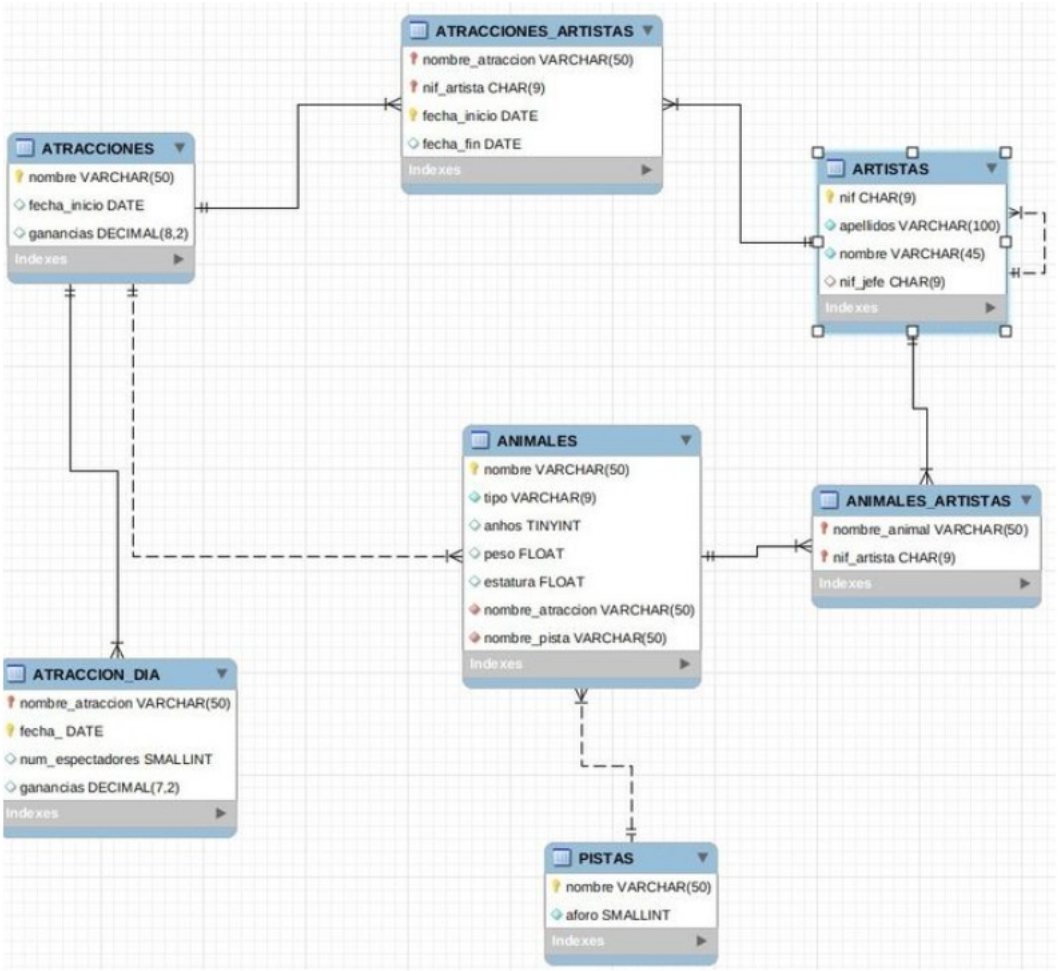
Exercises stored procedures

database circo

We are going to work with the following database called Circo:



Index

1. Exercises of stored procedures (SP)

1. Create a procedure named artistas_getList() that returns the first and last names of the artists separated by commas in the following format: lastname, firstname sorted in descending order.

```
USE CIRCO;

DROP PROCEDURE IF EXISTS artistas_getList;


DELIMITER $$


CREATE PROCEDURE artistas_getList()

COMMENT 'Devuelve el nombre completo de todos los artistas ordenado descendentemente'

BEGIN

   SELECT CONCAT(apellidos,', ',nombre) as nombreCompleto

   FROM ARTISTAS

   ORDER BY nombreCompleto DESC;

END$$


DELIMITER ;
```

Call example:

```
CALL artistas_getList();
```

2. Create a procedure named artists_getListAnimales() that returns the names of the artists along with their ID number as well as the name and weight of the animals that are attended by the artists, sorted by artist ID number and animal name.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS artistas_getListAnimales;

