

Evaluación Continua 2177

Ejercicios:

1- Trigger: Crea un trigger que se dispare **antes** de insertar una nueva banda en la tabla **Bandas** que verifique si el país de origen ya tiene una banda de igual género musical. Si es así, cancela la inserción y lanza un error.

Trigger

```
5 • 4 DROP TRIGGER Verifica_pais_y_genero;
6   DELIMITER //
7 • 5 CREATE TRIGGER Verifica_pais_y_genero
8   BEFORE INSERT ON Bandas
9   FOR EACH ROW
10  BEGIN
11  IF
12  EXISTS (SELECT 1 FROM Bandas WHERE Pais_origen = NEW.Pais_origen AND Genero_musical = Genero_musical)
13  THEN
14    SIGNAL SQLSTATE '45000'
15    SET MESSAGE_TEXT = 'Inserción cancelada, ya existe una banda en el país del mismo género';
16  END IF;
17
18  End;//
19
20
21  DELIMITER ;
22
```

Tests

```
4 • 4 INSERT INTO Bandas (Nombre,Genero_musical,Pais_origen,Fecha_creacion)
5   Values ('Muse', 'Rock', 'Reino Unido', now());
```

4 09:35:19 SELECT ... 20 row(s) returned
5 09:35:16 INSERT IN... Error Code: 1644. Inserción cancelada, ya existe una banda en el país del mismo género

2- Trigger: Crea un trigger que se dispare **después** de actualizar la tabla **Conciertos**, y que automáticamente inserte la fecha y el concierto modificado en una tabla de auditoría llamada **Auditoria_conciertos**.

TRIGGER

```
24 • 24 -- 2- Trigger: Crea un trigger que se dispare después de actualizar la tabla Conciertos, y que au
y el concierto modificado en una tabla de auditoría llamada Auditoria_conciertos.
25
26 DROP TRIGGER Actualiza_auditoria;
27 • 27 CREATE TABLE Auditoria_conciertos (
28   ID_Auditoria INT AUTO_INCREMENT NOT NULL,
29   Concierto_ID_concierto INT NOT NULL,
30   Fecha_modificacion DATETIME NOT NULL,
31   PRIMARY KEY (ID_Auditoria),
32   FOREIGN KEY (Concierto_ID_concierto) REFERENCES Conciertos(ID_concierto)
33 );
34
35 DELIMITER //
36 • 36 CREATE TRIGGER Actualiza_auditoria
37   AFTER UPDATE ON Conciertos
38   FOR EACH ROW
39   BEGIN
40
41   INSERT INTO Auditoria_conciertos (Concierto_ID_concierto,Fecha_modificacion)
42   VALUES (new.ID_concierto, NOW());
43
44   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Auditoria Creada';
45
46   End;//
47
48 DELIMITER ;
```

TESTS

```
7 • 7 UPDATE Conciertos
8   SET ID_banda = 3
9   WHERE ID_concierto = 1; -- cambiamos de los rolling a metálica el concierto de barcelona
```

12 10:20:59 CREATE TRIGGER ACTUALIZA_AUDITORIA AFTER UPDATE ON CONCIERTOS 0 row(s) affected
13 10:20:48 UPDATE Conciertos SET ID_banda = 3 WHERE ID_concierto = 1 Error Code: 1644. Auditoria Creada

ID_Auditoria	Concierto_ID_concie...	Fecha_modificacion
2	1	2024-10-02 10:23:36

ID_concierto	Fecha	Ciudad	ID_banda	Escenario	Fecha_creacion	Fecha_modificacion
1	2024-05-01	Barcelona	3	Escenario Principal	2024-09-25 10:07:02	2024-10-02 10:23:36
2	2024-05-02	Madrid	2	Escenario Pop	2024-09-25 10:07:02	2024-09-25 10:07:02
3	2024-05-03	París	3	Escenario Metal	2024-09-25 10:07:02	2024-09-25 10:07:02
4	2024-05-04	Londres	4	Escenario Pop Rock	2024-09-25 10:07:02	2024-09-25 10:07:02

3- Transaction con Control de Errores: Crea una transacción que registre una nueva venta de merchandising. Debe insertar una nueva fila en la tabla **Ventas_merchandising**, reducir la cantidad disponible del artículo en un inventario (CREA una tabla llamada **Inventario_merchandising** con los artículos disponibles), y si el artículo no tiene suficiente stock, debe abortar la transacción y lanzar un error. Si todo es correcto, la transacción debe completarse exitosamente.

