

+ - #ccc  
Step 1 of 6



## Hands-on Lab: Stored Procedures in MySQL using phpMyAdmin

**Estimated time needed:** 20 minutes

In this lab, you will learn how to create tables and load data in the MySQL database service using the phpMyAdmin graphical user interface (GUI) tool.

### Software Used in this Lab

In this lab, you will use [MySQL](#). MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.



To complete this lab you will utilize MySQL relational database service available as part of IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

### Database Used in this Lab

`mysql_learners` database has been used in this lab.

### Data Used in this Lab

The data used in this lab is internal data. You will be working on the **PETSALE** table.

| ID ▲ | ANIMAL   | SALEPRICE |
|------|----------|-----------|
| 1    | Cat      | 450.09    |
| 2    | Dog      | 666.66    |
| 3    | Parrot   | 50.00     |
| 4    | Hamster  | 60.60     |
| 5    | Goldfish | 48.48     |

This lab requires you to have the PETSALE table populated with sample data on mysql phpadmin interface. You might have created and populated a PETSALE table in a previous lab. But for this lab, it is recommended you download the `PETSALE-CREATE-v2.sql` script below, upload it to phpadmin console and run it. The script will create a new PETSALE table dropping any previous PETSALE table if exists, and will populate it with the required sample data.

- [PETSALE-CREATE-v2.sql](#)

### Objectives

After completing this lab, you will be able to:

- Create stored procedures
- Execute stored procedures

## Exercise 1

In this exercise, you will create and execute a stored procedure to read data from a table on mysql phpadmin using SQL.

1. Make sure you have created and populated the **PETSALE** table following the steps in the "**Data Used in this Lab**" section of this lab.

| ID ▲ | ANIMAL   | SALEPR |
|------|----------|--------|
| 1    | Cat      | 450.09 |
| 2    | Dog      | 666.66 |
| 3    | Parrot   | 50.00  |
| 4    | Hamster  | 60.60  |
| 5    | Goldfish | 48.48  |

2. \* You will create a stored procedure routine named **RETRIEVE\_ALL**.

- This **RETRIEVE\_ALL** routine will contain an SQL query to retrieve all the records from the PETSALE table, so you don't need to write the same query over and over again. You just call the stored procedure routine to execute the query everytime.
- To create the stored procedure routine, copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```
DELIMITER //
```

```
CREATE PROCEDURE RETRIEVE_ALL()
```

```
BEGIN
```

```
    SELECT * FROM PETSALE;
```

```
END //
```

```
DELIMITER ;
```

**Run SQL query/queries on database Mysql\_learners:**

```
1 DELIMITER //
2
3 CREATE PROCEDURE RETRIEVE_ALL()
4
5 BEGIN
6
7     SELECT * FROM PETALE;
8
9
10 END //
11
12 DELIMITER ;
```

Clear Format Get auto-saved query

☐ Bind parameters

[ Delimiter  ] ☐ Show this query here again ☐ Retain query box ☐ Rollback when finished ☒ Enable foreign key checks

Hide query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0064 seconds.)

CREATE PROCEDURE RETRIEVE\_ALL() BEGIN SELECT \* FROM PETALE; END

3. To call the RETRIEVE\_ALL routine, open another **SQL** tab by clicking **Open in new Tab**

The screenshot shows the phpMyAdmin web interface. On the left is a sidebar with a tree view of databases and tables. The main panel is titled 'Server: mysql.3306 » Database: HR » Table: EMPLOYEES'. It has tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', and 'Import'. The 'SQL' tab is active, and a right-click context menu is open over it. The menu options are: 'Open link in new tab' (highlighted with a red box), 'Open link in new window', 'Open link in incognito window', 'Save link as...', 'Copy link address', and 'Inspect'. Below the menu, the SQL query editor shows a single line: '1 SELECT \* FROM `EMPLOYEES`'. At the bottom of the editor are buttons for 'SELECT \*', 'SELECT', 'INSERT', 'UPDATE', 'DELETE', 'Clear', 'Format', and 'Get auto...'. There is also a checkbox for 'Bind parameters' and a section for query options including 'Delimiter' (set to semicolon), 'Show this query here again', 'Retain query box', and 'Rollback when finished'.

Delete the default line which appears so that you will get a blank window.

copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```
CALL RETRIEVE_ALL;
```

11 `CALL RETRIEVE_ALL;`

Clear Format Get auto-saved query

☐ Bind parameters

Delimiter `;` ☐ Show this query here again ☐ Retain query box ☐ Rollback when finished ☒ Enable foreign key checks

Hide query box

✓ Showing rows 0 - 4 (5 total, Query took 0.0010 seconds.)