DELIMITER $$

CREATE PROCEDURE artistas_getListAnimales()
COMMENT 'Devuelve el nombre, apellidos y nif de los artistas junto con el nombre y peso de los animales que atienden'
BEGIN
   SELECT nif, apellidos, ARTISTAS.nombre as nombreArtista, ANIMALES.nombre as
        nombreAnimal, peso
   FROM ARTISTAS INNER JOIN ANIMALES_ARTISTAS ON
        (ANIMALES_ARTISTAS.nif_artista = ARTISTAS.nif)
        INNER JOIN ANIMALES ON (ANIMALES_ARTISTAS.nombre_animal =
        ANIMALES.nombre)
   ORDER BY nif, nombreAnimal;

END$$

DELIMITER ;
```

Call example:

```
CALL artistas_getListAnimales();
```

3. Create a procedure named atracciones_getListConAntiguedad10() that returns data of attractions that started 10 years ago from the current date. You will have to make use of one of the date-time functions. Try to find out which one.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS atracciones_getListConAntiguedad10;
DELIMITER $$

CREATE PROCEDURE atracciones_getListConAntiguedad10()
COMMENT 'Devuelve todos los datos de las atracciones que comenzaron hace 10 años con respecto a la fecha actual'
BEGIN
   SELECT *
   FROM ATRACCIONES
   WHERE fecha_inicio BETWEEN DATE_SUB(curdate(), INTERVAL 10 YEAR) AND curdate()
   ORDER BY nombre;

END$$

DELIMITER ;
```

Call example:

```
CALL atracciones_getListConAntiguedad10();
```

4. Create a procedure named animales_Leo_getPista() that displays the data of the ring where the animal named 'Leo' works. Use a local variable that stores the name of the ring. Then query the obtained ring using that local variable.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS animales_Leo_getPista;
DELIMITER $$

CREATE PROCEDURE animales_Leo_getPista()
COMMENT 'Devuelve los datos de la pista donde trabaja el animal de nombre Leo'


BEGIN
   DECLARE v_nombrePista varchar(50) default '';


   SELECT nombre_pista
   INTO v_nombrePista
   FROM ANIMALES
   WHERE nombre = 'Leo';

   SELECT *
   FROM PISTAS
   WHERE nombre=v_nombrePista;


END$$

DELIMITER ;
```

Call example:

```
CALL animales_Leo_getPista;
```

5. Create a procedure named atracciones_getUltima() that obtains the data of the last attraction held (table ATTRACTION_DAY), using local variables. In order to do this, store in a variable the name of the last attraction held and look for the data of that attraction. Keep in mind to limit with the keyword LIMIT the number of rows returned by a query if you are not sure that it will return a single row and you are going to store the data in a variable.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS atracciones_getUltima;
DELIMITER $$

CREATE PROCEDURE atracciones_getUltima()
COMMENT 'Devuelve los datos de la última atracción celebrada'
BEGIN


   DECLARE v_nombreAtraccion varchar(50) default '';

   SELECT nombre_atraccion
   INTO v_nombreAtraccion
   FROM ATRACCION_DIA
   ORDER BY fecha DESC
   LIMIT 1;

   /*
   Si empleáramos subconsultas, podríamos hacer algo como lo siguiente
   SELECT nombre_atraccion
   INTO v_nombreAtraccion
         FROM ATRACCION_DIA
         WHERE fecha = (SELECT MAX(fecha)
                        FROM ATRACCION_DIA)
         LIMIT 1;
   */

   SELECT *                -- Devuelve una única fila. No hace falta order by
   FROM ATRACCIONES
   WHERE nombre=v_nombreAtraccion;

END$$
DELIMITER ;
```

Call example:

```
CALL atracciones_getUltima();
```

6. Create a procedure named atracciones_getArtistaUltima() that gets the data of the attraction and the artist working at the attraction whose start date has started later in time. It uses two variables: one that stores the artist's ID and another one that stores the name of the attraction.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS atracciones_getArtistaUltima;
DELIMITER $$

CREATE PROCEDURE atracciones_getArtistaUltima()
COMMENT 'Devuelve los datos de la atracción y del artista que trabaja en dicha atracción, cuya fecha de inicio ha empezado más tarde'
BEGIN
        DECLARE v_nombreAtraccion varchar(50) default '';
        DECLARE v_nifArtista char(9);                -- Al no llevar default el valor por defecto es null y sería lo que tendríamos que comparar en un if, por ejemplo.

