

UT3 - Prácticas- Procedimientos y funciones

Diagramas lógicos de B.D

Los ejercicios se realizarán sobre la base de datos ventas y ciclistas.

DIAGRAMA LÓGICO BASE DE DATOS CICLISTAS

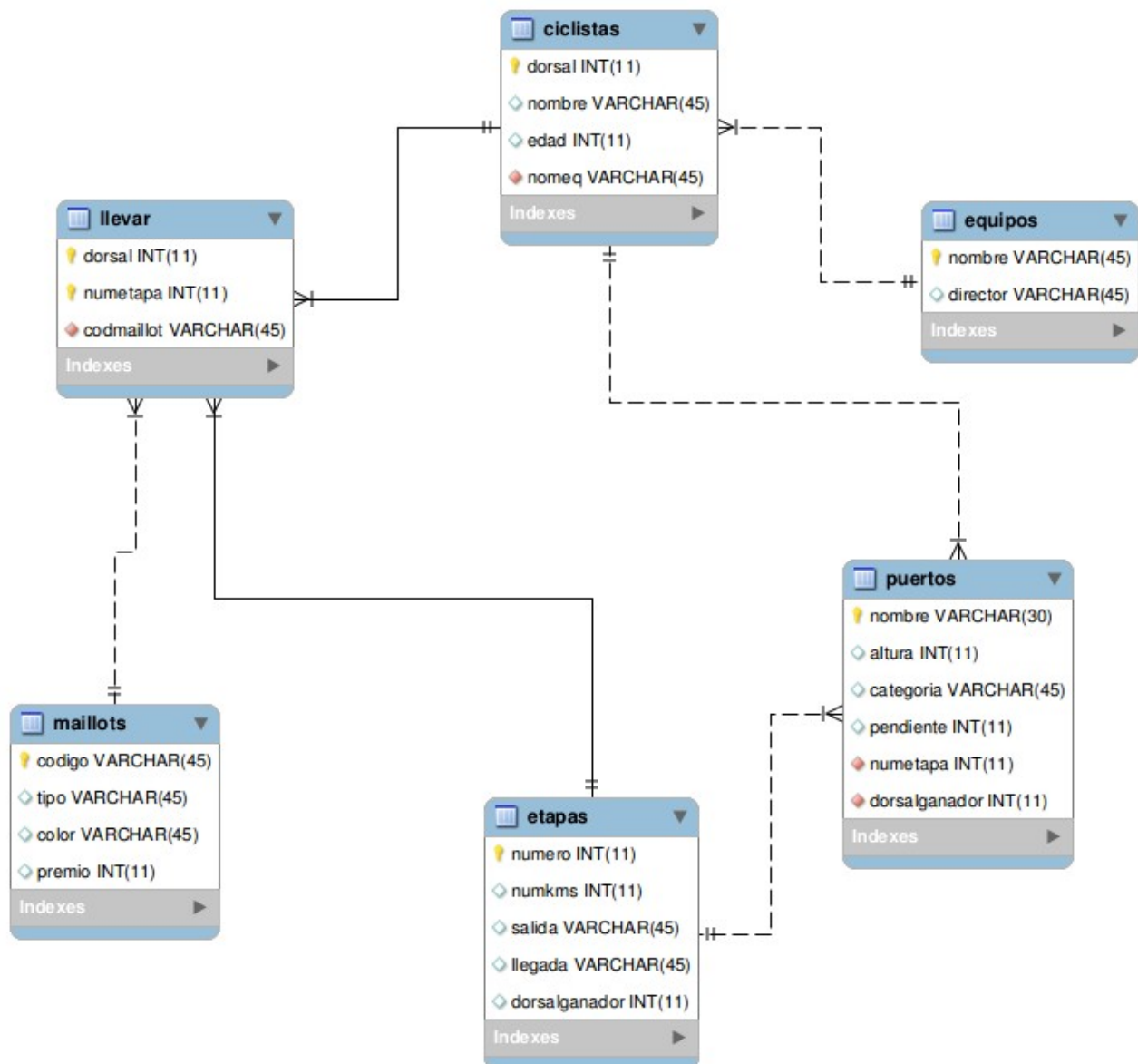
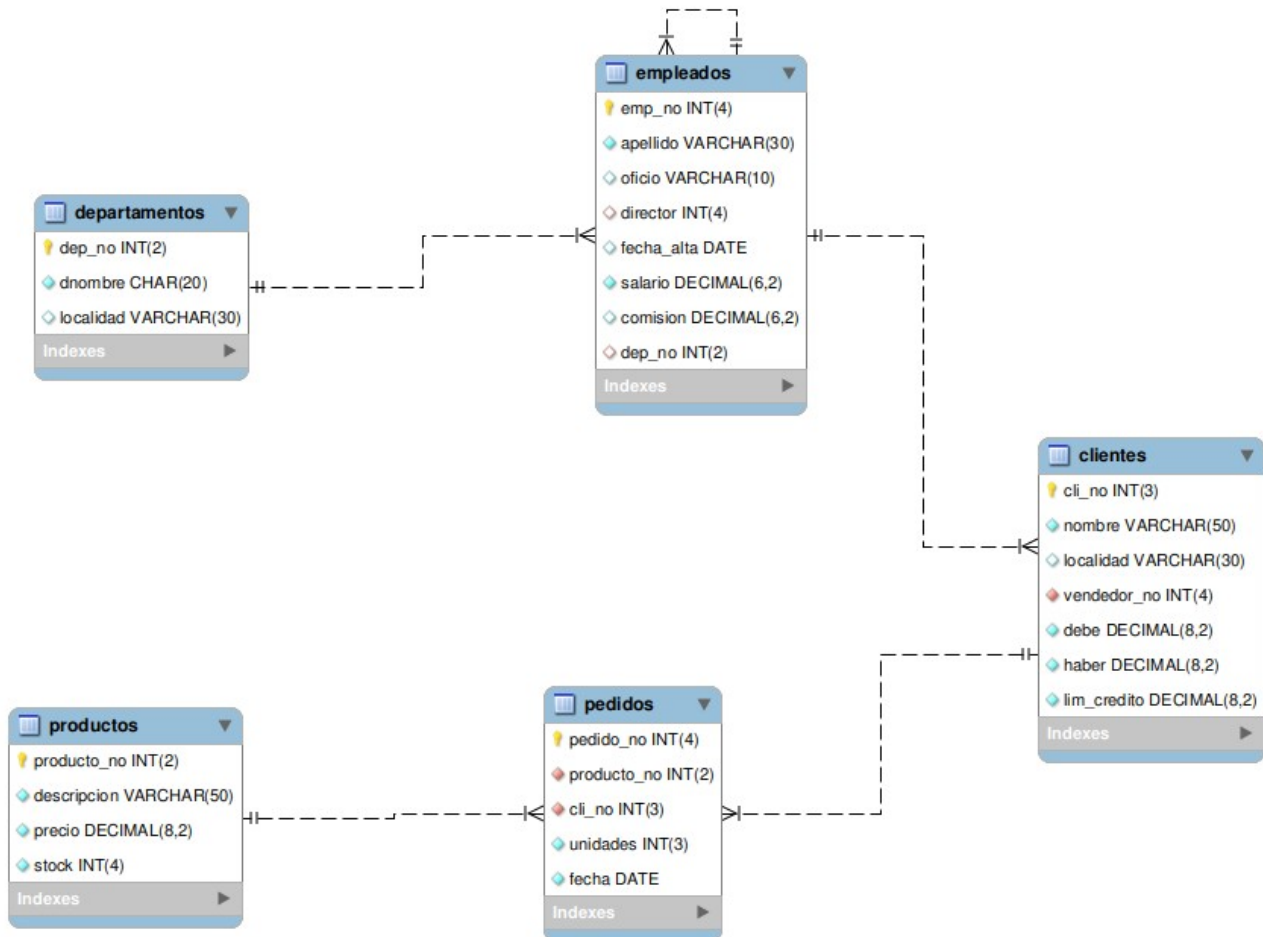


