



# Bases de datos

Act01. Ut07 Construcción de guiones. Parte 1

Francisco José García Cutillas | 1FPGS\_DAM



## Índice

Ejercicio 1 .....	3
Ejercicio 2 .....	3
Ejercicio 3 .....	4
Ejercicio 4 .....	5
Ejercicio 5 .....	5
Ejercicio 6 .....	6
Ejercicio 7 .....	7
Ejercicio 8 .....	8
Ejercicio 9 .....	9
Ejercicio 10 .....	10
Ejercicio 11 .....	11
Ejercicio 12 .....	11
Ejercicio 13 .....	12
Ejercicio 14 .....	13

## Ejercicio 1

Crea una función `f_es_vocal` que recibe una letra y retorna un `TRUE` si lo es y un `FALSE` en caso contrario

```
DELIMITER $$
drop function if exists f_es_vocal;
create function f_es_vocal (letra char) returns boolean
begin
    if letra in ( 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U') then
        return true;
    else
        return false;
    end if;
end;$$
```

```
select f_es_vocal('a') esVocal;
```

	esVocal
▶	1

```
select f_es_vocal('E') esVocal;
```

	esVocal
▶	1

```
select f_es_vocal('t') esVocal;
```

	esVocal
▶	0

## Ejercicio 2

Crea una función que tendrá como parámetros de entradas dos enteros y tendrá que devolver si el primer número es divisible de manera exacta por el segundo, es decir, si se pasa un 7 y un 3 deberá devolver `FALSE`, pero si se pasa un 9 y un 3 deberá devolver `TRUE`.

```
DELIMITER $$
drop function if exists f_divisible_exacto;$$
create function f_divisible_exacto (num1 int, num2 int) returns boolean
begin
    if (num1 % num2 = 0) then
        return true;
    else
        return false;
    end if;
end;$$
```

```
select f_divisible_exacto(7, 3) divisible;
```

	divisible
▶	0

```
select f_divisible_exacto(9, 3) divisible;
```

	divisible
▶	1

```
select f_divisible_exacto(12, 4) divisible;
```

	divisible
▶	1

### Ejercicio 3

Crea una función que, al pasar una cadena por parámetro, devuelve la misma en minúscula y al final tiene concatenado el tamaño de la longitud de la cadena. Por ejemplo, si se pasa hola, debería devolver hola4

```
DELIMITER $$
drop function if exists f_minuscula_mide;$$
create function f_minuscula_mide(cadena varchar (20)) returns varchar (20)
begin
    declare num_mide int;
    declare vc_salida varchar (20);
    set num_mide = length(cadena);
    set vc_salida = concat(lower(cadena), num_mide);
    return vc_salida;
end; $$
```

```
select f_minuscula_mide('cadena') min_mide;
```

	min_mide
▶	cadena6

```
select f_minuscula_mide('HOLA') min_mide;
```

	min_mide
▶	hola4

```
select f_minuscula_mide('PrUeBa') min_mide;
```

	min_mide
▶	prueba6

## Ejercicio 4

Crea un procedimiento que tenga tres parámetros, dos de entrada y uno de salida. El tercer parámetro tendrá el resultado de multiplicar los dos primeros. Este procedimiento se llamará `p_multiplica`

```
DELIMITER $$
drop procedure if exists p_multiplica; $$
create procedure p_multiplica (in num1 numeric, in num2 numeric, out resultado numeric)
begin
    set resultado = num1 * num2;
end; $$
```

```
call p_multiplica (5, 3, @salida);
select @salida resultado;
```

	resultado
▶	15

```
call p_multiplica (8, 8, @salida);
select @salida resultado;
```

	resultado
▶	64

```
call p_multiplica (-2, 3, @salida);
select @salida resultado;
```

	resultado
▶	-6

## Ejercicio 5

Utilizando el procedimiento creado anteriormente de nombre `p_multiplica` crea una función que se llamará `f_multiplica` que tendrá dos parámetros de entrada y devolverá el resultado de la multiplicación.

```
DELIMITER $$
drop function if exists f_multiplica; $$
create function f_multiplica (num1 numeric, num2 numeric) returns numeric
begin
    declare resultado numeric;
    call p_multiplica (num1, num2, resultado);
    return resultado;
end; $$
```

```
select f_multiplica (3,3) resultado;
```

resultado
9

```
select f_multiplica (6,3) resultado;
```

resultado
18

```
select f_multiplica (7,9) resultado;
```

resultado
63

Para realizar los siguientes apartados debemos hacer uso de la base de datos ligabaloncesto.