TRIGGER

```
5 DELIMITER //
6 • CREATE PROCEDURE Nueva_Venta_Actualizacion_Stock(
7     IN pArticulo VARCHAR(100),
8     IN pCantidad INT
9 )
10 BEGIN
11     -- Chivato de rollback super pro --
12     DECLARE pidArticulo INT;
13     DECLARE error_message VARCHAR(2000);
14     DECLARE error_code INT;
15     DECLARE EXIT HANDLER FOR SQLEXCEPTION
16         BEGIN
17             GET DIAGNOSTICS CONDITION 1 error_code = MYSQL_ERRNO;
18             ROLLBACK;
19             SET error_message = Concat('Me he vuelto al Rollback, He dado error: ', error_code);
20             SIGNAL SOLSTAS '45000';
21             SET MESSAGE_TEXT = error_message;
22         END;
23     START TRANSACTION;
24     -- Cambio Nombre del articulo por su Id para hacer safe update por Index de primary key
25     SET pidArticulo = (Select I.ID_Inventario from Inventario_merchandising I where I.Articulo = pArticulo);
26     -- Si hay Stock...
27     IF exists (SELECT 1 From Inventario_merchandising I where I.Articulo = pArticulo and I.Cantidad > 0 AND I.Disponible = 1)
28     THEN
29     -- Inserta datos en Ventas--
30         INSERT INTO Ventas_merchandising (ID_concierto,Articulo,Cantidad,Ingresos)
31         VALUES (?,pArticulo,pCantidad, 500);
32     -- actualiza inventario restando ventas.cantidad a inventario.cantidad
33 
```

TESTS

Mensaje

He hecho el update

ID_Inventario	Articulo	Precio	Talla	Cantidad	Disponible
1	Gorra	10	HULL	450	1
2	Camiseta Grupo	25	S	250	1
3	Camiseta Oficina	25	M	250	1

Segundo test: Sin Stock

```

I15 UPDATE Inventario_merchandising I SET I.Cantidad = (I.Cantidad - pCantidad)
I16 WHERE I.Articulo = pArticulo ;
I17 — Imprime mensaje confirmación
I18 SELECT 'He hecho el update' AS Mensaje;
I19
I20 COMMIT;
I21 ELSE
I22 — Imprime mensaje error
I23 SELECT CONCAT('no hay stock de ', pArticulo, ' suficiente') AS Mensaje;
I24 END IF;
I25 SET SQL_SAFE_UPDATES = 1;
I26 END //
I27
I28 DELIMITER ;
I29
I30 • CALL Nueva_Venta_Actualización_Stock('Gorra', 50);
I31 — TEST—
I32 • SELECT * FROM Ventas_merchandising;
I33

```

136	•	UPDATE	Inventory_merchandising	SET Disponible = 0	where ID_Inventory = 1;		
137	100%	◆	27:130				
Result Grid Filter Rows: <input type="text"/> Search							
Edit: Export/Import:							
ID_Inventory	Articulo	Precio	Talla	Cantidad	Disponible		
1	Gorra	10	NULL	450	0		
2	Camiseta Grupo	25	S	250	1		

Mensaje

no hay stock de Gorra suficiente

4- Informe con Procedure: Crea un procedimiento almacenado que genere un informe con los ingresos totales por merchandising para cada concierto, incluyendo un condicional que filtre solo los conciertos que han generado más de 5000 euros en ingresos. Utiliza un WHILE loop para recorrer los conciertos y sumar los ingresos.

CODIGO

```

DROP PROCEDURE Informe_Ingresos_Merchan;
DELIMITER //
CREATE PROCEDURE Informe_Ingresos_Merchan()
BEGIN
-- Variables
    DECLARE fin int default 0; -- > n.m: esta es para el cursor que si no está protesta.
    DECLARE pidConcierto INT;
    DECLARE pIngresos decimal (10,2) default 0;
    DECLARE error_message VARCHAR(2000);
    DECLARE error_code INT;

-- Mete un cursor... Porlo visto tiene que ir antes del Handler aunque sea el del rollback,
DECLARE cur_conciertos CURSOR FOR
    SELECT VM.ID_concierto, SUM(VM.Ingresos) as total_ingeros
    FROM Ventas_merchandising VM
    GROUP BY VM.ID_concierto
    HAVING total_ingeros > 5000
    ORDER BY VM.ID_concierto;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = 1;

-- Super combo rollback
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
    GET DIAGNOSTICS CONDITION 1 error_code = MYSQL_ERRNO;
    ROLLBACK;
    SET error_message = Concat('Me he vuelto al Rollback, He dado error: ', error_code);
    SIGNAL SQLSTATE '45000';
    SET MESSAGE_TEXT = error_message;
END;

-- Crear una tabla temporal para meter los resultados en algun sitio... y poder usarlos luegos ( meme de OBVIO)
CREATE TEMPORARY TABLE IF NOT EXISTS Informe_Ingresos (
    ID_concierto INT,
    Ingresos_Totales DECIMAL(10,2)
);
OPEN cur_conciertos;
-- THE WHILE: Por cada entrada suma los ingresos en merchan
read_loop: LOOP
    FETCH cur_conciertos INTO pidConcierto, pIngresos;
    IF fin THEN
        LEAVE read_loop; -- Esto tb es importante, si no acabas como con las entradas... con 13459282364.
    END IF;

    -- Insertar en la tabla temporal
    INSERT INTO Informe_Ingresos (ID_concierto, Ingresos_Totales)
    VALUES (pidConcierto, pIngresos);
END LOOP;
CLOSE cur_conciertos;
-- Printa el Informe
SELECT C.Ciudad, I.Ingresos_Totales
FROM Informe_Ingresos I
inner join Conciertos C ON C.ID_Concierto = I.ID_Concierto
ORDER BY Ingresos_Totales DESC;

-- Matar la tabla temporal
DROP TEMPORARY TABLE IF EXISTS Informe_Ingresos;
END; //

