

EXAMEN BASE DE DATOS 2ª EVALUACIÓN

Nombre: *AndreaVieira Hernández*

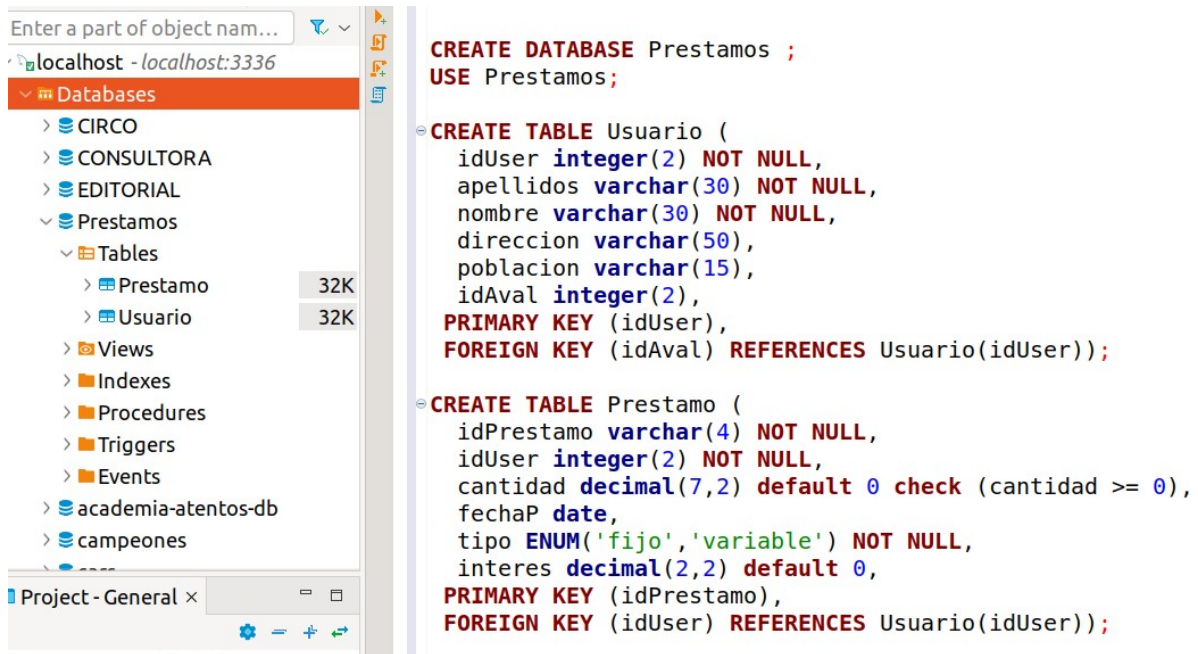
11/03/2022

1. (2 puntos)

Crear la base de datos llamada Prestamos con las siguientes tablas:

PRIMARY KEY * FOREIGN KEY (U) Único

USUARIO		PRESTAMO	
Campo	Tipo	Campo	Tipo
# idUser	numérico (2)	# idPrestamo	texto (4)
apellidos	texto (30)	* idUser	numérico (2)
nombre	texto (30)	cantidad	número real con 7 dígitos y 2 decimales
direccion	texto (50)	fechaP	fecha
poblacion	texto (15)	tipo	texto (2)
*Idaval	numérico (2)	interes	Número real de 2 dígitos con 2 decimales
Todos los campos son obligatorios salvo direccion, población e Idaval que pueden quedar vacíos.		El campo cantidad almacenará por defecto el valor 0 y, así mismo, debe comprobarse que almacene siempre un valor positivo.	
No se permitirá borrar un usuario si tiene préstamos asociados.		El tipo será fijo/variable	
El Idaval es un campo opcional que contiene el IdUser del que le avala (si es que existe).		Todos los campos son obligatorios	
		Por defecto, cantidad e interés valdrán 0.	



```
CREATE DATABASE Prestamos ;
USE Prestamos;
```

```
CREATE TABLE Usuario (
    idUser integer(2) NOT NULL,
    apellidos varchar(30) NOT NULL,
    nombre varchar(30) NOT NULL,
    direccion varchar(50),
    poblacion varchar(15),
    idAval integer(2),
    PRIMARY KEY (idUser),
    FOREIGN KEY (idAval) REFERENCES Usuario(idUser));
```

```
CREATE TABLE Prestamo (
    idPrestamo varchar(4) NOT NULL,
    idUser integer(2) NOT NULL,
    cantidad decimal(7,2) default 0 check (cantidad >= 0),
    fechaP date,
    tipo ENUM('fijo','variable') NOT NULL,
    interes decimal(2,2) default 0,
    PRIMARY KEY (idPrestamo),
    FOREIGN KEY (idUser) REFERENCES Usuario(idUser));
```

