



# Hands-on Lab: Stored Procedures

**Estimated time needed:** 10 minutes

In this lab, you will create and execute stored procedures on IBM Db2 using SQL. A stored procedure is a set of SQL statements that are stored and executed on the database server. So instead of sending multiple SQL statements from the client to the server, you encapsulate them in a stored procedure on the server and send one statement from the client to execute them. Also, stored procedures can be useful if you have an SQL query that you write over and over again. You can save it as a stored procedure, and then just call it to execute it. In stored procedures, you can also pass parameters so that a stored procedure can act based on the passed parameter values.

## Software Used in this Lab

In this lab, you will use an [IBM Db2 Database](#). Db2 is a Relational Database Management System (RDBMS) from IBM, designed to store, analyze and retrieve data efficiently.

To complete this lab you will utilize a Db2 database service on IBM Cloud. If you did not already complete this lab task earlier in this module, you will not yet have access to Db2 on IBM Cloud, and you will need to follow the lab below first:

- [Hands-on Lab : Sign up for IBM Cloud, Create Db2 service instance and Get started with the Db2 console](#)

## 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 Db2. 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 Db2 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](#)

Please go through the lab below to learn how to upload and run a script on Db2 console (for this case, you need don't need to know anything else other than how to upload and run a script):

- [Hands-on Lab : Create tables using SQL scripts and Load data into tables](#)

# Objectives

After completing this lab, you will be able to:

- Create stored procedures
- Execute stored procedures

## Instructions

When you approach the exercises in this lab, follow the instructions to run the queries on Db2:

- Go to the [Resource List](#) of IBM Cloud by logging in where you can find the Db2 service instance that you created in a previous lab under **Services** section. Click on the **Db2-xx service**. Next, open the Db2 Console by clicking on **Open Console** button. Click on the 3-bar menu icon in the top left corner and go to the **Run SQL** page. The Run SQL tool enables you to run SQL statements.
  - If needed, follow [Hands-on Lab : Sign up for IBM Cloud, Create Db2 service instance and Get started with the Db2 console](#)

## Exercise 1

In this exercise, you will create and execute a stored procedure to read data from 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 **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 textbox of the **Run SQL** page. Click **Run all**.

```

--#SET TERMINATOR @
CREATE PROCEDURE RETRIEVE_ALL      -- Name of this stored procedure routine

LANGUAGE SQL                      -- Language used in this routine
READS SQL DATA                  -- This routine will only read data from the table

DYNAMIC RESULT SETS 1            -- Maximum possible number of result-sets to be returned to the caller
query

BEGIN

    DECLARE C1 CURSOR              -- CURSOR C1 will handle the result-set by retrieving records row by row
from the table
    WITH RETURN FOR                -- This routine will return retrieved records as a result-set to the
caller query

    SELECT * FROM PETSALE;          -- Query to retrieve all the records from the table

    OPEN C1;                      -- Keeping the CURSOR C1 open so that result-set can be returned to the
caller query

END
@                                -- Routine termination character

```

Result - Dec 16, 2020 7:...

✓ CREATE PROCEDURE RETRIEVE\_All -- Name of this stored procedure routine ... Run time: 0.052 s

Status: **Success** | Affected Rows: 0

- To call the RETRIEVE\_ALL routine, copy the code below in a **new blank script** and paste it to the textbox of the **Run SQL** page. Click **Run all**. You will have all the records retrieved from the PETSALE table.

```
CALL RETRIEVE_ALL;      -- Caller query
```

Result set 1

| 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       |

- You can view the created stored procedure routine RETRIEVE\_ALL. Click on the 3-bar menu icon in the top left corner and click **EXPLORE > APPLICATION OBJECTS > Stored Procedures**. Find the procedure routine RETRIEVE\_ALL from Procedures by clicking **Select All**. Click on the procedure routine **RETRIEVE\_ALL**.

IBM Db2 on Cloud

Storage: 22%

Cookie Preferences

Discover

STORED PROCEDURES

Filter by schema name or procedure name

Select All

New implicit schema

AUDIT 2 procedures

ZJH17769 2 procedures

DB2INST1 1 procedure

ERRORSCHEMA 0 procedure

SQL74605 0 procedure

ST\_INFORMTN\_SCHEMA 0 procedure

Procedures

| NAME            | SCHEMA   | PROPERTIES |
|-----------------|----------|------------|
| CONNECT_CHE...  | DB2INST1 | ...        |
| LOAD            | AUDIT    | ...        |
| RETRIEVE_ALL    | ZJH17769 | ...        |
| UPDATE          | AUDIT    | ...        |
| UPDATE_SALEP... | ZJH17769 | ...        |

