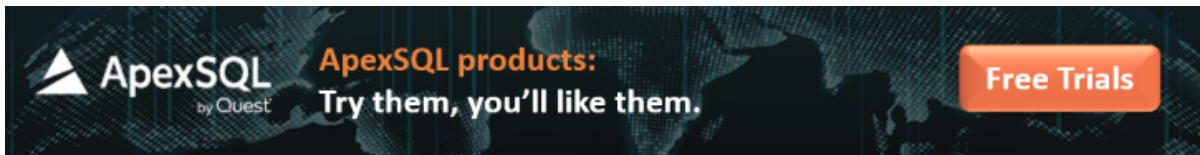




Learn MySQL: The Basics of MySQL Stored Procedures

January 8, 2021 by [Nisarg Upadhyay](#)



In this article, we are going to learn about the stored procedures in [MySQL](#). In this article, I am covering the basics of the stored procedure that includes the following

1. Summary of MySQL Stored Procedure
2. Create a stored procedure using Query and MySQL workbench
3. Create a Parameterized stored procedure
4. Drop the Stored Procedure using query and MySQL workbench

The stored procedure is SQL statements wrapped within the **CREATE PROCEDURE** statement. The stored procedure may contain a conditional statement like IF or CASE or the Loops. The stored procedure can also execute another stored procedure or a function that modularizes the code.

Following are the benefits of a stored procedure:

1. **Reduce the Network Traffic:** Multiple SQL Statements are encapsulated in a stored procedure. When you execute it, instead of sending multiple queries, we are sending only the name and the parameters of the stored procedure

This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

The syntax to create a MySQL Stored procedure is the following:

```
Create Procedure [Procedure Name] ([Parameter 1], [Parameter 2], [Parameter 3] )  
Begin  
SQL Queries..  
End
```

In the syntax:

1. The name of the procedure must be specified after the **Create Procedure** keyword
2. After the name of the procedure, the list of parameters must be specified in the parenthesis.
The parameter list must be comma-separated
3. The SQL Queries and code must be written between **BEGIN** and **END** keywords

To execute the store procedure, you can use the CALL keyword. Below is syntax:

```
CALL [Procedure Name] ([Parameters]..)
```

In the syntax:

1. The procedure name must be specified after the CALL keyword
2. If the procedure has the parameters, then the parameter values must be specified in the parenthesis

Let us create a basic stored procedure. For demonstration, I am using the **sakila** database.

Create a simple stored procedure

Suppose you want to populate the list of films. The output should contain film_id, title, description, release year, and rating column. The code of the procedure is the following:

```
DELIMITER //
```

This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

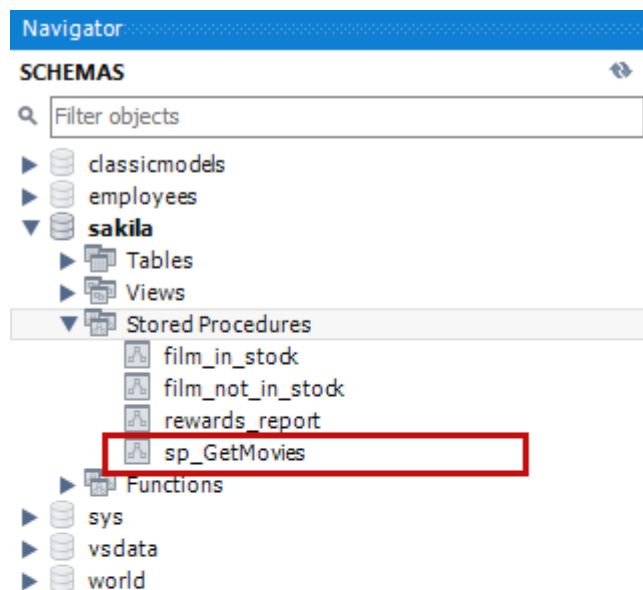
```
1 DELIMITER //
2
3 CREATE PROCEDURE sp_GetMovies()
4 BEGIN
5     select title,description,release_year,rating from film;
6 END //
7
8 DELIMITER ;
9
```

Output

Action Output

#	Time	Action	Message
✓ 1	15:04:17	CREATE PROCEDURE sp_GetMovies() BEGIN select title,description,release_year,rating from film; END	0 row(s) affected