- a. Añadir el campo edad al Usuario, que deberá de comprobar que sea mayor que 18 años.

Columns:

- idUser (int)
- apellidos (varchar(30))
- nombre (varchar(30))
- direccion (varchar(50))
- poblacion (varchar(10))
- idAval (int)
- edad (int)

Foreign Keys:

- ExerciciosE localhost 2
- ExerciciosE localhost
- empleados localhost
- examen11 localhost

Statistics 1 x

```
-- Añadir el campo edad al Usuario, que deberá de comprobar que sea mayor que 18 años.
ALTER TABLE Usuario ADD edad int check (edad >= 18);

-- Insertar 2 campos en cada tabla, con IdUser 1 y 2, e IdPrestamo 1 y 2.

-- Borra el idPrestamo 2.
```

Name	Value
Updated Rows	0
Query	-- Añadir el campo edad al Usuario, que deberá de comprobar que sea mayor que 18 años. ALTER TABLE Usuario ADD edad int check (edad >= 18)
Finish time	Fri Mar 11 10:41:36 CET 2022

ALTER TABLE Usuario ADD edad int check (edad >= 18);
Entiendo que solo tiene un tope mínimo.

b. Insertar 2 campos en cada tabla, con IdUser 1 y 2, e IdPrestamo 1 y 2.

```
-- Insertar 2 campos en cada tabla, con IdUser 1 y 2, e IdPrestamo
INSERT INTO Usuario VALUES
(1, 'Pérez', 'Luis', 'Calle Orfila, 4', 'Sevilla', null, 47),
(2, 'Mora', 'Ana', 'Calle Amor, 23', 'Sevilla', 1, 27);

-- Borra el idPrestamo 2.
```

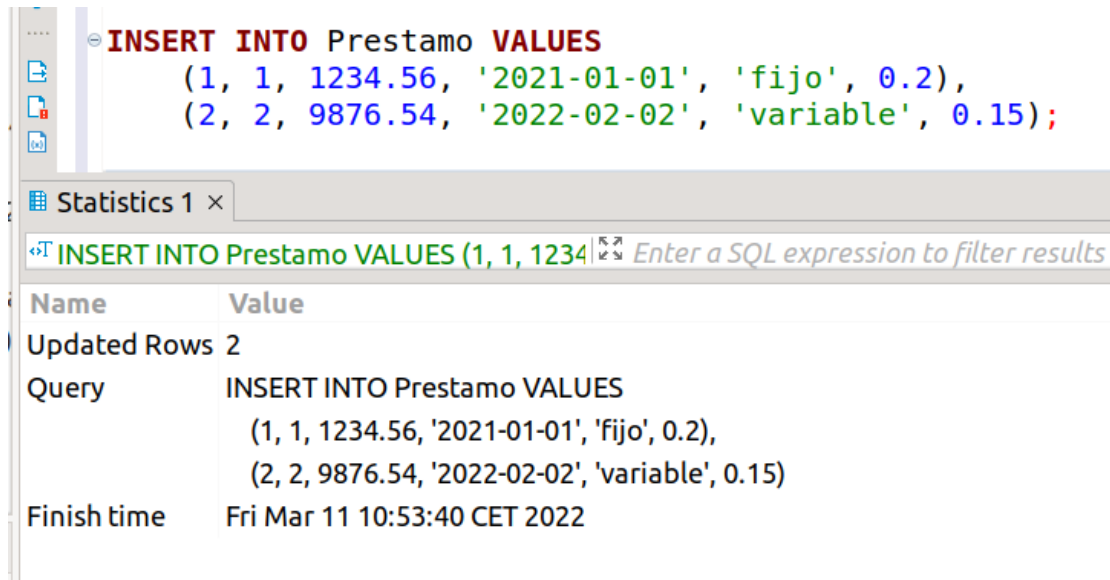
Name	Value
Updated Rows	2
Query	INSERT INTO Usuario VALUES (1, 'Pérez', 'Luis', 'Calle Orfila, 4', 'Sevilla', null, 47), (2, 'Mora', 'Ana', 'Calle Amor, 23', 'Sevilla', 1, 27)
Finish time	Fri Mar 11 10:46:26 CET 2022