```

217 ● **DROP PROCEDURE** Informe_Ingresos_Merchan;
-- TESTS
219 ● **CALL** Informe_Ingresos_Merchan();
220
221

75% 34:173 | 2 errors found

Result Grid Filter Rows: Search Export:

Ciudad	Ingresos_Totales
Nueva York	21000.00
Tokio	20000.00
Londres	16000.00
México DF	16000.00
Sidney	14000.00
Barcelona	12000.00
Berlín	12000.00
París	10000.00
Bogotá	10000.00
Madrid	8500.00

Result 51

Action Output

Time	Action	Response
2... 18:26:57	DROP PROCEDURE...	0 row(s) affected
2... 18:26:59	CREATE PROCEDU...	0 row(s) affected
2... 18:27:02	CALL Informe_Ingr...	10 row(s) returned

5- Procedure con Transaction y Control de Errores: Crea un procedimiento almacenado que inserte una nueva entrada en la tabla [Entradas](#). El procedimiento debe comenzar con una transacción y utilizar un [WHILE](#) loop para verificar si el asistente ya tiene entradas para otros conciertos de la misma banda en fechas diferentes. Si existe alguna entrada duplicada, debe abortar la transacción y lanzar un error.

CODE

```

14 • DROP PROCEDURE Entradas_Duplicadas;
DELIMITER //
CREATE PROCEDURE Entradas_Duplicadas(
    IN pidConcierto INT,
    IN pidAsistente INT
)
BEGIN
    DECLARE pCompra TinyINT DEFAULT false;
    DECLARE pidBanda INT;
    DECLARE pFecha datetime;

    DECLARE error_message VARCHAR(2000);
    DECLARE error_code INT;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        BEGIN
        --
        START TRANSACTION;
        --
        -- Seleccionar y guardar banda y fecha (no me ha dejado hacerlo en una sola... \_(`)_/)
        --
        SELECT ID_banda INTO pidBanda
        FROM Conciertos AS C
        WHERE C.ID_concierto = pidConcierto;
        SELECT Fecha INTO pFecha
        FROM Conciertos AS C
        WHERE C.ID_concierto = pidConcierto;

        -- El WHILE DICHOSO cuenta duplicados, si hay duplicados frena
        WHILE NOT pCompra DO
            IF EXISTS(
                SELECT 1
                FROM Entradas E
                WHERE E.ID_asistente = pidAsistente AND E.ID_concierto = pidConcierto )
            THEN
                SELECT 'Ya Tiene una entrada, no puede comprar'as Mensaje; -- Flag 1
                SET pCompra = true;
            ELSEIF EXISTS(
                SELECT 1
                FROM Entradas E
                INNER JOIN Conciertos C ON C.ID_concierto = E.ID_concierto
                WHERE E.ID_asistente = pidAsistente AND C.ID_banda = pidBanda AND C.Fecha = pFecha AND E.ID_concierto <> pidConcierto)
            THEN
                SET pCompra = true;

                SELECT 'Ya Tiene una entrada para la misma banda y fecha, no puede comprar'as Mensaje;
            ELSE
                INSERT INTO Entradas(ID_asistente, ID_concierto, Precio, Fecha_creacion)
                VALUES (pidAsistente, pidConcierto, 75, NOW());
                SELECT 'Enhorabuena, Ha comprador una entrada' AS Mensaje; -- Flag 2
                SET pCompra = true;
            END IF;
        END WHILE ;
        COMMIT;
    END;
DELIMITER ;

```

283 • -- Tests --
284 CALL Entradas_Duplicadas(2,6);
285 • SELECT * FROM Entradas;
286

75% 25:284 | 2 errors found

Result Grid Filter Rows: Search Edit: Export/Import

ID_entrada	ID_asistente	ID_concierto	Precio	Fecha_creacion	Fecha_modificacion
1175546	6	2	75.00	2024-10-02 17:34:29	2024-10-02 17:34:29

Entradas 44

Mensaje

Usted ya tiene una entrada, no puede comprar...

