The following is an example of a stored procedure in MySQL.

When you apply the complete procedure to the database, it becomes available to be called as a SQL statement (from within phpMyAdmin, or as part of an external application such as a PHP web page), using the syntax:

```
CALL class.find_expensive(price, supplier ID);  or  CALL `class`.`find_expensive`(price, supplier ID);
```

ex: `CALL `class`.`find_expensive`(3.00, 'Asp');`

```
DELIMITER $$

CREATE PROCEDURE `class`.`find_expensive`(var_CurrPrice DECIMAL(10,2), var_x VARCHAR(3))
BEGIN
  # You can add a multi-line comment
  # by including it between the BEGIN and END

  SELECT COUNT(Item_Number) AS `expensive items`
  FROM Item
  WHERE Item_Price > var_CurrPrice
  AND Supplier_ID LIKE var_x;

END $$
```

Defining a new delimiter (which can be any combination of characters), as in the first line, allows you to differentiate between the individual DDL statements used to DROP and CREATE the stored procedure, and the DML statements that are included within the contents of that procedure. In particular, it defines the entire "CREATE PROCEDURE .... END" statement as a single SQL command (since, in this example, there is no delimiter ($$) included to imply that the "SELECT" statement is a separate SQL statement from the rest of it).

It is simple to both create and edit such stored procedures within phpMyAdmin. In order to create one initially, you simply run the SQL, as given above, from any SQL window.
**View / Edit / Delete:**

To view an existing stored procedure in phpMyAdmin 4.0.4 or later, there is a "Routines" tab available (probably under "v More" at the right-hand side of the set of tabs) when you select your database.

To view an existing stored procedure in older versions of phpMyAdmin, you navigate to the database within which the procedure is defined, and select the Structure tab - you can then expand the "Routines" menu that should appear below the list of tables in the main window. You may then easily edit or delete the procedure by selecting the appropriate Action.

Note that in allowing editing of the procedure, MySQL uses the 'DROP Procedure' command first, and then recreates the original procedure from scratch. To define a new procedure while still keeping the old one, you simply need to rename the procedure to the new name (and then change its functionality, as needed) and then delete the 'DROP Procedure' command from the beginning of the statement before running the SQL.