**Tarea: TRIGGERS Y STORED PROCEDURES**

JORGE LUCAS BENAVIDES

Crear el siguiente esquema e insertar el siguiente contenido:

create database practica;

CREATE TABLE empleado

( id\_emp INTEGER,

nombre VARCHAR(100),

dep INTEGER,

PRIMARY KEY (id\_emp));

CREATE TABLE departamento

(nro\_dep INTEGER,

nombre VARCHAR(20),

PRIMARY KEY (nro\_dep));

INSERT INTO empleado VALUES(1, 'Arturo', 30);

INSERT INTO empleado VALUES(2, 'Eva', 20);

INSERT INTO empleado VALUES(3, 'Miguel', 10);

INSERT INTO empleado VALUES(4, 'Alejandro', 10);

INSERT INTO empleado VALUES(5, 'Elena', 50);

INSERT INTO empleado VALUES(6, 'Carmen', 40);

INSERT INTO empleado VALUES(7, 'Mario', 30);

INSERT INTO empleado VALUES(8, 'Esteban', 10);

INSERT INTO empleado VALUES(9, 'Paco', 20);

INSERT INTO empleado VALUES(10, 'Pili', 30);

INSERT INTO departamento VALUES(10, 'Informática');

INSERT INTO departamento VALUES(20, 'Ventas');

INSERT INTO departamento VALUES(30, 'Compras');

INSERT INTO departamento VALUES(40, 'Publicidad');

INSERT INTO departamento VALUES(50, 'Recursos Humanos');

**1. Crear un disparador “insertar\_empleado” para controlar la no inserción de una fila en la tabla empleado cuando no sea horario laboral. Para este ejercicio vamos a considerar que el horario laboral del negocio es de viernes a miércoles de 2 pm a 10 pm**

**1.1 Copie el código que utilizó para crear el disparador**

DROP TRIGGER IF EXISTS BEFORE\_INSERT\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_INSERT\_EMPLEADO BEFORE INSERT ON EMPLEADO FOR EACH ROW

BEGIN

IF NOT ((DAYOFWEEK(NOW()) BETWEEN 4 AND 6) AND (current\_time() between '14:00:00' AND '22:00:00')) THEN

SET NEW.ID\_EMP = NULL; -- Impider la operación.

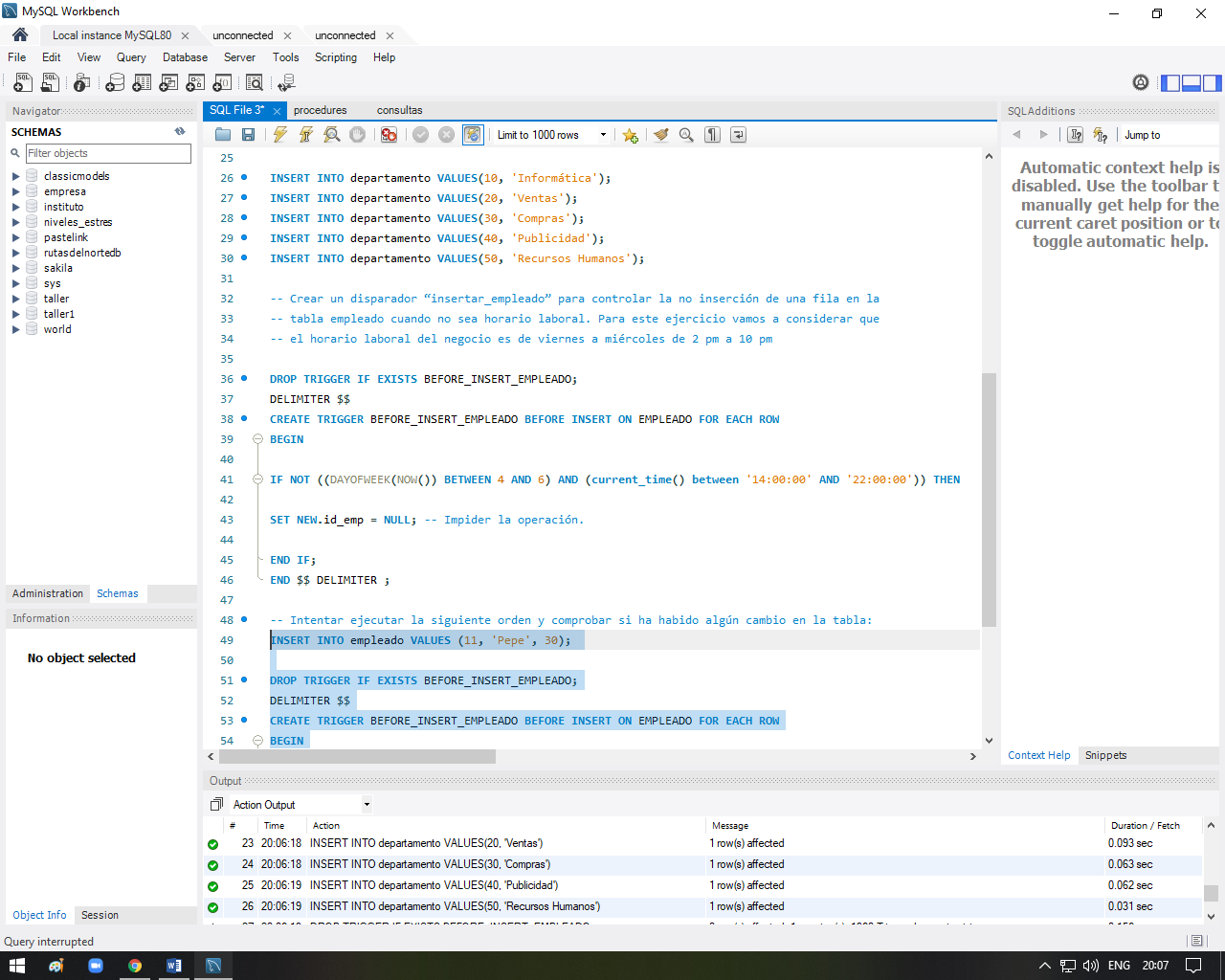
END IF;