Test 2

283 • -- Tests --
284 CALL Entradas_Duplicadas(2,7);
285 • SELECT * FROM Entradas;
286
287

75% 20:285 | 2 errors found

Result Grid Filter Rows: Search Edit: Export/Import

ID_entrada	ID_asistente	ID_concierto	Precio	Fecha_creacion	Fecha_modificacion
1175547	7	2	75.00	2024-10-02 17:38:02	2024-10-02 17:38:02

Mensaje

Enhorabuena, Ha comprador una entrada

6- Procedure con Control de Errores y Loop: Crea un procedimiento almacenado que inserte una nueva venta de merchandising en la tabla `Ventas_merchandising`. Si el concierto asociado no existe, el procedimiento debe lanzar un error y no realizar la inserción. Utiliza un `WHILE` loop para verificar si el artículo ya ha sido vendido previamente en ese concierto, y si es así, aumenta la cantidad en lugar de insertar una nueva fila.

```

I01 •  DROP PROCEDURE NUEVA_VENTA_MERCHAN;
I02 •  DELIMITER //
I03 •  CREATE PROCEDURE NUEVA_VENTA_MERCHAN (
I04    IN pConcierto INT,
I05    IN pArticulo VARCHAR (100),
I06    IN pCantidadVendida INT,
I07    IN pIngresos INT
I08
I09  )
I10  BEGIN
I11  -- VARIAS VARIABLES
I12
I13      DECLARE pVenta BOOLEAN DEFAULT false; -- testigo para el while
I14      DECLARE pCantidad_actual INT;      -- la cantidad que ya hay en la tabla
I15      DECLARE pArticuloVendido INT;     -- cuantos articulos se han vendido, para el if
I16      DECLARE pIngresos_actual INT;
I17      DECLARE error_message VARCHAR(2000);
I18      DECLARE error_code INT;
I19
I20      DECLARE EXIT HANDLER FOR SQLEXCEPTION -- El super Rollback
I21          BEGIN
I22          START TRANSACTION;
I23
I24          -- Comprueba que existe el concierto
I25          IF EXISTS ( Select 1 from Conciertos C WHERE C.ID_concierto = pConcierto ) -- IF 1
I26          THEN
I27              WHILE NOT pVenta DO
I28
I29                  -- Chequea que si existe el articulo y me guarda los valores de cuantos articulos y la cantidad en variables
I30                  SELECT COUNT(*)
I31                  INTO pArticuloVendido
I32                  FROM Ventas_merchandising VM
I33                  WHERE VM.ID_concierto = pConcierto AND VM.Articulo = pArticulo;
I34
I35                  SELECT Cantidad
I36                  INTO pCantidad_actual
I37                  FROM Ventas_merchandising VM
I38                  WHERE VM.ID_concierto = pConcierto AND VM.Articulo = pArticulo;
I39
I40                  SELECT Ingresos
I41                  INTO pIngresos_actual
I42                  FROM Ventas_merchandising VM
I43                  WHERE VM.ID_concierto = pConcierto AND VM.Articulo = pArticulo;
I44
I45
I46
I47
I48
I49      -- Si existe la venta actualiza valor cantidad
I50      IF pArticuloVendido > 0 THEN -- IF2
I51          -- Actualizar la venta existente
I52          UPDATE Ventas_merchandising VM
I53          SET Cantidad = pCantidad_actual + pCantidadVendida,
I54              Ingresos = pIngresos + pIngresos_actual
I55          WHERE VM.ID_concierto = pConcierto AND VM.Articulo = pArticulo;
I56
I57          SET pVenta = TRUE;
I58      ELSE -- Insertar la fila con la venta nueva y los valores de los IN
I59      INSERT INTO Ventas_merchandising (ID_concierto,Articulo,Cantidad,Ingresos)
I60      VALUES(pConcierto,pArticulo,pCantidadVendida,pIngresos);
I61      END IF;
I62      END WHILE;
I63
I64      -- SI no se cumple lo anterior , lanza error
I65      ELSE
I66          SELECT 'El concierto no existe'as Mensaje;
I67      END IF;
I68
I69      COMMIT;
I70      SELECT Concat('Has registrado correctamente la venta de ', pArticulo,' con la cantidad de ',pCantidadVendida) as Mensaje;
I71
I72  END //
I73  DELIMITER ;

```

5 / 4

375 • CALL NUEVA_VENTA_MERCHAN (1, 'Gorra', 50, 500);
376

75% 29:375

Result Grid Filter Rows: Search Export:

Mensaje					
Has registrado correctamente la venta de Gorra...					