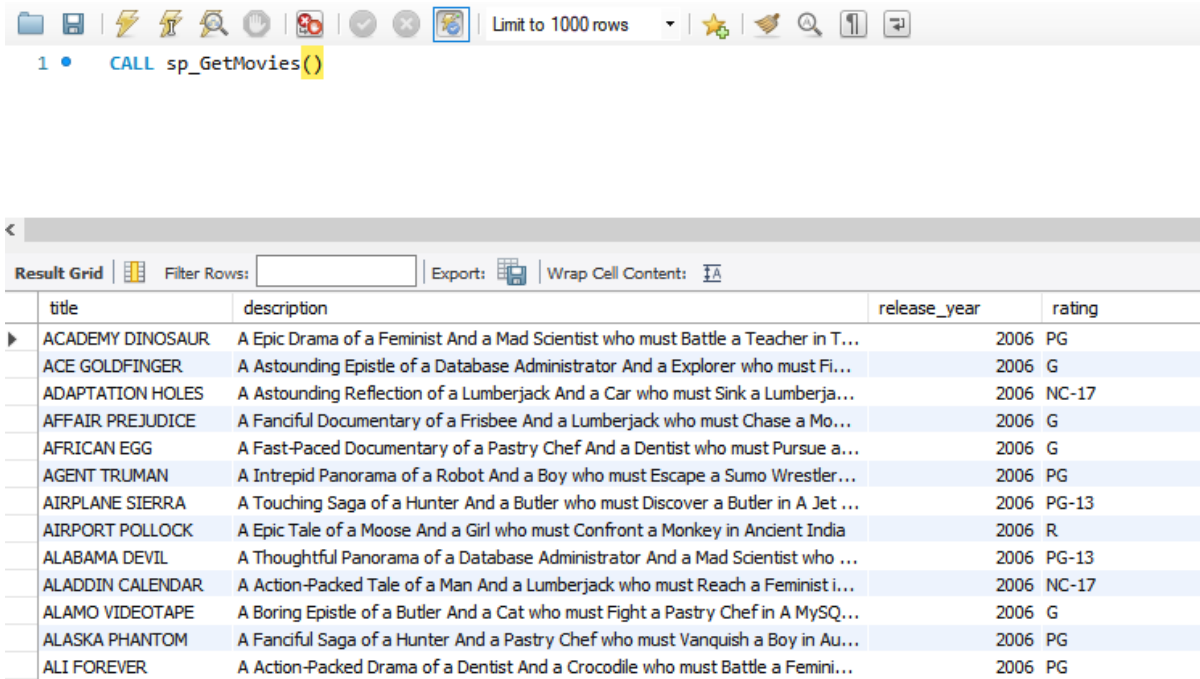
You can view the procedure under stored procedures. See the below screenshot.



This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

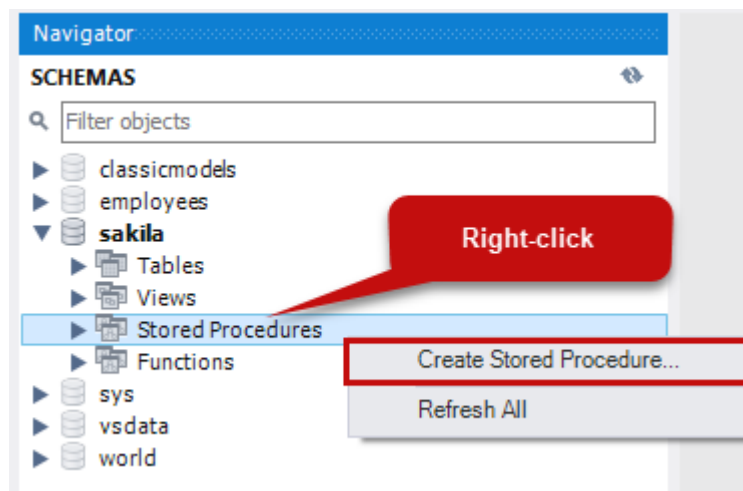


The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a SQL editor contains the query: `CALL sp_GetMovies()`. The results are displayed in a table with the following columns: title, description, release_year, and rating.

