

Hands-on Lab: Stored Procedures



**Skills
Network**

Estimated time needed: 20 minutes

Stored Procedures in SQL are a type of database object that allow you to encapsulate a series of SQL statements into a single routine. They are stored in the database data dictionary and can be invoked from an application program or from the database command interface. Stored procedures can accept input parameters and return multiple values of output parameters. They can also include control-of-flow constructs such as loops and conditional statements. Stored procedures offer several benefits including improved performance, higher productivity, ease of use, and increased scalability. They also provide a mechanism for enforcing business rules and data integrity in the database system.

Objectives

After completing this lab, you will be able to:

- Create stored procedures
- Execute stored procedures

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.

For this lab, you need to create a database PETS in the phpMyAdmin interface. Download the PETS-~~CREATE~~-v2.sql script below, upload it to console under the PETS database. Upon execution, the script will create a new PETS-~~CREATE~~ table dropping any previous PETS-~~CREATE~~ table if exists, and will populate it with the required sample data.

- [PETS-~~CREATE~~-v2.sql](#)

Stored Procedure: 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. 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 PETS-~~CREATE~~ 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**.


```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8

1. DELIMITER //
2.
3. CREATE PROCEDURE RETRIEVE_ALL()
4.
5. BEGIN
6.     SELECT * FROM PETS-CREATE;
7. END //
8. DELIMITER ;
```

Copied!

Run SQL query/queries on database Mysql_learners: 

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

☐ 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 PETS-CREATE; END
```

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

The screenshot shows the phpMyAdmin web interface. The left sidebar displays a tree view of databases and tables, including 'HR', 'DEPARTMENTS', 'EMPLOYEES', 'JOBS', 'JOB_HISTORY', 'LOCATIONS', 'information_schema', 'mysql', 'Mysql_learners', 'PETRESCUE', 'PETALE', 'performance_schema', and 'sys'. The main panel shows the 'SQL' tab for the 'EMPLOYEES' table. A context menu is open over the 'SQL' tab, with the option 'Open link in new tab' highlighted. The SQL query editor contains the query: `SELECT * FROM `EMPLOYEES``. Below the query editor, there are buttons for 'SELECT *', 'SELECT', 'INSERT', 'UPDATE', 'DELETE', 'Clear', 'Format', and 'Get auto-s'. There is also a checkbox for 'Bind parameters' and a section for query options: '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**.

```
1. 1  
1. CALL RETRIEVE_ALL;
```


Copied!

```
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

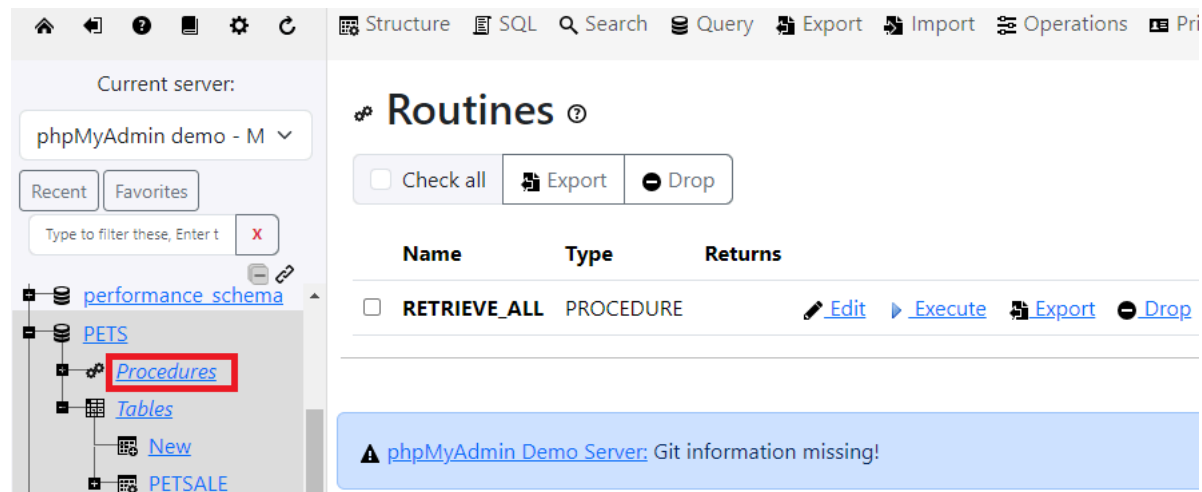
Filter rows:

Search this table

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

3. You can view the created stored procedure routine RETRIEVE_ALL. On the left panel, expand the **PETS** database option and click on **Procedures** to view the procedure.



