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

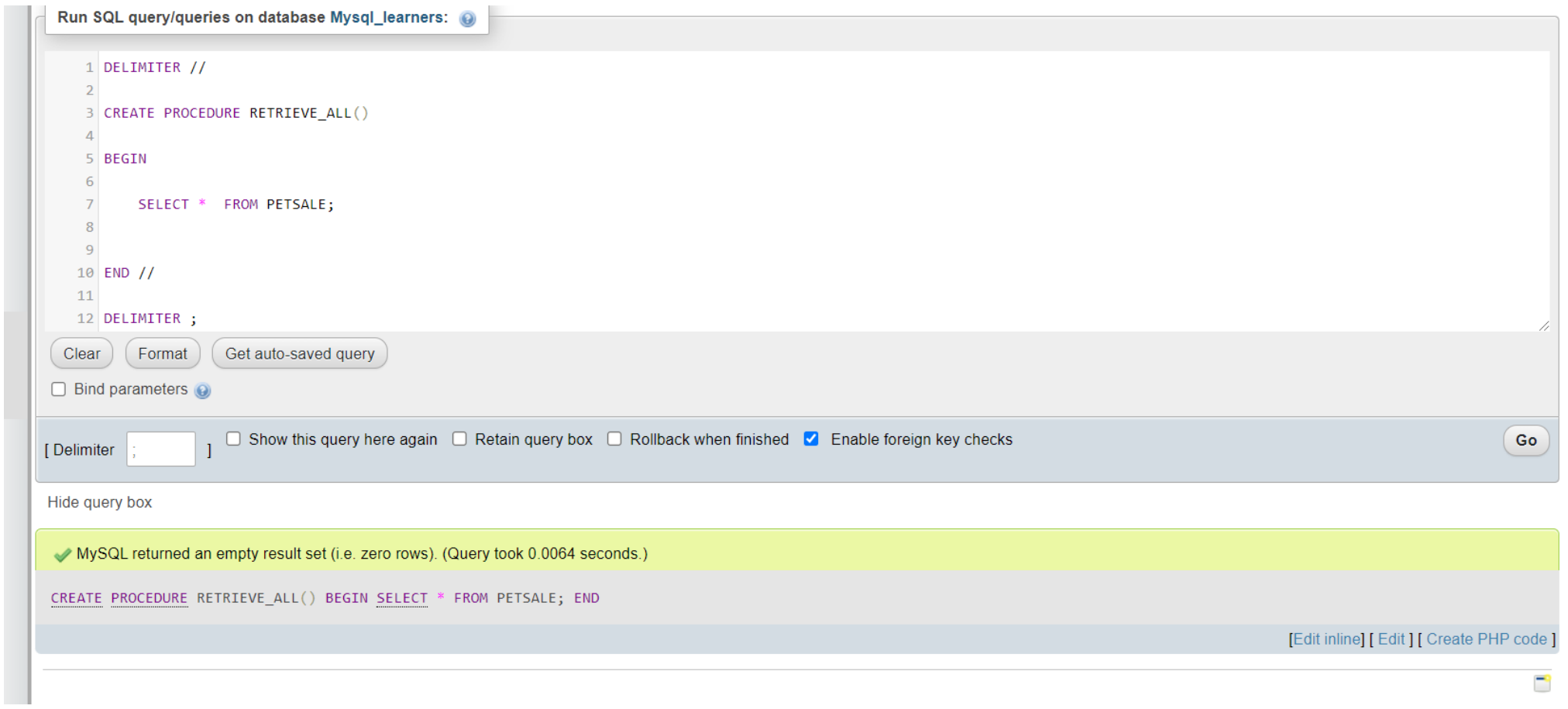
In this exercise, you will create and execute a stored procedure to read data from a table on mysql phpadmin using SQL.

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.

