

# Automatización de tareas

A.D.G.

Se parte del docker-compose del aula virtual que lanza un phpmyadmin y una base de datos MySQL.

## Ejercicio 1: Puesta a punto:

- Arrancar el docker de php y mysql
- Crea las siguientes tablas:

```
create table articulo(
  idarticulo int primary key,
  codigo varchar(50) null,
  nombre varchar(100) not null unique,
  precio_venta float not null,
  stock int not null,
  descripcion varchar(256) null
);
create table persona(
  idpersona int primary key ,
  nombre varchar(100) not null,
  dni varchar(9) null,
  direccion varchar(70) null,
  telefono varchar(20) null,
  email varchar(50) null
);
create table sospechosos_de_error(
  idarticulo int primary key,
  codigo varchar(50) null,
  nombre varchar(100) not null unique,
  precio_venta float not null,
  descripcion varchar(256) null
);
create table dinero_total(
  total float);
insert into dinero_total values (0);

insert into persona values(1, "Pablo", "12345678L", "calle falsa 123", "915555556", "p@g.com");
insert into persona values(2, "Cristina", "12365678L", "calle otra 12", "915565556", "c@g.com");
insert into persona values(3, "Titi", "12345378L", "calle dos 13", "915535556", "t@g.com");
insert into persona values(4, "Irene", "12345278L", "calle tres 23", "915535556", "i@g.com");
insert into persona values(5, "Miguel", "12645678L", "calle m 153", "915255556", "m@g.com");
insert into persona values(6, "Tomás", "12745678L", "calle do 53", "915555556", "t@g.com");
insert into persona values(7, "Álvaro", "12945678L", "calle re 63", "915555856", "al@g.com");
insert into persona values(8, "Ana", "12349678L", "calle mi 127", "915556856", "a@g.com");
insert into persona values(9, "Inés", "12336678L", "calle fa 138", "915455556", "ine@g.com");
insert into persona values(10, "Diego", "12445678L", "calle sol 132", "915556556", "dg@g.com");
```

## Ejercicio 2: Triggers

- Crea los siguientes triggers:

### 2.1.- Si se inserta un artículo se actualiza dinero total (sumándole el dinero x el stock añadido):

```
DELIMITER //
CREATE TRIGGER despuesdeinsertar_articulo_actualizardinerototal_21
AFTER INSERT ON articulo
FOR EACH ROW
BEGIN
  UPDATE dinero_total
  SET total = total + (NEW.precio_venta * NEW.stock);
END;
//
```

**2.2.- Si se actualiza un artículo se actualiza dinero total (sumándole el dinero x el stock añadido o restándolo si hay menos):**

```
DELIMITER //
CREATE TRIGGER despuesdeactualizar_articulo_actualizardinerototal_22
AFTER UPDATE ON articulo
FOR EACH ROW
BEGIN
    DECLARE dinero float;
    SET dinero = (NEW.stock - OLD.stock) * NEW.precio_venta;
    SET dinero = dinero + (NEW.precio_venta - OLD.precio_venta) * NEW.stock;
    UPDATE dinero_total SET total = total + dinero;
END;
//
```

**2.3.- Si se borra un artículo (restándole el dinero x el stock que tuviera):**

```
DELIMITER //
CREATE TRIGGER despuesdeborrar_articulo_restardinerototal_23
AFTER DELETE
ON articulo
FOR EACH ROW
BEGIN
    UPDATE dinero_total SET total = total - (OLD.precio_venta * OLD.stock);
END;
//
```

**2.4.- Si se ingresa un artículo con precio\_venta negativo se meterá en sospechosos\_de\_error y se ingresará en la tabla artículo con precio 0:**

```
DELIMITER //
CREATE TRIGGER antesdeinsertar_articulo_siprecionegativo_sospechoso_yasigna0_24
BEFORE INSERT ON articulo
FOR EACH ROW
BEGIN
    IF NEW.precio_venta < 0 THEN
        INSERT INTO sospechosos_de_error
        VALUES (NEW.idarticulo, NEW.codigo, NEW.nombre, NEW.precio_venta, NEW.descripcion);
        SET NEW.precio_venta = 0;
    END IF;
END;
//
```