END $$ DELIMITER ;

Intentar ejecutar la siguiente orden y comprobar si ha habido algún cambio en la tabla:

INSERT INTO empleado VALUES (11, 'Pepe', 30);

**1.2 Copie el resultado (de la consola) luego de tratar de ejecutar la orden anterior:**

INSERT INTO empleado VALUES (11, 'Pepe', 30);

DROP TRIGGER IF EXISTS BEFORE\_INSERT\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_INSERT\_EMPLEADO BEFORE INSERT ON EMPLEADO FOR EACH ROW

BEGIN

IF NOT ((DAYOFWEEK(NOW()) BETWEEN 4 AND 6) AND (current\_time() between '14:00:00' AND '22:00:00')) THEN

SIGNAL SQLSTATE '45000'; -- Impide la operación.

END IF;

END $$ DELIMITER ;

**2. Crear el mismo disparador pero utilizando un procedimiento en el cuerpo del disparador que controle si la hora y el día son correctos.**

**2.1 Copie el código que utilizó para crear el procedimiento.**

DROP PROCEDURE IF EXISTS VERIFICAR\_HORARIO;

DELIMITER //

CREATE PROCEDURE VERIFICAR\_HORARIO (OUT VERF BOOLEAN)

BEGIN

IF NOT ((DAYOFWEEK(NOW()) between 4 AND 6) AND (current\_time() between '14:00:00' AND '22:00:00')) THEN

SET VERF = 0;

ELSE

SET VERF = 1;

END IF;

END // DELIMITER ;

DROP TRIGGER IF EXISTS BEFORE\_INSERT\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_INSERT\_EMPLEADO BEFORE INSERT ON EMPLEADO FOR EACH ROW

BEGIN

CALL VERIFICAR\_HORARIO(@flag);

IF @flag = 0 THEN

SET NEW.ID\_EMP = NULL;

END IF;

END $$ DELIMITER ;

*Nota: Desde un disparador se puede llamar a un procedimiento mediante su nombre en el cuerpo del disparador utilizando la función* **CALL***.*

**2.2 Copie el código que utilizó para crear el disparador**

DROP TRIGGER IF EXISTS BEFORE\_INSERT\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_INSERT\_EMPLEADO BEFORE INSERT ON EMPLEADO FOR EACH ROW

BEGIN

CALL VERIFICAR\_HORARIO(@flag);

IF @flag = 0 THEN

SET NEW.ID\_EMP = NULL;

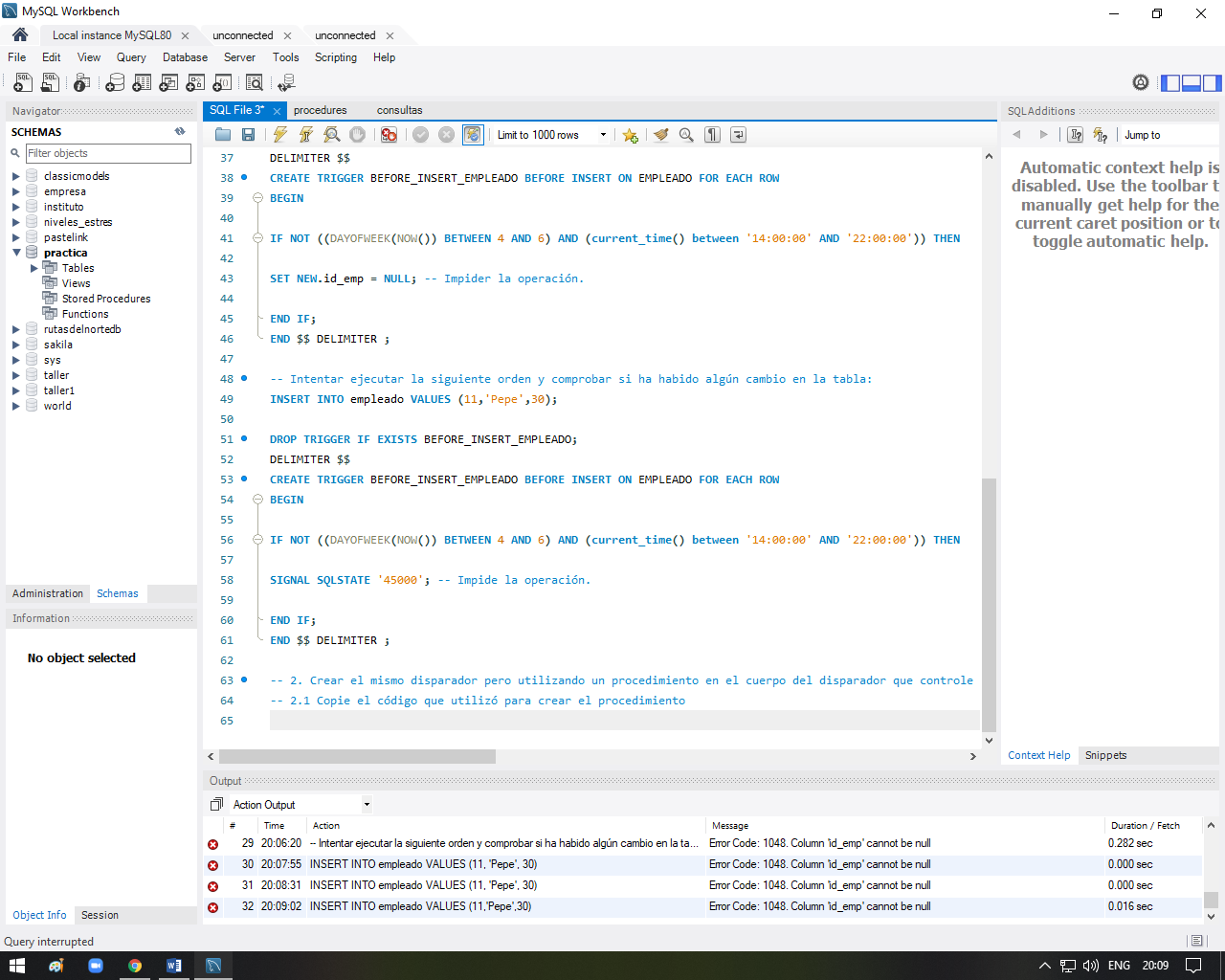
END IF;