title	description	release_year	rating
ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in T...	2006	PG
ACE GOLDFINGER	A Astounding Epistle of a Database Administrator And a Explorer who must Fi...	2006	G
ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberja...	2006	NC-17
AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Mo...	2006	G
AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a...	2006	G
AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who must Escape a Sumo Wrestler...	2006	PG
AIRPLANE SIERRA	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet ...	2006	PG-13
AIRPORT POLLOCK	A Epic Tale of a Moose And a Girl who must Confront a Monkey in Ancient India	2006	R
ALABAMA DEVIL	A Thoughtful Panorama of a Database Administrator And a Mad Scientist who ...	2006	PG-13
ALADDIN CALENDAR	A Action-Packed Tale of a Man And a Lumberjack who must Reach a Feminist i...	2006	NC-17
ALAMO VIDEOTAPE	A Boring Epistle of a Butler And a Cat who must Fight a Pastry Chef in A MySQL...	2006	G
ALASKA PHANTOM	A Fanciful Saga of a Hunter And a Pastry Chef who must Vanquish a Boy in Au...	2006	PG
ALI FOREVER	A Action-Packed Drama of a Dentist And a Crocodile who must Battle a Femini...	2006	PG

Create procedure using MySQL workbench wizard

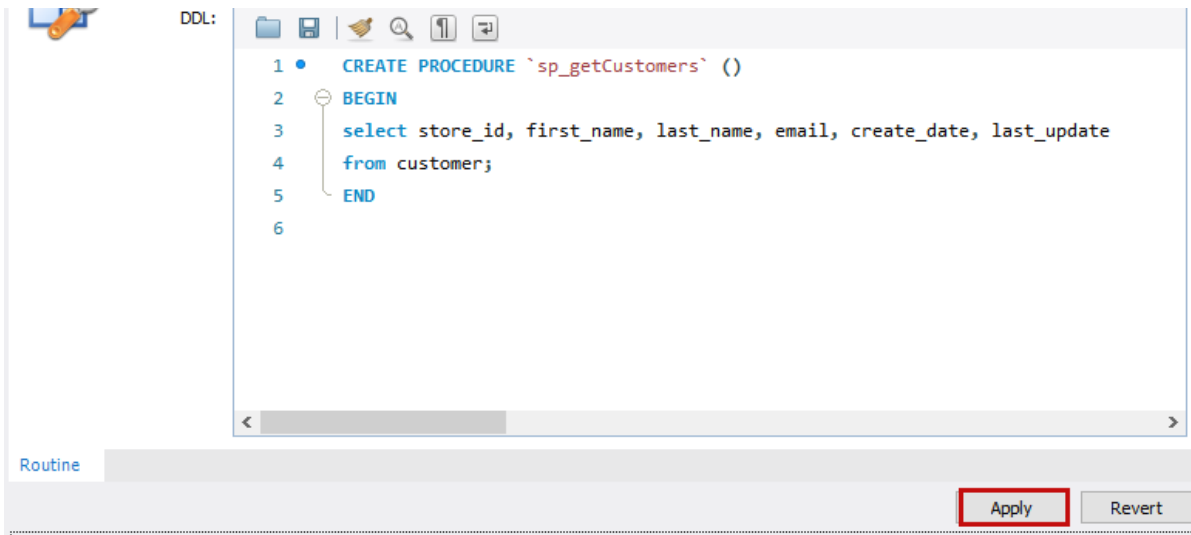
We can use the MySQL workbench wizard to create a stored procedure. Suppose you want to get the list of the customer from the sakila database. To do that, expand the **sakila** schema Right-click on **Stored Procedures** Select **Create a Stored procedure**.