    SELECT nombre_atraccion,nif_artista
    INTO v_nombreAtraccion,v_nifArtista
    FROM ATRACCIONES_ARTISTAS
    ORDER BY fecha_inicio DESC
    LIMIT 1;

    SELECT *                                    -- Devuelve una única fila. No hace falta order by
    FROM ATRACCIONES,ARTISTAS       -- Esta opción no es la adecuada ya que hace el producto cartesiano de todas las filas de las dos tablas.
    WHERE ATRACCIONES.nombre=v_nombreAtraccion AND
                ARTISTAS.nif = v_nifArtista;

   /* Mejor esta solución y en el resultado final se tendrían que leer dos conjuntos de resultados o bien hacer un INNER JOIN con atracciones_artistas.

    SELECT *
    FROM ARTISTAS
    WHERE ARTISTAS.nif = v_nifArtista;

    SELECT *
    FROM ATRACCIONES
    WHERE ATRACCIONES.nombre=v_nombreAtraccion;
   */

END$$

DELIMITER ;
```

Call example:

```
CALL atracciones_getArtistaUltima();
```

2. Exercises of SP with input parameters

1. Create a procedure named artistas_getAnimalesPorNif that returns the animals that an artist takes care of. It must take the artist's nif as a parameter.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS artistas_getAnimalesPorNif;
DELIMITER $$

CREATE PROCEDURE artistas_getAnimalesPorNif(p_nif char(9))
COMMENT 'Devuelve los animales que cuida un artista. Llevará como parámetro el nif de un artista.'
BEGIN
        SELECT distinct ANIMALES.*
        FROM ANIMALES INNER JOIN ANIMALES_ARTISTAS
                ON (ANIMALES.nombre = ANIMALES_ARTISTAS.nombre_animal)
        WHERE ANIMALES_ARTISTAS.nif_artista = p_nif
   ORDER BY nombre;
END$$
DELIMITER ;
```

Call examples:

```
USE CIRCO;

CALL artistas_getAnimalesPorNif('11111111A');
```

```
USE CIRCO;

CALL artistas_getAnimalesPorNif('22222222B');
```

```
USE CIRCO;

SET @nifBuscar='44444444D';

CALL artistas_getAnimalesPorNif(@nifBuscar);
```

2. Create a procedure named artistas_getAnimalesPorNombreApel that returns the animals that an artist takes care of. It shall take the artist's first and last name as a parameter. We assume that the first and last names form an alternative key.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS artistas_getAnimalesPorNombreApel;
DELIMITER $$

CREATE PROCEDURE artistas_getAnimalesPorNombreApel(p_nombre varchar(45),p_apellidos
varchar(100))


BEGIN
 DECLARE v_nifArtista char(9);              -- Si no ponemos valor por defecto, sería NULL

 SELECT nif
 INTO v_nifArtista
 FROM ARTISTAS
 WHERE apellidos = p_apellidos AND nombre = p_nombre;
 -- LIMIT 1;            Tendríamos que poner LIMIT 1 para poder emplear INTO en el caso de que
hubiera varias personas con el mismo nombre y apellidos

 SELECT distinct ANIMALES.*   -- Se podría emplear IN sobre animales_artistas
 FROM ANIMALES INNER JOIN ANIMALES_ARTISTAS
 ON (ANIMALES.nombre = ANIMALES_ARTISTAS.nombre_animal)
 WHERE ANIMALES_ARTISTAS.nif_artista = v_nifArtista
 ORDER BY nombre;
END$$
DELIMITER ;
```

Call example:

```
USE CIRCO;

CALL artistas_getAnimalesPorNombreApel('Luis','Sanchez');
```

3. Create a procedure named atracciones_getListConAntiguedad that returns data for attractions that started a number of years ago in relation to the current date. You will need to use time functions.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS atracciones_getListConAntiguedad;
DELIMITER $$

CREATE PROCEDURE atracciones_getListConAntiguedad(p_antiguedad tinyint)
    COMMENT 'Devuelve los datos de las atracciones que han comenzado hace un cierto número de años respecto a la fecha actual.'
BEGIN
 SELECT *
 FROM ATRACCIONES
 WHERE fecha_inicio BETWEEN DATE_SUB(curdate(), INTERVAL p_antiguedad YEAR) AND curdate()
 ORDER BY nombre;

END$$
DELIMITER ;
```

