***EJERCICIOS UT5***

**PROCEDIMIENTOS.**

1. **Escribe un procedimiento que reciba el nombre de un país como parámetro de entrada y realice una consulta sobre la tabla cliente para obtener todos los clientes que existen en la tabla de ese país.**

DELIMITER $$

CREATE PROCEDURE Nom\_1(IN PAIS VARCHAR(50))

BEGIN

SELECT \* FROM Clientes WHERE Pais=Pais;

END$$

CALL Nom\_1 (“Venezuela”)$$

1. **Escribe un procedimiento que reciba como parámetro de entrada una forma de pago, que será una cadena de caracteres (Ejemplo: PayPal, Transferencia, etc). Y devuelva como salida el pago de máximo valor realizado para esa forma de pago. Deberá hacer uso de la tabla pago de la base de datos jardineria.**

DELIMITER$$

CREATE PROCEDURE Nom\_2(IN VFPago (VARCHAR(50), OUT VMax\_Pago FLOAT)

BEGIN

SET VMax\_Pago = SELECT MAX(total) FROM Pago WHERE FPago=VFPago;

END$$

CALL Nom\_2(“Paypal, @VMAX\_Pago)$$

SELECT @VMAX\_Pago$$

1. **Escribe un procedimiento que reciba como parámetro de entrada una forma de pago, que será una cadena de caracteres (Ejemplo: PayPal, Transferencia, etc). Y devuelva como salida los siguientes valores teniendo en cuenta la forma de pago seleccionada como parámetro de entrada: • el pago de máximo valor, • el pago de mínimo valor, • el valor medio de los pagos realizados, • la suma de todos los pagos, • el número de pagos realizados para esa forma de pago. Deberá hacer uso de la tabla pago de la base de datos jardineria.**

DELIMITER $$

CREATE PROCEDURE Proc3 (IN VFPago VARCHAR (50), OUT VMaxValor FLOAT, OUT VMinValor FLOAT, OUT VValorMedio FLOAT, OUT VSumaPagos FLOAT, OUT NPagos FLOAT)

BEGIN

SET VMaxValor = SELECT MAX(Total) FROM Pagos WHERE FPago=VFPago;

SET VMinValor = SELECT MIN(Total) FROM Pagos WHERE FPago=VFPago;

SET VValorMedio = SELECT AVG(Total) FROM Pagos WHERE FPago=VFPago;

SET VSumaPagos = SELECT SUM(Total) FROM Pagos WHERE FPago=VFPago;

SET NPagos = SELECT COUNT(Total) FROM Pagos WHERE FPago=VFPago;

END$

CALL Proc3(“Paypal”,@VMaxValor,@VMinValor,@VValorMedio,@VSumaPagos,@NPagos)$$

SELECT(@VMaxValor,@VMinValor,@VValorMedio,@VSumaPagos,@NPagos)$$

**FUNCIONES.**

1. **Escribe una función para la base de datos jardineria que devuelva el número total de productos que hay en la tabla productos.**

DELIMITER$

CREATE FUNCTION Fun1() RETURNS INT

BEGIN

DECLARE NProductos INT;

SET NProductos = SELECT COUNT (Codigo) FROM Productos;

RETURN NProductos

END$

SELECT Fun1()$

1. **Escribe una función para la base de datos jardinería que devuelva el valor medio del precio de los productos de un determinado fabricante que se recibirá como parámetro de entrada. El parámetro de entrada será el nombre del fabricante.**

**MANEJO DE ERRORES**

1. **Crea una base de datos llamada test que contenga una tabla llamada alumno. La tabla debe tener cuatro columnas:**

**• id: entero sin signo (clave primaria).**

**• nombre: cadena de 50 caracteres.**

**• apellido1: cadena de 50 caracteres.**

**• apellido2: cadena de 50 caracteres.**

**Una vez creada la base de datos y la tabla deberá crear un procedimiento llamado insertar\_alumno con las siguientes características. El procedimiento recibe cuatro parámetros de entrada (id, nombre, apellido1, apellido2) y los insertará en la tabla alumno. El procedimiento devolverá como salida un parámetro llamado error que tendrá un valor igual a 0 si la operación se ha podido realizar con éxito y un valor igual a 1 en caso contrario.**

**Deberá manejar los errores que puedan ocurrir cuando se intenta insertar una fila que contiene una clave primaria repetida.**

**TRANSACCIONES CON PROCEDIMIENTOS ALMACENADOS.**

1. **Crea una base de datos llamada cine que contenga dos tablas con las siguientes columnas.**

**Tabla cuentas:**

**• id\_cuenta: entero sin signo (clave primaria).**

**• saldo: real sin signo.**

**Tabla entradas:**

**• id\_butaca: entero sin signo (clave primaria).**

**• nif: cadena de 9 caracteres.**

**Una vez creada la base de datos y las tablas deberá crear un procedimiento llamado comprar\_entrada con las siguientes características. El procedimiento recibe 3 parámetros de entrada (nif, id\_cuenta, id\_butaca) y devolverá como salida un parámetro llamado error que tendrá un valor igual a 0 si la compra de la entrada se ha podido realizar con éxito y un valor igual a 1 en caso contrario.**