END $$ DELIMITER ;

Intentar ejecutar la siguiente orden y comprobar si ha habido algún cambio en la tabla:

INSERT INTO empleado VALUES (20, 'Andres', 30);

**2.3 Copie el resultado (de la consola) luego de tratar de ejecutar la orden anterior:**

INSERT INTO empleado VALUES (11,'Pepe',30);

DROP TRIGGER IF EXISTS BEFORE\_INSERT\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_INSERT\_EMPLEADO BEFORE INSERT ON EMPLEADO FOR EACH ROW

BEGIN

IF NOT ((DAYOFWEEK(NOW()) BETWEEN 4 AND 6) AND (current\_time() between '14:00:00' AND '22:00:00')) THEN

SIGNAL SQLSTATE '45000'; -- Impide la operación.

END IF;

END $$ DELIMITER ;

**3. Crear un disparador que impida *cualquier operación* (insert, select, update, delete) en la tabla empleado fuera del horario laboral dando una indicación específica de la operación que no es realizable y porqué no es realizable (por ejemplo, que se encuentra fuera del horario laboral).**

**3.1 Copie el código que utilizó para crear el disparador**

DROP TRIGGER IF EXISTS BEFORE\_INSERT\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_INSERT\_EMPLEADO BEFORE INSERT ON EMPLEADO FOR EACH ROW

BEGIN

IF NOT ((DAYOFWEEK(NOW()) BETWEEN 4 AND 6) AND (current\_time() between '14:00:00' AND '22:00:00')) THEN

SIGNAL SQLSTATE '45000' -- Impide la operación.

SET MESSAGE\_TEXT = 'Fuera del horario laboral'; -- Pones el mensaje que quieras.

END IF;

END $$ DELIMITER ;

**3.2 Para probar si su disparador funciona cambie la fecha del sistema de tal forma que no permita realizar ninguna operación en la tabla. Copie el resultado (de la consola) al tratar de realizar: un *insert*, un *update*, un *select*, y un *delete* a la tabla.**

-- Para update

DROP TRIGGER IF EXISTS BEFORE\_UPDATE\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_UPDATE\_EMPLEADO BEFORE UPDATE ON EMPLEADO FOR EACH ROW

BEGIN

IF NOT ((DAYOFWEEK(NOW()) BETWEEN 4 AND 6) AND (current\_time() between '14:00:00' AND '22:00:00')) THEN

SIGNAL SQLSTATE '45000' -- Impide la operación.

SET MESSAGE\_TEXT = 'Fuera del horario laboral'; -- Pones el mensaje que quieras.

END IF;

END $$ DELIMITER ;

USE PRACTICA;

-- Para delete

DROP TRIGGER IF EXISTS BEFORE\_DELETE\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_DELETE\_EMPLEADO BEFORE DELETE ON EMPLEADO FOR EACH ROW

BEGIN

IF NOT ((DAYOFWEEK(NOW()) BETWEEN 4 AND 6) AND (current\_time() between '14:00:00' AND '22:00:00')) THEN

SIGNAL SQLSTATE '45000'; -- Impide la operación

SET MESSAGE\_TEXT = 'Fuera del horario laboral'; -- Pones el mensaje que quieras.

END IF;

END $$ DELIMITER ;

***Reestablezca la fecha del sistema para que pueda seguir trabajando en el resto de ejercicios.***

**4. Añadir a la tabla empleado una nueva columna salario. El valor por defecto debe ser el salario básico. Copie el código que utilizó para crear el nuevo campo.**

select \* from empleado;

ALTER TABLE EMPLEADO ADD SALARIO FLOAT DEFAULT 400;

**5. Cree un disparador que cada vez que se ingrese un empleado, su salario se modifique de la siguiente manera:**

* Si el empleado pertenece al departamento de ‘Informática’, su salario será el triple del salario básico.
* Si el empleado pertenece al departamento de ‘Ventas’, su salario será el doble del salario básico.
* Si el empleado pertenece al departamento de ‘Recursos Humanos’, su salario será el 1.5 del salario básico.

**5.1. Copie el código que utilizó para crear el disparador.**

DROP TRIGGER IF EXISTS BEFORE\_INSERT\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER BEFORE\_INSERT\_EMPLEADO BEFORE INSERT ON EMPLEADO FOR EACH ROW

BEGIN

-- Seteo el nombre del departamento que tiene ese id.

SET @nom\_depar = (SELECT NOMBRE FROM DEPARTAMENTO WHERE NRO\_DEP = NEW.DEP);

IF (@nom\_depar LIKE 'INFORMÁTICA') THEN

SET NEW.SALARIO = (NEW.SALARIO \* 3);

END IF;

IF (@nom\_depar LIKE 'VENTAS') THEN

SET NEW.SALARIO = NEW.SALARIO \* 2;

END IF;

IF (@nom\_depar LIKE 'RECURSOS HUMANOS') THEN

SET NEW.SALARIO = NEW.SALARIO \* 1.5;

END IF;

END $$ DELIMITER ;

**5.2. Verifique que los datos se ingresen normalmente. Ejecute las siguientes inserciones.**

INSERT INTO empleado VALUES (13, 'Antonella', 10);

INSERT INTO empleado VALUES (14, 'Jose', 20);