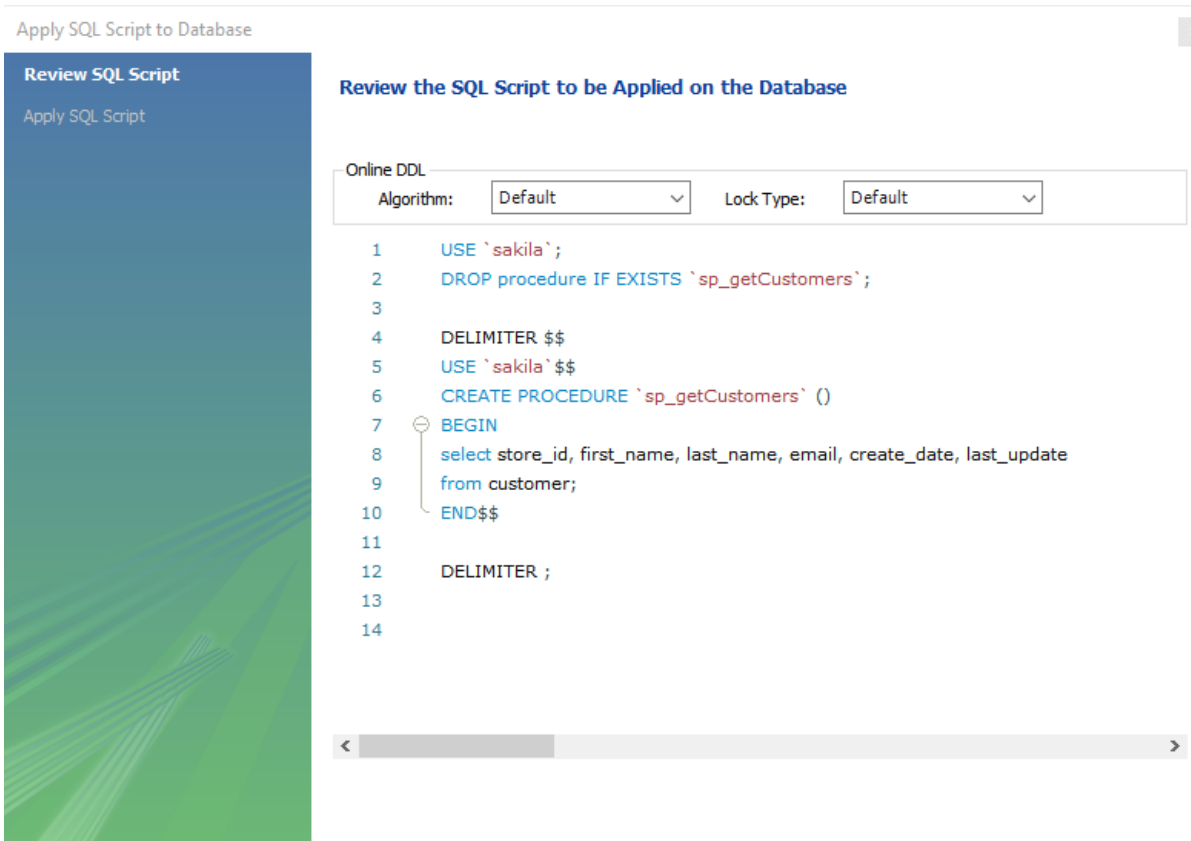
This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)



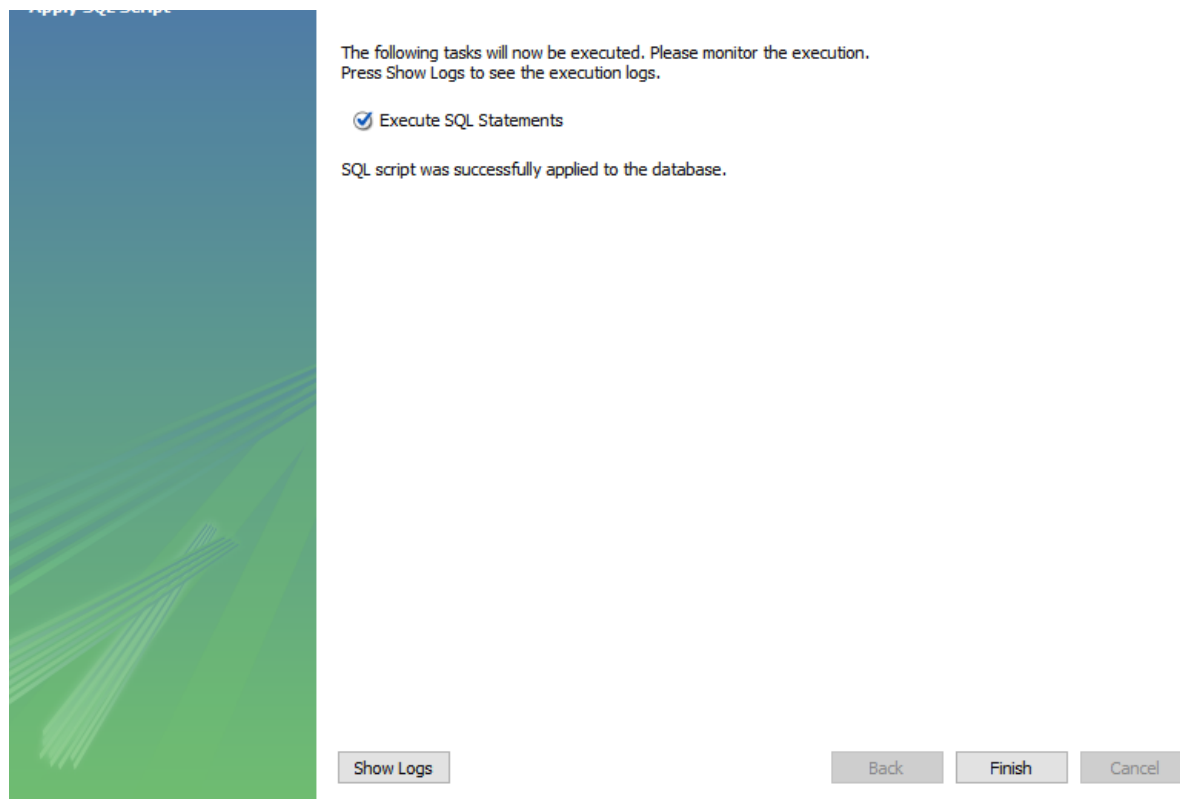
Click on Apply. A dialog box, Apply script to database opens. On the Review the script screen, you can view the code of the stored procedure. Click on Apply.