DIAGRAMA LÓGICO BASE DE DATOS VENTAS



Se deben poner también los pantallazos de los resultados.

1 EJERCICIOS PROCEDIMIENTOS

1.1 Sobre la base de ventas crea un procedimiento para mostrar el año actual.

Nota: Para obtener la fecha del sistema tenemos: CURRENT_DATE(), CURDATE(), NOW(). Para obtener el año YEAR(fecha)

Solución:

```
use ventasleon;  
drop procedure if exists anioactual;  
create procedure anioactual()  
    select year(now()) as anioactual;  
  
delimiter ;  
call anioactual();
```

Resultado:

```
CALL anioactual();  
2020
```

1.2 Crea un procedimiento que muestre las tres primeras letras de una cadena pasada como parámetro y en mayúsculas. Utiliza la base de datos Ventas.

Nota: Para obtener las n primeras letras de una cadena temenos: RIGTH(cad, n_letras). Para convertir una cadena en mayúsculas: UPPER(cadena)

Solución:

```
use ventasleon;  
drop procedure if exists anioactual;  
create procedure anioactual(in cadena varchar(100))  
    select left(upper(cadena),3) as tresLetras;  
call anioactual('Monteigueldo');
```

Resultado:

```
CALL anioactual('Monteigueldo');  
MON
```

1.3 Crea un procedimiento que muestre dos cadenas pasadas como parámetros concatenadas y en mayúsculas. Utiliza la base de datos Ventas.

Nota: Para concatenar dos cadenas: CONCAT(cad1, cad2 [, cadn]).

Solución:

```
use ventasleon;  
drop procedure if exists concatmayus;  
create procedure concatmayus(in cadena1 varchar(100), in cadena2  
varchar(100))  
    select upper(concat(cadena1, ' ', cadena2)) as palabras;  
call concatmayus('hola','mundo');
```

Resultado:

```
CALL concatmayus ('hola', 'mundo');  
HOLA MUNDO
```

1.4 Crea un procedimiento que sume uno a la variable que reciba como parámetro (para probarlo usa la variable de usuario anterior y ver que pasa cada vez que se ejecuta). Utiliza la base de datos Ventas.

Solución.

```
use ventasleon;  
drop procedure if exists sumauno;  
create procedure sumauno(inout contador int )  
    set contador = contador+1;  
set @a=1;  
call sumauno(@a);  
select @a;
```

Resultado:

```
SET @a=1;  
CALL sumauno (@a);
```

```
SELECT @a;  
2
```

1.5 Crea un procedimiento que devuelva el nombre de los ciclistas que pertenezcan al mismo equipo que un corredor que se le pasa por parámetro. Utiliza la base de datos Ciclistas.

Solución:

```
use CICLISTASCOMPLETA;  
drop procedure if exists CorredoresCompanero;  
create procedure CorredoresCompanero(in nombreCiclista varchar(100))  
    select nombre from ciclistas where nomeq = (select nomeq from  
ciclistas where nombre = nombreCiclista);  
CALL CorredoresCompanero('Miguel Induráin');
```

Resultado:

```
call CorredoresCompañero('Miguel Induráin');
```

```
+-----+  
| nombre |  
+-----+  
| Miguel Induráin |  
| Pedro Delgado |  
| Laurent Jalabert |  
| Per Pedersen |  
+-----+  
4 rows in set (0.00 sec)
```

1.6 Crea un procedimiento que devuelva el nombre de los ciclistas que han ganado más del número de puertos que se le pasa por parámetro. Utiliza la base de datos Ciclistas.

Solución:

```
use CICLISTASCOMPLETA;
drop procedure if exists CorredoresGanadoresPuertos;
create procedure CorredoresGanadoresPuertos(in numero int)
    select ciclistas.nombre, count(ciclistas.nombre ) as puertos from
    ciclistas join puertos on ciclistas.dorsal = puertos.dorsal ganador group by
    ciclistas.nombre having puertos >numero order by puertos,
    ciclistas.nombre;
call CorredoresGanadoresPuertos(1);
```

Resultado

```
call CorredoresGanadoresPuertos(1);
```

```
+-----+-----+
| nombre          | count(p.dorsal ganador) |
+-----+-----+
| Alex Zulle      | 2 |
| Miguel Induráin | 2 |
| Pedro Delgado   | 3 |
+-----+-----+
3 rows in set (0.58 sec)
```

1.7 Crea un procedimiento que devuelva el valor del atributo netapa de aquellas etapas tales que todos los puertos que están en ellas tenga más

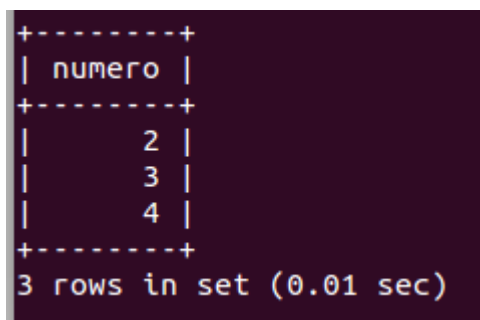
los metros que se le pasa por parámetro. Utiliza la base de datos Ciclistas.

