CASO PRÁCTICO 1

PROGRAMACIÓN EN BASE DE DATOS

Contexto

Como desarrollador, en tu empresa has realizado distintas bases de datos para poder optimizar y mejorar el trabajo en la compañía. Hace unos días, tus jefes te han pedido que elabores, para una de esas bases de datos, una serie de procedimientos almacenados y funciones, así como el código necesario para ejecutarlos.

```
1 •  CREATE DATABASE ud6;
2    USE ud6;
3
```

Cuestiones a resolver

Son las siguientes:

1. Crea un procedimiento que muestre 'Hola mundo' por pantalla.

```
DELIMITER //
CREATE PROCEDURE Saludo_HolaMundo()
BEGIN
   SELECT 'Hola mundo';
END;
//
DELIMITER ;
CALL Saludo_HolaMundo();
```

```
2 •  CREATE PROCEDURE Saludo_HolaMundo()
3    BEGIN
4        SELECT 'Hola mundo';
5    END;
6    //
7    DELIMITER ;
8 •  CALL Saludo_HolaMundo();
```

Result Grid | Filter Rows: | Export:

Hola mundo
Hola mundo

2. **Crea un procedimiento que pasándole una variable numérica cómo parámetro, muestre un mensaje diciendo si esa variable es mayor de 10 o no. Deberá mostrar un mensaje similar a este: "El número x es mayor que 10" o "El número x es menor que 10", siendo x el número pasado como parámetro.**

```
DELIMITER //
CREATE PROCEDURE Comparar_Numero(IN numero INT)
BEGIN
  IF numero > 10 THEN
     SELECT CONCAT('El número ', numero, ' es mayor que 10') AS Resultado;
  ELSE
     SELECT CONCAT('El número ', numero, ' es menor o igual que 10') AS Resultado;
  END IF;
END;
//
DELIMITER ;

CALL Comparar_Numero(5);
CALL Comparar_Numero(15);
```

```
1   DELIMITER //
2   CREATE PROCEDURE Comparar_Numero(IN numero INT)
3   BEGIN
4       IF numero > 10 THEN
5           SELECT CONCAT('El número ', numero, ' es mayor que 10') AS Resultado;
6       ELSE
7           SELECT CONCAT('El número ', numero, ' es menor o igual que 10') AS Resultado;
8       END IF;
9   END;
10  //
11  DELIMITER ;
12  CALL Comparar_Numero(5);
13
14
15
```

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

Resultado
El número 5 es menor o igual que 10

```
1   DELIMITER //
2   CREATE PROCEDURE Comparar_Numero(IN numero INT)
3   BEGIN
4       IF numero > 10 THEN
5           SELECT CONCAT('El número ', numero, ' es mayor que 10') AS Resultado;
6       ELSE
7           SELECT CONCAT('El número ', numero, ' es menor o igual que 10') AS Resultado;
8       END IF;
9   END;
10  //
11  DELIMITER ;
12  CALL Comparar_Numero(15);
13
14
15
```

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

Resultado
El número 15 es mayor que 10

3. Crea un procedimiento que pasándole una variable numérica devuelva su valor multiplicado por 2 en un parámetro de salida y muestre el valor por consola

```
DELIMITER //

CREATE PROCEDURE Multiplica_Por2(
  IN numero INT,
  OUT resultado INT
)
BEGIN
  SET resultado = numero * 2;
END;
//

DELIMITER ;

CALL Multiplica_Por2(7, @res);
SELECT @res;
```

```
1   DELIMITER //
2
3   CREATE PROCEDURE Multiplica_Por2(
4     IN numero INT,
5     OUT resultado INT
6   )
7   BEGIN
8     SET resultado = numero * 2;
9   END;
10  //
11
12  DELIMITER ;
13  CALL Multiplica_Por2(7, @res);
14  SELECT @res;
15
```

Result Grid | Filter Rows:

@res
14

4. **Crea un procedimiento que muestre los números del 1 al 100 con un WHILE.**

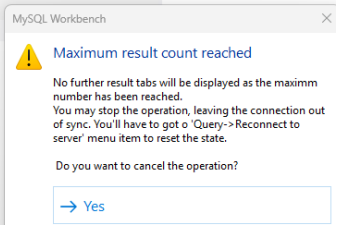
```
DELIMITER //

CREATE PROCEDURE Mostrar_1_100()
BEGIN
 DECLARE contador INT DEFAULT 1;

 WHILE contador <= 100 DO
  SELECT contador;
  SET contador = contador + 1;
 END WHILE;
END;
//

DELIMITER ;
CALL Mostrar_1_100();
```

```
1   DELIMITER //
2
3   CREATE PROCEDURE Mostrar_1_100()
4   BEGIN
5     DECLARE contador INT DEFAULT 1;
6
7     WHILE contador <= 100 DO
8       SELECT contador;
9       SET contador = contador + 1;
10    END WHILE;
11  END;
12  //
13
14  DELIMITER ;
15  CALL Mostrar_1_100();
16
```

Automati

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

contador
50



5. Crea un procedimiento para mostrar los números pares del 1 al 100 con un FOR

```
DELIMITER //

CREATE PROCEDURE Mostrar_Pares()
BEGIN
  DECLARE i INT DEFAULT 2;

  WHILE i <= 100 DO
    SELECT i;
    SET i = i + 2;
  END WHILE;
END;
//

DELIMITER ;

CALL Mostrar_Pares();
```

Nota: En MySQL no existe un bucle FOR como tal, por lo que se ha simulado su comportamiento mediante un bucle WHILE, incrementando el contador en pasos de 2 para mostrar únicamente los números pares del 1 al 100.

```
1   DELIMITER //
2
3   CREATE PROCEDURE Mostrar_Pares()
4   BEGIN
5     DECLARE i INT DEFAULT 2;
6
7     WHILE i <= 100 DO
8       SELECT i;
9       SET i = i + 2;
10    END WHILE;
11  END;
12  //
13
14  DELIMITER ;
15
16  CALL Mostrar_Pares();
17
```

Automatic

Tu Cer

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

i
100

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

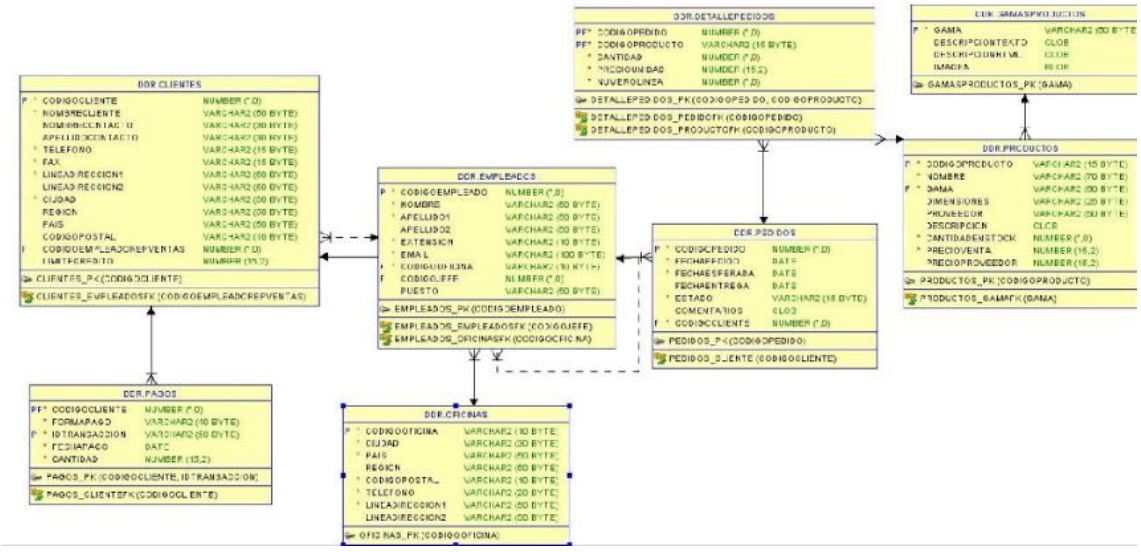
Result 96 | Result 97 | Result 98 | Result 99 | Result 100 | Result 101 | Result 102 | Result 103 | Result 104 | Result 105 | Result 106 | Result 1(0) | Read Only | Context Help | Snippets