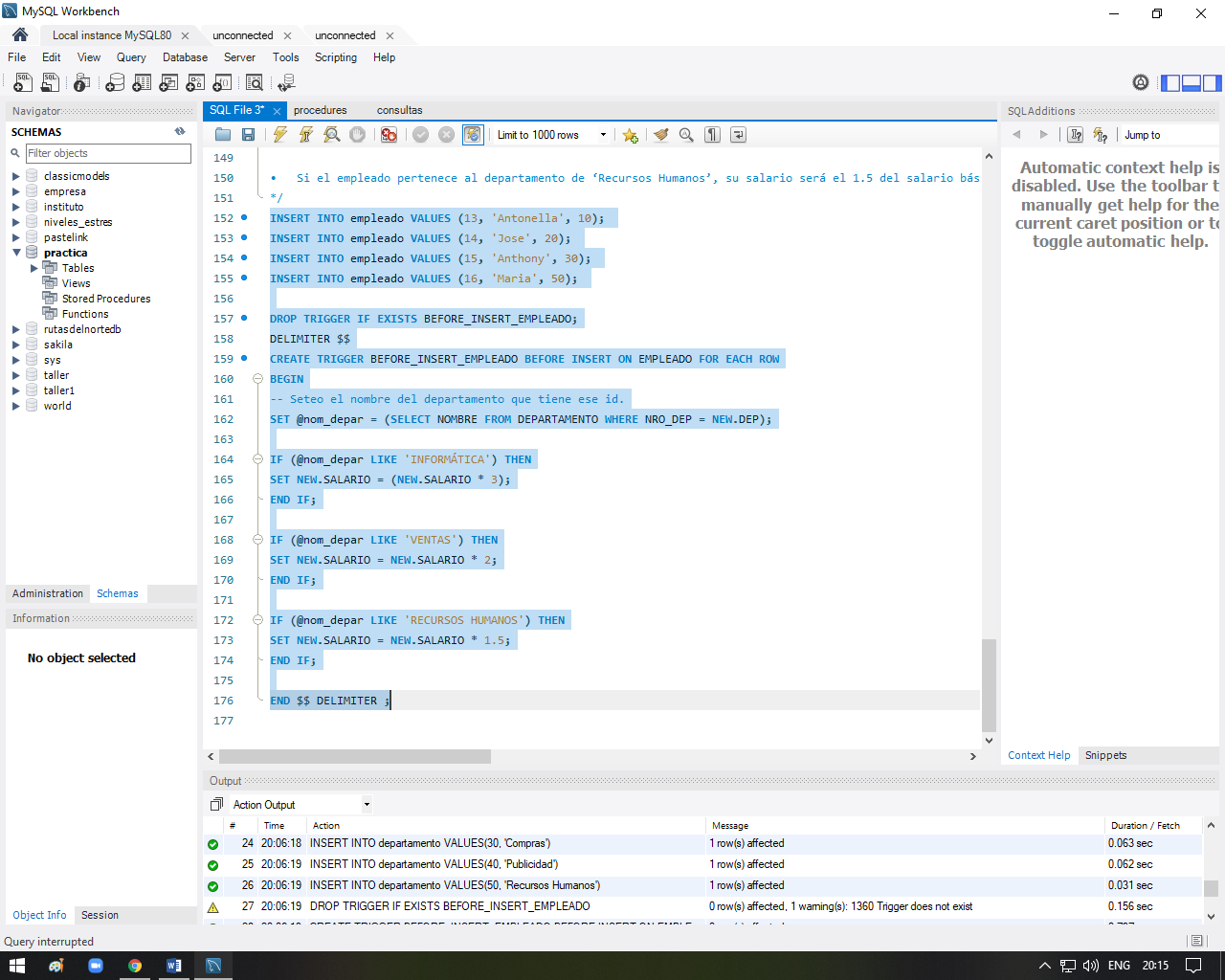
INSERT INTO empleado VALUES (15, 'Anthony', 30);

INSERT INTO empleado VALUES (16, 'Maria', 50);

***Ingrese sus datos en la tabla empleado (por ejemplo: un numero, su nombre, y el codigo del departamento de informática.***

**Pegue el resultado (de la consola) al ejecutar las inserciones. Además, pegue el resultado (captura de pantalla) luego de hacer un *Select \* from empleado.***

**6. Crear una tabla llamada *Cambios* dónde se almacenarán, mediante la utilización de un disparador, los cambios que se han llevado a cabo en la tabla empleado de forma que ante cada cambio de departamento o de salario de un empleado, la nueva tabla almacenará el identificador del empleado, el antiguo y el nuevo salario (a\_salario, n\_salario), o el antiguo y el nuevo departamento (a\_dep, n\_dep) junto con la fecha y hora en que el cambio tuvo lugar.**



**Tabla: Cambios**

**- id\_empleado (NOT null)**

**- v\_salario (null)**

**- n\_salario (null)**

**- v\_dep (null)**

**- n\_dep (null)**

**- Fecha (datetime) NOT NULL**

**6.1 Copie código que utilizó para crear la tabla “Cambios”**

DROP TABLE IF EXISTS CAMBIOS;

CREATE TABLE CAMBIOS (

ID\_CAMBIOS INT PRIMARY KEY NOT NULL UNIQUE AUTO\_INCREMENT,

ID\_EMPLEADO INT NOT NULL,

V\_SALARIO FLOAT NULL,

N\_SALARIO FLOAT NULL,

V\_DEP INT NULL,

N\_DEP INT NULL,

FECHA DATETIME NOT NULL

);

DROP TRIGGER IF EXISTS AFTER\_UPDATE\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER AFTER\_UPDATE\_EMPLEADO AFTER UPDATE ON EMPLEADO FOR EACH ROW

BEGIN

INSERT INTO CAMBIOS (ID\_EMPLEADO,V\_SALARIO,N\_SALARIO,V\_DEP,N\_DEP,FECHA) VALUES (OLD.ID\_EMP,OLD.SALARIO,NEW.SALARIO,OLD.DEP,NEW.DEP,NOW());