Solución:

```
use CICLISTASCOMPLETA;
drop procedure if exists EtapasPuertosAltura;
create procedure EtapasPuertosAltura(in alturaPuerto int)
    select etapas.numero from etapas where etapas.numero not in
(select numetapa from puertos where altura<=alturaPuerto) and
etapas.numero in (select numetapa from puertos where altura
>alturaPuerto) order by etapas.numero;
call EtapasPuertosAltura(700);
```

Resultado

```
call EtapasPuertosAltura(700);
```



```
+-----+
| numero |
+-----+
|      2 |
|      3 |
|      4 |
+-----+
3 rows in set (0.01 sec)
```

1.8 Crear un procedimiento que devuelva el nombre y el director de los equipos tales que todos sus ciclistas son mayores de una edad que se pasa por parámetro. Utiliza la base de datos Ciclistas.

Solución:

```
use CICLISTASCOMPLETA;
drop procedure if exists EdadEquipos;
create procedure EdadEquipos(in edadCiclista int)
    select nombre, director from equipos where nombre not in (select
    nomeq from ciclistas where edad < edadCiclista) and nombre in (select
    nomeq from ciclistas where edad >= edadCiclista) order by nombre;
call EdadEquipos(20);
```

Resultado

```
call EdadEquipos(20);
```

```
+-----+-----+
| nombre      | director      |
+-----+-----+
| Banesto     | Miguel Echeverría |
| Bresciale-Refin | Pietro Armani  |
| Carrera     | Luigi Petroni   |
| Gatorade     | Gian Luca Pacelli |
| Kelme       | Álvaro Pino     |
| Mapei-Clas   | Juan Fernández  |
| Navigare     | Lorenzo Sciacchi |
| TVM         | Steevens Henk   |
+-----+-----+
8 rows in set (0.00 sec)
```

1.9 Crear un procedimiento que devuelva el valor del atributo netapa, la ciudad de salida y la ciudad de llegada de las etapas de más de un determinado número de kilómetros y un mínimo de puertos que se pasan por parámetros. Utiliza la base de datos Ciclistas.

Solución:

```
use CICLISTASCOMPLETA;
drop procedure if exists EtapaKilometrosPuertos;
create procedure EtapaKilometrosPuertos(in km int, in numPuertos int)

    select  numero,  salida,  llegada  from  (select  etapas.numero,
count(etapas.numero) as num_etapas , etapas.salida , etapas.llegada from
etapas join puertos on etapas.numero = puertos.numetapa where
puertos.altura >km group by etapas.numero having
num_etapas>=numPuertos) as sel1;
call EtapaKilometrosPuertos(190,2);
```

Resultado

```
call EtapaKilometrosPuertos(190,2);
```

```
+-----+-----+-----+
| numero | salida  | llegada |
+-----+-----+-----+
|      4 | Córdoba | Granada |
+-----+-----+-----+
1 row in set (0.00 sec)
```

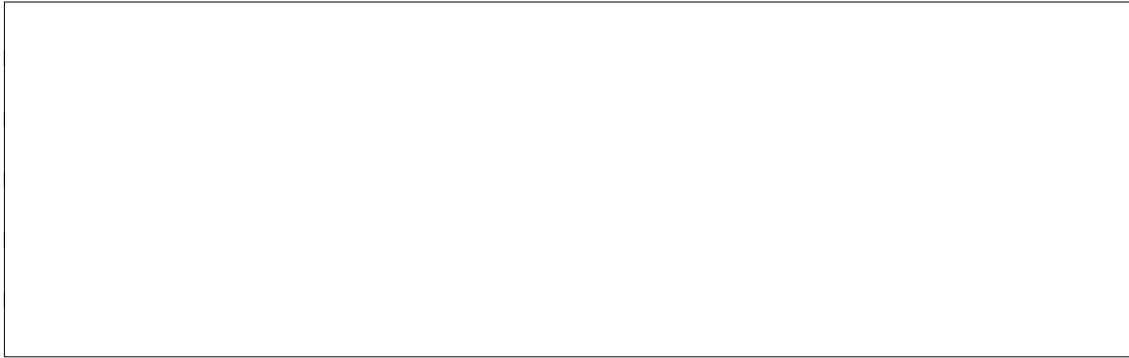
2 EJERCICIOS FUNCIONES

2.1 Crea una función que devuelva el valor de la hipotenusa de un triángulo a partir de los valores de sus lados. Utiliza la base de datos Ventas.

