PRACTICA 20

Seguimos usando la base de datos ACME.

1.- Corrige los errores y explica:

```
CREATE PROCEDURE s_max()
BEGIN
Declare sal_pre dec(8,2)
SELECT max(salario) into @s  FROM empleados;
SELECT salario into sal_pre FROM empleados where oficio=PRESIDENTE;
Select @s as ‘Maximo’, sal_pre as ‘PRESIDENTE’;
IF (@s=sal_pre)
Select “Gana mas el presidente”;
Else
Select “No gana mas el presidente”;
End if;
END//
```

1. Guarda en la variable @s el salario máximo de los empleados.
2. Guarda en la variable sal_pre en salario del del empleado con oficio de presidente.
3. Imprime el valor de las dos variables.
4. Compara si el valor es el mismo, si el valor es el mismo significa que el que mas gana el es empleado con oficio de presidente y devuelve una cadena.

```
MariaDB [acme]> DELIMITER //
MariaDB [acme]> CREATE PROCEDURE s_max( )
    -> BEGIN
    ->     DECLARE sal_pre dec(8,2);
    ->     SELECT max(salario) into @s FROM empleados;
    ->     SELECT salario into sal_pre FROM empleados WHERE oficio="PRESIDENTE";
    ->     SELECT @s AS 'Maximo', sal_pre AS 'PRESIDENTE';
    ->     IF (@s=sal_pre) THEN
    ->         SELECT "Gana mas el presidente";
    ->     ELSE
    ->         SELECT "No gana mas el presidente";
    ->     END IF;
    -> END//
Query OK, 0 rows affected (0.023 sec)

MariaDB [acme]> DELIMITER ;
```

PRACTICA 20

```
MariaDB [acme]> CALL s_max();
+--------+------------+
| Maximo | PRESIDENTE |
+--------+------------+
|   6000 |    6000.00 |
+--------+------------+
1 row in set (0.058 sec)

+------------------------+
| Gana mas el presidente |
+------------------------+
| Gana mas el presidente |
+------------------------+
1 row in set (0.061 sec)

Query OK, 2 rows affected (0.063 sec)
```

2.- Haz un procedimiento que escriba el salario medio y nos diga si este salario medio de los empleados está por encima o por debajo de 1000€.

```
MariaDB [acme]> DELIMITER //
MariaDB [acme]> CREATE PROCEDURE s_medio()
    -> BEGIN
    ->     DECLARE media DEC(8,2);
    ->     SELECT AVG(salario) INTO media FROM empleados;
    ->     SELECT media AS "Euros de media de sueldo";
    ->     IF media<1000 THEN
    ->         SELECT "Media por debajo de 1000 euros" AS "Media de sueldo";
    ->     ELSE
    ->         SELECT "Media por encima de 1000 euros" AS "Media de sueldo";
    ->     END IF;
    -> END//
Query OK, 0 rows affected (0.007 sec)

MariaDB [acme]> DELIMITER ;
MariaDB [acme]> CALL s_medio();
+------------------------+
| Euros de media de sueldo |
+------------------------+
|                2281.69 |
+------------------------+
1 row in set (0.001 sec)

+------------------------------+
| Media de sueldo              |
+------------------------------+
| Media por encima de 1000 euros |
+------------------------------+
1 row in set (0.004 sec)

Query OK, 1 row affected, 1 warning (0.007 sec)
```

PRACTICA 20

3.- Haz una función que le pasemos la comisión y nos devuelva el tipo de vendedor:

- **Buen vendedor** si la comisión es >= 1000
- **Vendedor medio** si la comisión es >500 y < 1000
- **Mal vendedor** si la comisión es <500

```
MariaDB [acme]> DELIMITER //
MariaDB [acme]> CREATE FUNCTION tip_ven(comi FLOAT(6,2)) RETURNS VARCHAR(15)
    -> BEGIN
    ->     DECLARE text VARCHAR(15);
    ->     SET text =
    ->         CASE
    ->             WHEN comi < 999.99 THEN "Buen vendedor"
    ->             WHEN comi < 500 THEN "Mal vendedor"
    ->             ELSE "Vendedor medio"
    ->         END;
    ->     RETURN text;
    -> END//
Query OK, 0 rows affected (0.011 sec)

MariaDB [acme]> DELIMITER ;
```

4.- Escribe el nombre de los vendedores de la tabla empleados y el tipo usando la función anterior.

```
MariaDB [acme]> SELECT apellido,tip_ven(comision) FROM empleados;
+----------+-------------------+
| apellido | tip_ven(comision) |
+----------+-------------------+
| Garcia   | Vendedor medio    |
| Barrio   | Vendedor medio    |
| ALONSO   | Buen vendedor     |
| LOPEZ    | Vendedor medio    |
| MARTIN   | Vendedor medio    |
| GARRIDO  | Vendedor medio    |
| MARTINEZ | Vendedor medio    |
| REY      | Vendedor medio    |
| CALVO    | Buen vendedor     |
| GIL      | Vendedor medio    |
| JIMENEZ  | Vendedor medio    |
| ROMERALE | Vendedor medio    |
| LOBATO   | Vendedor medio    |
+----------+-------------------+
13 rows in set (0.065 sec)
```

PRACTICA 20

5.- Crea una **tabla temporal** que contenga solo un número. Con un procedimiento que se le pase como parámetros el valor inicial (valor que lo decrementamos de 5 en 5, no se almacenaran números negativos). Usa **WHILE**.