Call example:

```
USE CIRCO;
CALL atracciones_getListConAntiguedad(10);
```

4. Create a procedure named artistas_getListMasAnimalesCuida that returns the data of the artist(s) that take care of more animals than the indicated number (parameter sent to it). Hint: As the query can return more than one artist, we cannot use INTO.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS artistas_getListMasAnimalesCuida;
DELIMITER $$

CREATE PROCEDURE artistas_getListMasAnimalesCuida(p_numAnimales tinyint)
   COMMENT 'Devuelve los datos del/os artista/s que cuida a más animales de los indicados.
p_numAnimales: indica a cuantos animales tiene que cuidar el artista para salir en el listado'
BEGIN
   SELECT *
   FROM ARTISTAS
   WHERE nif IN (SELECT nif_artista
                 FROM ANIMALES_ARTISTAS
                 GROUP BY nif_artista
                 HAVING COUNT(*) > p_numAnimales)
   ORDER BY nif;

END$$
DELIMITER ;
```

Call examples:

```
USE CIRCO;

CALL artistas_getListMasAnimalesCuida(2);
```

5. Create a procedure named atracciones_getListPorFecha that returns the data of the attractions that have started from the specified date.

Hint: Remember that dates are treated as strings and note the formatting.

Add a new attraction with the current start date.

Call the procedure using the current date minus 3 days (make use of the DATE_SUB and CURDATE functions).

```
USE CIRCO;
DROP PROCEDURE IF EXISTS atracciones_getListPorFecha;
DELIMITER $$

CREATE PROCEDURE atracciones_getListPorFecha(p_fecha char(10)) -- aaaa-mm-dd
 COMMENT 'Devuelve los datos de las atracciones que han comenzado a partir de la fecha
indicada. p_fecha: Formato aaaa-mm-dd'
BEGIN
 SELECT *
 FROM ATRACCIONES
 WHERE fecha_inicio > p_fecha
 ORDER BY nombre;

END$$
DELIMITER ;
```

Call examples:

```
USE CIRCO;
CALL atracciones_getListPorFecha('2001-01-01');
```

```
USE CIRCO;
CALL atracciones_getListPorFecha(DATE_SUB(curdate(), INTERVAL 8 YEAR));
```

6. Create a procedure named pistas_add that adds a new track.

Regarding data validation, we could have ‘if conditions’ and check if the capacity is greater than zero.

You can use the ROW_COUNT() function to find out how many rows were added, deleted or modified.
Important: The parameters must have the same data type and size as the one defined at column level in the PISTAS table.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS pistas_add;
DELIMITER $$

CREATE PROCEDURE pistas_add(p_nombre varchar(50),p_aforo smallint)
   COMMENT 'Añade una nueva pista'
   BEGIN
   INSERT INTO PISTAS (nombre, aforo)
   VALUES (p_nombre,p_aforo);

   SELECT ROW_COUNT();

END$$
DELIMITER ;
```

Call example:

```
USE CIRCO;
CALL pistas_add('El gran misil',134);
```

7. Create a procedure named atracciones_update that allows modifying the data of an attraction (it is not allowed to update its primary key).

Modify the start date of the attraction 'El gran felino' and set it one day after the current date.

Hint: You will have to save the earnings and the start date it has in order to send that data to the procedure.

Check that when you call the method with an attraction that doesn't exist, row_count will return 0.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS atracciones_update;
DELIMITER $$

CREATE PROCEDURE atracciones_update(p_nombre varchar(50),p_fecha date, p_ganancias
decimal(8,2))
   COMMENT 'Modifica una atracción'
   BEGIN

   UPDATE ATRACCIONES
   SET fecha_inicio = p_fecha,
      ganancias = p_ganancias
   WHERE nombre = p_nombre;

   SELECT ROW_COUNT();

END$$
DELIMITER ;
```

Call examples:

```
USE CIRCO;
SELECT ganancias,fecha_inicio
INTO @ganancias,@fecha_inicio
FROM ATRACCIONES
WHERE nombre = 'El gran felino';
CALL atracciones_update('El gran felino',DATE_ADD(@fecha_inicio,INTERVAL 1 DAY),
@ganancias);
```