Output

Action Output

#	Time	Action	Message
124	14:09:00	CREATE PROCEDURE Mostrar_Pares() BEGIN DECLARE i INT DEFAULT 2; WHILE i <= 100 DO SELECT i; SET i = i + 2; END WHILE; END;	0 row(s) affected
125	14:09:12	CALL Mostrar_Pares()	1 row(s) returned
126	14:09:12	CALL Mostrar_Pares()	1 row(s) returned
127	14:09:12	CALL Mostrar_Pares()	1 row(s) returned
128	14:09:12	CALL Mostrar_Pares()	1 row(s) returned
129	14:09:12	CALL Mostrar_Pares()	1 row(s) returned

Dado el siguiente esquema de base de datos:



6. **Crea un TRIGGER para actualizar la tabla T_PRODUCTOS. Cuando se haga un INSERT en la tabla T_DETALLEPEDIDOS, se debe actualizar el campo CANTIDADENSTOCK de la tabla T_ PRODUCTOS. Por ejemplo, si hago el siguiente INSERT:**


```
Insert into T_DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD, PRECIOUNIDAD, NUMEROLINEA) values (2,'FR-67',5,70,3);
```

al producto con código FR-67 se le debe restar 5 al campo CANTIDADENSTOCK de la tabla T_PRODUCTOS. Sin embargo, si lo que hago es un borrado de la tabla T_DETALLEPEDIDOS, lo que debe hacer el trigger es sumar esa cantidad. Por ejemplo: después de la inserción anterior, tenemos lo siguiente en la tabla T_DETALLEPEDIDOS para el producto FR-67.

La última línea es la que acabamos de insertar.

CODIGOPEDIDO	CODIGOPRODUCTO	CANTIDAD	PRECIOUNIDAD	NUMEROLINEA
1	FR-67	10	70	3
74	FR-67	15	70	1
2	FR-67	5	70	3

Si la borramos, lo que debe hacer el trigger es volver a sumar esa cantidad (5), en el campo CANTIDADENSTOCK de la tabla T_PRODUCTOS.

```
CREATE TABLE T_PRODUCTOS (
 CODIGOPRODUCTO VARCHAR(10) PRIMARY KEY,
 NOMBRE VARCHAR(50),
 CANTIDADENSTOCK INT
);

CREATE TABLE T_DETALLEPEDIDOS (
 CODIGOPEDIDO INT,
 CODIGOPRODUCTO VARCHAR(10),
 CANTIDAD INT,
 PRECIOUNIDAD DECIMAL(10,2),
 NUMEROLINEA INT,
```

```
FOREIGN KEY (CODIGOPRODUCTO) REFERENCES T_PRODUCTOS(CODIGOPRODUCTO)
);

INSERT INTO T_PRODUCTOS (CODIGOPRODUCTO, NOMBRE, CANTIDADENSTOCK)
VALUES ('FR-67', 'Producto Prueba', 100);
```

```sql
CREATE TABLE T_PRODUCTOS (
  CODIGOPRODUCTO VARCHAR(10) PRIMARY KEY,
  NOMBRE VARCHAR(50),
  CANTIDADENSTOCK INT
);

CREATE TABLE T_DETALLEPEDIDOS (
  CODIGOPEDIDO INT,
  CODIGOPRODUCTO VARCHAR(10),
  CANTIDAD INT,
  PRECIOUNIDAD DECIMAL(10,2),
  NUMEROLINEA INT,
  FOREIGN KEY (CODIGOPRODUCTO) REFERENCES T_PRODUCTOS(CODIGOPRODUCTO)
);
```

```
175 14:13:40 CREATE TABLE T_PRODUCTOS ( CODIGOPRODUCTO VARCHAR(10) PRIMARY KEY, NOMBRE VARCHAR(50), CANTIDADENSTOCK INT ) 0 row(s) affected
176 14:13:40 CREATE TABLE T_DETALLEPEDIDOS ( CODIGOPEDIDO INT, CODIGOPRODUCTO VARCHAR(10), CANTIDAD INT, PRECIOUNIDAD DECIMAL(10,2), NUMEROLINEA INT, FOREIGN KEY (CODIGOPROD... 0 row(s) affected
```

```sql
CREATE TABLE T_PRODUCTOS (
  CODIGOPRODUCTO VARCHAR(10) PRIMARY KEY,
  NOMBRE VARCHAR(50),
  CANTIDADENSTOCK INT
);

CREATE TABLE T_DETALLEPEDIDOS (
  CODIGOPEDIDO INT,
  CODIGOPRODUCTO VARCHAR(10),
  CANTIDAD INT,
  PRECIOUNIDAD DECIMAL(10,2),
  NUMEROLINEA INT,
  FOREIGN KEY (CODIGOPRODUCTO) REFERENCES T_PRODUCTOS(CODIGOPRODUCTO)
);
INSERT INTO T_PRODUCTOS (CODIGOPRODUCTO, NOMBRE, CANTIDADENSTOCK)
VALUES ('FR-67', 'Producto Prueba', 100);
```