**2.5.- Se hará lo mismo que en el trigger4 si se actualiza un precio a 0:**

```
DELIMITER //
CREATE TRIGGER antesdeactualizar_articulo_siprecioa0_sospechoso_25
BEFORE UPDATE ON articulo
FOR EACH ROW
BEGIN
    IF NEW.precio_venta = 0 THEN
        INSERT INTO sospechosos_de_error (idarticulo, codigo, nombre, precio_venta, descripcion)
        VALUES (NEW.idarticulo, NEW.codigo, NEW.nombre, NEW.precio_venta, NEW.descripcion);
    END IF;
END;
//
```

## Entrega del código de los 5. Puedes mostrar ejemplo de funcionamiento:

(Testee el funcionamiento de todos, quiero simplificar el doc, en lo siguiente se puede ver igualmente algunos al ejecutar las operaciones)

## Después de crear los triggers (puedes haber insertado y borrado elementos) borra todo lo que hayas

### Insertado:

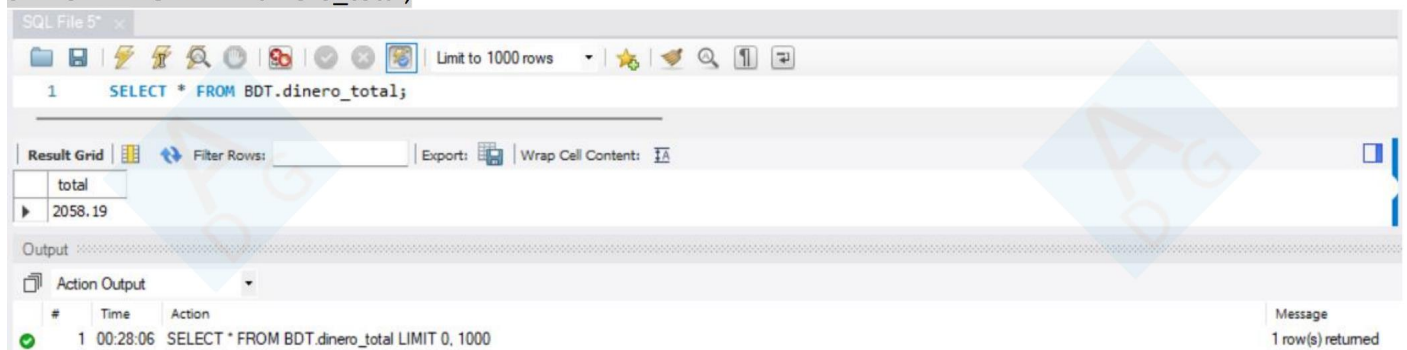
```
DELETE FROM articulo;  
DELETE FROM persona;  
DELETE FROM sospechosos_de_error;  
DELETE FROM dinero_total;  
INSERT INTO dinero_total VALUES (0);
```

### Inserta:

```
insert into articulo values(1, "XXXXXXXX1", "Ejemplo1", 12.54, 1, "xxxxxxxxx1");  
insert into articulo values(2, "XXXXXXXX2", "Ejemplo2", 2.5, 2, "xxxxxxxxx2");  
insert into articulo values(3, "XXXXXXXX3", "Ejemplo3", 15.24, 5, "xxxxxxxxx3");  
insert into articulo values(4, "XXXXXXXX4", "Ejemplo4", 7.24, -10, "xxxxxxxxx4");  
insert into articulo values(5, "XXXXXXXX5", "Ejemplo5", 8.54, 7, "xxxxxxxxx5");  
insert into articulo values(6, "XXXXXXXX6", "Ejemplo6", 16.99, 2, "xxxxxxxxx6");  
insert into articulo values(7, "XXXXXXXX7", "Ejemplo7", 25.99, 1, "xxxxxxxxx7");  
insert into articulo values(8, "XXXXXXXX8", "Ejemplo8", 70.3, 3, "xxxxxxxxx8");  
insert into articulo values(9, "XXXXXXXX9", "Ejemplo9", 80, -1, "xxxxxxxxx9");  
insert into articulo values(10, "XXXXXXXX10", "Ejemplo10", 80, 2, "xxxxxxxxx10");  
insert into articulo values(11, "XXXXXXXX11", "Ejemplo11", 26, 34, "xxxxxxxxx11");  
insert into articulo values(12, "XXXXXXXX12", "Ejemplo12", 10.2, 11, "xxxxxxxxx12");  
insert into articulo values(13, "XXXXXXXX13", "Ejemplo13", 63, 10, "xxxxxxxxx13");
```

