EJERCICIOS PROCEDIMIENTOS Y FUNCIONES.

Sobre la base de datos EMPRESA realiza los siguientes apartados:

1. Realiza un procedimiento ***mostrar_empleados_hijos,*** que muestre todos los empleados que tienen hijos.

```
DELIMITER //

CREATE PROCEDURE mostrar_empleados_hijos()
BEGIN
   SELECT e.*
   FROM empleado e
   WHERE e.CodEmp IN (SELECT DISTINCT CodEmp FROM hijo);
END //

DELIMITER ;
```

2. Realiza un procedimiento ***contar_empleados*** que muestre el número de empleados.

```
DELIMITER //

CREATE PROCEDURE contar_empleados()
BEGIN
   SELECT COUNT(*) AS total_empleados FROM empleado;
END //

DELIMITER ;
```

3. Realiza una función que realice lo mismo que el procedimiento anterior (***contar_empleados)***

```
DELIMITER //

CREATE FUNCTION total_empleados() RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE total INT;
```

```
   SELECT COUNT(*) INTO total FROM empleado;
   RETURN total;
END //

DELIMITER ;
```

4. Realiza un procedimiento ***contarEmpleadosDpto*** que devuelva la cantidad de empleados de un determinado departamento (introducido como parámetro de entrada) en un parámetro de salida.

```
DELIMITER //

CREATE PROCEDURE contarEmpleadosDpto(IN cod_dep CHAR(5), OUT total INT)
BEGIN
   SELECT COUNT(*) INTO total
   FROM empleado
   WHERE CodDep = cod_dep;
END //

DELIMITER ;
```

5. Realiza una función que realice lo mismo que el procedimiento anterior (***contarEmpleadosDpto)***

```
DELIMITER //

CREATE FUNCTION total_empleados_dpto(cod_dep CHAR(5)) RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE total INT;
   SELECT COUNT(*) INTO total FROM empleado WHERE CodDep = cod_dep;
   RETURN total;
END //

DELIMITER ;
```

6. Realiza un procedimiento ***habilidades*** que reciba como entrada un entero con una cantidad, e inserte tantas habilidades como el parámetro recibido, asignando como código **BD-NNN** y descripción **Habilidad NNN**, sustituyendo NNN por un número entero secuencial.
   Ejemplo: Si el parámetro es 3, insertará las habilidades BD-1, BD-2 y BD-3 y las descripciones Habilidad 1, Habilidad 2 y Habilidad 3.
   Controlar que el código de habilidad a insertar no exista ya en la tabla, ya que daría un error de clave duplicada.

```
DELIMITER //

CREATE PROCEDURE habilidades(IN cantidad INT)
BEGIN
   DECLARE i INT DEFAULT 1;
   DECLARE codigo VARCHAR(10);

   WHILE i <= cantidad DO
      SET codigo = CONCAT('BD-', i);
      IF NOT EXISTS (SELECT 1 FROM habilidad WHERE CodHab = codigo) THEN
         INSERT INTO habilidad(CodHab, DesHab)
         VALUES (codigo, CONCAT('Habilidad ', i));
      END IF;
      SET i = i + 1;
   END WHILE;
END //

DELIMITER ;
```

7. Realiza un procedimiento ***habilidadesInicioFin*** que reciba como entrada un entero que indique el número de inicio y otro con el tope (o fin), e inserte habilidades cuyos valores vayan desde inicio hasta fin, con la misma nomenclatura que el apartado anterior.

```
DELIMITER //

CREATE PROCEDURE habilidadesInicioFin(IN inicio INT, IN fin INT)
BEGIN
   DECLARE i INT;
   SET i = inicio;

   WHILE i <= fin DO
```

```
      SET @codigo = CONCAT('BD-', i);
      IF NOT EXISTS (SELECT 1 FROM habilidad WHERE CodHab = @codigo) THEN
         INSERT INTO habilidad(CodHab, DesHab)
         VALUES (@codigo, CONCAT('Habilidad ', i));
      END IF;
      SET i = i + 1;
   END WHILE;
END //

DELIMITER ;
```

8. Realiza una función ***presupuestoCentro*** que, a partir del código de un centro, devuelva su presupuesto (calculado como la suma de los presupuestos de sus departamentos).

```
DELIMITER //

CREATE FUNCTION presupuestoCentro(codcen CHAR(4)) RETURNS DECIMAL(12,2)
DETERMINISTIC
BEGIN
   DECLARE total DECIMAL(12,2);
   SELECT SUM(PreAnu) INTO total
   FROM departamento
   WHERE CodCen = codcen;
   RETURN IFNULL(total, 0);
END //

DELIMITER ;
```