Procedure Parameters

RETRIEVE\_ALL

```

CREATE PROCEDURE RETRIEVE_All
-- Name of this stored procedure routine

LANGUAGE SQL
-- Language used in this routine
READS SQL DATA
-- This routine will only read data from
table

DYNAMIC RESULT SETS 1
-- Maximum possible number of result-sets
to be returned to the caller

BEGIN

DECLARE C1 CURSOR
-- CURSOR C1 will handle the result-set by
retrieving records row by row from the table
WITH RETURN FOR
-- This routine will return retrieved
records as result-set to the caller

SELECT * FROM PETSale;
-- Query to retrieve all the records from
the table

OPEN C1;
-- Keeping the CURSOR C1 open so that
result-set can be returned to the caller

END

```

PARAMETER

DATA TYPE

MODE

LENGTH

SCALE

LOCAT

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

**DROP PROCEDURE RETRIEVE\_ALL;**

**CALL RETRIEVE\_ALL;**

1 DROP PROCEDURE RETRIEVE\_All;

2

3 CALL RETRIEVE\_ALL;

4

Syntax assistant

Result - Dec 16, 2020 5:...

✓ DROP PROCEDURE RETRIEVE\_All

Run time: 0.023 s

Status: Success | Affected Rows: 0

✗ CALL RETRIEVE\_ALL

Run time: 0.008 s

Status: Failed

**Error message**  
 No authorized routine named "RETRIEVE\_ALL" of type "PROCEDURE" having compatible arguments was found.  
 SQLCODE=-440, SQLSTATE=42884, DRIVER=4.26.14  
[Learn more about this error](#)

## Exercise 2

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

- 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       |

- 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 -
    - 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 textbox of the **Run SQL** page. Click **Run all**.

```
--#SET TERMINATOR @
CREATE PROCEDURE UPDATE_SALEPRICE (
    IN Animal_ID INTEGER, IN Animal_Health VARCHAR(5) )    -- ( { IN/OUT type } { parameter-name } { data-
type }, ... )

LANGUAGE SQL                                           -- Language used in this routine
MODIFIES SQL DATA                                     -- This routine will only write/modify data in
the table

BEGIN

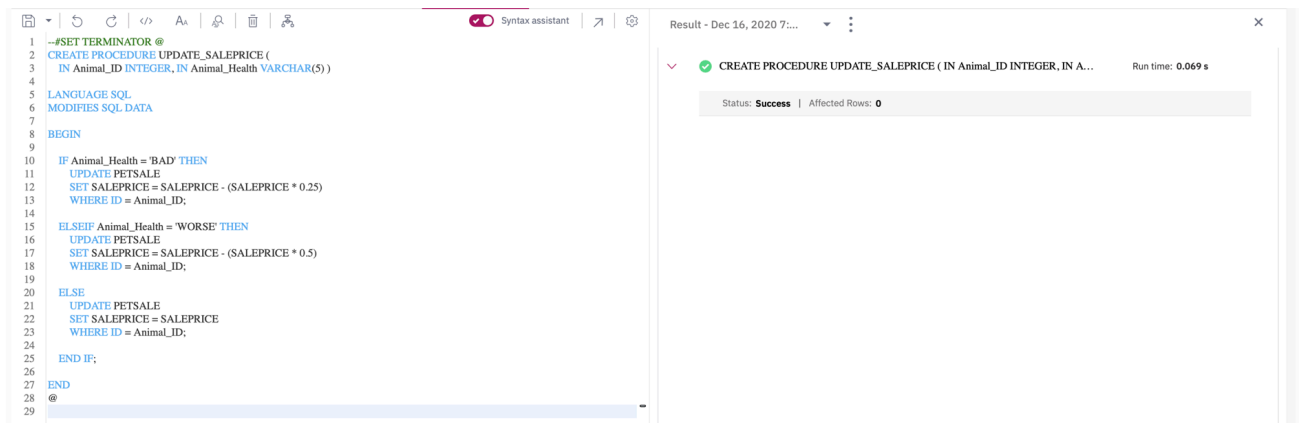
    IF Animal_Health = 'BAD' THEN                        -- Start of conditional statement
        UPDATE PETALE
        SET SALEPRICE = SALEPRICE - (SALEPRICE * 0.25)
        WHERE ID = Animal_ID;

    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 of conditional statement

END
@                                                         -- Routine termination character
```



The screenshot shows a SQL IDE interface. On the left, the SQL code for the `UPDATE_SALEPRICE` procedure is displayed with line numbers 1 through 29. The code includes comments and syntax highlighting. On the right, the 'Result' pane shows the execution status: 'CREATE PROCEDURE UPDATE\_SALEPRICE ( IN Animal\_ID INTEGER, IN A...' with a run time of 0.069 s. Below this, it indicates 'Status: Success' and 'Affected Rows: 0'.