## Muestra el valor de Dinero\_total y de sospechosos\_de\_error:

```
SELECT * FROM BDT.dinero_total;
```



SQL File 5

1 SELECT \* FROM BDT.dinero\_total;

Result Grid

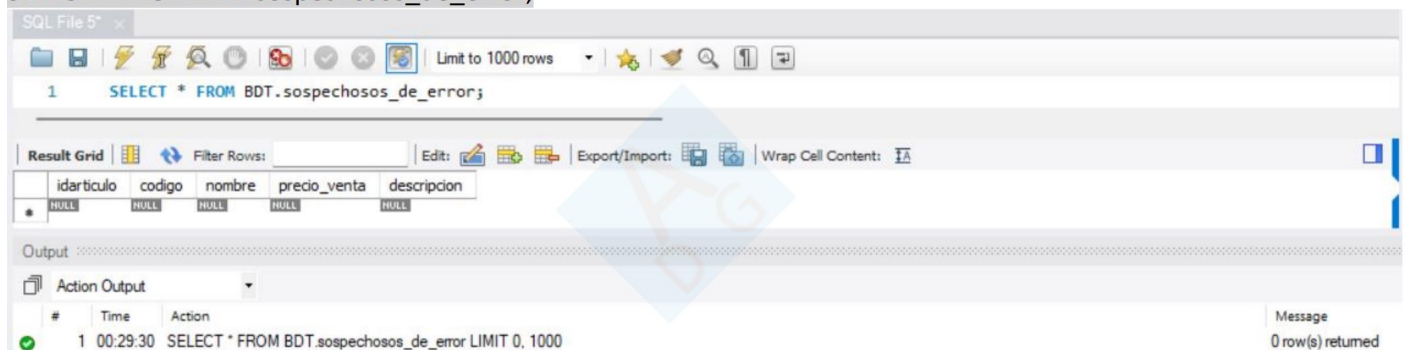
total
2058.19

Output

Action Output

#	Time	Action	Message
1	00:28:06	SELECT * FROM BDT.dinero_total LIMIT 0, 1000	1 row(s) returned

```
SELECT * FROM BDT.sospechosos_de_error;
```