END $$ DELIMITER ;

**6.2 Copie el código que utilizó para crear el disparador que permite guardar los cambios de la tabla “Empleado”.**

DROP TRIGGER IF EXISTS AFTER\_UPDATE\_EMPLEADO;

DELIMITER $$

CREATE TRIGGER AFTER\_UPDATE\_EMPLEADO AFTER UPDATE ON EMPLEADO FOR EACH ROW

BEGIN

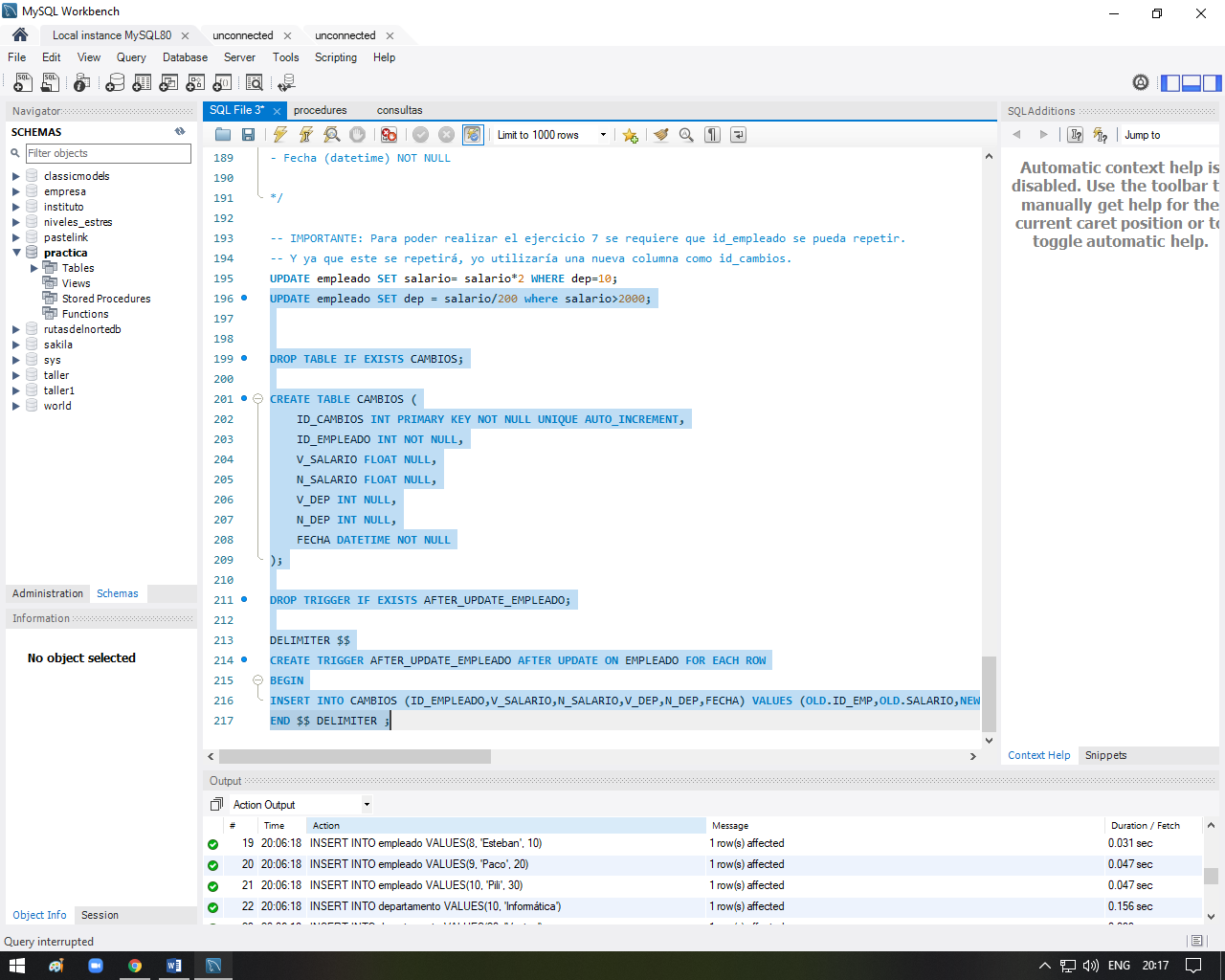
INSERT INTO CAMBIOS (ID\_EMPLEADO,V\_SALARIO,N\_SALARIO,V\_DEP,N\_DEP,FECHA) VALUES (OLD.ID\_EMP,OLD.SALARIO,NEW.SALARIO,OLD.DEP,NEW.DEP,NOW());

END $$ DELIMITER ;

