



# 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	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

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	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

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 PETSALE;
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
Go

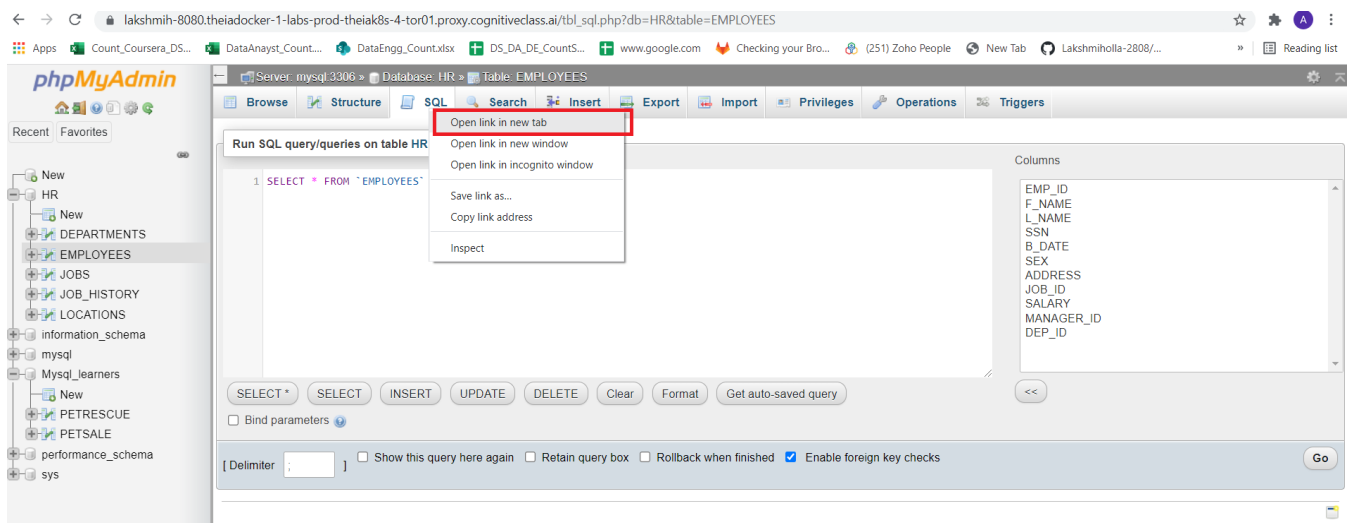
Hide query box

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

```
CREATE PROCEDURE RETRIEVE_ALL() BEGIN SELECT * FROM PETSALE; END
```

[Edit inline]
[Edit]
[Create PHP code]

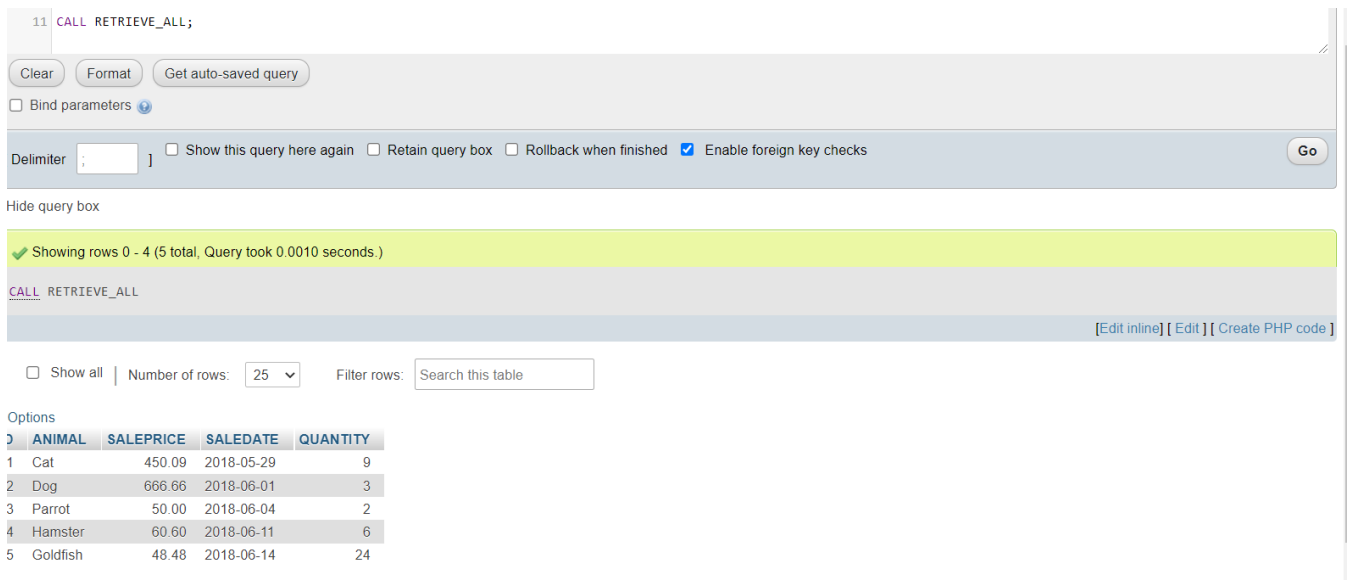
3. To call the RETRIEVE\_ALL routine, open another **SQL** tab by clicking **Open in new Tab**