177 14:14:29 INSERT INTO T_PRODUCTOS (CODIGOPRODUCTO, NOMBRE, CANTIDADENSTOCK) VALUES ('FR-67', 'Producto Prueba', 100) 1 row(s) affected

7. Crea un procedimiento que use un cursor para ver todos los productos comprados en un pedido (pasa el código de pedido como parámetro). Muestra también la cantidad de cada uno de los artículos.


```
DELIMITER //

CREATE TRIGGER trg_actualiza_stock
AFTER INSERT ON T_DETALLEPEDIDOS
FOR EACH ROW
BEGIN
  UPDATE T_PRODUCTOS
  SET CANTIDADENSTOCK = CANTIDADENSTOCK - NEW.CANTIDAD
  WHERE CODIGOPRODUCTO = NEW.CODIGOPRODUCTO;
END;
//

CREATE TRIGGER trg_restaura_stock
AFTER DELETE ON T_DETALLEPEDIDOS
FOR EACH ROW
BEGIN
  UPDATE T_PRODUCTOS
  SET CANTIDADENSTOCK = CANTIDADENSTOCK + OLD.CANTIDAD
  WHERE CODIGOPRODUCTO = OLD.CODIGOPRODUCTO;
END;
//

DELIMITER ;
INSERT INTO T_DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD,
PRECIOUNIDAD, NUMEROLINEA)
VALUES (4, 'FR-67', 5, 70, 5);

SELECT * FROM T_PRODUCTOS;

DELETE FROM T_DETALLEPEDIDOS
WHERE CODIGOPEDIDO = 4 AND CODIGOPRODUCTO = 'FR-67';
SELECT * FROM T_PRODUCTOS;
```

```sql
DELIMITER //

CREATE TRIGGER trg_actualiza_stock
AFTER INSERT ON T_DETALLEPEDIDOS
FOR EACH ROW
BEGIN
  UPDATE T_PRODUCTOS
  SET CANTIDADENSTOCK = CANTIDADENSTOCK - NEW.CANTIDAD
  WHERE CODIGOPRODUCTO = NEW.CODIGOPRODUCTO;
END;
//

CREATE TRIGGER trg_restaura_stock
AFTER DELETE ON T_DETALLEPEDIDOS
FOR EACH ROW
BEGIN
  UPDATE T_PRODUCTOS
  SET CANTIDADENSTOCK = CANTIDADENSTOCK + OLD.CANTIDAD
  WHERE CODIGOPRODUCTO = OLD.CODIGOPRODUCTO;
END;
//

DELIMITER ;
INSERT INTO T_DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD, PRECIOUNIDAD, NUMEROLINEA)
VALUES (4, 'FR-67', 5, 70, 5);

SELECT * FROM T_PRODUCTOS;
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CODIGOPRODUCTO	NOMBRE	CANTIDADENSTOCK
FR-67	Producto Prueba	95
NULL	NULL	NULL

```
DELIMITER //

CREATE TRIGGER trg_actualiza_stock
AFTER INSERT ON T_DETALLEPEDIDOS
FOR EACH ROW
BEGIN
  UPDATE T_PRODUCTOS
  SET CANTIDADENSTOCK = CANTIDADENSTOCK - NEW.CANTIDAD
  WHERE CODIGOPRODUCTO = NEW.CODIGOPRODUCTO;
END;
//