Es decir, si el valor inicial es 17, almacenaremos en la tabla: 17, 12, 7, 2.

Las tablas temporales son muy útiles para mantener datos temporales.

La tabla temporal se destruirá automáticamente cuando finalice la sesión o se cierre la conexión. El usuario también puede borrar la tabla temporal. Ejemplo:

```
CREATE TEMPORARY TABLE tempTable1(
    id INT NOT NULL AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
    PRIMARY KEY ( id )  );
```

Es decir se crean como una tabla normal anteponiendo simplemente la palabra TEMPORARY. Para insertar valores y demás no cambia nada.

```
MariaDB [acme]> DELIMITER //
MariaDB [acme]> CREATE PROCEDURE num_tab(num INT)
    -> BEGIN
    ->     CREATE TEMPORARY TABLE tempNum(
    ->         num INT NOT NULL,
    ->         PRIMARY KEY (num)
    ->     );
    ->     WHILE num > 0 DO
    ->         INSERT INTO tempNum VALUES(num);
    ->         SET num = num - 5;
    ->     END WHILE;
    -> END//
Query OK, 0 rows affected (0.009 sec)

MariaDB [acme]> DELIMITER ;
```

```
MariaDB [acme]> CALL num_tab(21);
Query OK, 5 rows affected (0.002 sec)

MariaDB [acme]> SELECT * FROM tempnum;
+-----+
| num |
+-----+
|   1 |
|   6 |
|  11 |
|  16 |
|  21 |
+-----+
5 rows in set (0.000 sec)
```

PRACTICA 20

6.- (valor del decremento). Usa **REPEAT … UNTIL**.

```
MariaDB [acme]> DELIMITER //
MariaDB [acme]> CREATE PROCEDURE numtab2(num INT, sal INT)
    -> BEGIN
    ->     CREATE TEMPORARY TABLE tnum(
    ->         num INT NOT NULL,
    ->         PRIMARY KEY (num)
    ->     );
    ->     REPEAT
    ->         INSERT INTO tnum VALUES(num);
    ->         SET num = num - sal;
    ->     UNTIL num < 0
    ->     END REPEAT;
    -> END//
Query OK, 0 rows affected (0.006 sec)

MariaDB [acme]> DELIMITER ;
```

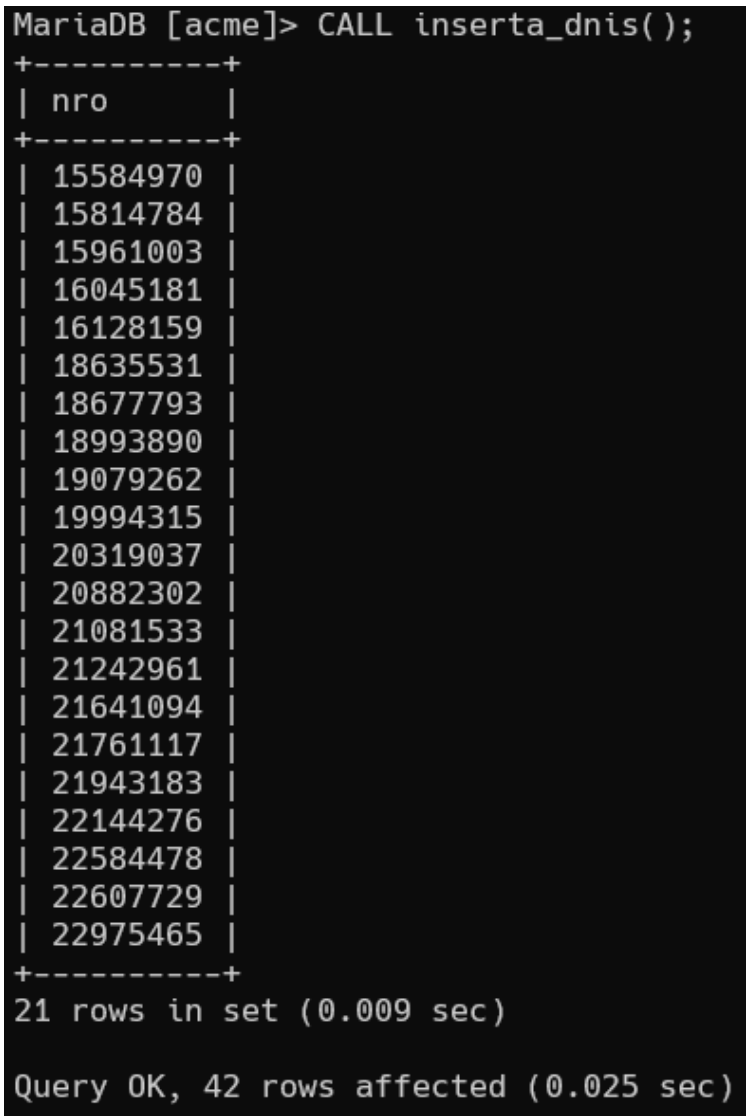
```
MariaDB [acme]> CALL numtab2(17,5);
Query OK, 4 rows affected (0.002 sec)

MariaDB [acme]> SELECT * FROM tnum;
+-----+
| num |
+-----+
|   2 |
|   7 |
|  12 |
|  17 |
+-----+
4 rows in set (0.000 sec)
```