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. Copy the code below in a **new blank script** and paste it to the textbox of the **Run SQL** page. Click **Run all**. You will have all the records retrieved from the `PETALE` table.

```
CALL RETRIEVE_ALL;

CALL UPDATE_SALEPRICE(1, 'BAD');    -- Caller query

CALL RETRIEVE_ALL;
```

SQL IDE Interface showing a query execution result.

Query: `CALL RETRIEVE_ALL;`

Result set 1:

| 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       |

Run time: 0.027 s

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 textbox of the **Run SQL** page. Click **Run all**. You will have all the records retrieved from the `PETSALE` table.

```
CALL RETRIEVE_ALL;
```

```
CALL UPDATE_SALEPRICE(3, 'WORSE');    -- Caller query
```

```
CALL RETRIEVE_ALL;
```

SQL IDE Interface showing a query execution result.

Query: `CALL RETRIEVE_ALL;`

Result set 1:

| ID | ANIMAL   | SALEPRICE | SALEDATE   | QUANTITY |
|----|----------|-----------|------------|----------|
| 1  | Cat      | 337.56    | 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       |

Run time: 0.020 s

5. You can view the created stored procedure routine `UPDATE_SALEPRICE`. Click on the 3-bar menu icon in the top left corner and click **EXPLORE > APPLICATION OBJECTS > Stored Procedures**. Find the procedure routine `UPDATE_SALEPRICE` from Procedures by clicking **Select All**. Click on the procedure routine **UPDATE\_SALEPRICE**.

IBM Db2 on Cloud
Storage: 22%

[Cookie Preferences](#)
[Discover](#)

### STORED PROCEDURES

**Schemas**

- ☒ Select All + New implicit schema
- ☒ AUDIT 2 procedures
- ☒ ZJH17769 2 procedures
- ☒ DB2INST1 1 procedure
- ☒ ERRORSHEMA 0 procedure
- ☒ SQL74605 0 procedure
- ☒ ST\_INFORMTN\_SCHEMA 0 procedure

**Procedures**

| NAME  | SCHEMA   | PROPERTIES |
|---|----------|------------|
| <input type="checkbox"/> CONNECT_CHE...             | DB2INST1 | ...        |
| <input type="checkbox"/> LOAD                       | AUDIT    | ...        |
| <input type="checkbox"/> RETRIEVE_ALL               | ZJH17769 | ...        |
| <input type="checkbox"/> UPDATE                     | AUDIT    | ...        |
| <input checked="" type="checkbox"/> UPDATE_SALEP... | ZJH17769 | ...        |

**Procedure Parameters**

**UPDATE\_SALEPRICE**

```
CREATE PROCEDURE UPDATE_SALEPRICE (
  IN Animal_ID INTEGER, IN Animal_Health VARCHAR(5) ) -- {
  input/output type parameter {{ parameter-name }} data-type }

LANGUAGE SQL
```

Show more

| PARAMETER     | DATA TYPE | MODE | LENGTH | SCALE | LOCAT |
|---------------|-----------|------|--------|-------|-------|
| ANIMAL_ID     | INTEGER   | IN   | 4      | 0     | No    |
| ANIMAL_HEA... | VARCHAR   | IN   | 5      | 0     | No    |

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

DROP **PROCEDURE UPDATE\_SALEPRICE;**

Syntax assistant

```

1 DROP PROCEDURE UPDATE_SALEPRICE;
2
3 CALL UPDATE_SALEPRICE(5,'BAD');
4

```

Result - Dec 16, 2020 8:...

**DROP PROCEDURE UPDATE\_SALEPRICE**
Run time: 0.024 s

Status: **Success** | Affected Rows: **0**

**CALL UPDATE\_SALEPRICE(5,'BAD')**
Run time: 0.008 s

Status: **Failed**

**Error message**  
No authorized routine named "UPDATE\_SALEPRICE" of type "PROCEDURE" having compatible arguments was found..  
SQLCODE=-440, SQLSTATE=42884, DRIVER=4.26.14  
[Learn more about this error](#)

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

## Author(s)

- [Sandip Saha Joy](#)

## Other Contributor(s)

-

## Changelog

| Date       | Version | Changed by      | Change Description      |
|------------|---------|-----------------|-------------------------|
| 2020-12-25 | 1.1     | Steve Ryan      | ID Reviewed             |
| 2020-12-14 | 1.0     | Sandip Saha Joy | Created initial version |

© IBM Corporation 2020. All rights reserved.