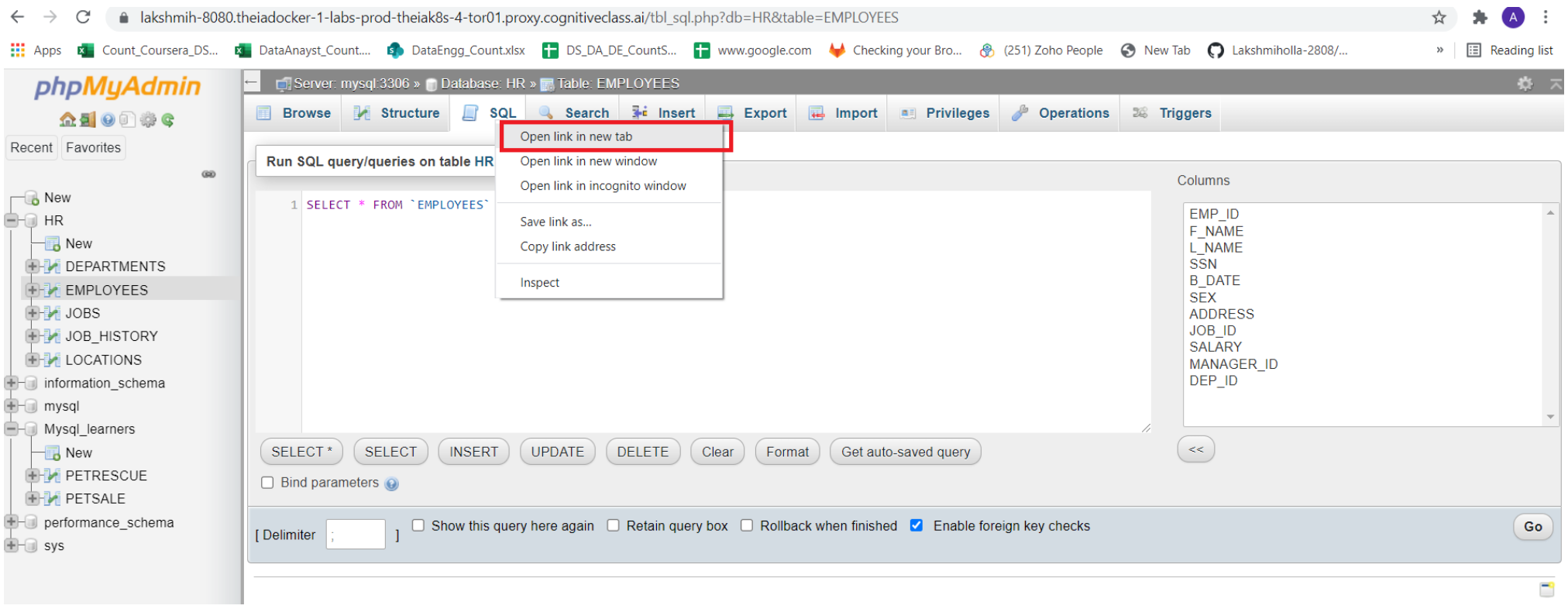
ID	ANIMAL	SALEPRICE	SALEDATE	QUANTITY
1	Cat	450.09	2018-05-29	9
2	Dog	666.66	2018-06-01	3
3	Parrot	50.00	2018-06-04	2
4	Hamster	60.60	2018-06-11	6
5	Goldfish	48.48	2018-06-14	24