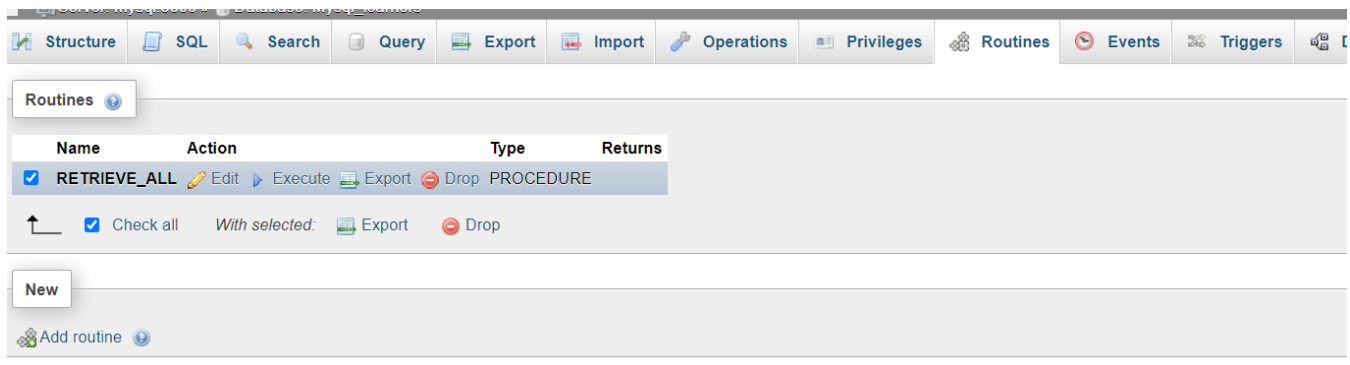
Delete the default line which appears so that you will get a blank window.

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

```
CALL RETRIEVE_ALL;
```



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



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. The top toolbar includes buttons for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, and Designer. The main query editor area contains the following SQL code:

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

Below the query editor, there are buttons for 'Clear', 'Format', and 'Get auto-saved query'. A checkbox for 'Bind parameters' is also present. At the bottom, there is a 'Go' button and a status bar with the following options: 'Show this query here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks' (checked).

An error message is displayed in a red box at the bottom of the IDE:

**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	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

2.
  - o You will create a stored procedure routine named **UPDATE\_SALEPRICE** with parameters **Animal\_ID** 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 PETALE
        SET SALEPRICE = SALEPRICE - (SALEPRICE * 0.25)
        WHERE ID = Animal_ID;

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

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

    END IF;