```
USE CIRCO;
CALL atracciones_update('El gran felino_no_existe','1890-01-03',1000.34); -- no devolverá nada
```

8. Create a procedure named pistas_delete that deletes a ring by name.

Base the deletion on the pattern name (using the like function).

Delete the attraction you added in exercise 6 by sending the first letters (be careful that there is no other attraction stored with those letters at the beginning).

Hint: Use the CONCAT function to help you with the LIKE operation.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS pistas_delete;
DELIMITER $$

CREATE PROCEDURE pistas_delete(p_nombre varchar(50))
 COMMENT 'Borra una pista'
 BEGIN

 DELETE FROM PISTAS
 WHERE nombre LIKE CONCAT(p_nombre,'%');

 SELECT ROW_COUNT();

 END$$
DELIMITER ;
```

Call example:

```
USE CIRCO;
CALL pistas_delete('El gran m');
```

3. Exercises of SP with output parameters

1. Create a procedure named pistas_getAforo where the name of a ring is passed and its capacity is returned as an output parameter.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS pistas_getAforo;
DELIMITER $$


CREATE PROCEDURE pistas_getAforo(p_nombre varchar(50), OUT p_aforo smallint)
   COMMENT 'Devuelve en p_aforo el aforo de la pista indicada por p_nombre'
BEGIN

   SELECT aforo
   INTO p_aforo
   FROM PISTAS
   WHERE nombre = p_nombre;

END$$
DELIMITER ;
```

Call examples:

```
USE CIRCO;
CALL pistas_getAforo('LATERAL1',@aforo);
SELECT @aforo;
```

2. Create a procedure named artistas_getNumAnimalesCuida for which the nif of an artist is passed and returns as an output parameter how many animals are being cared of.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS artistas_getNumAnimalesCuida;
DELIMITER $$
CREATE PROCEDURE artistas_getNumAnimalesCuida(p_nif char(9), OUT p_numAnimales
tinyint)
   COMMENT 'Devuelve en p_numAnimales el número de animales que cuida el artista indicado
        por p_nif'
BEGIN

   SELECT COUNT(*)
   INTO p_numAnimales
   FROM ANIMALES_ARTISTAS
   WHERE nif_artista = p_nif;

END$$
DELIMITER ;
```

Call examples:

```
USE CIRCO;
CALL artistas_getNumAnimalesCuida('11111111A',@numAnimales);
SELECT @numAnimales;
```

3. Create a procedure with the name animales_getNombreAforo which receives the name of an animal and returns, using an output parameter and making use of the procedure created in exercise 1, a string with the following format: AnimalName:weight:track:capacity.

Hint: Use the CONCAT function.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS animales_getNombreAforo;
DELIMITER $$
CREATE PROCEDURE animales_getNombreAforo(p_nombre varchar(50), OUT p_cadena
varchar(150))
   COMMENT 'Devuelve en p_cadena una cadena con el formato: NombreAnimal:peso:pista:aforo
en base al nombre del animal p_nombre'
BEGIN
   DECLARE v_peso float;
   DECLARE v_aforo smallint;
   DECLARE v_nombrePista varchar(50);

   SELECT nombre_pista,peso
   INTO v_nombrePista,v_peso
   FROM ANIMALES
   WHERE nombre = p_nombre;

   CALL pistas_getAforo(v_nombrePista, v_aforo); -- Devuelve en v_aforo el aforo de la pista

   SET p_cadena = CONCAT(p_nombre,':',v_peso,':',v_nombrePista,':',v_aforo);

END$$
DELIMITER ;
```

Call examples:

```
USE CIRCO;
CALL animales_getNombreAforo('Leo',@datos);
SELECT @datos;
```

4. Create a procedure named artistas_getNumAtracAnimal that receives the surname and first name of an artist and returns, using an output parameter, the number of attractions in which he works and the number of animals he takes care of (using the procedure of exercise 2) with the following format: nif: NumAtractions: NumAnimals.

Note: We assume that there are no artists with the same first and last name.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS artistas_getNumAtracAnimal;
DELIMITER $$
CREATE PROCEDURE artistas_getNumAtracAnimal(p_nombre varchar(45),p_apellidos
varchar(100), OUT p_cadena varchar(20))
   COMMENT 'Devuelve en p_cadena una cadena con el formato: nif:NumAtracciones:NumAnimales'
BEGIN
   DECLARE v_nif char(9);
   DECLARE v_numAnimales tinyint default 0;              -- Ponemos default para en el caso de
                                       que el SELECT no encuentre el artista no aparezca null
   DECLARE v_numAtracciones tinyint default 0;           -- Ponemos default para en el caso de
                                       que el SELECT no encuentre el artista no aparezca null

   SELECT nif
   INTO v_nif
   FROM ARTISTAS
   WHERE nombre = p_nombre AND apellidos = p_apellidos;

   CALL artistas_getNumAnimalesCuida(v_nif,v_numAnimales);          -- El método devuelve en
                                       v_numAnimales el número de animales que cuida el artista


   SELECT COUNT(*)
   INTO v_numAtracciones
   FROM ATRACCIONES_ARTISTAS
   WHERE fecha_fin IS NULL AND nif_artista = v_nif;

   SET p_cadena = CONCAT(v_nif,':',v_numAtracciones,':',v_numAnimales);
END$$
DELIMITER ;
```

Call examples:

```
USE CIRCO;
CALL artistas_getNumAtracAnimal('Carlos','Perez',@datos);
SELECT @datos;
```

4. Exercises of SP with input and output parameters

1. Create a procedure called pistas_addAforo that receives as parameters the name of the ring and an amount that represents the increase of the capacity.

The procedure must return in the same parameter the new capacity of the ring.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS pistas_addAforo;
DELIMITER $$
CREATE PROCEDURE pistas_addAforo(p_nombre varchar(50),INOUT p_incAforo smallint)
        COMMENT 'Devuelve en p_incAforo el nuevo aforo incrementado en la pista p_nombre'
BEGIN
   UPDATE PISTAS
   SET aforo = aforo + p_incAforo
   WHERE nombre = p_nombre;

   SELECT aforo
   INTO p_incAforo
   FROM PISTAS
   WHERE nombre = p_nombre;

END$$
DELIMITER ;
```

Call example:

```
USE CIRCO;
SET @dato = 50;       -- Incremento de aforo
CALL pistas_addAforo('LATERAL1',@dato);
SELECT @dato;
```

2. Create a procedure with the name artistas_getNombreCompleto that receives the nif as parameter and returns in the same parameter the full name with the format: surname, name.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS artistas_getNombreCompleto;
DELIMITER $$
CREATE PROCEDURE artistas_getNombreCompleto(INOUT p_dato varchar(147))
   COMMENT 'Devuelve el nombre y apellidos con el formato apellidos, nombre. p_dato tiene el
        nif del artista y se modifica con el nombre y apellidos'
BEGIN

   SELECT CONCAT(apellidos,', ',nombre)
   INTO p_dato
   FROM ARTISTAS
   WHERE nif = p_dato;

END$$
DELIMITER ;
```

Call example:

```
USE CIRCO;
SET @dato = '11111111A';
CALL artistas_getNombreCompleto(@dato);
SELECT @dato;
```

3. Create a procedure with the name animales_addAforo where the parameters received are the name of the animal and the increase in the capacity of the ring where the animal is working. It must make use of the stored procedure created in exercise 1 and must return using the two previous parameters, the name of the ring and its new capacity.

```
USE CIRCO;
DROP PROCEDURE IF EXISTS animales_addAforo;
DELIMITER $$
CREATE PROCEDURE animales_addAforo(INOUT p_nombre varchar(50),INOUT p_aforo
SMALLINT)
   COMMENT 'Devuelve el nombre de la pista y el nuevo aforo en el que trabaja el animal.
p_nombre indica el nombre del animal p_aforo indica el incremento del aforo'
   BEGIN

   SELECT nombre_pista
   INTO p_nombre
   FROM ANIMALES
   WHERE nombre = p_nombre;

   CALL pistas_addAforo(p_nombre, p_aforo);
END$$
DELIMITER ;
```

Call example:

```
USE CIRCO;

SET @nombre = 'Princesa1';
SET @aforo = 10;

CALL animales_addAforo(@nombre,@aforo);
SELECT @nombre,@aforo;
```