374

375 • CALL NUEVA_VENTA_MERCHAN (1, 'Gorra', 50, 500);
376 • SELECT * FROM Ventas_merchandising;

75% 28:375

Result Grid Filter Rows: Search Edit: Export/Import:

ID_venta	ID_concierto	Articulo	Cantidad	Ingresos	Fecha_creacion	Fecha_modificacion
32	2	Gorra	50	500.00	2024-10-02 11:41:41	2024-10-02 11:41:41
33	1	Gorra	150	1500.00	2024-10-03 10:21:29	2024-10-03 10:22:00

376

377 • CALL NUEVA_VENTA_MERCHAN (36, 'Gorra', 50, 500);

5% 20:377

Result Grid Filter Rows: Search Export:

Mensaje					
El concierto no existe					

7- Formulario con Procedure y Loop: Crea un procedimiento almacenado que reciba como entrada el nombre de una banda, el género musical, el país de origen, y verifique si ya existe una banda en ese país con el mismo género. Si no existe, inserta la nueva banda. Utiliza un `WHILE` loop para verificar que los nombres de las bandas similares no coinciden con la nueva banda antes de hacer la inserción.

Código

```

CREATE PROCEDURE NUEVA_BANDA SIN REPETIR (
    IN pNombreBanda VARCHAR(100),
    IN pGenero varchar(100),
    IN pPais Varchar(100))

BEGIN
    -- Variables
    DECLARE pidBanda INT;
    DECLARE banda_existe BOOLEAN DEFAULT FALSE;
    DECLARE nombre_similar VARCHAR(100);
    DECLARE PalWhile BOOLEAN DEFAULT FALSE;
    DECLARE error_message VARCHAR(2000);
    DECLARE error_code INT;
    -- Cursor para comparar los nombres uno a uno
    DECLARE curBanda CURSOR FOR
    SELECT Nombre
    FROM Bandas B
    WHERE B.Pais_origen = pPais AND B.Genero_musical = pGenero;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET PalWhile = TRUE;

    /*DECLARE EXIT HANDLER FOR SQLEXCEPTION -- El super Rollback
    BEGIN
        GET DIAGNOSTICS CONDITION 1 error_code = MYSQL_ERRNO;
        ROLLBACK;
        SET error_message = Concat('Me he vuelto al Rollback, He dado error: ', error_code);
        SIGNAL SQLSTATE '45000';
        SET MESSAGE_TEXT = error_message;
    END;*/



    START TRANSACTION;

    -- Modifica valores en texto a ID--
    SET pidBanda = (Select B.ID_Banda from Bandas B where B.Nombre = pNombreBanda);
    -- Verifica si existe una banda de igual género en el país -- B.ID_banda = pidBanda or IF Exists (Select 1 from Bandas B WHERE (B.Pais_origen = pPais AND B.Genero_musical = pGenero)) -- IF1
    THEN
        -- Si existe comprobamos el nombre si suena parecido tb

        OPEN curBanda;
        read_loop: WHILE NOT PalWhile DO
            FETCH curBanda INTO nombre_similar;
            IF NOT PalWhile THEN -- revisa que los nombres no suenen parecido
                IF SOUNDEX(nombre_similar) = SOUNDEX(pNombreBanda) -- Pues resulta que existe una función que compara fonéticamente los nombres
                THEN
                    SET banda_existe = TRUE; -- si suenan parecidos la toma como que existe la banda y la manda al IF3
                    LEAVE read_loop;
                END IF;
            END IF;
        END WHILE;
        CLOSE curBanda;

        IF banda_existe -- si definitivamente existe Error no insert -- IF3
        THEN
            SELECT CONCAT('Lo siento Prenda!, ya existe una banda similar (' , nombre_similar, ') en ', pPais, ' con el género ', pGenero) AS Mensaje;
        ELSE
            -- Si no existe, Inserta la banda nueva
            INSERT INTO Bandas (Nombre, Genero_musical, Pais_origen, Fecha_creacion)
            VALUES (pNombreBanda, pGenero, pPais, now());
            SELECT Concat(pNombreBanda, ' ha sido registrada, Bienvenido al Tour') as Mensaje; -- mensaje de confirmación
        END IF;
    ELSE
        -- Si no Existe, la crea en el registro, ale pa dentro del tour
        INSERT INTO Bandas (Nombre, Genero_musical, Pais_origen, Fecha_creacion)
        VALUES (pNombreBanda, pGenero, pPais, now());
        SELECT Concat(pNombreBanda, ' ha sido registrada, Bienvenido al Tour') as Mensaje; -- mensaje de confirmación
    END IF;
    Commit;
END // DELIMITER ;