**El procedimiento de compra realiza los siguientes pasos:**

**• Inicia una transacción.**

**• Actualiza la columna saldo de la tabla cuentas cobrando 5 euros a la cuenta con el id\_cuenta adecuado.**

**• Inserta una una fila en la tabla entradas indicando la butaca (id\_butaca) que acaba de comprar el usuario (nif).**

**• Comprueba si ha ocurrido algún error en las operaciones anteriores. Si no ocurre ningún error entonces aplica un COMMIT a la transacción y si ha ocurrido algún error aplica un ROLLBACK.**

**Deberá manejar los siguientes errores que puedan ocurrir durante el proceso.**

**• ERROR 1264 (Out of range value)**

**• ERROR 1062 (Duplicate entry for PRIMARY KEY)**

DELIMITER $

CREATE PROCEDURE Comprar\_Entrada(IN Vnif VARCHAR(6), IN Vid\_cuenta INT UNSIGNED, IN Vid\_butaca UNSIGNED, OUT error TINYINT)

BEGIN

DECLARE CONTINUE HANDLER FOR 1264,1062

BEGIN

SET error=1;

END;

START TRANSACTION;

SET error=0;

UPDATE Cuentas SET saldo-5 WHERE id\_cuenta=Vid\_cuenta;

INSERT INTO Entradas VALUES (Vid\_butaca,Vnif);

IF error=0 THEN COMMIT;

ELSE ROLLBACK;

END IF;

END$

**CURSORES.**

1. **Escribe las sentencias SQL necesarias para crear una base de datos llamada test, una tabla llamada alumnos y 4 sentencias de inserción para inicializar la tabla. La tabla alumnos está formada por las siguientes columnas:**

**• id (entero sin signo y clave primaria)**

**• nombre (cadena de caracteres)**

**• apellido1 (cadena de caracteres)**

**• apellido2 (cadena de caracteres**

**• fecha\_nacimiento (fecha)**

**Una vez creada la tabla se decide añadir una nueva columna a la tabla llamada edad que será un valor calculado a partir de la columna fecha\_nacimiento. Escriba la sentencia SQL necesaria para modificarla tabla y añadir la nueva columna.**

**Escriba una función llamada calcular\_edad que reciba una fecha y devuelva el número de años que han pasado desde la fecha actual hasta la fecha pasada como parámetro:**

**• Función: calcular\_edad**

**• Entrada: Fecha**

**• Salida: Número de años (entero)**

**Ahora escriba un procedimiento que permita calcular la edad de todos los alumnmos que ya existen en la tabla.**

**Para esto será necesario crear un procedimiento llamado actualizar\_columna\_edad que calcule la edad de cada alumno y actualice la tabla. Este procedimiento hará uso de la función calcular\_edad que hemos creado en el paso anterior.**

1. **Modifica la tabla alumnos del ejercicio anterior para añadir una nueva columna email. Una vez que hemos modificado la tabla necesitamos asignarle una dirección de correo electrónico de forma automática.**

**Escriba un procedimiento llamado crear\_email que dados los parámetros de entrada: nombre, apellido1, apellido2 y dominio, cree una dirección de email y la devuelva como salida.**

**• Procedimiento: crear\_email**

**• Entrada:**

**– nombre (cadena de caracteres)**

**– apellido1 (cadena de caracteres)**

**– apellido2 (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 apellido1.**

**• Los tres primeros caracteres del parámetro apellido2.**

**• El carácter @.**

**• El dominio pasado como parámetro.**

**Ahora escriba un procedimiento que permita crear un email para todos los alumnmos que ya existen en la tabla.**

**Para esto será necesario crear un procedimiento llamado actualizar\_columna\_email que actualice la columna email de la tabla alumnos. Este procedimiento hará uso del procedimiento crear\_email que hemos creado en el paso anterior.**

**TRIGGERS.**

1. **Crea una base de datos llamada test que contenga una tabla llamada alumnos con las siguientes columnas.**

**Tabla alumnos:**

**• id (entero sin signo)**

**• nombre (cadena de caracteres)**

**• apellido1 (cadena de caracteres)**

**• apellido2 (cadena de caracteres)**

**• nota (número real)**

**Una vez creada la tabla, escriba dos triggers con las siguientes características:**

**• Trigger 1: trigger\_check\_nota\_before\_insert**

**– Se ejecuta sobre la tabla alumnos.**

**– Se ejecuta antes de una operación de inserción.**

**– Si el nuevo valor de la nota que se quiere insertar es negativo, se guarda como 0.**

**– Si el nuevo valor de la nota que se quiere insertar es mayor que 10, se guarda como 10.**

**• Trigger2 : trigger\_check\_nota\_before\_update**

**– Se ejecuta sobre la tabla alumnos.**

**– Se ejecuta antes de una operación de actualización.**

**– Si el nuevo valor de la nota que se quiere actualizar es negativo, se guarda como 0.**

**– Si el nuevo valor de la nota que se quiere actualizar es mayor que 10, se guarda como 10.**

**Una vez creados los triggers escriba varias sentencias de inserción y actualización sobre la tabla alumnos y verifica que los triggers se están ejecutando correctamente.**