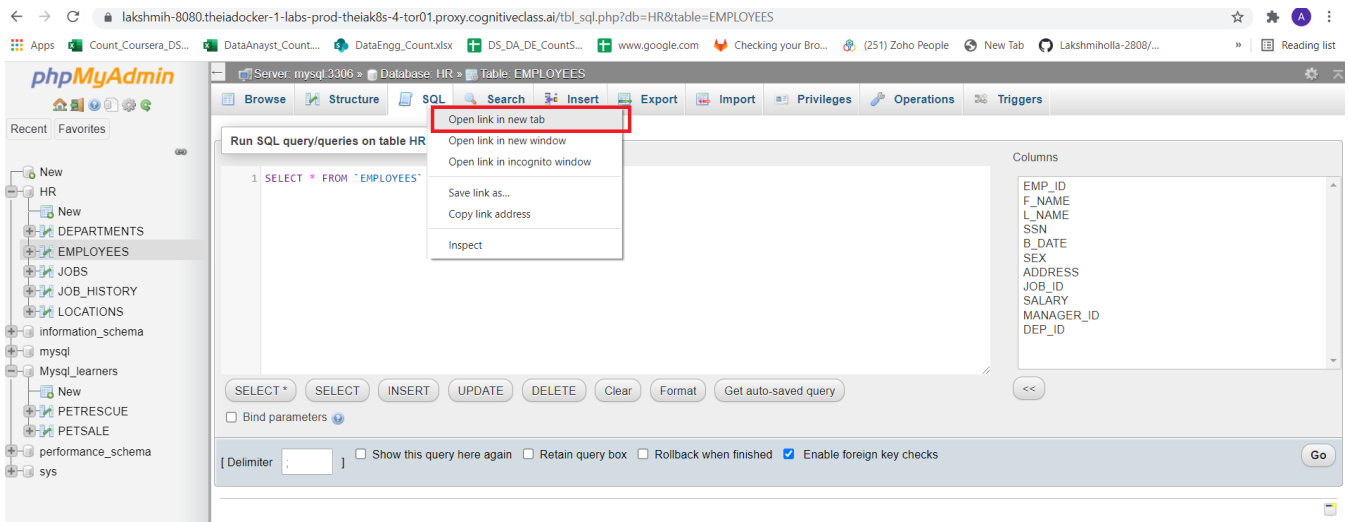
END @

DELIMITER ;

```

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with icons for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, and Designer. Below the toolbar, a tab is open titled "Run SQL query/queries on database Mysql\_learners:". The main area displays the SQL code from the previous block, with line numbers 15 to 26. Below the code editor, there are buttons for "Clear", "Format", and "Get auto-saved query". There's also a checkbox for "Bind parameters" which is unchecked. At the bottom, there's a status bar showing "[ Delimiter : ]" and several checkboxes: "Show this query here again" (unchecked), "Retain query box" (unchecked), "Rollback when finished" (unchecked), and "Enable foreign key checks" (checked). A "Go" button is on the right. Below the status bar, a green message box states: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0214 seconds.)". At the bottom, there's a detailed view of the executed query, showing the full CREATE PROCEDURE statement with syntax highlighting. Links for "[Edit inline]", "[Edit]", and "[Create PHP code]" are at the bottom right.

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**



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 RETRIEVE_ALL;

CALL UPDATE_SALEPRICE(1, 'BAD');

CALL RETRIEVE_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 Data truncated for column 'SALEPRICE' at row 1

Showing rows 0 - 4 (5 total, Query took 0.0015 seconds.)

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

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

Showing rows 0 - 4 (5 total, Query took 0.0005 seconds.)

CALL RETRIEVE\_ALL

Options

	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

Options

	ANIMAL	SALEPRICE	SALEDATE	QUANTITY
1	Cat	337.57	2018-05-29	9
2	Dog	666.66	2018-06-01	3
3	Parrot	25.00	2018-06-04	2
4	Hamster	60.60	2018-06-11	6
5	Goldfish	48.48	2018-06-14	24

Query results operations

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

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Designer

Routines

Name	Action	Type	Returns
RETRIEVE_ALL	Edit Execute Export Drop	PROCEDURE	
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;
```

```
7
8
9 DROP PROCEDURE UPDATE_SALEPRICE;
10
11 CALL UPDATE_SALEPRICE;
```

Clear Format Get auto-saved query

Bind parameters

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

Hide query box

**Error**

SQL query: Copy

```
DROP PROCEDURE UPDATE_SALEPRICE
```

MySQL said:

```
#1305 - PROCEDURE Mysql_learners.UPDATE_SALEPRICE does not exist
```

**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
2021-08-09	0.2	Sathya Priya	Updated HTML tags and SQL link
2021-11-01	0.1	Lakshmi Holla, Malika Singla	Initial Version

© IBM Corporation 2021. All rights reserved.