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 ;
```



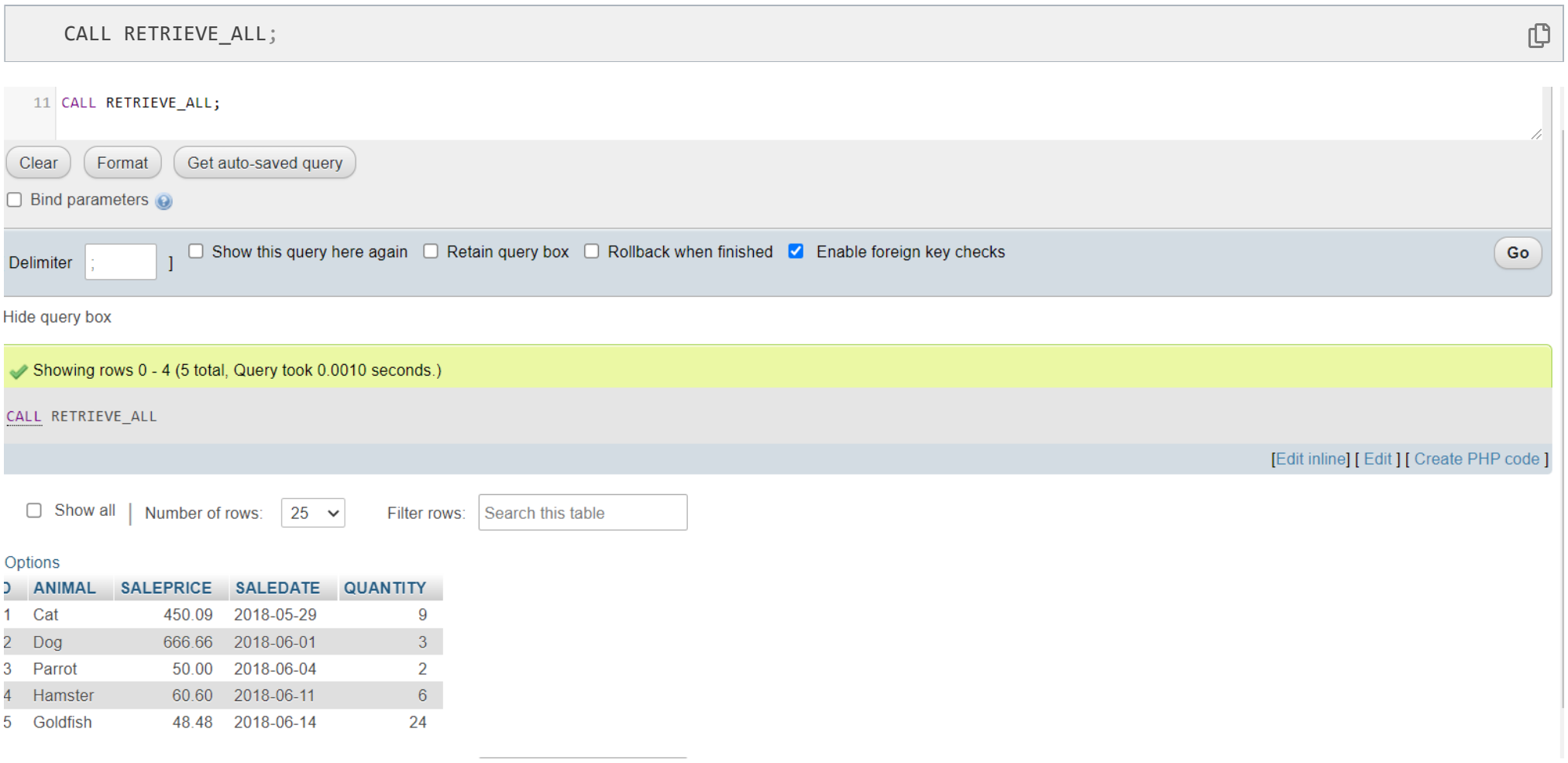
3. To call the RETRIEVE_ALL routine, open another **SQL** tab by clicking **Open in new Tab**



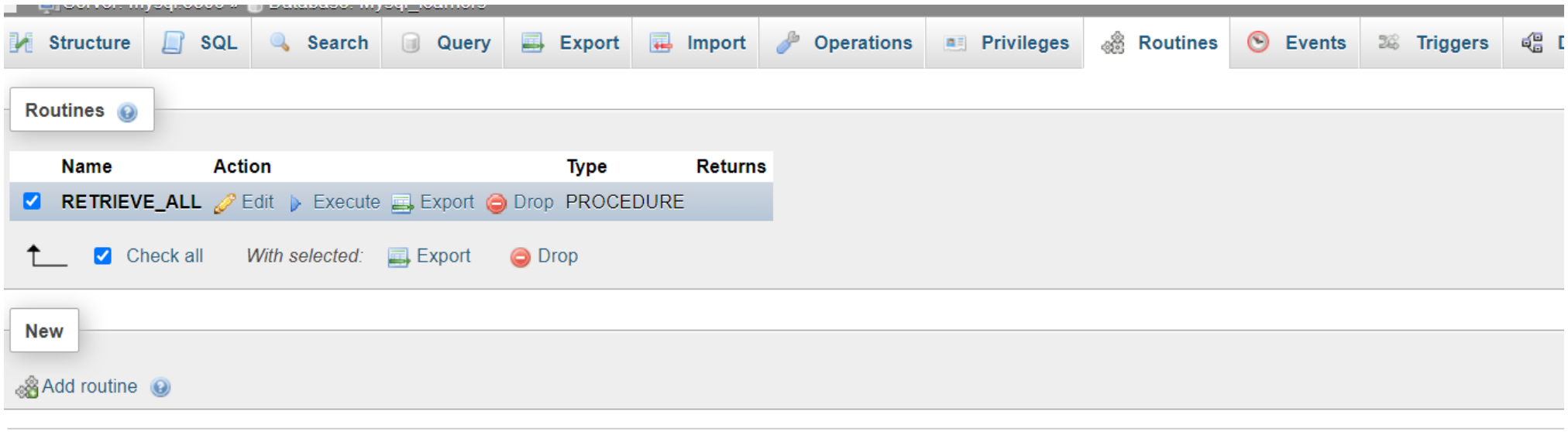
Delete the default line which appears so that you will get a blank window.

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;
```

