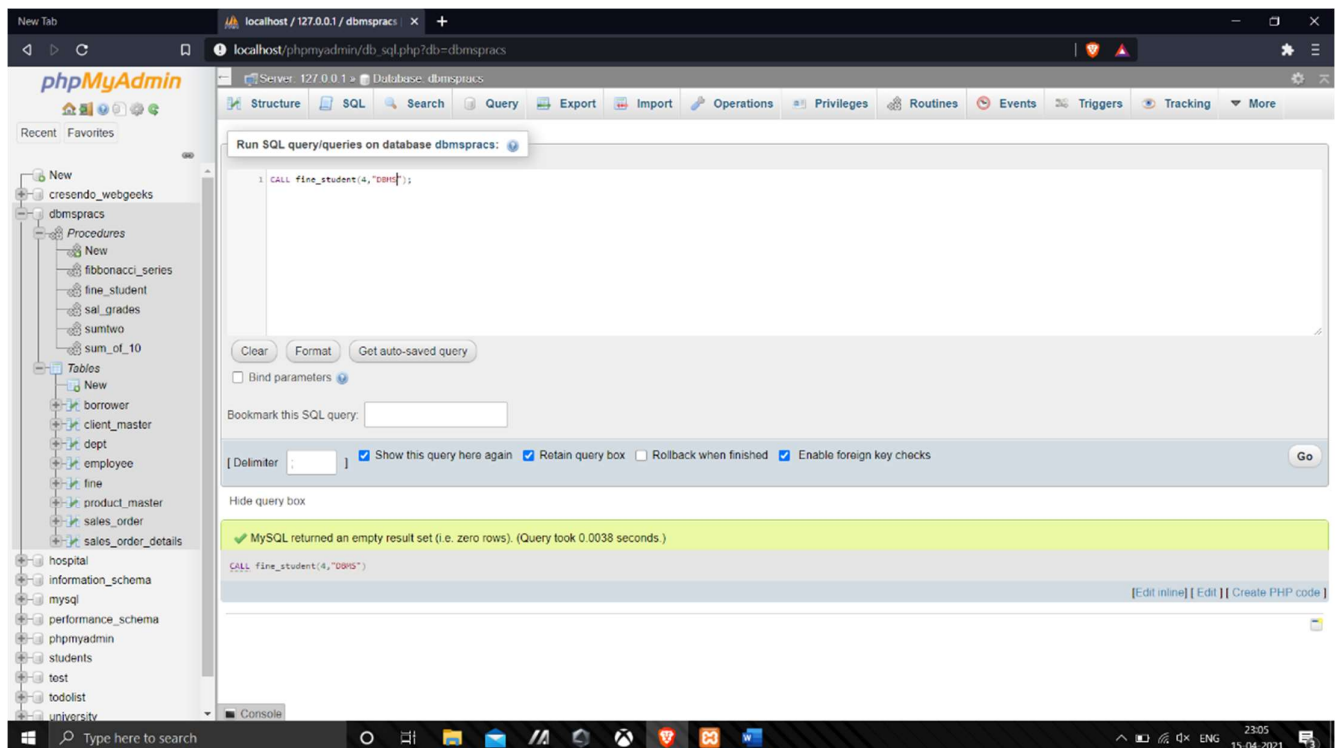
Before Calling fine_student procedure:



The screenshot shows the phpMyAdmin interface for the 'dbmspracs' database. The 'Tables' section is expanded, and the 'borrower' table is selected. The table structure is shown with columns: roll_no, name, Dateofissue, book_name, and status. The table contains 6 rows of data.

roll_no	name	Dateofissue	book_name	status
1	Amrita	2021-04-01	Java	I
2	Sonakshi	2021-04-02	Networking	I
3	Nira	2021-04-03	MySQL	I
4	Jagdish	2021-04-04	DBMS	I
5	Jayashree	2021-04-05	MySQL	I
6	Kiran	2021-03-01	Java	I

Calling fine_student procedure:



The screenshot shows the phpMyAdmin interface for the 'dbmspracs' database. The 'Routines' section is expanded, and the 'fine_student' procedure is selected. The 'Run SQL query/queries on database dbmspracs:' section is active, showing the execution of the procedure with the argument '4, "DBMS"'. The result shows that MySQL returned an empty result set (i.e. zero rows).

```
CALL fine_student(4, "DBMS");
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0038 seconds)

After Calling fine_student procedure:

The screenshot shows the phpMyAdmin interface with the 'borrower' table selected. The table contains 6 rows of data. The SQL query 'SELECT * FROM `borrower`' is displayed at the top. The table structure and query results are shown below.

roll_no	name	Dateofissue	book_name	status
1	Amrita	2021-04-01	Java	I
2	Sonakshi	2021-04-02	Networking	I
3	Nira	2021-04-03	MySQL	I
4	Jagdish	2021-04-04	DBMS	R
5	Jayashree	2021-04-05	MySQL	I
6	Kiran	2021-03-01	Java	I

The screenshot shows the phpMyAdmin interface with the 'fine' table selected. The table contains 1 row of data. The SQL query 'SELECT * FROM `fine`' is displayed at the top. The table structure and query results are shown below.

roll_no	Date_of_return	Amt
4	2021-04-15	0

Exp - 8 - Postlab

II Advantages of PL/SQL vs SQL

Ans

- (a) PL/SQL is a database programming language using SQL, while SQL is only database query language.
- (b) PL/SQL allows _____, declaration of data variable while SQL does not.
- (c) Control structures are available like, For loop, while loop whereas SQL does not support control structures.
- (d) PL/SQL block performs group of operations as single block whereas SQL performs single query operations.
- (e) It does not interact directly with the database server and PL/SQL is application oriented language in contrast SQL is Data oriented language and it directly interacts with database server.
- (f) PL/SQL is accustomed write program blocks, functions, procedures, triggers and packages. in contrast SQL is used to write queries, DDL and DML statements.

Q2 Explain data types of PL/SQL

Ans

1) NUMERIC DATATYPES:-

(a) INT

ANSI specifies integer type with maximum precision of 38 decimal digits

(b) FLOAT

ANSI and IBM specific floating point type with maximum precision of ¹²⁶ binary digits

2) CHARACTER DATATYPES:-

Varchar2:- Variable-length character string with maximum size of 32,767 bytes

3) DATETIME and INTERVAL DATATYPES:-

DAY

01 to 31 (valid by month & year)

MONTH

01 to 12

YEAR

- 4712 to 9999

LARGE OBJECT (LOB) DATATYPES

BLOB

Used to store large binary object upto 128 TB.