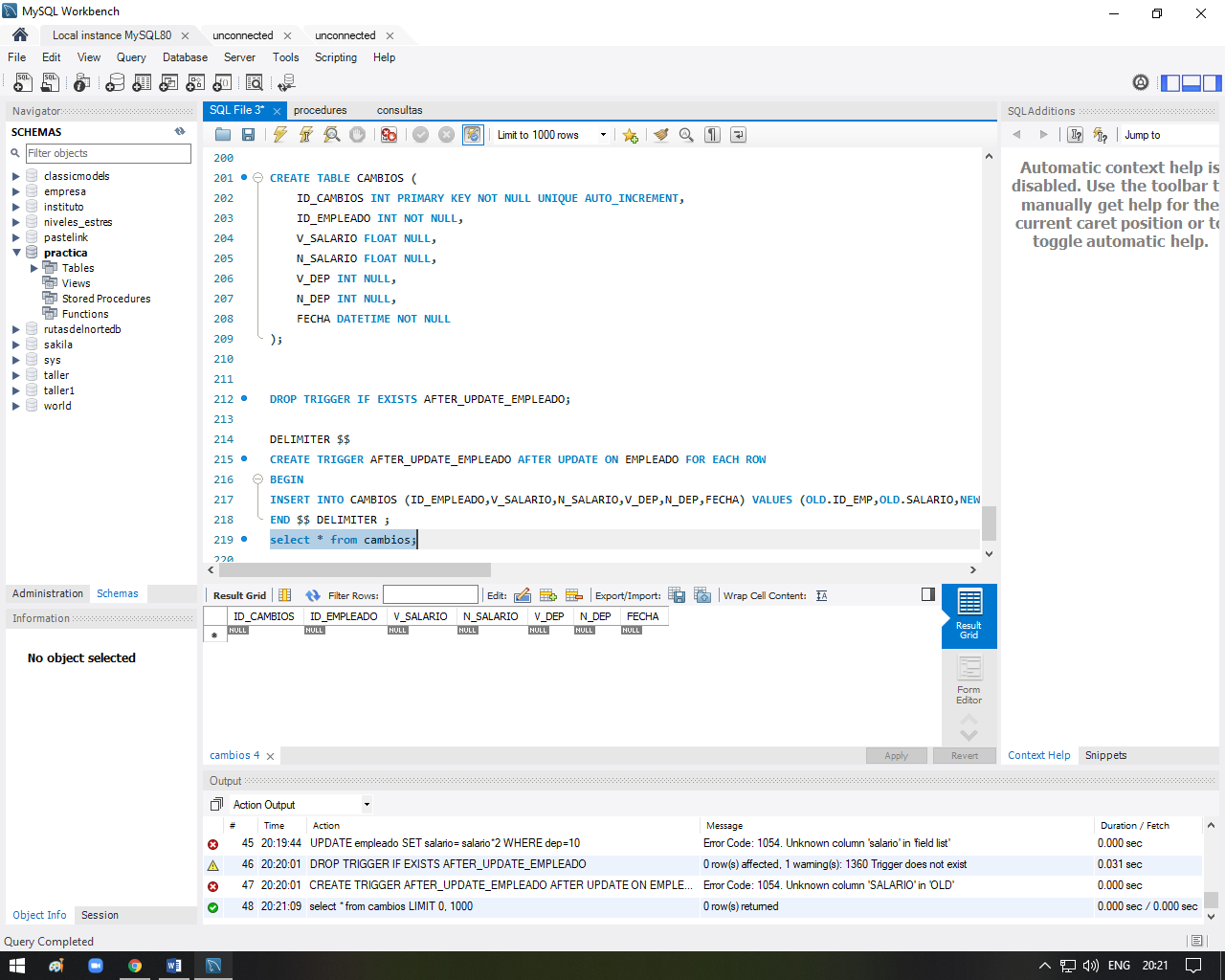
**6.3. Probar las siguientes consultas y mostrar los resultados obtenidos.**

1. UPDATE empleado SET salario= salario\*2 WHERE dep=10;
2. UPDATE empleado SET dep = salario/200 where salario>2000;
3. Para el siguiente update, utilice sus datos (su nombre). Modifique su registro de tal manera que el salario nuevo represente el 12% del valor actual.

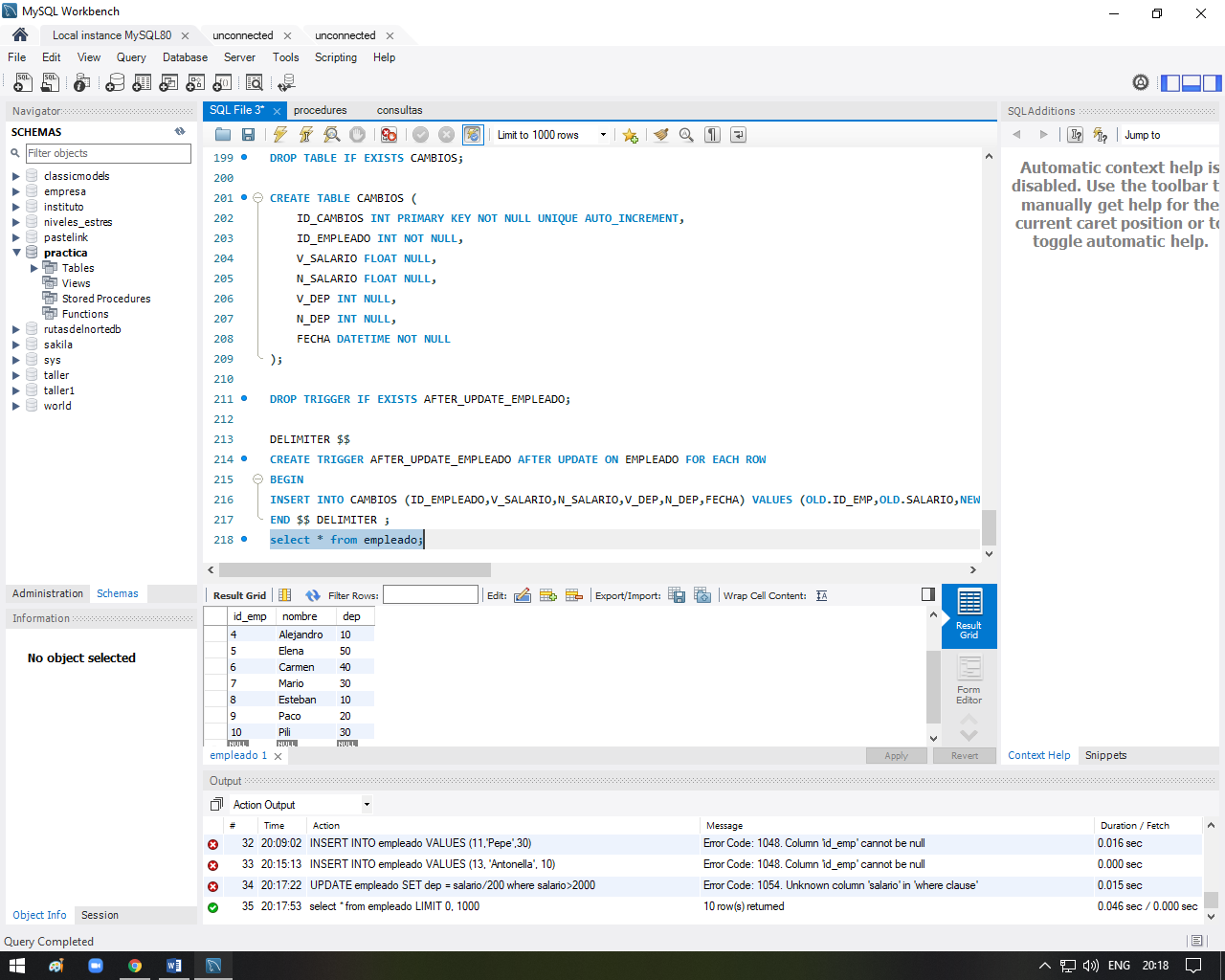
**6.4 Copie el resultado (consola) de la tabla Cambios luego de haber hecho el update 1, 2 y 3.**