INSERT INTO Usuario VALUES

```

(1, 'Pérez', 'Luis', 'Calle Orfila, 4', 'Sevilla',
null, 47),
(2, 'Mora', 'Ana', 'Calle Amor, 23', 'Sevilla', 1, 27);

```



The screenshot shows a SQL IDE with a query editor and a statistics window. The query editor contains the following SQL statement:

```

INSERT INTO Prestamo VALUES
(1, 1, 1234.56, '2021-01-01', 'fijo', 0.2),
(2, 2, 9876.54, '2022-02-02', 'variable', 0.15);

```

The statistics window, titled "Statistics 1", displays the following information:

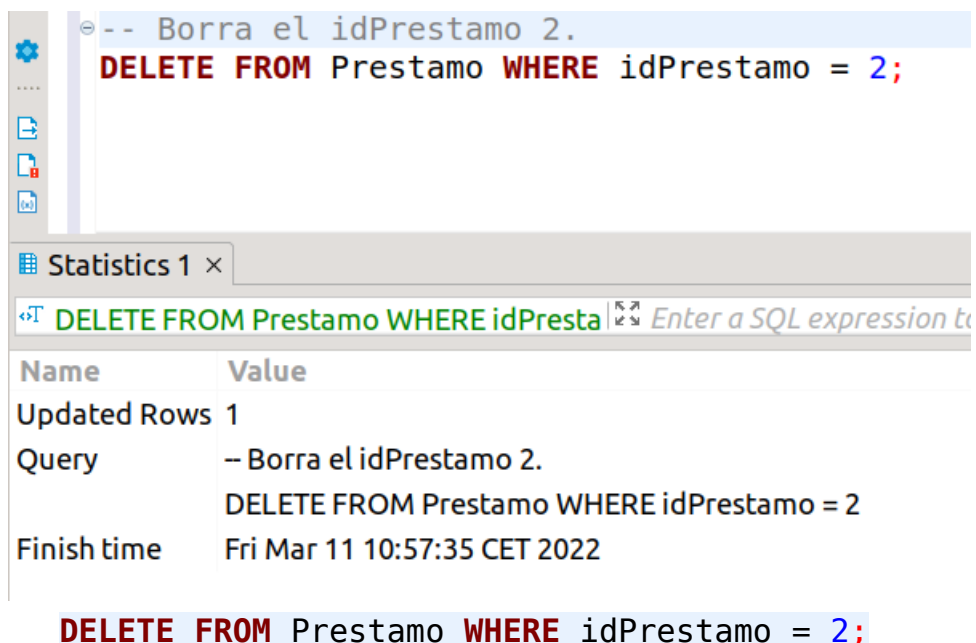
Name	Value
Updated Rows	2
Query	INSERT INTO Prestamo VALUES (1, 1, 1234.56, '2021-01-01', 'fijo', 0.2), (2, 2, 9876.54, '2022-02-02', 'variable', 0.15)
Finish time	Fri Mar 11 10:53:40 CET 2022

```

INSERT INTO Prestamo VALUES
(1, 1, 1234.56, '2021-01-01', 'fijo', 0.2),
(2, 2, 9876.54, '2022-02-02', 'variable', 0.15);

```

c. Borra el idPrestamo 2.