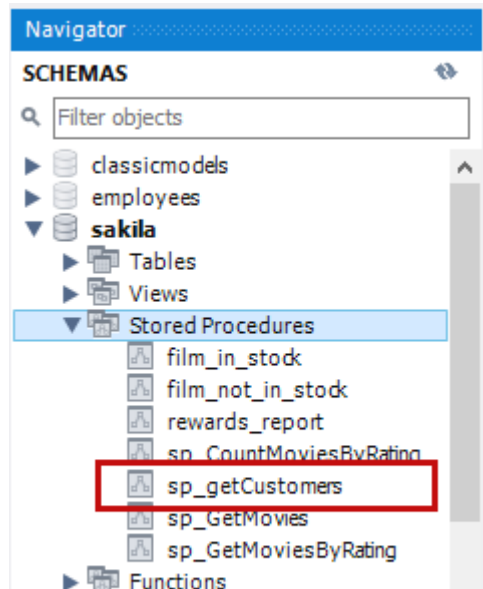
This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)



In MySQL Workbench, You can view the stored procedure under the **Stored Procedures** folder of the sakila schema.



This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

Statements. When we declare the INOUT type parameter, the application has to pass an argument, and based on the input argument; the procedure returns the output to the application.

When we create a stored procedure, the parameters must be specified within the parenthesis. The syntax is following:

```
(IN | OUT | INOUT) (Parameter Name [datatype(length)])
```

In the syntax:

1. Specify the type of the parameter. It can be IN, OUT or INOUT
2. Specify the name and data type of the parameter

Example of IN parameter

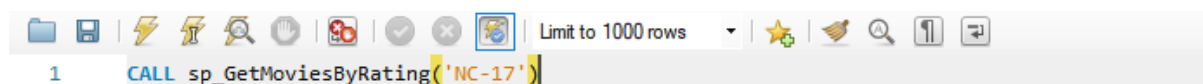
Suppose we want to get the list of films based on the rating. The **param_rating** is an input parameter, and the data type is varchar. The code of the procedure is the following:

```
DELIMITER //  
CREATE PROCEDURE sp_GetMoviesByRating(IN rating varchar(50))  
BEGIN  
    select title,description,release_year,rating from film where rating=rating;  
END //  
DELIMITER ;
```

To populate the list of the films with an **NC-17** rating, we pass the **NC-17** value to the **sp_getMoviesByRating()** procedure.

```
CALL sp_GetMoviesByRating('NC-17')
```

Output:



The screenshot shows a MySQL command window with a toolbar at the top. The command line contains the text: `1 CALL sp_GetMoviesByRating('NC-17')`. The window title bar indicates "Limit to 1000 rows".

This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

ALAMO VIDEOTAPE	A Boring Epistle of a Butler And a Cat who must Fight a Pastry Chef in A MySQL...	2006	NC-17
-----------------	---	------	-------

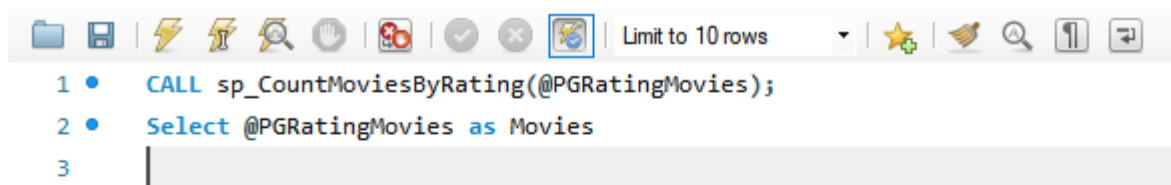
Example of OUT parameter

Suppose we want to get the count of the films that have a **PG-13** rating. The **Total_Movies** is an output parameter, and the data type is an integer. The count of the movies is assigned to the **OUT** variable (**Total_Movies**) using the INTO keyword. The code of the procedure is the following:

```
DELIMITER //
CREATE PROCEDURE sp_CountMoviesByRating(OUT Total_Movies int)
BEGIN
    select count(title) INTO Total_Movies from film where rating='PG-13';
END //
DELIMITER ;
```

