My first stored procedures database gestion

We are going to create our first stored procedures. In MySQL Workbench, we are going to write the following scripts:

1. Create an SP that lists all clients of database gestion.

1. The first line will be dedicated as usual to connect to the desired database, in this case gestion:

   USE gestion;

2. Next, we are going to create a stored procedure that lists all the clients in our database. For that reason, we write:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS listar_clientes$$
CREATE PROCEDURE listar_clientes()
BEGIN
 SELECT *
 FROM clientes;
END
$$
```

   You just need to take into account that a delimiter must be set. Then, we add a line in order to delete our SP (that´s important because if we want to change something, we can execute the script again without any problems). Then, we write the SP header with the sentence: CREATE PROCEDURE, its name and its parameters (here we don´t have any). Afterwards, we find between the sentences BEGIN and END, the code of the SP. In this case, we have only an easy query that lists all the information about the clients. And finally, we close it with the delimiter.

3. In order to execute it, we must write:

```
USE gestion;
CALL listar_clientes();
```

2. Now, create an SP that lists all employees. It´s very similar to the previous one.

3. List all orders for a specific customer.

```
USE gestion;
DELIMITER $$
DROP PROCEDURE IF EXISTS pedidos_cliente$$
CREATE PROCEDURE pedidos_cliente(IN cli int)
BEGIN
 SELECT *
 FROM pedidos
 WHERE clie = cli;
END
$$
```

Calling the SP:

```
USE gestion;

CALL pedidos_cliente(2107);
```

4. List all orders of a specific representative (very similar to the previous one).

5. Create an oficinas_reg SP that lists the offices in a given region. Hint: what is different from the previous exercise is the data type of the input parameter, as its datatype is now a string.

```
USE gestion;
DELIMITER $$
DROP PROCEDURE IF EXISTS oficinas_reg$$
CREATE PROCEDURE oficinas_reg(IN reg VARCHAR(20))
BEGIN
 SELECT *
 FROM oficinas
 WHERE region = reg;
END
$$
```

Calling the SP:

```
USE gestion;

CALL oficinas_reg('este');


USE gestion;

CALL oficinas_reg('esta'); --returns no values
```

6. Create a clientes_rep procedure that lists the client data assigned to the code of a given representative that will be passed as an input parameter.

```
USE gestion;
DELIMITER $$
DROP PROCEDURE IF EXISTS clientes_rep$$
CREATE PROCEDURE clientes_rep(IN rep INT)
BEGIN
 SELECT *
 FROM clientes
 WHERE repclie = rep;
END
$$
```

Calling the SP:

```
USE gestion;

CALL clientes_rep(106);
```

7. List the orders placed for a given customer by a given representative.

```
USE gestion;
DELIMITER $$
DROP PROCEDURE IF EXISTS pedidos_cliente_rep$$
CREATE PROCEDURE pedidos_cliente_rep(IN cli int, IN r int)
BEGIN
 SELECT *
 FROM pedidos
 WHERE clie = cli AND rep = r;
END
$$
```

Calling the SP:

```
USE gestion;

CALL pedidos_cliente_rep(2103, 105);
```

8. List the orders in which a certain customer has purchased products from a certain manufacturer.

```
USE gestion;
DELIMITER $$
DROP PROCEDURE IF EXISTS pedidos_cliente_fab$$
CREATE PROCEDURE pedidos_cliente_fab(IN cli int, IN f VARCHAR(20))
BEGIN
 SELECT *
 FROM pedidos
 WHERE clie = cli AND fab = f;
END
$$
```

Calling the SP:

```
USE gestion;

CALL pedidos_cliente_fab(2103, 'aci');
```

9. List the subordinates of a given employee.

```
USE gestion;
DELIMITER $$
DROP PROCEDURE IF EXISTS empleados_jefe$$
CREATE PROCEDURE empleados_jefe(IN j int)
BEGIN
 SELECT *
 FROM empleados
 WHERE jefe = j;
END
$$
```

Calling the SP:

```
USE gestion;

CALL empleados_jefe(108);
```

10. List the employees asigned to a given office. It is very similar to the previous exercise.