CREATE TRIGGER trg_restaura_stock
AFTER DELETE ON T_DETALLEPEDIDOS
FOR EACH ROW
BEGIN
  UPDATE T_PRODUCTOS
  SET CANTIDADENSTOCK = CANTIDADENSTOCK + OLD.CANTIDAD
  WHERE CODIGOPRODUCTO = OLD.CODIGOPRODUCTO;
END;
//

DELIMITER ;
INSERT INTO T_DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD, PRECIOUNIDAD, NUMEROLINEA)
VALUES (4, 'FR-67', 5, 70, 5);

SELECT * FROM T_PRODUCTOS;

DELETE FROM T_DETALLEPEDIDOS
WHERE CODIGOPEDIDO = 4 AND CODIGOPRODUCTO = 'FR-67';
SELECT * FROM T_PRODUCTOS;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CODIGOPRODUCTO	NOMBRE	CANTIDADENSTOCK
FR-67	Producto Prueba	100
NULL	NULL	NULL

8. **Crea un procedimiento almacenado para dar de alta un nuevo producto. Para ello, deberás pasar como parámetros los siguientes valores como mínimo: código, nombre, gama, cantidadstock y precioventa.**

```
DELIMITER //

CREATE PROCEDURE ProductosPorPedido(IN pedido INT)
BEGIN
  DECLARE fin INT DEFAULT FALSE;
  DECLARE codProd VARCHAR(10);
  DECLARE cant INT;
```

```
  DECLARE cur CURSOR FOR
    SELECT CODIGOPRODUCTO, CANTIDAD
    FROM T_DETALLEPEDIDOS
    WHERE CODIGOPEDIDO = pedido;

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;
  OPEN cur;

  bucle: LOOP
    FETCH cur INTO codProd, cant;
    IF fin THEN
      LEAVE bucle;
    END IF;
    SELECT CONCAT('Producto: ', codProd, ' | Cantidad: ', cant) AS Detalle;
  END LOOP;

  CLOSE cur;
END;
//

DELIMITER ;
CALL ProductosPorPedido(10);
```

```
1   DELIMITER //
2
3   CREATE PROCEDURE ProductosPorPedido(IN pedido INT)
4   BEGIN
5     DECLARE fin INT DEFAULT FALSE;
6     DECLARE codProd VARCHAR(10);
7     DECLARE cant INT;
8
9     DECLARE cur CURSOR FOR
10      SELECT CODIGOPRODUCTO, CANTIDAD
11      FROM T_DETALLEPEDIDOS
12      WHERE CODIGOPEDIDO = pedido;
13
14    DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;
15
16    OPEN cur;
17
18    bucle: LOOP
19      FETCH cur INTO codProd, cant;
20      IF fin THEN
21        LEAVE bucle;
22      END IF;
23      SELECT CONCAT('Producto: ', codProd, ' | Cantidad: ', cant) AS Detalle;
24    END LOOP;
25
26    CLOSE cur;
27  END;
28  //
29
30  DELIMITER ;
31  CALL ProductosPorPedido(10);
32
33
```