## Ejercicio 6

Crea una función que se llame `f_media_equipo` que devuelva la media del salario del código de equipo que se pasa como parámetro.

```
DELIMITER $$
drop function if exists f_media_equipo; $$
create function f_media_equipo(cod_equipo int) returns decimal(10,2)
begin
    declare dec_salida decimal(10,2);
    select avg(j.salario) into dec_salida
        from jugador j inner join equipo e
            on j.num_equipo = e.id_equipo
        where e.id_equipo = cod_equipo
        group by e.id_equipo;
    return dec_salida;
end; $$
```

```
select f_media_equipo(1) salario_medio;
```

salario_medio
105000.00

```
select f_media_equipo(5) salario_medio;
```

salario_medio
77500.00

```
select f_media_equipo(6) salario_medio;
```

salario_medio
68333.33

## Ejercicio 7

Crea una función que se llame `f_puesto_equipo` que devuelva la posición en la que se encuentra ese equipo en la liga (se pasará el código). Ya sabéis que para saber su posición se debe revisar la columna puntos de la tabla equipo.

```
DELIMITER $$
drop function if exists f_puesto_equipo; $$
create function f_puesto_equipo(codigo int) returns int
begin
    declare int_salida int;
    select e2.clasificacion into int_salida
        from equipo e1 inner join (select id_equipo, row_number() over (order by puntos desc) clasificacion
                                   from equipo) e2
        on e1.id_equipo = e2.id_equipo
        where e1.id_equipo = codigo;
    return int_salida;
end; $$
```

```
select f_puesto_equipo (7) puesto_clasificacion;
```

	puesto_clasificacion
▶	1

```
select f_puesto_equipo (2) puesto_clasificacion;
```

	puesto_clasificacion
▶	7

```
select f_puesto_equipo (5) puesto_clasificacion;
```

	puesto_clasificacion
▶	4

## Ejercicio 8

Crea un procedimiento que pasando el código de un equipo devuelva en dos parámetros de salida el número de partidos que ha jugado como local y los que ha jugado como visitante. El procedimiento `p_partidos_equipo`

```
DELIMITER $$
drop procedure if exists p_partidos_equipo; $$
create procedure p_partidos_equipo (in codigo int, out num_partidos_loc int, out num_partidos_vis int)
begin
    declare n_local int;
    declare n_visitante int;

    select count(e.id_equipo) into n_local
        from equipo e inner join partido p
            on e.id_equipo = p.elocal
        where e.id_equipo = codigo
        group by e.id_equipo;

    select count(e.id_equipo) into n_visitante
        from equipo e inner join partido p
            on e.id_equipo = p.evisitante
        where e.id_equipo = codigo
        group by e.id_equipo;

    set num_partidos_loc = ifnull(n_local, '0');
    set num_partidos_vis = ifnull(n_visitante, '0');
end; $$
```

```
call p_partidos_equipo (1, @p_loc, @p_vis);
select @p_loc partidos_local, @p_vis partidos_visitante;
```

	partidos_local	partidos_visitante
▶	2	0

```
call p_partidos_equipo (2, @p_loc, @p_vis);
select @p_loc partidos_local, @p_vis partidos_visitante;
```

	partidos_local	partidos_visitante
▶	3	1

```
call p_partidos_equipo (6, @p_loc, @p_vis);
select @p_loc partidos_local, @p_vis partidos_visitante;
```

	partidos_local	partidos_visitante
▶	1	2



## Ejercicio 9

Crea una función que se llame `f_num_partidos_equipos` que devuelva el número total de partidos que ha jugado ese equipo (se pasará el NOMBRE del equipo). Se debe tener en cuenta tanto los partidos que ha jugado como local como visitante por lo que se aconseja hacer uso del procedimiento `p_partidos_equipo`

```
DELIMITER $$
drop function if exists f_num_partidos_equipos; $$
create function f_num_partidos_equipos (nom_equipo varchar(20)) returns numeric
> begin
    declare p_local int;
    declare p_visitante int;

    > call p_partidos_equipo ((select id_equipo
                             from equipo
                             where nombre = nom_equipo),
    ~                             p_local, p_visitante);
    ~ return ifnull(p_local, '0') + ifnull(p_visitante, '0');
    ~ end; $$
```

```
select f_num_partidos_equipos ('REGAL BARCELONA') total_partidos;
```

	total_partidos
▶	2

```
select f_num_partidos_equipos ('REAL MADRID') total_partidos;
```

	total_partidos
▶	4

```
select f_num_partidos_equipos ('GRAN CANARIA') total_partidos;
```