The screenshot shows a SQL IDE with a query editor and a statistics window. The query editor contains the following SQL statement:

```

-- Borra el idPrestamo 2.
DELETE FROM Prestamo WHERE idPrestamo = 2;

```

The statistics window, titled "Statistics 1", displays the following information:

Name	Value
Updated Rows	1
Query	-- Borra el idPrestamo 2. DELETE FROM Prestamo WHERE idPrestamo = 2
Finish time	Fri Mar 11 10:57:35 CET 2022

Below the statistics window, the SQL statement is repeated:

```

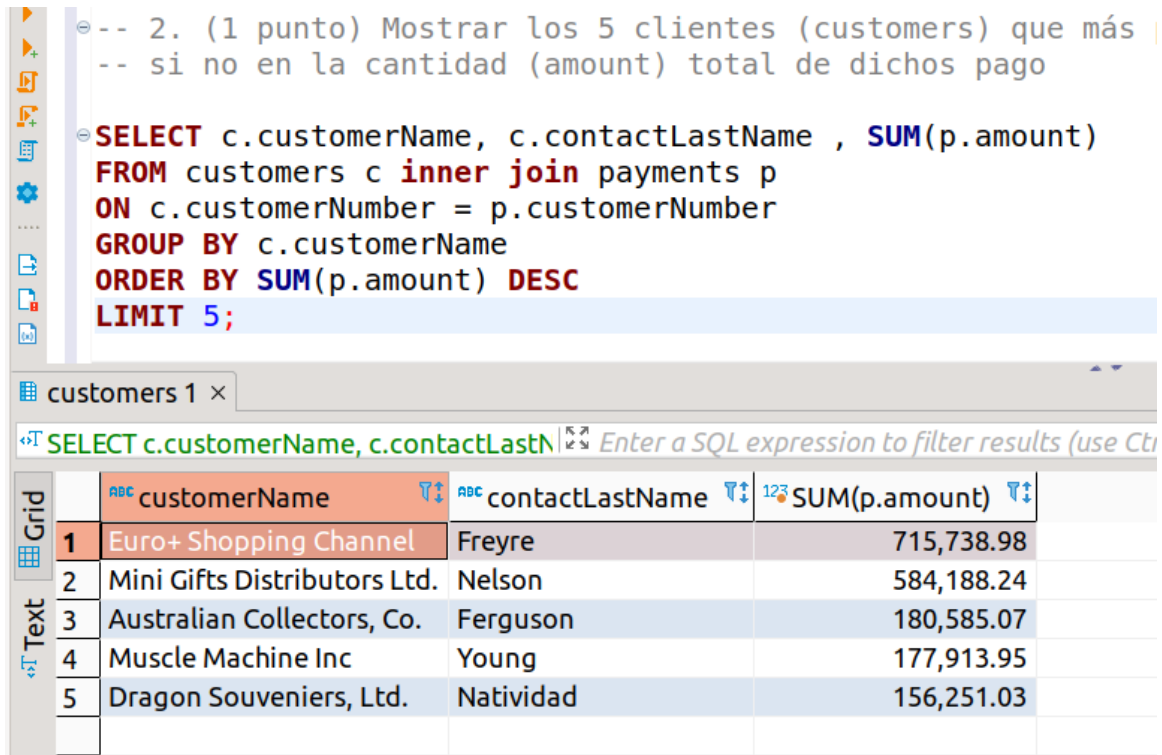
DELETE FROM Prestamo WHERE idPrestamo = 2;

```

CONSULTAS SQL

Dada la BD [classicModels](#) instalarla en vuestro servidor MySql y realizar las siguientes consultas a la BD:

- (1 punto) Mostrar los 5 clientes (customers) que más pagos (payments) han realizado, no en número de pagos, si no en la cantidad (amount) total de dichos pago



The screenshot shows a MySQL IDE interface. At the top, a SQL query is entered in a text area. Below the query, a tab labeled 'customers 1' is active, displaying the results of the query in a table grid. The table has three columns: 'customerName', 'contactLastName', and 'SUM(p.amount)'. The results are sorted by the total amount in descending order, showing the top 5 customers.

```
-- 2. (1 punto) Mostrar los 5 clientes (customers) que más  
-- si no en la cantidad (amount) total de dichos pago  
  
SELECT c.customerName, c.contactLastName , SUM(p.amount)  
FROM customers c inner join payments p  
ON c.customerNumber = p.customerNumber  
GROUP BY c.customerName  
ORDER BY SUM(p.amount) DESC  
LIMIT 5;
```

	customerName	contactLastName	SUM(p.amount)
1	Euro+ Shopping Channel	Freyre	715,738.98
2	Mini Gifts Distributors Ltd.	Nelson	584,188.24
3	Australian Collectors, Co.	Ferguson	180,585.07
4	Muscle Machine Inc	Young	177,913.95
5	Dragon Souvenirs, Ltd.	Natividad	156,251.03

```
SELECT c.customerName, c.contactLastName , SUM(p.amount)  
FROM customers c inner join payments p  
ON c.customerNumber = p.customerNumber  
GROUP BY c.customerName  
ORDER BY SUM(p.amount) DESC  
LIMIT 5;
```

- (1 punto) Mostrar los datos de los pedidos (orders) realizados por clientes que estén atendidos por empleados (employees) de la oficina (offices) de París.

```
-- 3. (1 punto) Mostrar los datos de los pedidos (orders) realizados por clientes que estén atendidos por empleados
-- (employees) de la oficina (offices) de París.
SELECT o.*, o2.city
FROM orders o inner join customers c inner join employees e inner join offices o2
on o.customerNumber = c.customerNumber
and c.salesRepEmployeeNumber = e.employeeNumber
and e.officeCode = o2.officeCode
WHERE e.officeCode = (SELECT o3.officeCode
FROM offices o3
WHERE o3.city = 'Paris');
```

```
SELECT o.*, o2.city
FROM orders o inner join customers c inner join employees e
inner join offices o2
on o.customerNumber = c.customerNumber
and c.salesRepEmployeeNumber = e.employeeNumber
and e.officeCode = o2.officeCode
WHERE e.officeCode = (SELECT o3.officeCode
FROM offices o3
WHERE o3.city = 'Paris');
```

4. (1 punto) Mostrar el precio un resumen por meses y años del número de pedidos realizados en los últimos 20 años (a partir de la fecha actual).

```
-- 4. (1 punto) Mostrar el precio un resumen por meses y años del número de pedidos realizados en los últimos 2
-- (a partir de la fecha actual).
SELECT COUNT(*) Pedidos, o.*
FROM orders o
WHERE (CURDATE()-o.orderDate)<=20
GROUP BY MONTH(o.orderDate), YEAR(o.orderDate);
```

Como los pedidos tienen como fecha a partir del 2003, no hay ninguno que sea anterior a 20 años., por este motivo no devuelve nada la consulta.

```
SELECT COUNT(*) Pedidos, o.*
```

```

FROM orders o
WHERE (CURDATE() - o.orderDate) <= 20
GROUP BY MONTH(o.orderDate), YEAR(o.orderDate);

```

5. (1 punto) Mostrar los datos de los clientes (customers) que han realizado entre 5 y 10 pedidos (orders)

-- 5. (1 punto) Mostrar los datos de los clientes (customers) que han realizado entre 5 y 10 pedidos (orders)

```

SELECT c.customerName, c.contactFirstName, c.contactLastName, COUNT(*)
FROM customers c INNER JOIN orders o
on c.customerNumber = o.customerNumber
GROUP BY c.customerNumber
HAVING COUNT(*) >= 5 AND COUNT(*) <= 10;