Exercise 2

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

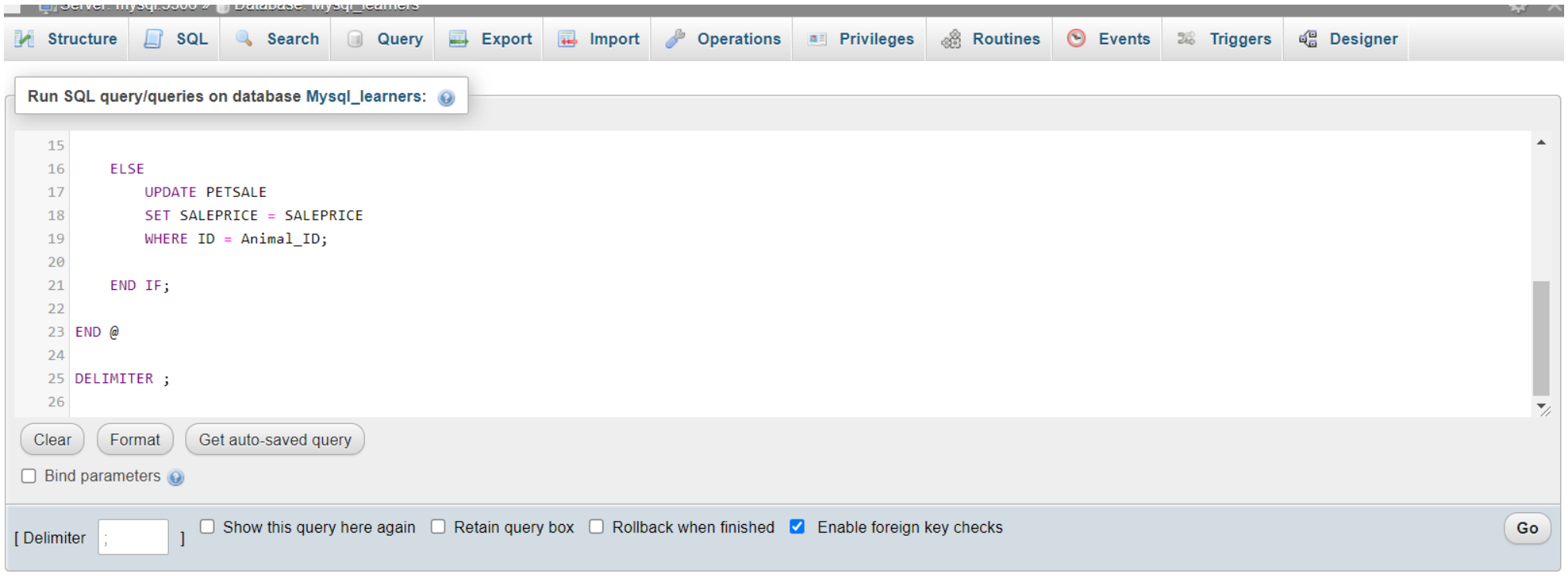
1. Make sure you have created and populated the **PETSALE** table following the steps in the **"Data Used in this Lab"** section of this lab.