```

Tests

438 • CALL NUEVA_BANDA SIN REPETIR ('Los inhumanos', 'Rock', 'España')

439

440

441

442 — 8- Function con Loop: Crea una función que, dado el ID de un concierto, devuelva el total de ingresos por ventas de merchandising para ese concierto. Utiliza un WHILE loop para sumar cada una de las ventas de merchandising asociadas al concierto. La función debe realizar cálculos matemáticos para sumar los ingresos y retornar el valor, manejando los posibles casos en los que no haya ventas.

Result Grid Filter Rows: Search Export:

Mensaje

JLo siento Prenda!, ya existe una banda similar (Los inhumanos) en España con el género pop

438 • CALL NUEVA_BANDA SIN REPETIR ('Los peruanos', 'pop', 'España')

439

75% 51438 |

Action Output

	Time	Action	Response
3...	11:32:01	CALL NUEVA_BANDA SIN REPETIR ('Los peruanos', 'pop', 'España')	1 row(s) returned
3...	11:33:15	Select * from Bandas	21 row(s) returned
3...	11:34:49	CALL NUEVA_BANDA SIN REPETIR ('Los peruanos', 'pop', 'España')	Error Code: 1644. Inserción cancelada, ya existe una banda en el país del mismo género

ID_banda	Nombre	Genero_musical	Pais_origen	F
1	The Rolling Stones	Rock	Reino Unido	21
2	Beyoncé	Pop	Estados Unidos	21
3	Metallica	Metal	Estados Unidos	21
4	Coldplay	Pop Rock	Reino Unido	21
5	Daft Punk	Electrónica	Francia	21
6	Shakira	Pop	Colombia	21
7	Arctic Monkeys	Indie Rock	Reino Unido	21
8	AC/DC	Rock	Australia	21
9	Muse	Rock Alternativo	Reino Unido	21
10	Daddy Yankee	Reggaeton	Puerto Rico	21
21	Los inhumanos	pop	España	21
NULL	NULL	NULL	NULL	1

457 • CALL NUEVA_BANDA SIN REPETIR (' Manolos', 'Sevillanas', 'España');

5% 24:457 | 1 error found

Result Grid Filter Rows: Search Export:

Mensaje

Manolos ha sido registrada, Bienvenido al Tour

8- Function con Loop: Crea una función que, dado el ID de un concierto, devuelva el total de ingresos por ventas de merchandising para ese concierto. Utiliza un WHILE loop para sumar cada una de las ventas de merchandising asociadas al concierto. La función debe realizar cálculos matemáticos para sumar los ingresos y retornar el valor, manejando los posibles casos en los que no haya ventas.

Codigo

```
DROP FUNCTION Ingresos_Concierto;
DELIMITER //
CREATE FUNCTION Ingresos_Concierto (pIdConcierto INT) RETURNS VARCHAR (100)
READS SQL DATA
BEGIN
-- Varias Variables
DECLARE Total_Ventas DECIMAL (10,2) DEFAULT 0;
DECLARE Ingresos_actuales DECIMAL (10,2);
DECLARE freno BOOLEAN DEFAULT FALSE;
DECLARE devolucion varchar(100);

-- Cursor para recorrer Ventas_Merchan
DECLARE curIngresos CURSOR FOR
    SELECT Ingresos
    FROM Ventas_merchandising VM
    WHERE VM.ID_concierto = pIdConcierto;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET freno = TRUE;

    OPEN curIngresos;
    WHILE NOT freno DO
        FETCH curIngresos INTO Ingresos_actuales;
        IF NOT freno
        THEN
            set Total_Ventas = Total_Ventas + Ingresos_Actuales;
        END IF;
    END WHILE;
    CLOSE curIngresos;

    IF Total_Ventas > 0 THEN
        set devolucion = CONCAT('Total ingresos: ', Total_Ventas);
        RETURN devolucion;
    ELSE
        RETURN 'No hay ventas registradas para este concierto';
    END IF;
END //

DELIMITER ;
Select Ingresos_Concierto(2) as Ingresos ;
```

TEST

The screenshot shows the MySQL Workbench interface with two panes. The left pane displays the SQL code for the stored procedure. The right pane shows the results of the execution.