```

customers 1 x

SELECT c.customerName, c.contactFirstN... Enter a SQL expression to filter results (use Ctrl+Space)

	customerName	contactFirstName	contactLastName	COUNT(*)
1	Australian Collectors, Co.	Peter	Ferguson	5
2	Danish Wholesale Imports	Jytte	Petersen	5
3	Dragon Souvenirs, Ltd.	Eric	Natividad	5
4	Down Under Souvenirs, In	Mike	Graham	5
5	Reims Collectables	Paul	Henriot	5

```

SELECT          c.customerName,                c.contactFirstName,
c.contactLastName, COUNT(*)
FROM customers c INNER JOIN orders o
on c.customerNumber = o.customerNumber
GROUP BY c.customerNumber
HAVING COUNT(*) >= 5 AND COUNT(*) <= 10;

```


DELIMITER \$\$

```
CREATE PROCEDURE mejorCliente(p_codClient int)
BEGIN
    declare v_cantPedidos int default 0;

    SELECT COUNT(*) into v_cantPedidos
    FROM customers c INNER JOIN orders o
    on c.customerNumber = o.customerNumber
    WHERE c.customerNumber = p_codClient;

    IF (v_cantPedidos > 0) THEN
        SELECT 'El cliente se encuentra entre los 5 mejores';
    ELSE
        SELECT 'El cliente no ha realizado ningún pago';
    END IF;
END$$
DELIMITER ;
```

Statistics 1 ×

SQL Enter a SQL expression to filter results (use Ctrl+Space)

Name	Value
Queries	3
Updated Rows	0
Execute time (ms)	8
Fetch time (ms)	0
Total time (ms)	8
Finish time	2022-03-11 11:49:12.619

CALL mejorCliente(103);

Results 1 ×

CALL mejorCliente(103) Enter a SQL expression to filter

Grid	1
	ABC El cliente se encuentra entre los 5 mejores
1	El cliente se encuentra entre los 5 mejores

```
DELIMITER $$
CREATE PROCEDURE mejorCliente(p_codClient int)
BEGIN
    declare v_cantPedidos int default 0;

    SELECT COUNT(*) into v_cantPedidos
    FROM customers c INNER JOIN orders o
    on c.customerNumber = o.customerNumber
    WHERE c.customerNumber = p_codClient;
```

```

        IF (v_cantPedidos > 0) THEN
            SELECT 'El cliente se encuentra entre los 5
mejores';
        ELSE
            SELECT 'El cliente no ha realizado ningún pago';
        END IF;
    END$$
DELIMITER ;

CALL mejorCliente(103);

```

7. (1 punto) Escriba una función llamada crear_email que dados los parámetros de entrada: nombre, apellidos y dominio, cree una dirección de email y la devuelva como salida:

- Procedimiento: crear_email
- Entrada:
 - nombre (cadena de caracteres)
 - apellidos (cadena de caracteres)
 - dominio (cadena de caracteres)
- Salida:
 - email (cadena de caracteres)

devuelva una dirección de correo electrónico con el siguiente formato:

- El primer carácter del parámetro nombre.
- Los tres primeros caracteres del parámetro apellidos.
- El carácter @.
- El dominio pasado como parámetro.

Una vez creado el procedimiento verifica que funcione correctamente.

```
DELIMITER $$
CREATE FUNCTION crear_email(p_nombre varchar(50), p_apellidos varchar(80), p_dominio varchar(50))
RETURNS varchar(200)
DETERMINISTIC
BEGIN
    Declare v_email varchar(200) default '';

    SET v_email = concat(substr(p_nombre, 1, 1), substr(p_apellidos, 1, 3), '@', p_dominio);

    RETURN v_email;

END $$
DELIMITER ;

SELECT crear_email('andrea', 'vieira', 'alixar.org');
```

Results 1 x

SELECT crear_email('andrea', 'vieira', 'alixar.org')