SQL File 5

1 SELECT \* FROM BDT.sospechosos\_de\_error;

Result Grid

idarticulo	codigo	nombre	precio_venta	descripcion
NULL	NULL	NULL	NULL	NULL

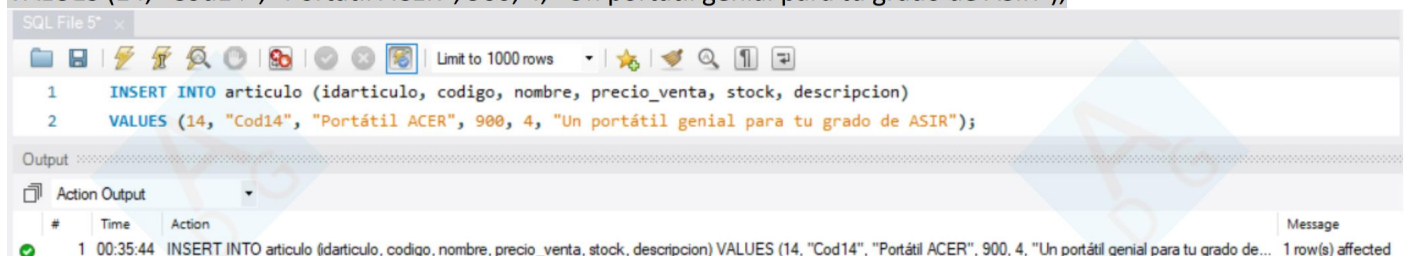
Output

Action Output

#	Time	Action	Message
1	00:29:30	SELECT * FROM BDT.sospechosos_de_error LIMIT 0, 1000	0 row(s) returned

## Inserta un artículo (14) y muestra que se actualiza dinero total:

```
INSERT INTO articulo (idarticulo, codigo, nombre, precio_venta, stock, descripcion)  
VALUES (14, "Cod14", "Portátil ACER", 900, 4, "Un portátil genial para tu grado de ASIR");
```



SQL File 5

1 INSERT INTO articulo (idarticulo, codigo, nombre, precio\_venta, stock, descripcion)  
2 VALUES (14, "Cod14", "Portátil ACER", 900, 4, "Un portátil genial para tu grado de ASIR");

Output

Action Output

#	Time	Action	Message
1	00:35:44	INSERT INTO articulo (idarticulo, codigo, nombre, precio_venta, stock, descripcion) VALUES (14, "Cod14", "Portátil ACER", 900, 4, "Un portátil genial para tu grado de...	1 row(s) affected



SELECT \* FROM BDT.dinero\_total;

SQL File 5\* x

1 • SELECT \* FROM BDT.dinero\_total;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

total
5658.19

Output

Action Output

#	Time	Action	Message
1	00:37:18	SELECT * FROM BDT.dinero_total LIMIT 0, 1000	1 row(s) returned

**Borra el artículo 6 y muestra cómo se ha actualizado:**

DELETE FROM articulo WHERE idarticulo = 6;

SQL File 5\* x

1 • DELETE FROM articulo WHERE idarticulo = 6;

Output

Action Output

#	Time	Action	Message
1	00:39:17	DELETE FROM articulo WHERE idarticulo = 6	1 row(s) affected

SELECT \* FROM BDT.dinero\_total;

SQL File 5\* x

1 • SELECT \* FROM BDT.dinero\_total;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

total
5624.21

Output

Action Output

#	Time	Action	Message
1	00:40:44	SELECT * FROM BDT.dinero_total LIMIT 0, 1000	1 row(s) returned

**Actualiza el 9 para ponerle 5 de stock y muestra cómo se ha actualizado:**

UPDATE articulo SET stock = 5 WHERE idarticulo = 9;

SQL File 5\* x

1 • UPDATE articulo SET stock = 5 WHERE idarticulo = 9;

Output

Action Output

#	Time	Action	Message
1	00:43:44	UPDATE articulo SET stock = 5 WHERE idarticulo = 9	1 row(s) affected

SELECT \* FROM BDT.dinero\_total;

SQL File 6\* x

1 • SELECT \* FROM BDT.dinero\_total;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

total
6104.21

Output

Action Output

#	Time	Action	Message
1	00:48:48	SELECT * FROM BDT.dinero_total LIMIT 0, 1000	1 row(s) returned

## Ejercicio 3: Procedimientos y funciones

### 3.1 Crea una función que dada una cadena la devuelva pasada a letras mayúsculas:

```
CREATE PROCEDURE convertir_texto_a_mayusculas_31(cadenatexto VARCHAR(50))
BEGIN
    DECLARE convertir VARCHAR(50);
    SET convertir = "";
    SET convertir = UPPER(cadenatexto);
    SELECT convertir;
END;
```

The screenshot shows a database client interface with a toolbar at the top. Below the toolbar, a SQL statement is entered: `call BDT.convertir_texto_a_mayusculas_31('Hola, esto es una prueba ASIR');`. The 'Result Grid' tab is active, displaying a single row with the column 'convertir' and the value 'HOLA, ESTO ES UNA PRUEBA ASIR'. Below the result grid, the 'Output' tab is active, showing a table with columns '#', 'Time', and 'Action'. The first row shows the execution of the call at 23:57:57. A 'Message' pane on the right indicates '1 row(s) returned'.