Current server: phpMyAdmin demo - M

Recent Favorites

Type to filter these, Enter t

performance_schema

PETS



Procedures





Tables


New

PETSALE

Routines

☐ Check all
  Export
  Drop

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

 phpMyAdmin Demo Server: Git information missing!

4. 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**.

```
1. 1
2. 2
3. 3

1. DROP PROCEDURE RETRIEVE_ALL;
2.
3. CALL RETRIEVE_ALL;
```

Copied!

Structure

SQL

Search

Query

Export

Import

Operations

Privileges

Routines

1

2

3

4

5

6

DROP PROCEDURE RETRIEVE_ALL;

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

Error

SQL query: [Copy](#)

CALL RETRIEVE_ALL

MySQL said:

#1305 - PROCEDURE Mysql_learners.RETRIEVE_ALL does not exist

Stored Procedure: Exercise 2

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

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 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 that:
 - 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**.

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19

```

```

1. DELIMITER @
2. CREATE PROCEDURE UPDATE_SALEPRICE (IN Animal_ID INTEGER, IN Animal_Health VARCHAR(5))
3. BEGIN
4.     IF Animal_Health = 'BAD' THEN
5.         UPDATE PETALE
6.         SET SALEPRICE = SALEPRICE - (SALEPRICE * 0.25)
7.         WHERE ID = Animal_ID;
8.     ELSEIF Animal_Health = 'WORSE' THEN
9.         UPDATE PETALE
10.        SET SALEPRICE = SALEPRICE - (SALEPRICE * 0.5)
11.        WHERE ID = Animal_ID;
12.    ELSE
13.        UPDATE PETALE
14.        SET SALEPRICE = SALEPRICE
15.        WHERE ID = Animal_ID;
16.    END IF;
17. END @
18.
19. DELIMITER ;

```

Copied!

Server: mysql:5.6.0 / 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' THEN
(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

```

- 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 'Open link in new tab' highlighted. The SQL query editor contains the query: `1 SELECT * FROM `EMPLOYEES``. Below the query editor are buttons for 'SELECT *', 'SELECT', 'INSERT', 'UPDATE', 'DELETE', 'Clear', 'Format', and 'Get auto-s'. There is also a checkbox for 'Bind parameters' and a section for query options: '[Delimiter ;]', '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.

```

1. 1
2. 2
3. 3
4. 4
5. 5

1. CALL RETRIEVE_ALL;
2.
3. CALL UPDATE_SALEPRICE(1, 'BAD');
4.
5. CALL RETRIEVE_ALL;

```

Copied!

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

CALL RETRIEVE_ALL

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

+ 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

The screenshot shows the application interface with the following elements:

- A blue box containing a yellow warning triangle icon and the text "Note: #1265 D".
- A green box containing a green checkmark icon and the text "Showing rows".
- A purple text label "CALL" followed by the text "RETRIEVE_A".
- A checkbox labeled "Show all" with a vertical line to its right.
- A section header "+ Options".
- A table with the following structure:

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

2. 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.

1. 1
2. 2
3. 3
4. 4
5. 5

```

1.  CALL RETRIEVE_ALL;
2.
3.  CALL UPDATE_SALEPRICE(3, 'WORSE');
4.
5.  CALL RETRIEVE_ALL;

```

Copied!

CALL RETRIEVE_ALL

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

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:

Query results operations

Showing rows 0 - 4 (5 total, Query results)

CALL RETRIEVE_ALL

☐ Show all | Number of rows:

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:

3. 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

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

4. 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**.

1. 1
 2. 2
 3. 3
1. DROP PROCEDURE UPDATE_SALEPRICE;
 - 2.
 3. CALL UPDATE_SALEPRICE;

Copied!

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

Hide query box

Error

SQL query: [Copy](#)

```
DROP PROCEDURE UPDATE_SALEPRICE
```

MySQL said:

#1305 - PROCEDURE Mysql_learners.UPDATE_SALEPRICE does not exist

Conclusion

Congratulations! You have completed this lab on creating stored procedures in MySQL.

You are now able to:

- Write a stored procedure as per requirement
- Call or Execute a stored procedure

- Drop a stored procedure once its utility is over

Author(s)

[Lakshmi Holla](#)

[Malika Singla](#)

[Abhishek Gagneja](#)

Changelog

Date	Version	Changed by	Change Description
2023-10-31	0.4	Mercedes Schneider	QA Edits
2023-10-16	0.3	Abhishek Gagneja	Updated the instructions
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 2023. All rights reserved.