Nota: POW (x, y) o POWER (x, y) obtiene el valor de x elevado a la potencia y. SQRT(x) calcula la raíz cuadrada de un número.

Solución

```
use ventasleon;
drop function if exists hipotenusa;
delimiter //
create function hipotenusa(a int, b int)
returns double
deterministic
begin
    declare resultado double;
    set resultado = sqrt(pow(a,2)+pow(b,2));
    return resultado;
end//
delimiter ;
select hipotenusa(5,7);
```



Resultado:

```
select hipotenusa (5,7);  
8.602325439453125
```

2.2 Crea una función que calcule el total de puntos en un partido de baloncesto tomando como entrada el resultado en formato 'xxx-xxx'. Utiliza la base de datos Ventas.

Nota: SUBSTRING_INDEX(str,delim,count) extrae cadenas en función de delimitadores. Devuelve la subcadena de str anterior a la aparición de count veces el delimitador delim. Si count es positivo, se retorna todo lo que haya a la izquierda del delimitador final (contando desde la izquierda). Si count es negativo, se devuelve todo lo que haya a la derecha del delimitador final (contando desde la derecha).

Solución:

```
use ventasleon;  
drop procedure if exists total;  
delimiter //  
create function total(puntos varchar(10))  
returns int  
deterministic  
begin  
    declare resultado int;  
    set resultado = substring_index(puntos,"-", 1) +  
substring_index(puntos,"-", -1);  
    return resultado;  
end//
```

```
delimiter ;  
select total('104-20');
```

Resultado

```
select total('102-127');  
229
```

2.3 Crea una función que calcule el número de años transcurridos entre 2 fechas donde la primera es menor que la segunda (para probarlo calcula tu edad introduciendo tu fecha de nacimiento y la fecha actual).

Solución:

```
use ventasleon;  
drop function datesDiff;  
delimiter //  
create function datesDiff(fecha1 varchar(20), fecha2 date)  
returns int  
deterministic  
begin  
    declare anios int;  
    set anios = timestampdiff(year,fecha1,fecha2);  
    return anios;  
end//  
delimiter ;  
select datesDiff('1997-01-18',curdate());
```

Resultado

```
select datesDiff("1993-12-09", curdate());
```

2.4 Crea una función que devuelva la edad media de los ciclistas que han ganado alguna etapa. Utiliza la base de datos Ciclistas.

Solución:

```

use CICLISTASCOMPLETA;
drop function EdadMedia;
delimiter //
create function EdadMedia()
returns float
deterministic
begin
    declare anios float;
    select avg(distinct ciclistas.edad) into anios from ciclistas join puertos on
ciclistas.dorsal = puertos.dorsalganador;
    return anios;
end//
delimiter ;
select EdadMedia();

```

Resultado

```
call CorredoresGanadoresPuertos(1);
```

```

+-----+-----+
| nombre          | count(p.dorsalganador) |
+-----+-----+
| Alex Zulle      | 2 |
| Miguel Induráin | 2 |
| Pedro Delgado   | 3 |
+-----+-----+
3 rows in set (0.58 sec)

```

3 EJERCICIOS IF Y CASE

3.1 Crea una función que reciba 2 números enteros y devuelva 1, si el primer número es divisible por el otro y 0, en caso contrario. Utiliza la base de datos Ventas.

Nota: MOD(x,y) ó x%y ó xMODy es el resto de una división entre x e y.

Solución

```
use ventasleon;
drop function divisible;
delimiter //
create function divisible(a int, b int)
returns int
deterministic
begin
    declare resto int;
    set resto = MOD(a,b);
    if resto = 0 then
        return 1;
    else
        return 0;
    end if;
end//
delimiter ;
select divisible(6,5);
```

Resultado

```
SELECT divisible(6,5);
```

3.2 Crea una función, usando las estructuras condicionales, que devuelva el literal del día de la semana según un valor de entrada numérico, 1 para lunes, 2 martes, etc. Utiliza la base de datos Ventas.