### 3.2 Crea un procedimiento que reciba un id y muestre la información de la persona de la tabla personas con ese id si existe. (utilizad el formato del ejemplo):

```
CREATE PROCEDURE mostrar_informacion_delapersona_porid_32(IN id INT)
BEGIN
```

```
    DECLARE sunombre varchar(100);
    DECLARE sudni varchar(9);
    DECLARE sudireccion varchar(70);
    DECLARE sutelefono varchar(20);
    DECLARE suemail varchar(50);
```

```
    SELECT nombre, dni, direccion, telefono, email
    INTO sunombre, sudni, sudireccion, sutelefono, suemail
    FROM persona
    WHERE idpersona = id;
```

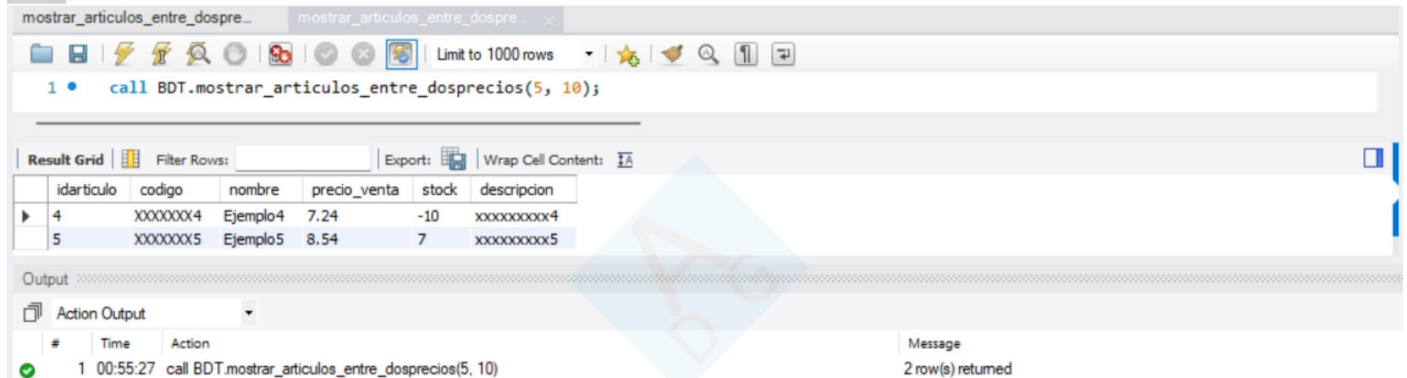
```
    SET @resultado = CONCAT("La información solicitada es de ", sunombre, ", con DNI ", sudni, ", con domicilio en ", sudireccion, ", correo ", suemail, " y número de teléfono ", sutelefono);
    SELECT @resultado;
```

END;

The screenshot shows a database client interface with a toolbar at the top. Below the toolbar, a SQL statement is entered: `call BDT.mostrar_informacion_delapersona_porid(1);`. The 'Result Grid' tab is active, displaying a single row with the column '@resultado' and the value 'La información solicitada es de Pablo, con DNI 12345678L, con domicilio en calle falsa 123, correo p@g.com y número de teléfono 915555556'. Below the result grid, the 'Output' tab is active, showing a table with columns '#', 'Time', and 'Action'. The first row shows the execution of the call at 00:53:00. A 'Message' pane on the right indicates '1 row(s) returned'.

### 3.3 Crea un procedimiento que muestre los artículos que tengan un precio entre 2 valores dados:

```
CREATE PROCEDURE mostrar_articulos_entre_dosprecios_33 (IN precio_minimo FLOAT, IN precio_maximo FLOAT)
BEGIN
    SELECT * FROM articulo WHERE precio_venta BETWEEN precio_minimo AND precio_maximo;
END;
```



mostrar\_articulos\_entre\_dospre... mostrar\_articulos\_entre\_dospre... x

Limit to 1000 rows

1 • call BDT.mostrar\_articulos\_entre\_dosprecios(5, 10);

	idarticulo	codigo	nombre	precio_venta	stock	descripcion
▶	4	XXXXXXXX4	Ejemplo4	7.24	-10	xxxxxxxxxx4
	5	XXXXXXXX5	Ejemplo5	8.54	7	xxxxxxxxxx5

Output

Action Output

#	Time	Action	Message
✓ 1	00:55:27	call BDT.mostrar_articulos_entre_dosprecios(5, 10)	2 row(s) returned

### 3.4 Crea un procedimiento que reciba 2 valores de descuento (entre 0 y 100%).

Le rebajará al precio de los artículos el primer valor si su precio actual es menor o igual a 20 y el segundo valor si es mayor de 20.