**6.5 Pegue el resultado (captura de pantalla) al hacer *un Select \* from Cambios;***



**6.6 Pegue el resultado (captura de pantalla) al hacer *un Select \* from empleado;***



**7. Añadir a la tabla *empleado* una nueva columna denominada media (tipo de dato decimal, valor por defecto 0) que contendrá́ la media del salario que el trabajador ha tenido en la empresa. Esta columna tendrá́ por defecto el salario actual del trabajador hasta que este salario se modifique.**

**7.1 Copie el código que utilizó para crear la nueva columna y la modificación del valor de la nueva columna con el salario actual del empleado.**

ALTER TABLE EMPLEADO ADD MEDIA DECIMAL DEFAULT 0; -- Nueva columna, por defecto 0

UPDATE EMPLEADO SET MEDIA = SALARIO; -- Otro por defecto xd. El salario que tiene actualmente.

**7.2 Cuando se salario se modifique, media pasará a tener el valor correspondiente la suma del salario anterior, el salario actual dividido para dos. Modifique el disparador que se ejecuta en la acción *before update* de la tabla empleado, de tal forma que guarde el nuevo valor en la columna media.**

**Copie el código que utilizó para crear el disparador.**

*DROP TRIGGER IF EXISTS MEDIA\_SALARIO;*

*DELIMITER $$*

*CREATE TRIGGER MEDIA\_SALARIO BEFORE UPDATE ON EMPLEADO FOR EACH ROW*

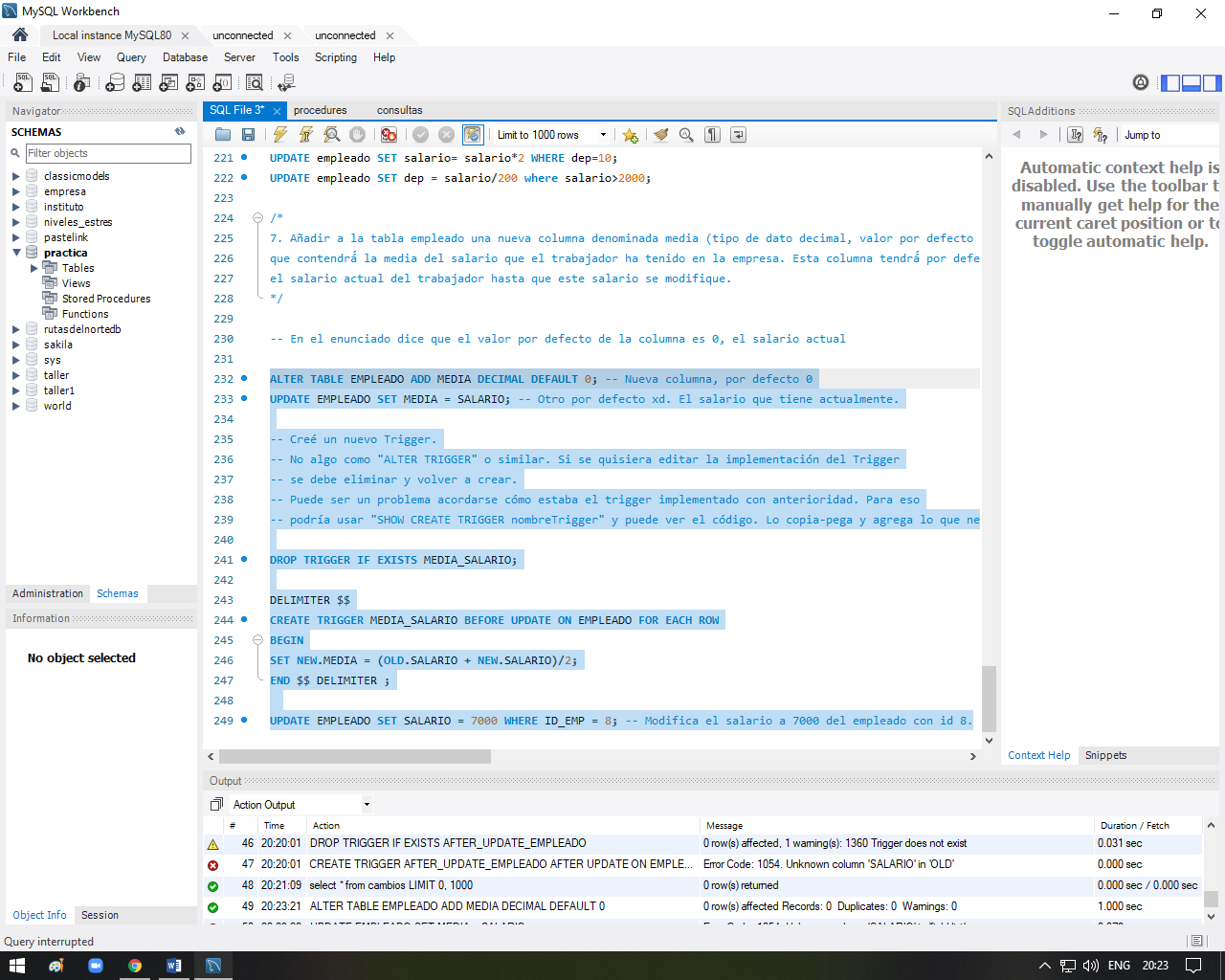
*BEGIN*

*SET NEW.MEDIA = (OLD.SALARIO + NEW.SALARIO)/2;*

*END $$ DELIMITER ;*

*UPDATE EMPLEADO SET SALARIO = 7000 WHERE ID\_EMP = 8; -- Modifica el salario a 7000 del empleado con id 8.*