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

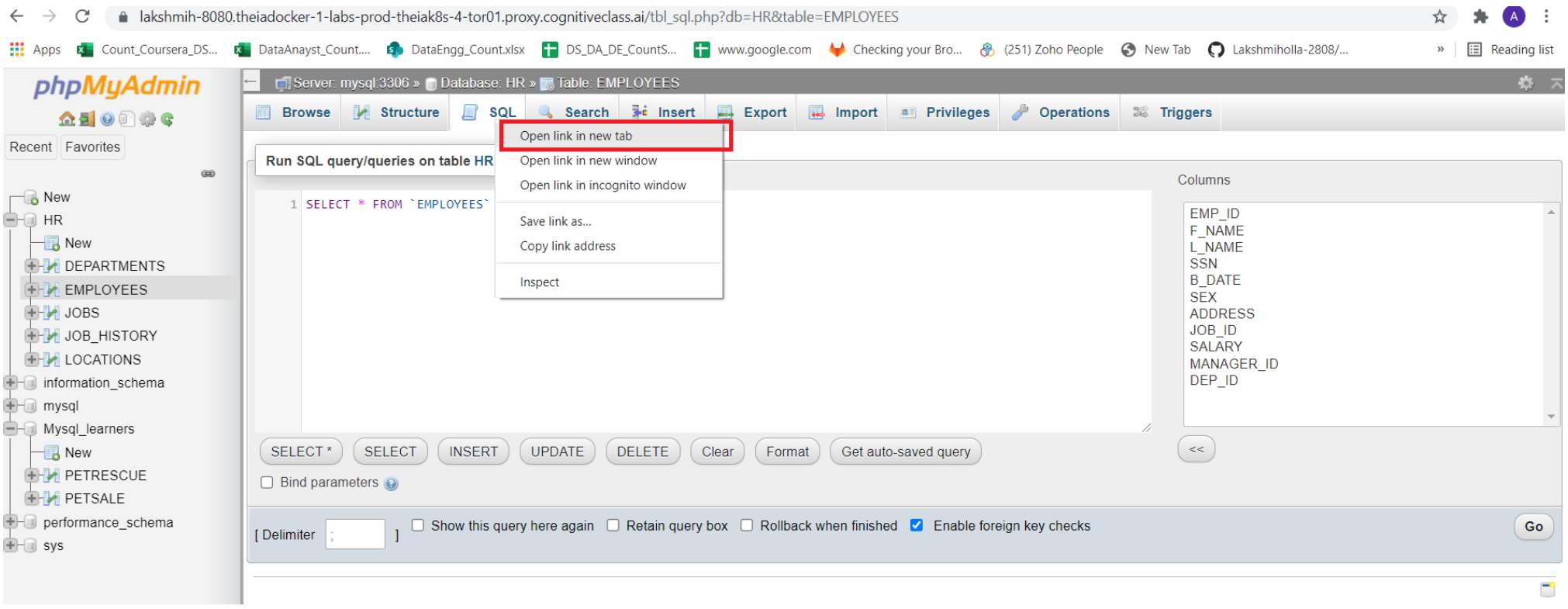
2.
    - You will create a stored procedure routine named **UPDATE_SALEPRICE** with parameters **Animal_ID** and **Animal_Health**.
    - This **UPDATE_SALEPRICE** routine will contain SQL queries to update the sale price of the animals in the PETSALE table depending on their health conditions, **BAD** or **WORSE**.
    - This procedure routine will take animal ID and health conditon 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 ;
```



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 PETSALE table. open another **SQL** tab by clicking **Open in new Tab**

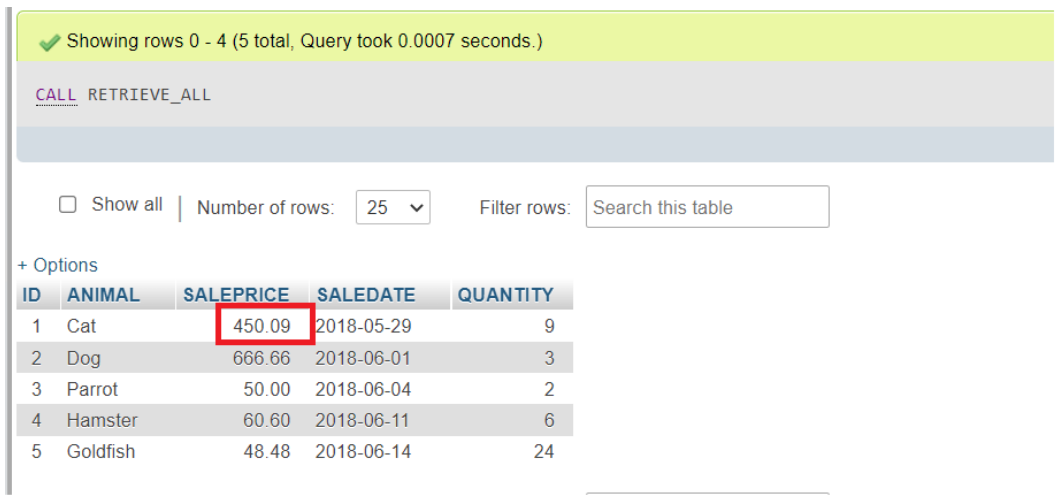


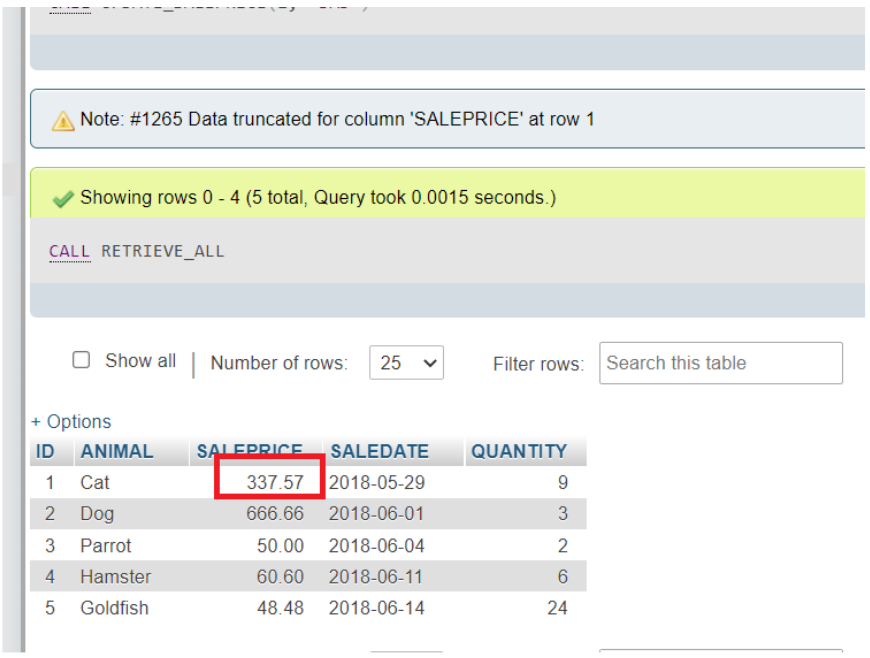
Delete the default line which appears so that you will get a blank window.

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;
```