To store the value returned by the procedure, pass a session variable named **@PGRatingMovies**.

```
CALL sp_CountMoviesByRating(@PGRatingMovies)
Select @PGRatingMovies as Movies
```



Result Grid	
Movies	223

Example of an INOUT parameter

Suppose we want to get the total count of movies based on the rating. The input parameter is param_rating in the procedure, and the data type is **varchar(10)**. The output parameter is **Movies_count**, and the data type is an **integer**.

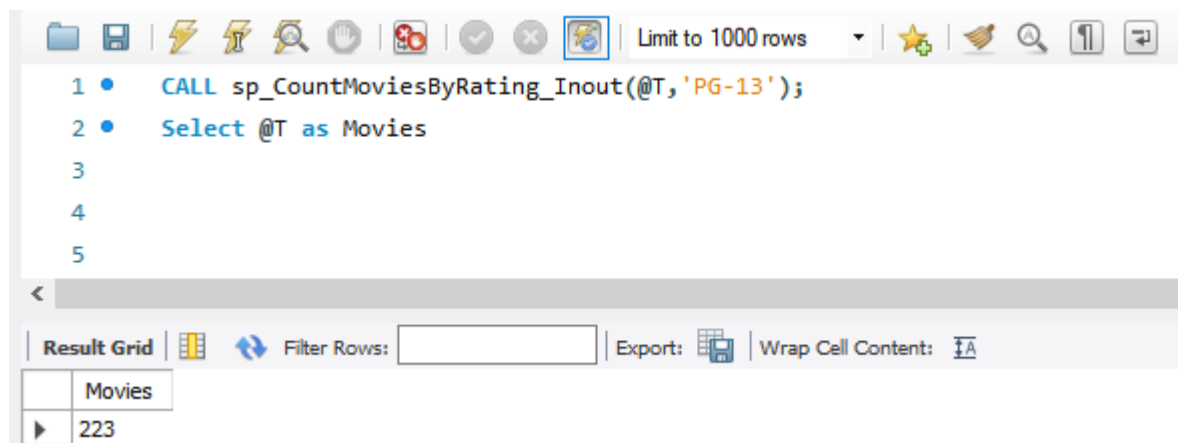
This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

Execute the procedure using **CALL** keyword and save the output in session variable named **@MoviesCount**

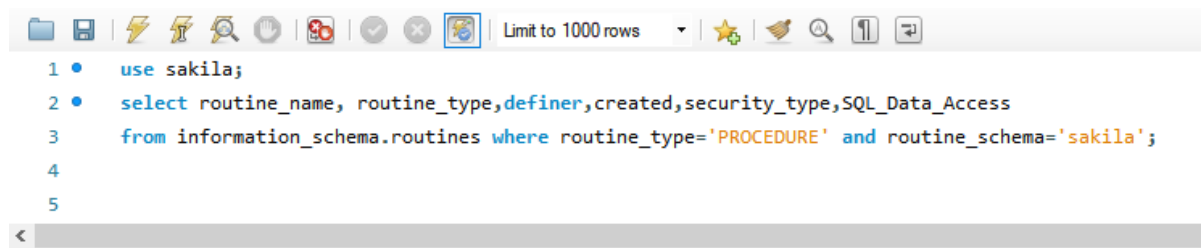
```
CALL sp_CountMoviesByRating_Inout(@T, 'PG-13');  
Select @T as Movies
```



View the list of stored procedure in a database using a query

To view the list of the stored procedure, you can query the `information_schema.routines` table. It contains the list of the stored procedure and stored functions created on the database. To view the list of the **stored procedure** created in a sakila database, run the following query. Moreover, it also provides the **owner**, **created date**, **security type**, and **SQL data access** to the stored procedures.

```
select routine_name, routine_type, definer, created, security_type, SQL_Data_Access  
from information_schema.routines where routine_type='PROCEDURE' and routine_schema  
='sakila';
```



This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

Drop a Stored Procedure

To drop the stored procedure, you can use the drop procedure command. The syntax is following

```
Drop procedure [IF EXISTS] <Procedure Name>
```