1	avie@alixar.org
---	-----------------

DELIMITER \$\$

CREATE FUNCTION crear_email(p_nombre **varchar**(50), p_apellidos **varchar**(80), p_dominio **varchar**(50))

RETURNS varchar(200)

DETERMINISTIC

BEGIN

Declare v_email **varchar**(200) **default** '';

SET v_email = **concat**(**substr**(p_nombre, 1, 1),
substr(p_apellidos, 1, 3), '@', p_dominio);

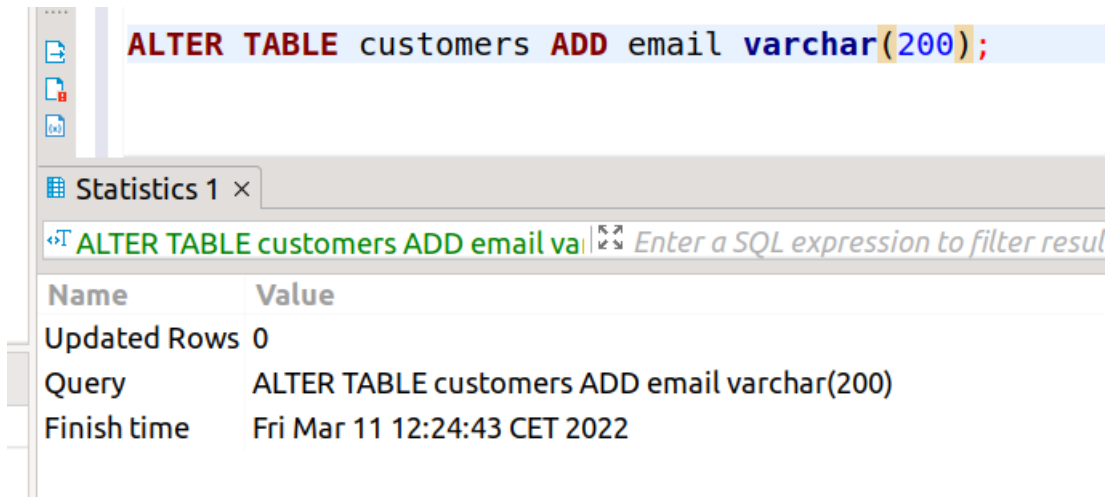
RETURN v_email;

END \$\$

DELIMITER ;

SELECT crear_email('andrea', 'vieira', 'alixar.org');

8. (1 punto) Añadir un campo de email para la tabla customers (tipo texto 100).
Crear un procedimiento llamado actualizar_columna_email que actualice la columna email de la tabla customers en el dominio @empresa.net. Este procedimiento hará uso del procedimiento crear_email que hemos creado en el paso anterior.



DELIMITER \$\$

```
CREATE PROCEDURE actualizar_columna_email()  
BEGIN
```

```
    declare v_nombre varchar(200) default '';  
    declare v_apellido varchar(200) default '';  
    declare v_dominio varchar(200) default 'empresa.net';
```

```
    SELECT c.contactFirstName into v_nombre  
    FROM customers c;
```

```
    SELECT c.contactLastName into v_apellido  
    FROM customers c;
```

```
    UPDATE customers SET email = crear_email(v_nombre,  
v_apellido, v_dominio);  
END$$
```

DELIMITER ;

9. (1 punto) Crea un *trigger* trigger_guardar_email_after_update con las siguientes características:
- Se ejecuta sobre la tabla customers.
 - Se ejecuta *después* de una operación de *actualización*.
 - Cada vez que un alumno modifique su dirección de email se deberá insertar un nuevo registro en una **nueva tabla** llamada log_cambios_email.

La tabla log_cambios_email contiene los siguientes campos:

- customerNumber: clave primaria (entero autonumérico)

- `customerName`: nombre cliente(texto)
- `fecha_hora`: marca de tiempo con el instante del cambio (fecha y hora)
- `old_email`: valor anterior del email (cadena de caracteres)
- `new_email`: nuevo valor con el que se ha actualizado