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

Detalle
Producto: FR-67 | Cantidad: 3

```
DELIMITER //

CREATE PROCEDURE Alta_Producto(
  IN p_codigo VARCHAR(10),
  IN p_nombre VARCHAR(50),
  IN p_gama VARCHAR(50),
  IN p_cantidad INT,
```

```
  IN p_precioVenta DECIMAL(10,2)
)
BEGIN
  DECLARE existe INT;

  SELECT COUNT(*) INTO existe
  FROM T_PRODUCTOS
  WHERE CODIGOPRODUCTO = p_codigo;

  IF existe > 0 THEN
    SELECT 'El producto ya existe' AS Resultado;
  ELSE
    INSERT INTO T_PRODUCTOS (CODIGOPRODUCTO, NOMBRE,
CANTIDADENSTOCK)
    VALUES (p_codigo, p_nombre, p_cantidad);

    -- Nota: omitimos GAMA y PRECIOVENTA porque no existen aún en la tabla que
creamos, ¡pero puedes añadirlos si los agregas!
    SELECT 'Producto insertado correctamente' AS Resultado;
  END IF;
END;
//

DELIMITER ;

CALL Alta_Producto('FR-99', 'Producto Nuevo', 'Oficina', 20, 55.00);
CALL Alta_Producto('FR-99', 'Producto Nuevo', 'Oficina', 20, 55.00);
```

```sql
DELIMITER //

CREATE PROCEDURE Alta_Producto(
  IN p_codigo VARCHAR(10),
  IN p_nombre VARCHAR(50),
  IN p_gama VARCHAR(50),
  IN p_cantidad INT,
  IN p_precioVenta DECIMAL(10,2)
 )
BEGIN
  DECLARE existe INT;

  SELECT COUNT(*) INTO existe
  FROM T_PRODUCTOS
  WHERE CODIGOPRODUCTO = p_codigo;

  IF existe > 0 THEN
    SELECT 'El producto ya existe' AS Resultado;
  ELSE
    INSERT INTO T_PRODUCTOS (CODIGOPRODUCTO, NOMBRE, CANTIDADENSTOCK)
    VALUES (p_codigo, p_nombre, p_cantidad);

    -- Nota: omitimos GAMA y PRECIOVENTA porque no existen aún en la tabla que creamos, ¡pero puedes añadirlos si los agregas!
    SELECT 'Producto insertado correctamente' AS Resultado;
  END IF;
 END;
 //

 DELIMITER ;
 CALL Alta_Producto('FR-99', 'Producto Nuevo', 'Oficina', 20, 55.00);
 CALL Alta_Producto('FR-99', 'Producto Nuevo', 'Oficina', 20, 55.00);
```

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

Resultado
El producto ya existe

200	14:35:11	CREATE PROCEDURE ProductosPorPedido(IN pedido INT) BEGIN DECLARE fin INT DEFAULT FALSE; DECLARE codProd VARCHAR(10); DECLARE cant INT; DECLARE cur CURSOR FOR SELECT CODI...	0 row(s) affected
201	14:35:21	CALL ProductosPorPedido(10)	1 row(s) returned
202	14:38:10	CREATE PROCEDURE Alta_Producto(IN p_codigo VARCHAR(10), IN p_nombre VARCHAR(50), IN p_gama VARCHAR(50), IN p_cantidad INT, IN p_precioVenta DECIMAL(10,2)) BEGIN DECLARE existe INT...	0 row(s) affected
203	14:38:22	CALL Alta_Producto('FR-99', 'Producto Nuevo', 'Oficina', 20, 55.00)	1 row(s) returned

Antes de dar de alta el nuevo producto, el procedimiento debe asegurarse de que ese producto no existe en la tabla. Si existe, deberá mostrar un mensaje avisando de que ese producto ya está dado de alta. Por el contrario, si no existe, procederá con la inserción y mostrará un mensaje de éxito. Hazlo sin usar excepciones.

Conclusión

Este caso práctico me ha ayudado a comprender cómo automatizar operaciones dentro de una base de datos mediante procedimientos, cursores y triggers. He aprendido a usar estructuras de control, validar datos sin necesidad de excepciones y adaptar el código a las posibilidades del sistema gestor (como en el caso del FOR en MySQL). Este ejercicio ha reforzado mis habilidades prácticas y mi comprensión del lenguaje SQL.

Recursos

Se deberá consultar el contenido de la unidad, internet, libros, revistas y utilizar medios informáticos para la presentación del caso práctico (Word, Power-Point…)

Objetivos

Reconocer e identificar formas alternativas de automatizar tareas. Definir y utilizar guiones para automatizar tareas. Hacer uso de las funciones proporcionadas por el sistema gestor.

Resultados de aprendizaje y criterios de evaluación

Desarrolla procedimientos almacenados evaluando y utilizando las sentencias del lenguaje incorporado en el sistema gestor de bases de datos.