`CALL RETRIEVE_ALL`

☐ Show all | Number of rows: 25  Filter rows:

Options

|   | ANIMAL   | SALEPRICE | SALEDATE   | QUANTITY |
|---|----------|-----------|------------|----------|
| 1 | Cat      | 450.09    | 2018-05-29 | 9        |
| 2 | Dog      | 666.66    | 2018-06-01 | 3        |
| 3 | Parrot   | 50.00     | 2018-06-04 | 2        |
| 4 | Hamster  | 60.60     | 2018-06-11 | 6        |
| 5 | Goldfish | 48.48     | 2018-06-14 | 24       |

4. You can view the created stored procedure routine RETRIEVE\_ALL. Click on the **Routines** and view the procedure.

Structure SQL Search Query Export Import Operations Privileges Routines

Routines

| Name   | Action                      | Type      | Returns |
|--|-----------------------------|-----------|---------|
| <input checked="" type="checkbox"/> RETRIEVE_ALL | Edit  Execute  Export  Drop | PROCEDURE |         |

☒ Check all With selected: Export Drop

New

Add routine

5. If you wish to drop the stored procedure routine RETRIEVE\_ALL, copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```
DROP PROCEDURE RETRIEVE_ALL;

CALL RETRIEVE_ALL;
```

The screenshot shows a SQL IDE interface with a menu bar (Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines) and a toolbar. The main editor contains the following SQL code:

```
1  
2 DROP PROCEDURE RETRIEVE_ALL;  
3  
4 CALL RETRIEVE_ALL;  
5  
6
```

Below the editor are buttons for "Clear", "Format", and "Get auto-saved query". There is a checkbox for "Bind parameters" and a "Delimiter" dropdown set to ";". At the bottom of the toolbar are checkboxes for "Show this query here again", "Retain query box", "Rollback when finished", and "Enable foreign key checks" (checked).

An error message is displayed in a pink box:

**Error**

SQL query: [Copy](#)

```
CALL RETRIEVE_ALL
```

MySQL said: ⓘ

#1305 - PROCEDURE Mysql\_learners.RETRIEVE\_ALL does not exist

## Exercise 2

In this exercise, you will create and execute a stored procedure to write/modify data in a table on Db2 using SQL.

1. Make sure you have created and populated the **PETSALE** table following the steps in the "**Data Used in this Lab**" section of this lab.

| ID ▲ | ANIMAL   | SALEPRICE |
|------|----------|-----------|
| 1    | Cat      | 450.09    |
| 2    | Dog      | 666.66    |
| 3    | Parrot   | 50.00     |
| 4    | Hamster  | 60.60     |
| 5    | Goldfish | 48.48     |

2. \* You will create a stored procedure routine named **UPDATE\_SALEPRICE** with parameters **AnimalID** and **Animal\_Health**.
- o This **UPDATE\_SALEPRICE** routine will contain SQL queries to update the sale price of the animals in the PETSale table depending on their health conditions, **BAD** or **WORSE**.
  - o This procedure routine will take animal ID and health condition as parameters which will be used to update the sale price of animal in the PETSale table by an amount depending on their health condition. Suppose -
    - For animal with ID XX having BAD health condition, the sale price will be reduced further by 25%.
    - For animal with ID YY having WORSE health condition, the sale price will be reduced further by 50%.
    - For animal with ID ZZ having other health condition, the sale price won't change.
- To create the stored procedure routine, copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```

DELIMITER @
CREATE PROCEDURE UPDATE_SALEPRICE (
  IN Animal_ID INTEGER, IN Animal_Health VARCHAR(5) )
BEGIN

  IF Animal_Health = 'BAD' THEN
    UPDATE PETSale
    SET SALEPRICE = SALEPRICE - (SALEPRICE * 0.25)
    WHERE ID = Animal_ID;

  ELSEIF Animal_Health = 'WORSE' THEN
    UPDATE PETSale
    SET SALEPRICE = SALEPRICE - (SALEPRICE * 0.5)
    WHERE ID = Animal_ID;

  ELSE
    UPDATE PETSale
    SET SALEPRICE = SALEPRICE
    WHERE ID = Animal_ID;

  END IF;

END @
DELIMITER ;

```

Server: MySQL5502 / Database: MySQL\_learners

Structure SQL Search Query Export Import Operations Privileges Routines

Run SQL query/queries on database MySQL\_learners:

```
15
16 ELSE
17     UPDATE PETALE
18     SET SALEPRICE = SALEPRICE
19     WHERE ID = Animal_ID;
20
21 END IF;
22
23 END @
24
25 DELIMITER ;
26
```

Clear Format Get auto-saved query

☐ Bind parameters

[ Delimiter ; ] ☐ Show this query here again ☐ Retain query box ☐ Rollback when finished ☒ Enable foreign key checks

Hide query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0214 seconds.)

```
CREATE PROCEDURE UPDATE_SALEPRICE ( IN Animal_ID INTEGER, IN Animal_Health VARCHAR(5) ) BEGIN IF Animal_Health = 'BAD'
(SALEPRICE * 0.25) WHERE ID = Animal_ID; ELSEIF Animal_Health = 'WORSE' THEN UPDATE PETALE SET SALEPRICE = SALEPRICE
PETALE SET SALEPRICE = SALEPRICE WHERE ID = Animal_ID; END IF; END
```