	total_partidos
▶	4

## Ejercicio 10

Crea una función que se llame `f_encima_media_eq` que indique para un código de equipo el número de jugadores que cobran más de la media del salario de dicho equipo. Se debe hacer uso de la función `f_media_equipo` que hemos creado anteriormente.

```
DELIMITER $$
drop function if exists f_encima_media_eq; $$
create function f_encima_media_eq (codigo int) returns int
begin
    declare n_salida int;
    select count(id_jugador) into n_salida
        from jugador
        where salario > f_media_equipo (codigo)
        and num_equipo = codigo;
    return n_salida;
end; $$
```

```
select f_encima_media_eq (1) total_jug_sueldo_mayor_media_eq;
```

	total_jug_sueldo_mayor_media_eq
▶	1

```
select f_encima_media_eq (2) total_jug_sueldo_mayor_media_eq;
```

	total_jug_sueldo_mayor_media_eq
▶	2

```
select f_encima_media_eq (4) total_jug_sueldo_mayor_media_eq;
```

	total_jug_sueldo_mayor_media_eq
▶	1

## Ejercicio 11

Crea un procedimiento llamado `p_jug_capitan` que muestre por pantalla todos los jugadores que tienen como capitán al jugador que se pasa como parámetro (se pasa el código).

```
DELIMITER $$
drop procedure if exists p_jug_capitan; $$
create procedure p_jug_capitan (id_jugador int)
begin
    select nombre, apellido
    from jugador
    where id_capitan = id_jugador;
end; $$
```

```
call p_jug_capitan (6);
```

	nombre	apellido
▶	FERNANDO	SAN EMETERIO
	MIRZA	TELETOVIC

```
call p_jug_capitan (2);
```

	nombre	apellido
▶	FELIPE	REYES
	SERGIO	LLULL
	CARLOS	SUÁREZ

```
call p_jug_capitan (3);
```

	nombre	apellido
▶	VÍCTOR	CLAVER
	RAFA	MARTÍNEZ

## Ejercicio 12

Crea una función que dado un código de equipo devuelva su nombre. Pon de nombre `f_nom_equipo`.

```
DELIMITER $$
drop function if exists f_nom_equipo; $$
create function f_nom_equipo (codigo int) returns varchar (20)
begin
    declare vc_salida varchar (20);
    select nombre into vc_salida
    from equipo
    where id_equipo = codigo;
    return vc_salida;
end; $$
```

```
select f_nom_equipo (1) nombre_Equipo;
```

nombre_Equipo
▶ REGAL BARCELONA

```
select f_nom_equipo (4) nombre_Equipo;
```

nombre_Equipo
▶ CAJA LABORAL

```
select f_nom_equipo (7) nombre_Equipo;
```

nombre_Equipo
▶ BALONCESTO MURCIA

## Ejercicio 13

Crea una función que dado un código de equipo y un salario (ambos son parámetros) nos indique el número de jugadores que para ese equipo cobran más de ese salario.

```
DELIMITER $$
drop function if exists f_cobran_mas_que; $$
create function f_cobran_mas_que (codigo int, in_salario decimal) returns int
begin
    declare n_salida int;
    select count(*) into n_salida
        from jugador
        where num_equipo = codigo
            and salario > in_salario;
    return n_salida;
end; $$
```

```
select f_cobran_mas_que(1, 70000) jug_sueldo_mayor;
```

jug_sueldo_mayor
▶ 2

```
select f_cobran_mas_que(2, 110000) jug_sueldo_mayor;
```

jug_sueldo_mayor
▶ 1

```
select f_cobran_mas_que(4, 80000) jug_sueldo_mayor;
```

jug_sueldo_mayor
▶ 0

## Ejercicio 14

Crea una función `f_fecha_equipo` que recibiendo el `NOMBRE` de un equipo nos indique el número de partidos que ha jugado con fecha anterior a la fecha que también va a recibir como parámetro.

```
DELIMITER $$
drop function if exists f_fecha_equipo; $$
create function f_fecha_equipo (nombre varchar(20), fecha date) returns int
begin
    declare n_num_partidos int;
    select count(*) into n_num_partidos
        from equipo e inner join partido p
            on e.id_equipo = p.elocal
            or e.id_equipo = p.evisitante
        where e.nombre = nombre
            and p.fecha < fecha;
    return n_num_partidos;
end; $$
```

```
select f_fecha_equipo ('REAL MADRID', '2018-01-01') num_part_anteriores;
```

	num_part_anteriores
▶	4

```
select f_fecha_equipo ('VALENCIA BASKET', '2011-12-01') num_part_anteriores;
```

	num_part_anteriores
▶	2

```
select f_fecha_equipo ('CAJA LABORAL', '2012-02-01') num_part_anteriores;
```

	num_part_anteriores
▶	3