In the syntax, the name of the stored procedure must be followed by the **Drop Procedure** keyword. If you want to drop the **sp_getCustomers** procedure from the sakila database, you can run the following query.

```
Drop procedure sp_getCustomers
```

When you try to drop the procedure that does not exist on a database, the query shows an error:

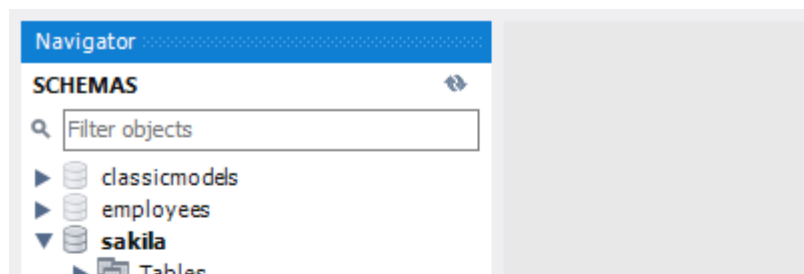
```
ERROR 1305 (42000): PROCEDURE sakila.getCustomer does not exist
```

To avoid this, you can include the [IF EXISTS] option in the drop procedure command. When you include the IF EXISTS keyword, instead of an error, the query returns a warning:

```
Query OK, 0 rows affected, 1 warning (0.01 sec) 1305 PROCEDURE  
sakila.getCustomer does not exist
```

Drop a Stored Procedure using MySQL workbench wizard

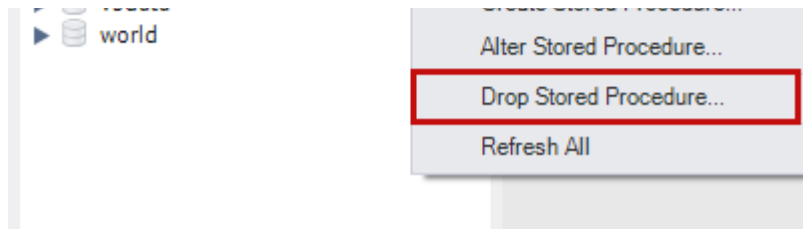
You can use the MySQL workbench wizard to drop the procedure. To drop any procedure, expand **sakila** schema Expand **Stored Procedures** Right-click on **sp_GetMovies** Click on **Drop Stored Procedure**.



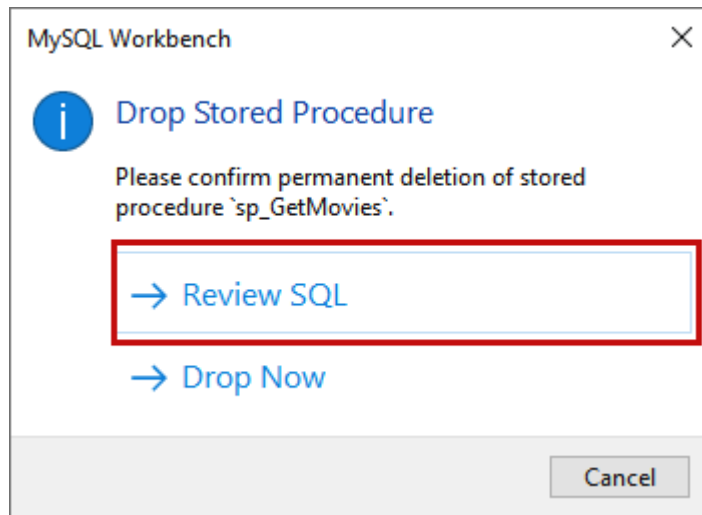
This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

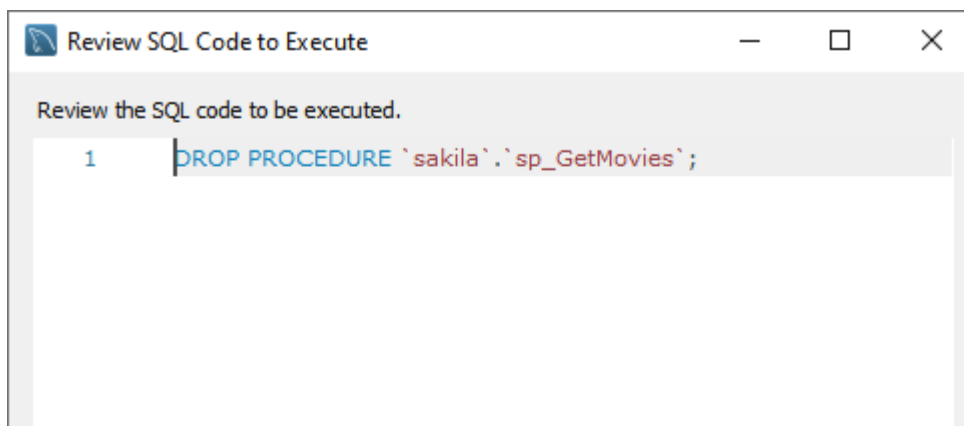
Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)



A dialog box opens. You can choose to review the procedure before dropping it, or you can drop it without reviewing it. It is good practice to review the database object before dropping it, so choose Review SQL.