- Se han identificado las diversas formas de automatizar tareas.
- Se han reconocido los métodos de ejecución de guiones.
- Se han identificado las herramientas disponibles para editar guiones.
- Se han definido y utilizado guiones para automatizar tareas.
- Se han utilizado estructuras de control de flujo.
- Se ha hecho uso de las funciones proporcionadas por el sistema gestor.
- Se han definido procedimientos y funciones de usuario.
- Se han definido disparadores.
- Se han utilizado cursores.
- Se han utilizado excepciones.

RUBRICA

CRITERIOS	Excelente	Notable	Satisfactorio	Insuficiente
Correcta Implementación de Procedimientos y Funciones 20%	Implementación perfecta de todos los procedimientos y funciones. Código optimizado y eficiente. Cumple exactamente con los requisitos de la tarea.	Implementación correcta con pequeños errores o ineficiencias. Cumple con la mayoría de los requisitos de la tarea.	Implementación básica que cumple con los requisitos mínimos. Errores o ineficiencias que no impiden la funcionalidad.	Implementación con errores significativos. No cumple con los requisitos básicos de la tarea.

	2.0 puntos	1.5 puntos	1.0 puntos	0 puntos
Uso Efectivo de Estructuras de Control 20%	Uso avanzado y eficiente de estructuras de control y cursores. Muestra una comprensión profunda de su aplicación práctica.	Uso correcto de estructuras de control y cursores, con pequeñas ineficiencias o errores que no afectan significativamente el resultado final.	Uso básico de estructuras de control y cursores. Errores o ineficiencias evidentes, pero la funcionalidad general se mantiene.	Uso inadecuado o incorrecto de estructuras de control y cursores, lo que resulta en una funcionalidad deficiente o errores significativos.
	2.0 puntos	1.5 puntos	1.0 puntos	0 puntos
Creación y Manejo de Triggers 20%	Creación de TRIGGERs perfecta, con una lógica impecable y un manejo adecuado de los eventos de la base de datos.	Creación de TRIGGERs correcta con pequeñas ineficiencias o errores menores en la lógica.	Creación básica de TRIGGERs que cumple con los requisitos mínimos. Errores o ineficiencias notables.	Creación inadecuada o incorrecta de TRIGGERs, con errores significativos en la lógica o en el manejo de eventos de la base de datos.
	2.0 puntos	1.5 puntos	1.0 punto	0 puntos
Manejo de Excepciones y Validaciones 20%	Manejo de excepciones y validaciones realizado de manera experta, mostrando una comprensión avanzada y un enfoque proactivo.	Manejo de excepciones y validaciones correcto, aunque con espacio para una mayor refinación o eficiencia.	Manejo básico de excepciones y validaciones, cumpliendo con los requisitos mínimos, pero con margen de mejora.	Manejo inadecuado o incorrecto de excepciones y validaciones, con errores significativos o falta de implementación.
	2 puntos	1.5 puntos	1 punto	0 puntos
Presentación, redacción y ortografía 10%	Presenta un discurso ordenado y comprensible, profundizando en todos los conceptos. Además, no se aprecian errores de gramática,	Presenta un discurso ordenado y comprensible, pero aparecen algunos errores de gramática, ortografía o puntuación.	Presenta un discurso comprensible, aunque algo desordenado y sin profundizar en conceptos e ideas y/o se observan varios errores	Presenta un discurso desordenado que dificulta la comprensión de los conceptos e ideas que se exponen y/o aparecen

	ortografía o puntuación.		ortográficos, de gramática o de puntuación.	numerosos errores gramaticales, de ortografía o puntuación.
	1 punto	0.75 puntos	0.5 puntos	0 puntos
Uso de recursos adicionales y creatividad 10%	Responde con gran originalidad, expresando ideas creativas e ingeniosas. Utiliza numerosas fuentes de información relevantes, fiables y actualizadas. Aporta imágenes, gráficos y recursos que clarifican la respuesta.	Muestra cierta originalidad en su respuesta. Utiliza diversas fuentes de información, aunque no todas son relevantes. Aporta alguna imagen, gráfico o recurso adicional.	Utiliza alguna fuente de información. Hace uso de cita de ideas de autores, pero no se aportan ideas y puntos de vista originales propios.	No hace uso de fuentes fiables ni añade recursos o se utilizan ideas de otros autores sin citar.
	1 punto	0.75 puntos	0.5 puntos	0 puntos