PRACTICA 20

7.- Crea una tabla temporal llamada **DNIS** con un campo **nro** de tipo INT clave principal. Después **Ejecuta** este procedimiento y **explica que hace**.

```
Create procedure INSERTA_DNIS()
Begin
Declare n int;
Declare c smallint;
Truncate DNIS;
repeat
        Set n= Floor(rand()*8000000)+15000001;
# Multiplicamos por 8 millones y sumamos 15 millones
        Select count(*) into c from DNIS;
        Insert into DNIS values(n);
Until  c=20
End repeat;
Select * from DNIS;
End //
```

Es un procedimiento que inserta dnis aleatorios a partir de 15000001 hasta llegar a 20 dnis en la tabla de dnis.



```
MariaDB [acme]> CALL inserta_dnis();
+----------+
| nro      |
+----------+
| 15584970 |
| 15814784 |
| 15961003 |
| 16045181 |
| 16128159 |
| 18635531 |
| 18677793 |
| 18993890 |
| 19079262 |
| 19994315 |
| 20319037 |
| 20882302 |
| 21081533 |
| 21242961 |
| 21641094 |
| 21761117 |
| 21943183 |
| 22144276 |
| 22584478 |
| 22607729 |
| 22975465 |
+----------+
21 rows in set (0.009 sec)

Query OK, 42 rows affected (0.025 sec)
```

PRACTICA 20

8.- Crea la función letra_nif que reciba un DNI y calcule la letra del NIF correspondiente (La función debe de devolver la letra).

La fórmula para calcular la letra del DNI y obtener el NIF es la siguiente:

Tomamos el número completo de hasta 8 cifras de nuestro DNI, lo dividimos entre 23 y nos quedamos con el resto de dicha división, o dicho de otro modo, calculamos el módulo 23 del DNI.

El resultado anterior es un número entre 0 y 22. A cada uno de estos posibles números le corresponde una letra, según la siguiente tabla:

RESTO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LETRA	T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

```
MariaDB [acme]> DELIMITER //
MariaDB [acme]> CREATE FUNCTION letra_nif(nif INT) RETURNS CHAR(1)
    -> BEGIN
    ->     DECLARE res INT;
    ->     SET res = nif % 23;
    ->     RETURN ELT(res+1,"T","R","W","A","G","M","Y","F","P","D","X","B","N","J","Z","S","Q","V","H","L","C","K","E");
    -> END//
Query OK, 0 rows affected (0.009 sec)

MariaDB [acme]> DELIMITER ;
```

Utiliza la función MID() y la cadena: "TRWAGMYFPDXBNJZSQVHLCKE")

Función MySQL MID ()
Ejemplo

Extraiga una subcadena de una cadena (comience en la posición 4, extraiga 6 caracteres):
SELECT MID("SQL es muy interesante", 4, 6) AS CadenaExtraida;

```
MariaDB [acme]> DELIMITER //
MariaDB [acme]> CREATE FUNCTION letra_nif(nif INT) RETURNS CHAR(1)
    -> BEGIN
    ->     RETURN MID("TRWAGMYFPDXBNJZSQVHLCKE",nif % 23,1);
    -> END//
Query OK, 0 rows affected (0.009 sec)

MariaDB [acme]> DELIMITER ;
```

```
MariaDB [acme]> SELECT letra_nif(99999999);
+---------------------+
| letra_nif(99999999) |
+---------------------+
| R                   |
+---------------------+
1 row in set (0.000 sec)
```

PRACTICA 20

9.- Usa la función anterior mostrar los números de la tabla **DNIS** con su letra correspondiente.

```
MariaDB [acme]> SELECT CONCAT(nro,letra_nif(nro)) AS DNI FROM dnis;
+-----------+
| DNI       |
+-----------+
| 15584970D |
| 15814784F |
| 15961003S |
| 16045181J |
| 16128159F |
| 18635531B |
| 18677793E |
| 18993890F |
| 19079262A |
| 19994315R |
| 20319037D |
| 20882302G |
| 21081533D |
| 21242961T |
| 21641094A |
| 21761117N |
| 21943183X |
| 22144276Z |
| 22584478L |
| 22607729V |
| 22975465Y |
+-----------+
21 rows in set (0.001 sec)
```