Solución:

```
use ventaseon;
drop function if exists diasemana;
delimiter //
create function diasemana(dia int)
returns varchar(10)
deterministic
begin
    if dia=1 then
        return "Lunes";
    elseif dia=2 then
        return "Martes";
    elseif dia=3 then
        return "Miercoles";
    elseif dia=4 then
        return "Jueves";
    elseif dia=5 then
        return "Viernes";
    elseif dia=6 then
        return "Sábado";
    else
        return "Domingo";
    end if;
end//
delimiter ;
select diasemana(3);
```

Resultado:

```
SELECT diasemana(3);
Martes
```


| |
|--|
| |
|--|

Solución CASE

```
use ventasleon;
drop function if exists diasemana;
delimiter //
create function diasemana(dia int)
returns varchar(10)
deterministic
begin
    case dia
    when 1 then
        return 'Lunes';
    when 2 then
        return 'Martes';
    when 3 then
        return 'Miercoles';
    when 4 then
        return 'Jueves';
    when 5 then
        return 'Viernes';
    when 6 then
        return 'Sábado';
    else
        return 'Domingo';
    end case;
end//
delimiter ;
select diasemana(4);
```

3.3 Crea una función que devuelva el mayor de tres números pasados como parámetros, sin usar la función GREATEST. Utiliza la base de datos Ventas.

Solucion

```
use ventasleon;
drop function if exists mayor;
delimiter //
create function mayor(a int, b int, c int)
returns int
deterministic
begin
    if a=b and b=c then
        return a;
    elseif a > b then
        if a > c then
            return a;
        else
            return c;
        end if;
    elseif b > a then
        if b > c then
            return b;
        else
            return c;
        end if;
    else
        return c;
    end if;
end//
delimiter ;
select mayor(4,5,2);
```

Resultado

```
select mayor(34,56,4);  
  
56
```

3.4 Crea una función que devuelva 1 si ganó el de casa, 2 el visitante y 0 si empataron. El parámetro de entrada es el resultado con el formato 'xxx-xxx'. Utiliza la base de datos Ventas.

Solución:

```
use ventasleon;  
drop function ganador;  
delimiter //  
create function ganador(puntos varchar(7))  
returns int  
deterministic  
begin  
    declare casa int;  
    declare visitante int;  
    set casa = substring_index(puntos,'-',1);  
    set visitante = substring_index(puntos,'-',-1);  
    if casa=visitante then  
        return 0;  
    elseif casa>visitante then  
        return 1;  
    else  
        return 2;  
    end if;  
end//  
delimiter ;  
select ganador('111-23');
```

Resultado:

```
select ganador('102-102');  
0
```

3.5 Crea una función que diga si una palabra, pasada como parámetro, es palíndroma (por ejemplo: radar, reconocer, sometemos, ...). Devuelve 1 si es palíndroma y 0 en caso contrario. Utiliza la base de datos Ventas.

Nota: REVERSE (cad) invierte el orden de los caracteres de la cadena cad.

Solución:

```
use ventasleon;  
drop function palindrome;  
delimiter //  
create function palindrome(palabra varchar(30))  
returns int  
deterministic  
begin  
    declare reverso varchar(30);  
    set reverso = reverse(palabra);  
    if palabra = reverso then  
        return 1;  
    else  
        return 0;  
    end if;  
end//  
delimiter ;  
select palindrome('aua');
```

Resultado

```
SELECT palindrome('aua');
```

```
1
```

3.6 Crea una función que devuelva el valor “viejo” si la media de edad de los corredores de un equipo que se le pasa por parámetro es mayor que la media total de todos los corredores. Devolverá “joven” si la media del equipo es menor.

Solución:

```
use CICLISTASCOMPLETA;
drop function if exists EdadEquipo;
delimiter //
create function EdadEquipo(nombreEquipo varchar(20))
returns varchar(5)
deterministic
begin
    declare mediaEquipo float;
    declare mediaTotal float;
    select avg(edad) into mediaEquipo from ciclistas where
nomeq=nombreEquipo;
    select avg(edad) into mediaTotal from ciclistas;
    if mediaEquipo > mediaTotal then
        return 'viejo';
    elseif mediaEquipo < mediaTotal then
        return 'joven';
    else
        return 'error';
    end if;
end//
delimiter ;
select EdadEquipo('TVM');
```

Resultado:

```
mysql> select EdadEquipo('TVM');
+-----+
| EdadEquipo('TVM') |
+-----+
| Viejo             |
+-----+
1 row in set (0.00 sec)
```

```
mysql> select EdadEquipo('Amore Vita');
+-----+
| EdadEquipo('Amore Vita') |
+-----+
| Joven                    |
+-----+
1 row in set (0.01 sec)
```