Por ejemplo:

call Procedimiento 70 30

Los artículos con precio inferior a 20 se les baja un 70% su precio.

Los artículos con precio superior a 20 se les baja un 30% su precio.

```
CREATE PROCEDURE AplicarDescuento(IN descuento1 FLOAT, IN descuento2 FLOAT)
BEGIN
```

```
    DECLARE precio_articulo FLOAT;
```

```
    UPDATE articulo
```

```
    SET precio_venta = CASE
```

```
        WHEN precio_venta <= 20 THEN precio_venta * (1 - (descuento1 / 100))
```

```
        ELSE precio_venta
```

```
    END;
```

```
    UPDATE articulo
```

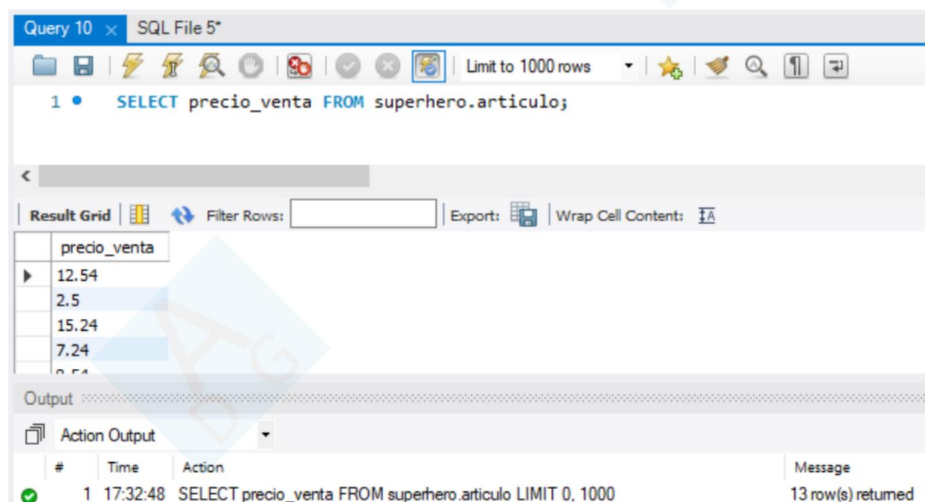
```
    SET precio_venta = CASE
```

```
        WHEN precio_venta > 20 THEN precio_venta * (1 - (descuento2 / 100))
```

```
        ELSE precio_venta
```

```
    END;
```

```
END;
```



Query 10 x SQL File 5\*

Limit to 1000 rows

1 • SELECT precio\_venta FROM superhero.articulo;

precio_venta
12.54
2.5
15.24
7.24
8.54

Output

Action Output

#	Time	Action	Message
✓ 1	17:32:48	SELECT precio_venta FROM superhero.articulo LIMIT 0, 1000	13 row(s) returned

SQL File 5\* x

Limit to 1000 rows

```
1 • CALL AplicarDescuento(70, 30);
2 • SELECT precio_venta FROM superhero.articulo;
```

Result Grid

precio_venta
3.762
0.75
4.572
2.172
2.562
5.097

Output

Action Output

#	Time	Action	Message
✓ 1	17:32:48	SELECT precio_venta FROM superhero.articulo LIMIT 0, 1000	13 row(s) returned
✓ 2	17:35:40	CALL AplicarDescuento(70, 30)	6 row(s) affected
✓ 3	17:36:27	SELECT precio_venta FROM superhero.articulo LIMIT 0, 1000	13 row(s) returned

Se entregará el código de cada apartado y una muestra de su ejecución