)

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

	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

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

```
CALL RETRIEVE_ALL
```

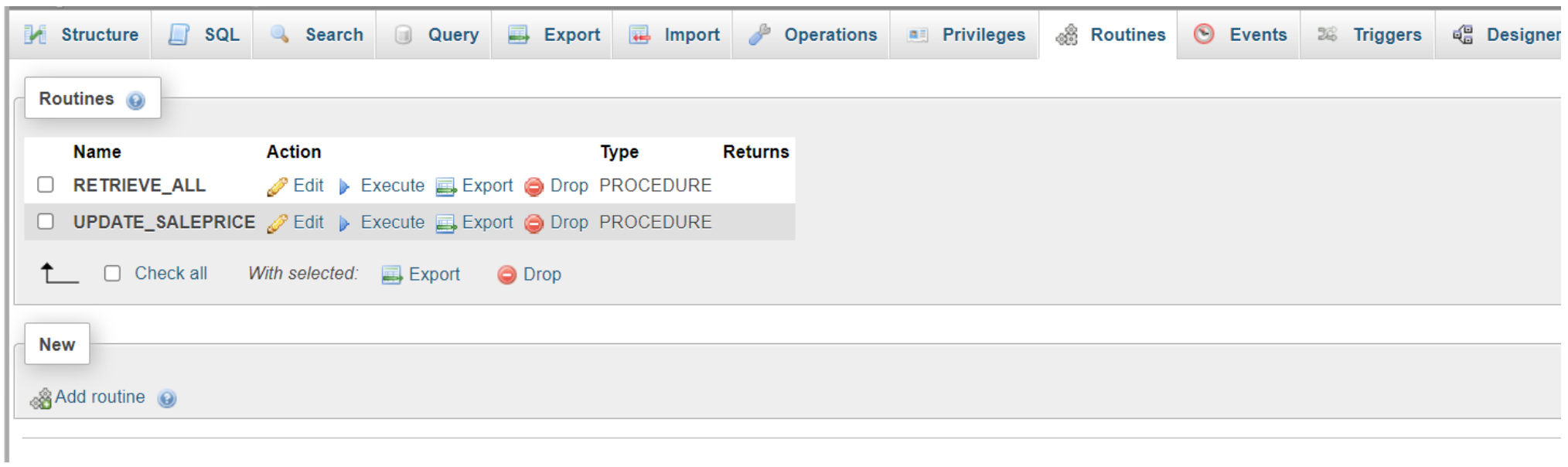
Show all | Number of rows: 25 Filter rows: Search this table