In **Review SQL Code to Execute** dialog box, you can review the drop statement and the object name.



This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

Summary

In this article, we learned the basics of MySQL Stored procedure. I have covered following topics:

1. Syntax to create a MySQL Stored Procedure and how to create them us Create Procedure statement and MySQL workbench wizard
2. How to create a parameterized MySQL Stored Procedure

In next article, we are going to learn about the MySQL Views.

Table of contents

[Learn MySQL: Querying data from MySQL server using the SELECT statement](#)

[Learn MySQL: What is pagination](#)

[Learn MySQL: Sorting and Filtering data in a table](#)

[Learn MySQL: Add data in tables using the INSERT statement](#)

[Learn MySQL: Create and drop temp tables](#)

[Learn MySQL: Delete and Update Statements](#)

[Learn MySQL: The Basics of MySQL Stored Procedures](#)

[Learn MySQL: The Basics of MySQL Views](#)

[Learn MySQL: An overview of MySQL Binary Logs](#)

[Learn MySQL: An overview of the mysqlbinlog utility](#)

[Learn MySQL: Run multiple instances of MySQL Server on Windows 10](#)

[Learn MySQL: MySQL String Functions](#)

This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

ApexSQL Database Power Tools for VS Code is an extension for VS Code which allows users to [connect to MySQL and MariaDB instances](#), run queries and display results, search for objects, export query results into several standard formats, generate DDL and DML scripts from object explorer on existing platforms like Windows, Linux, macOS



MySQL and MariaDB development and management



Nisarg Upadhyay

Nisarg Upadhyay is a SQL Server Database Administrator and Microsoft certified professional who has more than 8 years of experience with SQL Server administration and 2 years with Oracle 10g database administration.

He has expertise in database design, performance tuning, backup and recovery, HA and DR setup, database migrations and upgrades. He has completed the B.Tech from Ganpat University. He can be reached on nisargupadhyay87@outlook.com

Related Posts:

1. [Learn MySQL: The Basics of MySQL Views](#)
2. [Learn MySQL: Install MySQL server 8.0.19 using a noinstall Zip archive](#)
3. [Learn MySQL: Run multiple instances of MySQL Server on Windows 10](#)
4. [Learn MySQL: MySQL String Functions](#)
5. [MySQL Recursive Queries](#)

Development, MySQL, Stored procedures

This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)

ALSO ON SQL SHACK

**Understanding the
OPENQUERY ...**

5 months ago • 1 comment

This article will show the
OPENQUERY function and
its usage.

**How to write readable
T-SQL queries**

6 months ago • 1 comment

This article intends to give
some beneficial suggestions
that help to write a more ...

SQL Subtract dates

20 days ago • 1 comment

This article will show how to
subtract dates using SQL
Server and also help you ...

**Gettin
SQL :**

a year &

This ar
to crea
functio

1 Comment

 **Robert Holland** ▾

Join the discussion...

Sort by Best ▾



5

**Luke Krell** • 7 months ago

In the article you say:

"Secure: The stored procedures are more secure than the AdHoc queries. The permission can be granted to the user to execute the stored procedure without giving permission to the tables used in the stored procedure."

I can't seem to get this to work. I want a user to be able to use my stored procedures which are comprised of a series of select statements. But I don't want the user to have permissions to run select statements outside of my stored procedures. This sounds possible from your statement, but I can't get it to work.

Can you help me? Thanks.

^ | ▾ • Reply • Share >

© 2022 Quest Software Inc. ALL RIGHTS RESERVED. | [GDPR](#) | [Terms of Use](#) | [Privacy](#)

This website uses cookies. By continuing to use this site and/or clicking the "Accept" button you are providing consent

Accept

Quest Software and its affiliates do NOT sell the Personal Data you provide to us either when you register on our websites or when you do business with us. For more information about our [Privacy Policy](#) and our data protection efforts, please visit [GDPR-HQ](#)