4 EJERCICIOS LOOP

4.1 Crea un procedimiento que muestre por pantalla los números del 1 al 10.

```
use ventasleon;
drop procedure if exists mostrarnumeros;
delimiter //
create procedure mostrarnumeros()
begin
    declare count int;
    declare numeros varchar(50);
    set count=1;
    set numeros = "->";
    bucle:LOOP
    if count>10 then LEAVE bucle;
    end if;
    set numeros = concat(numeros,' ', count);
```

```
        set count = count +1;
    end LOOP bucle;
    select numeros;
end//
delimiter ;
call mostrarnumeros();
```

4.2 Crea un procedimiento que reciba dos números y haga la cuenta atrás desde el mayor hasta el menor. El orden de los números es aleatorio, hay que comprobar cual es el mayor y el menor.

Solución

```
use ventasleon;
drop procedure cuenta_atras;
delimiter //
create procedure cuenta_atras(in x int, in y int)
begin
    declare mayor, menor, contador int;
    select GREATEST(x,y) into mayor;
    select least(x,y) into menor;
    set contador = mayor;
    bucle: LOOP
        if contador<menor then leave bucle;
        end if;
        select contador as max;
        set contador = contador-1;
    end loop bucle;
end//
delimiter ;
```

```
call cuenta_atras(4,9);
```

Resultado

```
mysql> call cuenta_atras(6,4);
+-----+
| max |
+-----+
|    6 |
+-----+
1 row in set (0.00 sec)

+-----+
| max |
+-----+
|    5 |
+-----+
1 row in set (0.00 sec)

+-----+
| max |
+-----+
|    4 |
+-----+
1 row in set (0.00 sec)
```

4.3 Necesitamos tener una tabla que contenga, al menos, los siguientes campos:

- uno que almacene las fechas correspondientes desde el día de hoy hasta dentro de X días (dato que nos pasarán por parámetro),
- otro campo que sea el día de la semana al que corresponde (Monday, Tuesday, ...)
- otro campo que sea el número del día del año que corresponde (1 a 366).

¿Qué se te ocurre hacer?

Solución.


```

use ventasleon;
drop procedure if exists genera_fechas_en;
delimiter //
create procedure genera_fechas_en(in dias int)
begin
    declare contador int default 0;
    drop table if exists fechas_en;
    create table fechas_en(Id int primary key auto_increment,
        Fecha date, Nombre varchar(10), NumDia int);
    bucle: LOOP
        if contador =dias then LEAVE bucle;
        end if;
        insert into fechas_en(Fecha, Nombre, NumDia)
values(date_add(curdate(), interval contador day), date_format(
    date_add(curdate(), interval contador day),"%W"),
    date_format( date_add(curdate(), interval contador day),'%j' ));
        set contador = contador +1;
    end loop bucle;
    select * from fechas_en;
end //
delimiter ;
call genera_fechas_en(5);

```

Resultado

```

mysql> call genera_fechas_en(5);
Query OK, 1 row affected (0.60 sec)

mysql> select * from fechas_en;
+----+-----+-----+-----+
| Id | Fecha       | Nombre    | NumDia |
+----+-----+-----+-----+
| 1  | 2021-01-12 | Tuesday   | 12     |
| 2  | 2021-01-13 | Wednesday | 13     |
| 3  | 2021-01-14 | Thursday  | 14     |
| 4  | 2021-01-15 | Friday    | 15     |
| 5  | 2021-01-16 | Saturday  | 16     |
+----+-----+-----+-----+
5 rows in set (0.00 sec)

```

5 EJERCICIOS ITERATE

5.1 Dado un número, saca una lista con todos los números inferiores a él que son divisibles por 8. Por ejemplo, si el número pasado como parámetro es el 32, nos deberá devolver lo siguiente: 24|16|8|.

```
use ventasleon;
drop function if exists divisibles;
delimiter //
create function divisibles(numero int)
returns varchar(100)
deterministic
begin
    declare numeros varchar(100) default "";
    bucle: loop
        set numero = numero -1;
        if MOD(numero,8)=0 and numero >0 then
            set numeros = concat(numeros, numero, '|');
        end if;
        if numero>0 then iterate bucle;
        else leave bucle;
        end if;
    end loop bucle;
    return numeros;
end//
delimiter ;
select divisibles(32);
```