9. Realiza una función ***totalHabilidadesEmpleado*** que, a partir de un código de un empleado, devuelva cuantas habilidades tiene.

```
DELIMITER //

CREATE FUNCTION totalHabilidadesEmpleado(codemp INT) RETURNS INT
DETERMINISTIC
BEGIN
```

```
    DECLARE total INT;
    SELECT COUNT(*) INTO total
    FROM habemp
    WHERE CodEmp = codemp;
    RETURN total;
END //

DELIMITER ;
```

10. Realiza un procedimiento ***EmpleadosHabilidad*** que, a partir del nombre del empleado(parámetro de entrada), devuelve las habilidades (código y descripción) de dicho empleado.

```
DELIMITER //

CREATE PROCEDURE EmpleadosHabilidad(IN nombre_empleado VARCHAR(100))
BEGIN
    SELECT h.CodHab, h.DesHab
    FROM empleado e
    JOIN habemp he ON e.CodEmp = he.CodEmp
    JOIN habilidad h ON h.CodHab = he.CodHab
    WHERE e.NomEmp = nombre_empleado;
END //

DELIMITER ;
```

11. Realiza una función ***directorCentro*** que, a partir del código de un centro, devuelva el nombre de su director.

```
DELIMITER //

CREATE FUNCTION directorCentro(codcen CHAR(4)) RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
    DECLARE nombre VARCHAR(100);
    SELECT e.NomEmp INTO nombre
    FROM centro c
    JOIN empleado e ON c.CodEmpDir = e.CodEmp
```

```
    WHERE c.CodCen = codcen;
    RETURN nombre;
END //

DELIMITER ;
```

12. Realiza una función ***emailEmpleado*** que, a partir de un código de empleado, devuelva su email con la siguiente nomenclatura: CodEmp@CodDep.CodCen.com

```
DELIMITER //

CREATE FUNCTION emailEmpleado(codemp INT) RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
    DECLARE coddep CHAR(5);
    DECLARE codcen CHAR(4);
    SELECT e.CodDep, d.CodCen
    INTO coddep, codcen
    FROM empleado e
    JOIN departamento d ON e.CodDep = d.CodDep
    WHERE e.CodEmp = codemp;

    RETURN CONCAT(codemp, '@', coddep, '.', codcen, '.com');
END //

DELIMITER ;
```

13. Modifica la función anterior, para que en lugar del código del empleado devuelva su email con el nombre en minúsculas y sin acentos (hacer uso de la función creada en el ejercicio anterior **quitar_acentos)**

Ejemplo: SELECT emailEmpleado(1);
Devuelve: ***saladinomandamasaugusto@DIRGE.DIGE.com***

```
DELIMITER //

CREATE FUNCTION quitar_acentos(texto VARCHAR(255)) RETURNS VARCHAR(255)
DETERMINISTIC
BEGIN
    SET texto = REPLACE(texto, 'á', 'a');
```

```
    SET texto = REPLACE(texto, 'é', 'e');
    SET texto = REPLACE(texto, 'í', 'i');
    SET texto = REPLACE(texto, 'ó', 'o');
    SET texto = REPLACE(texto, 'ú', 'u');
    SET texto = REPLACE(texto, 'Á', 'A');
    SET texto = REPLACE(texto, 'É', 'E');
    SET texto = REPLACE(texto, 'Í', 'I');
    SET texto = REPLACE(texto, 'Ó', 'O');
    SET texto = REPLACE(texto, 'Ú', 'U');
    RETURN texto;
END //

DELIMITER ;

DELIMITER //

CREATE FUNCTION emailEmpleadoNombre(codemp INT) RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
    DECLARE nombre VARCHAR(100);
    DECLARE coddep CHAR(5);
    DECLARE codcen CHAR(4);

    SELECT quitar_acentos(LOWER(REPLACE(e.NomEmp, ' ', ''))),
        e.CodDep,
        d.CodCen
    INTO nombre, coddep, codcen
    FROM empleado e
    JOIN departamento d ON e.CodDep = d.CodDep
    WHERE e.CodEmp = codemp;

    RETURN CONCAT(nombre, '@', coddep, '.', codcen, '.com');
END //

DELIMITER ;
```

14. Comprueba las funciones y procedimientos existentes en la base de datos **empresa**.

```
SHOW PROCEDURE STATUS WHERE Db = 'EMPRESA';
```

```
SHOW FUNCTION STATUS WHERE Db = 'EMPRESA';
```