Options

D	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

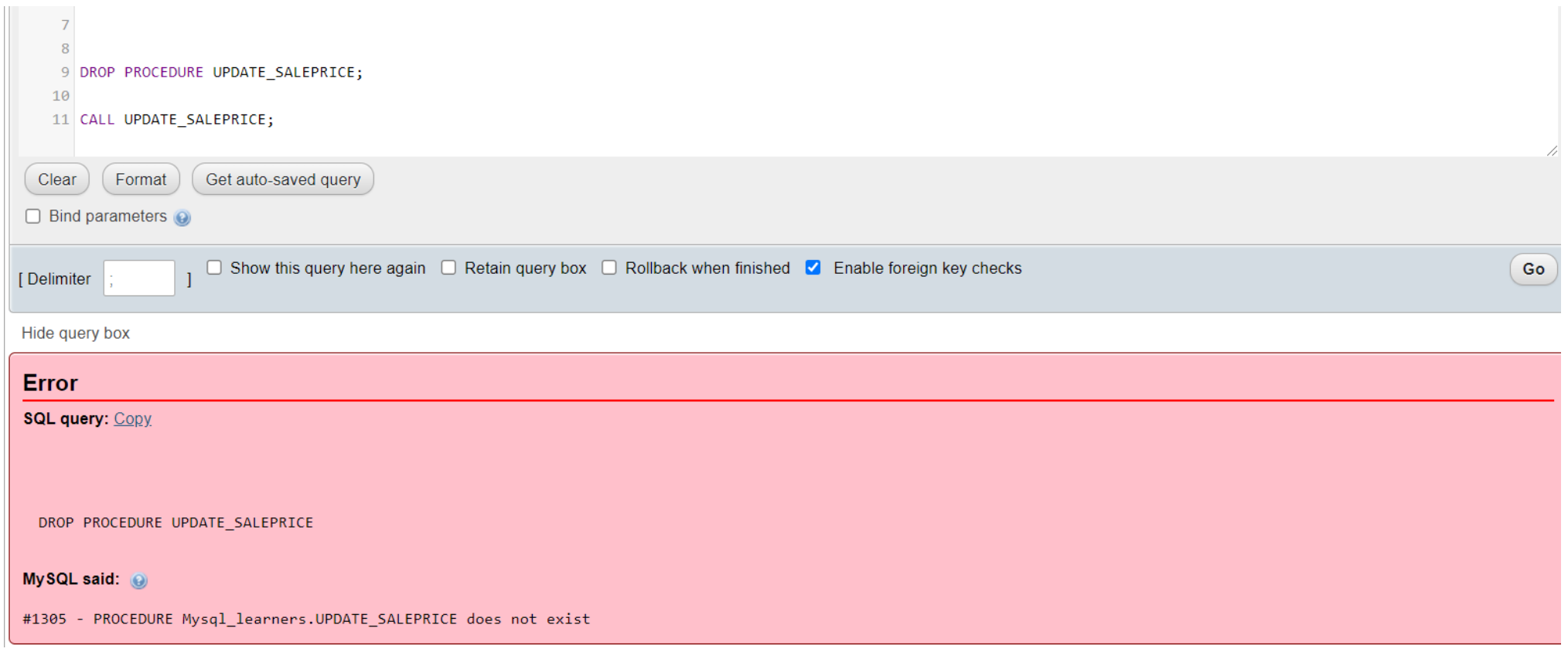
Show all | Number of rows: 25 Filter rows: Search this table

5. You can view the created stored procedure routine UPDATE_SALEPRICE. Click on the **Routines** and view the procedure.



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