5. Exercises of functions

1. Create a function named utilidades_getMesEnLetra that receives a number and returns the name of the month. In case the number does not correspond to any month, it must return ‘No existe’.

```
USE CIRCO;
DROP FUNCTION IF EXISTS utilidades_getMesEnLetra;

DELIMITER $$
CREATE FUNCTION utilidades_getMesEnLetra (p_mes tinyint)
RETURNS VARCHAR(10) DETERMINISTIC NO SQL
   COMMENT 'Devuelve el mes en letra que se corresponde con el número de mes'
BEGIN
        DECLARE v_mesEnLetra varchar(10);               -- Valor por defecto NULL

        CASE p_mes
                WHEN 1 THEN SET v_mesEnLetra = 'ENERO';
                WHEN 2 THEN       SET v_mesEnLetra = 'FEBRERO';
                WHEN 3 THEN SET v_mesEnLetra = 'MARZO';
                WHEN 4 THEN SET v_mesEnLetra = 'ABRIL';
                WHEN 5 THEN SET v_mesEnLetra = 'MAYO';
                WHEN 6 THEN SET v_mesEnLetra = 'JUNIO';
                WHEN 7 THEN SET v_mesEnLetra = 'JULIO';
                WHEN 8 THEN SET v_mesEnLetra = 'AGOSTO';
                WHEN 9 THEN SET v_mesEnLetra = 'SEPTIEMBRE';
                WHEN 10 THEN SET v_mesEnLetra = 'OCTUBRE';
                WHEN 11 THEN SET v_mesEnLetra = 'NOVIEMBRE';
                WHEN 12 THEN SET v_mesEnLetra = 'DICIEMBRE';
      ELSE SET v_mesEnLetra = 'No existe';
        END CASE;

        RETURN v_mesEnLetra;
END $$
DELIMITER ;
```

Call examples:

- Call the function directly and store it on a session variable:

```
USE CIRCO;
SET @mes = utilidades_getMesEnLetra(10);
SELECT @mes;
```

```
USE CIRCO;
SET @mes = utilidades_getMesEnLetra(18);
SELECT @mes;
```

- Call the function so that it shows in letters in which months the attraction ‘El gran felino’ took place:

```
USE CIRCO;
SELECT DISTINCT MONTH(fecha) as mesNumero, utilidades_getMesEnLetra(MONTH(fecha)) as mesCadena
FROM ATRACCION_DIA
WHERE nombre_atraccion='El gran felino'
ORDER BY mesNumero;
```

2. Modify a previous exercise in which we created the procedure named 'atracciones_getNumPorMes' and create a function named utilidades_getMesEnNumero where the name of a month is passed and returns the number that corresponds to that name. If the month does not exist it should return -1. Modify the procedure so that it makes use of the function.

```
USE CIRCO;
DROP FUNCTION IF EXISTS utilidades_getMesEnNumero;

DELIMITER $$
CREATE FUNCTION utilidades_getMesEnNumero (p_mes varchar(10))
RETURNS INTEGER DETERMINISTIC NO SQL
   COMMENT 'Devuelve el número de mes que se corresponde con el nombre del mes'
BEGIN
        DECLARE v_mesEnNumero int default -1;           -- Valor por defecto de -1

        CASE p_mes
                WHEN 'ENERO' THEN SET v_mesEnNumero = 1;
                WHEN 'FEBRERO' THEN   SET v_mesEnNumero = 2;
                WHEN 'MARZO' THEN SET v_mesEnNumero = 3;
                WHEN 'ABRIL' THEN SET v_mesEnNumero = 4;
                WHEN 'MAYO' THEN SET v_mesEnNumero = 5;
                WHEN 'JUNIO' THEN SET v_mesEnNumero = 6;
                WHEN 'JULIO' THEN SET v_mesEnNumero = 7;
                WHEN 'AGOSTO' THEN SET v_mesEnNumero = 8;
                WHEN 'SEPTIEMBRE' THEN SET v_mesEnNumero = 9;
                WHEN 'OCTUBRE' THEN SET v_mesEnNumero = 10;
                WHEN 'NOVIEMBRE' THEN SET v_mesEnNumero = 11;
                WHEN 'DICIEMBRE' THEN SET v_mesEnNumero = 12;
        END CASE;

        RETURN v_mesEnNumero;
END $$
DELIMITER ;
```

Call examples:

```
USE CIRCO;
 SET @mes = utilidades_getMesEnNumero(‘ENERO’);

SELECT @mes;
```

3. Create a function named animales_getEstadoPorAnhos that returns the following string:

If type = Lion

years < 2: 'JOVEN'

years >=2 and <=5: 'MADURO'

years > 5: 'VIEJO'