Execution Results:

Ingresos
Total ingresos: 4000.00

Logs:

Time	Action	Response
13:57:38	CREATE FUNCTION Ingresos_Concierto (pIdConcierto INT) RETURNS VARCHAR (100) ...	0 row(s) affected
13:57:41	Select Ingresos_Concierto(2) as Ingresos LIMIT 0, 1000	1 row(s) returned

Result Grid:

Ingresos
No hay ventas registradas para este concierto

9- Formulario con Procedure y Loop: Crea un procedimiento almacenado que permita la creación de una nueva entrada en la tabla Entradas, comprobando que la edad del asistente sea mayor de 18 años. Si el asistente es menor de 18 años, lanza un error. Utiliza un WHILE loop para recorrer la lista de asistentes menores de edad que intenten registrarse, y si se encuentran, se deben rechazar todas las entradas.

Código

```
CREATE PROCEDURE Crear_Entrada_Adultos(
    IN p_id_concierto INT,
    IN p_id_asistente INT,
    IN p_precio DECIMAL(10,2)
)
BEGIN
    DECLARE pedad INT;
    DECLARE menores_encontrados BOOLEAN DEFAULT FALSE;
    DECLARE asistente_actual INT;
    DECLARE freno BOOLEAN DEFAULT FALSE;

    DECLARE curAsistentes CURSOR FOR
        SELECT ID_asistente
        FROM Asistentes
        WHERE Edad < 18;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET freno = TRUE;

    -- Verificar la edad del asistente
    SELECT Edad INTO pedad
    FROM Asistentes
    WHERE ID_asistente = p_id_asistente;

    IF pedad < 18 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: El asistente es menor de 18 años.';
    ELSE
        -- Verificar si hay menores intentando registrarse
        OPEN curAsistentes;
        check_menores: WHILE NOT freno DO
            FETCH curAsistentes INTO asistente_actual;
            IF NOT freno THEN
                SET menores_encontrados = TRUE;
            END IF;
        END WHILE;

        CLOSE curAsistentes;

        IF menores_encontrados THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Error: Se encontraron menores de edad intentando Colarse. Todas las entradas han sido rechazadas.';
        ELSE
            -- Insertar la nueva entrada
            INSERT INTO Entradas (ID_concierto, ID_asistente, Precio, Fecha_creacion)
            VALUES (p_id_concierto, p_id_asistente, p_precio, NOW());

            SELECT 'Bien!! Entrada comprada Difruta del concierto' AS Mensaje;
        END IF;
    END IF;
END // /
```

DELIMITER :

TEST

568 ● CALL Crear_Entrada_Adultos (1, 2, 75);
75% 29:568

Result Grid Filter Rows: Search Export

Mensaje

Bien!! Entrada comprada Difruta del concierto

TEST menores

565 ● CALL Crear_Entrada_Adultos (21, 2, 75);
75% 19:565

Action Output

Time	A	Response
5... 13:16:04	...	1292: Incorrect datetime value: '' for column 'Fecha_creacion' at row 1
5... 13:17:39	...	Error Code: 1644. Error: Se encontraron menores de edad intentando Colarse. Todas las entradas han sido rechazadas.

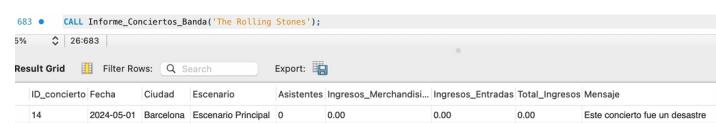
ID	Nombre	Edad	Nacionalidad	FECTIVO USDT	FECTIVO USDT
20	Anna Müller	30	Alemania	2024-10-02 09:14:03	2024-10-02 09:14:03
21	Maria José	17	noruega	2024-10-03 13:15:12	2024-10-03 13:15:12
22	asdsdd	14	NULL	2024-10-03 13:16:42	2024-10-03 13:16:42
NULL	NULL	NULL	NULL	NULL	NULL

10- Informe con Procedure y Loop: Crea un procedimiento que genere un informe detallado de todos los conciertos de una banda específica, incluyendo la cantidad de asistentes y el total de ingresos generados por merchandising y entradas. Utiliza un `WHILE` loop para recorrer cada concierto de la banda, sumar los ingresos, y verificar que haya asistentes para cada concierto. Si un concierto no tiene asistentes o ventas, genera un mensaje de advertencia.