5.2 Realiza el ejercicio anterior controlando que el número introducido sea superior a 8.

```
use ventasleon;
drop function if exists divisibles;
```

```
delimiter //
create function divisibles(numero int)
returns varchar(100)
deterministic
begin
    declare numeros varchar(100) default "";
    if numero<=8 then
        set numeros = "Introduce un numero mayor";
    else
        bucle: loop
            set numero = numero -1;
            if MOD(numero,8)=0 and numero >0 then
                set numeros = concat(numeros, numero, '|');
            end if;
            if numero>0 then iterate bucle;
            else leave bucle;
            end if;
        end loop bucle;
    end if;

    return numeros;
end//
delimiter ;
select divisibles(32);
```

5.3 Realiza el ejercicio anterior pero que en vez de ser la división entre 8, sea entre cualquier número que se desee y se pase por parámetro.

```
use ventasleon;
drop function if exists divisibles;
delimiter //
create function divisibles(numero int, divisor int)
```

```
returns varchar(100)
deterministic
begin
    declare numeros varchar(100) default "";
    if numero<=divisor then
        set numeros = "Introduce un dividendo mayor";
    else
        bucle: loop
            set numero = numero -1;
            if MOD(numero,divisor)=0 and numero >0 then
                set numeros = concat(numeros, numero, '|');
            end if;
            if numero>0 then iterate bucle;
            else leave bucle;
            end if;
        end loop bucle;
    end if;

    return numeros;
end//
delimiter ;
select divisibles(32);
```

6 EJERCICIOS REPEAT

6.1 Crea un procedimiento que muestre los “n” primeros números enteros, siendo “n” un parámetro de entrada.

```
use ventasleon;
drop procedure if exists nprimeros;
delimiter //
create procedure nprimeros(in n int)
```

```
begin
    declare resultado varchar(150) default '';
    declare contador int default 1;
    repeat
        set resultado = concat(resultado, contador, ' ');
        set contador = contador + 1;
    until contador > n
    end repeat;
    select resultado;
end//
delimiter ;
call nprimeros(5);
```

7 EJERCICIOS WHILE

7.1 Crea un procedimiento que muestre los “n” primeros números enteros, siendo “n” un parámetro de entrada.

```
use ventasleon;
drop procedure if exists nwhile;
delimiter //
create procedure nwhile(in n int)
begin
    declare resultado varchar(150) default '';
    declare contador int default 1;
    while contador <= n DO
        set resultado = concat(resultado, contador, ' ');
        set contador = contador + 1;
    end while;
    select resultado;
end//
delimiter ;
```

```
call nwhile(10);
```

7.2 Crea un procedimiento que muestre la suma de los primeros n números enteros, siendo “n” un parámetro de entrada.

```
use ventasleon;
drop procedure if exists sumarPrimeros;
delimiter //
create procedure sumarPrimeros(in n int)
    begin
        declare resultado int default 0;
        declare contador int default 1;
        while contador <=n DO
            set resultado = resultado + contador;
            set contador = contador +1;
        end while;
        select resultado;
    end//
delimiter ;
call sumarPrimeros(2);
```

7.3 Crea una función que determine si un número es primo (un número es primo cuando no es divisible entre ningún otro número menor que él, salvo sí mismo y la unidad) y devuelva un mensaje de texto indicándolo.

```
use ventasleon;
drop function if exists comprobarPrimo;
delimiter //
create function comprobarPrimo(primo int)
returns varchar(20)
deterministic
begin
```

```

declare contador int;
declare isPrimo int default 0;
set contador = primo;
while (isPrimo <3) and (contador>0) do
    if mod(primo, contador)=0 then
        set isPrimo = isPrimo +1 ;
    end if;
    set contador = contador -1;
end while;
if isPrimo >=3 then
    return "No es primo";
else
    return "Es primo";
end if;
end//
delimiter ;
select comprobarPrimo(8);

```

7.4 Crea una función que determine si un número es primo (un número es primo cuando no es divisible entre ningún otro número menor que él, salvo sí mismo y la unidad) y devuelva un booleano indicándolo.

```

use ventasleon;
drop function if exists comprobarPrimo;
delimiter //
create function comprobarPrimo(primo int)
returns boolean
deterministic
begin
    declare contador int;
    declare isPrimo int default 0;
    set contador = primo;

```

```
while (isPrimo <3) and (contador>0) do
    if mod(primo, contador)=0 then
        set isPrimo = isPrimo +1 ;
    end if;
    set contador = contador -1;
end while;
if isPrimo >=3 then
    return false;
else
    return true;
end if;
end//
delimiter ;
select comprobarPrimo(8);
```