3. Let's call the UPDATE\_SALEPRICE routine. We want to update the sale price of animal with ID 1 having **BAD** health condition in the PETALE table. open another SQL tab by clicking **Open in new Tab**



The screenshot shows the phpMyAdmin web interface. On the left is the database navigation tree with 'HR' selected. The main panel shows the 'SQL' tab for the 'EMPLOYEES' table in the 'HR' database. A context menu is open over the 'SQL' tab, with the option 'Open link in new tab' highlighted. The SQL query editor contains the text: `1 SELECT * FROM `EMPLOYEES``. Below the query editor are buttons for 'SELECT \*', 'SELECT', 'INSERT', 'UPDATE', 'DELETE', 'Clear', 'Format', and 'Get auto...'. There is also a checkbox for 'Bind parameters' and a row of checkboxes for 'Show this query here again', 'Retain query box', and 'Rollback when finished'.

Delete the default line which appears so that you will get a blank window.

copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

Note if you have dropped RETREIVE\_ALL procedure rerun the creation script of that procedure before executing these lines.

```
CALL RETREIVE_ALL;  
CALL UPDATE_SALEPRICE(1, 'BAD');  
CALL RETREIVE_ALL;
```

✓ Showing rows 0 - 4 (5 total, Query took 0.0007 seconds.)

CALL RETRIEVE\_ALL

☐ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

| ID | ANIMAL   | SALEPRICE | SALEDATE   | QUANTITY |
|----|----------|-----------|------------|----------|
| 1  | Cat      | 450.09    | 2018-05-29 | 9        |
| 2  | Dog      | 666.66    | 2018-06-01 | 3        |
| 3  | Parrot   | 50.00     | 2018-06-04 | 2        |
| 4  | Hamster  | 60.60     | 2018-06-11 | 6        |
| 5  | Goldfish | 48.48     | 2018-06-14 | 24       |

Note: #1265

✓ Showing rows 0 - 4 (5 total, Query took 0.0007 seconds.)

CALL RETRIEVE\_ALL

☐ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

| ID | ANIMAL   |
|----|----------|
| 1  | Cat      |
| 2  | Dog      |
| 3  | Parrot   |
| 4  | Hamster  |
| 5  | Goldfish |

4. Let's call the UPDATE\_SALEPRICE routine once again. We want to update the sale price of animal with ID 3 having **WORSE** health condition in the PETSale table. copy the code below and paste it to the textarea of the SQL page. Click **Go**. You will have all the records retrieved from the PETSale table.

```
CALL RETRIEVE_ALL;
CALL UPDATE_SALEPRICE(3, 'WORSE');
CALL RETRIEVE_ALL;
```

CALL RETRIEVE\_ALL

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Options

| ID | ANIMAL   | SALEPRICE | SALEDATE   | QUANTITY |
|----|----------|-----------|------------|----------|
| 1  | Cat      | 337.57    | 2018-05-29 | 9        |
| 2  | Dog      | 666.66    | 2018-06-01 | 3        |
| 3  | Parrot   | 50.00     | 2018-06-04 | 2        |
| 4  | Hamster  | 60.60     | 2018-06-11 | 6        |
| 5  | Goldfish | 48.48     | 2018-06-14 | 24       |

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

CALL RETRIEVE\_ALL

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Options

| ID | ANIMAL   | SALEPRICE |
|----|----------|-----------|
| 1  | Cat      | 337.57    |
| 2  | Dog      | 666.66    |
| 3  | Parrot   | 25.00     |
| 4  | Hamster  | 60.60     |
| 5  | Goldfish | 48.48     |

☐ Show all | Number of rows: 25 | Filter rows: Search this table

5. You can view the created stored procedure routine UPDATE\_SALEPRICE. Click on the **Routines** and view the procedure.

StructureSQLSearchQueryExportImportOperationsPrivilegesRoutines

Routines

| Name                                      | Action                      | Type      | Returns |
|---|-----------------------------|-----------|---------|
| <input type="checkbox"/> RETRIEVE_ALL     | Edit  Execute  Export  Drop | PROCEDURE |         |
| <input type="checkbox"/> UPDATE_SALEPRICE | Edit  Execute  Export  Drop | PROCEDURE |         |

☐ Check all    With selected: Export Drop

New

Add routine

6. If you wish to drop the stored procedure routine UPDATE\_SALEPRICE, copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```
DROP PROCEDURE UPDATE_SALEPRICE;  
  
CALL UPDATE_SALEPRICE;
```



**Congratulations! You have completed this lab, and you are ready for the next topic.**

# Author(s)

[Lakshmi Holla](#)  
[Malika Singla](#)

# Changelog

| Date       | Version | Changed by                   | Change Description |
|------------|---------|------------------------------|--------------------|
| 2021-11-01 | 0.1     | Lakshmi Holla, Malika Singla | Initial Version    |

© IBM Corporation 2021. All rights reserved.

Continue