Codigo

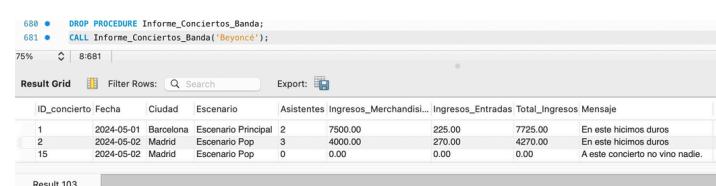
```
--  
582 • CREATE PROCEDURE Informe_Conciertos_Banda(  
583     IN pNombre_banda VARCHAR(100))  
584 BEGIN  
585     -- Variables  
586     DECLARE freno INT DEFAULT FALSE;  
587     DECLARE pID_concierto INT;  
588     DECLARE pFecha DATE;  
589     DECLARE pCiudad VARCHAR(100);  
590     DECLARE pEscenario VARCHAR(100);  
591     DECLARE pAsistentes INT;  
592     DECLARE pIngresos_merc DECIMAL(10,2);  
593     DECLARE pIngresos_entradas DECIMAL(10,2);  
594     DECLARE pTotal_ingenros DECIMAL(10,2);  
595     DECLARE pMensaje VARCHAR(1000);  
596     DECLARE pID_banda INT;  
597  
598     -- Cursor para recorrer los conciertos de la banda  
599     DECLARE cur_conciertos CURSOR FOR  
600         SELECT C.ID_concierto, C.Fecha, C.Ciudad, C.Escenario  
601             FROM Conciertos C  
602             JOIN Bandas B ON C.ID_banda = B.ID_banda  
603             WHERE B.Nombre = pNombre_banda  
604             ORDER BY C.Fecha;  
605  
606     DECLARE CONTINUE HANDLER FOR NOT FOUND SET freno = TRUE;  
607  
608     -- Obtener el ID de la banda desde el nombre  
609     SELECT ID_banda INTO pID_banda  
610     FROM Bandas B  
611     WHERE B.Nombre = pNombre_banda;  
612  
  
IF pID_banda IS NULL THEN  
    SELECT 'La banda especificada no existe' AS Mensaje;  
ELSE  
    -- Crear tabla temporal para almacenar resultados  
    CREATE TEMPORARY TABLE IF NOT EXISTS Informe_Temporal (  
        ID_concierto INT,  
        Fecha DATE,  
        Ciudad VARCHAR(100),  
        Escenario VARCHAR(100),  
        Asistentes INT,  
        Ingresos_Merchandising DECIMAL(10,2),  
        Ingresos_Entradas DECIMAL(10,2),  
        Total_Ingresos DECIMAL(10,2),  
        Mensaje VARCHAR(1000)  
    );  
  
    OPEN cur_conciertos;  
  
    WHILE NOT freno DO  
        FETCH cur_conciertos INTO pID_concierto, pFecha, pCiudad, pEscenario;  
        IF NOT freno THEN  
            -- Contar asistentes  
            SELECT COUNT(DISTINCT E.ID_asistente) INTO pAsistentes  
            FROM Entradas E  
            WHERE E.ID_concierto = pID_concierto;  
  
            -- Calcular ingresos por merchandising  
            SELECT IFNULL(SUM(VM.Ingresos), 0) INTO pIngresos_merc  
            FROM Ventas_merchandising VM  
            WHERE VM.ID_concierto = pID_concierto;  
  
            -- Calcular ingresos por entradas  
            SELECT IFNULL(SUM(E.Precio), 0) INTO pIngresos_entradas  
            FROM Entradas E  
            WHERE E.ID_concierto = pID_concierto;  
  
            SET pTotal_ingenros = pIngresos_merc + pIngresos_entradas;  
        END IF;  
    END WHILE;  
END IF;
```

Tests KO



ID_concierto	Fecha	Ciudad	Escenario	Asistentes	Ingresos_Merchandising	Ingresos_Entradas	Total_Ingresos	Mensaje
14	2024-05-01	Barcelona	Escenario Principal	0	0.00	0.00	0.00	Este concierto fue un desastre

Test OK



ID_concierto	Fecha	Ciudad	Escenario	Asistentes	Ingresos_Merchandising	Ingresos_Entradas	Total_Ingresos	Mensaje
1	2024-05-01	Barcelona	Escenario Principal	2	7500.00	225.00	7725.00	En este himnos duros
2	2024-05-02	Madrid	Escenario Pop	3	4000.00	270.00	4270.00	En este himnos duros
15	2024-05-02	Madrid	Escenario Pop	0	0.00	0.00	0.00	A este concierto no vino nadie

```
    SET pTotal_ingenros = pIngresos_merc + pIngresos_entradas;  
END IF;
```

-- Chivatos de Errores
TF lnTotal_inraene = @1 AND lnhectontac = @2 THEN