Any other type:

years < 1: 'JOVEN'

years >=1 and <=3: 'MADURO'

years > 3: 'VIEJO'

Call the function to show the status by year of each of the CIRCUS animals.

```
USE CIRCO;
DROP FUNCTION IF EXISTS animales_getEstadoPorAnhos;
DELIMITER $$
CREATE FUNCTION animales_getEstadoPorAnhos (p_tipo varchar(9), p_anhos tinyint)
RETURNS CHAR(6) DETERMINISTIC NO SQL
   COMMENT 'Devuelve una cadena indicativa de la edad en función de la edad y tipo de animal'
BEGIN
        DECLARE v_cadena char(6) default '';

        IF (p_tipo='León') THEN
                CASE
                   WHEN p_anhos < 2 THEN SET v_cadena = 'JOVEN';
                   WHEN p_anhos >= 2 AND p_anhos <= 5 THEN SET v_cadena = 'MADURO';
                   WHEN p_anhos > 5 THEN SET v_cadena = 'VIEJO';
                END CASE;
        ELSE
                CASE
                    WHEN p_anhos < 1 THEN SET v_cadena = 'JOVEN';
                    WHEN p_anhos >= 1 AND p_anhos <= 3 THEN SET v_cadena = 'MADURO';
                    WHEN p_anhos > 3 THEN SET v_cadena = 'VIEJO';
                END CASE;
        END IF;

        RETURN v_cadena;
END $$
DELIMITER ;
```

Call examples:

```
USE CIRCO;
SELECT *,animales_getEstadoPorAnhos(tipo, anhos) as estado
FROM ANIMALES
ORDER BY nombre;
```

4. Create a function named pistas_getDiferenciaAforo in which you pass the new capacity of a ring and it returns the difference between the new capacity and the previous capacity.

- **If difference < 100 it should return the string 'PEQUEÑA menor que 100'.**
- **If the difference is between 100 and 500 it should return the string 'REGULAR entre 100 y 500'.**
- **If the difference > 500 it should return the string 'ABISMAL mayor que 500'.**

Example: TRACK1, 150 => If the ring currently has a capacity of 100, it should return 150-100 = 50 => 'PEQUEÑA menor que 100'. If the ring does not exist, it should return NULL.

Remember to add the appropriate modifiers when creating the function.

```
USE CIRCO;
DROP FUNCTION IF EXISTS pistas_getDiferenciaAforo;

DELIMITER $$
CREATE FUNCTION pistas_getDiferenciaAforo (p_nombrePista varchar(50), p_aforo smallint)
        -- El tipo se corresponde con el de la tabla
RETURNS varchar(100) READS SQL DATA
   COMMENT 'Devuelve la diferencia entre el nuevo aforo y el antiguo'
BEGIN
   DECLARE v_aforoAntiguo smallint default -1;
   DECLARE v_cadena varchar(100);
   DECLARE v_diferenciaAforo smallint default 0;

   SELECT aforo
   INTO v_aforoAntiguo
   FROM PISTAS
   WHERE nombre = p_nombrePista;

   IF (v_aforoAntiguo=-1) THEN     -- La pista no existe
      RETURN v_cadena;
   END IF;

   SET v_diferenciaAforo = p_aforo-v_aforoAntiguo;
   CASE
      WHEN v_diferenciaAforo < 100 THEN
         SET v_cadena = 'PEQUEÑA menor que 100';
      WHEN v_diferenciaAforo >= 100 AND v_diferenciaAforo <= 500  THEN
         SET v_cadena = 'REGULAR entre 100 y 500';
      WHEN v_diferenciaAforo > 500 THEN
      SET v_cadena = 'ABISMAL mayor que 500';
   END CASE;
```

```
 RETURN v_cadena;

END $$
DELIMITER ;
```

Call examples:

- Show the data of all rings along with the difference in capacity using the previous function and sending a new capacity of 600:

```
USE CIRCO;
SELECT *,pistas_getDiferenciaAforo(nombre,600) as estado
FROM PISTAS
ORDER BY nombre;
```

- Show the rings that have an ABISMAL difference (search for the chain with the INSTR function with a proposed capacity of 1000).

```
USE CIRCO;
SELECT *
FROM PISTAS
WHERE INSTR (pistas_getDiferenciaAforo(nombre,1000), 'ABISMAL' COLLATE
'utf8mb4_spanish2_ci') > 0
ORDER BY nombre;
```