**7.3 Modifique el salario del empleado cuyo id es igual a 8. El nuevo salario del empleado será igual a 7000. Copie los resultados obtenidos (captura de pantalla) al hacer un *select \* from empleado* luego de haber realizado dicho cambio.**



**8. Crear una vista llamada *PromedioSalarioporDepartamento* en dónde se muestre el identificador del departamento y el promedio del salario de los empleados por departamento. *Nota: La tabla empleado tiene el id del departamento.***

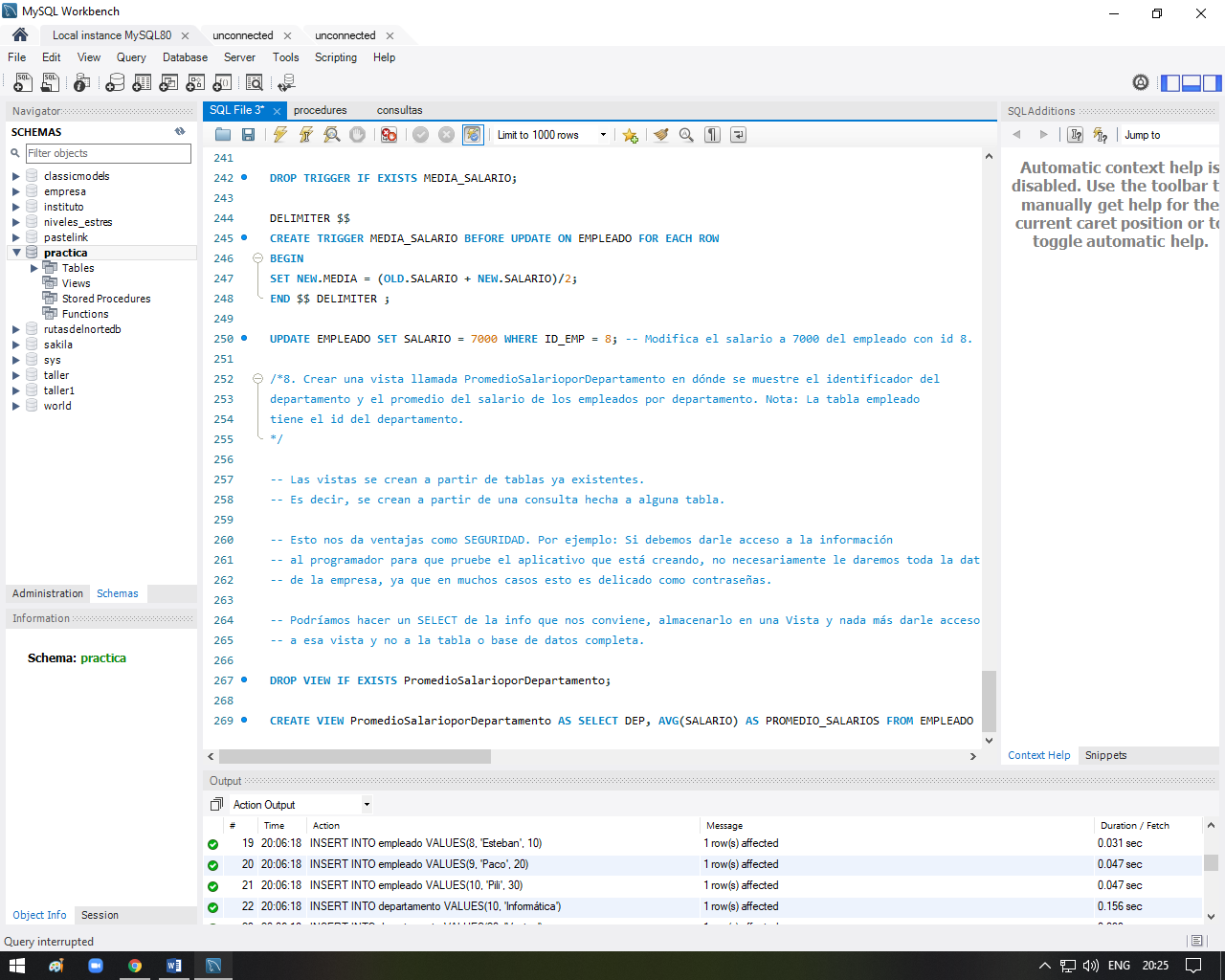
**8.1 Copie el código que utilizó para crear la vista.**

DROP VIEW IF EXISTS PromedioSalarioporDepartamento;

CREATE VIEW PromedioSalarioporDepartamento AS SELECT DEP, AVG(SALARIO) AS PROMEDIO\_SALARIOS FROM EMPLEADO GROUP BY DEP;

**8.2 Copie los resultados (captura de pantalla) luego de haber ejecutado la siguiente instrucción:**

**SELECT \* FROM *PromedioSalarioporDepartamento;***



**9. Ejecute la siguiente instrucción:**

**DELETE FROM *PromedioSalarioporDepartamento* WHERE idep=20;**

**Se puede ejecutar? Si? No? Por qué?**

No se puede ejecutar, Porque el SELECT con el cuál fue creada la VISTA

-- contenía una función de agregación AVG() y GROUP BY.