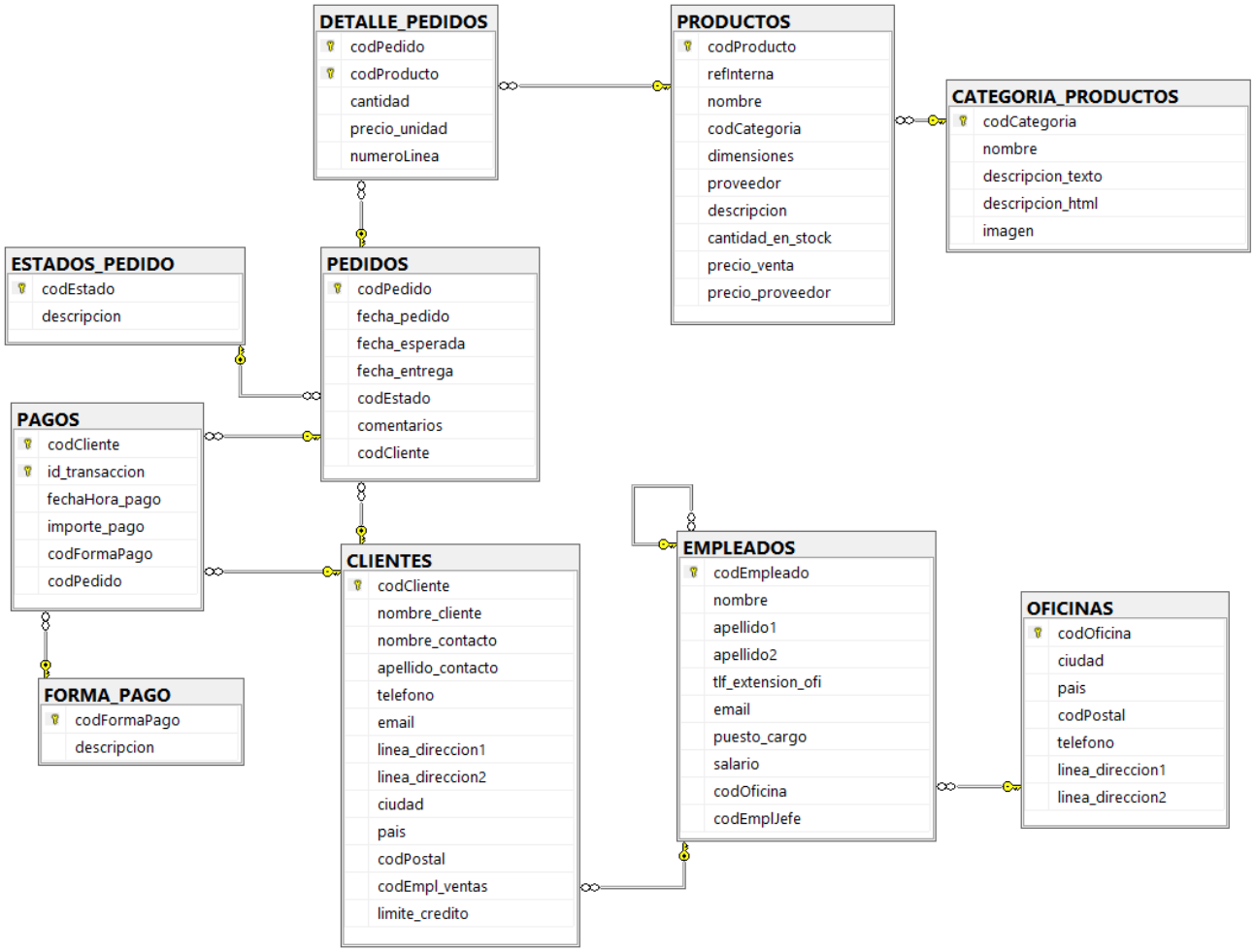
EXAMEN TERCERA EVALUACIÓN

Nombre: _____Leo Coves Guzman

Utilizando la base de datos JARDINERIA que hemos trabajado durante el curso, resuelve las siguientes cuestiones:



EXAMEN TERCERA EVALUACIÓN

1.- Gestión de cursores (2 puntos)

Crea un script que itere por cada uno de los registros de la tabla **OFICINAS**

Para cada oficina deberá mostrarse **una línea** con la siguiente información:

*"La oficina **XXX**, ubicada en la ciudad **YYY**, tiene un total de **ZZZ** empleados"*

Siendo:

- ***XXX**: el código de la oficina*
- ***YYY**: la ciudad de la oficina*
- ***ZZZ**: el número de empleados que trabajan en ella*

Pega aquí tu código con la respuesta

```
USE JARDINERIA
GO
DECLARE @codOficina CHAR(6)
DECLARE @ciudad VARCHAR(40)

DECLARE recorrer_Oficinas CURSOR FOR
SELECT codOficina, ciudad
       FROM OFICINAS

OPEN recorrer_Oficinas
FETCH NEXT FROM recorrer_Oficinas INTO @codOficina, @ciudad

WHILE @@FETCH_STATUS = 0
BEGIN
       DECLARE @numEmpleados INT
       SET @numEmpleados = (SELECT COUNT(codEmpleado)
                                   FROM EMPLEADOS
                                   WHERE codOficina = @codOficina)

       PRINT CONCAT('La oficina ', @codOficina, ', ubicada en la ciudad ', @ciudad, ', tiene
       un total de ', @numEmpleados, ' empleados.')
       FETCH NEXT FROM recorrer_Oficinas INTO @codOficina, @ciudad
END

CLOSE recorrer_Oficinas
DEALLOCATE recorrer_Oficinas
```

EXAMEN TERCERA EVALUACIÓN

2.- Implementación de funciones y llamada (3 puntos)

Crear una función llamada **cuentaProductosCategoria** que reciba como parámetros un **codCategoria**, un **minPrecio** y un **maxPrecio;** y devuelva el número de productos incluidos en ella cuyo precio esté comprendido entre minPrecio y maxPrecio.

Crear una función llamada **obtenerCostePedido** que reciba como parámetro un **codPedido** y devuelva el coste total de dicho pedido (no se permite el uso del campo totalLinea, debe calcularse dentro de la función).

Implementa, también, dos SELECTs en las que pruebes el correcto funcionamiento de las funciones **cuentaProductosCategoria** y **obtenerCostePedido**.

Pega aquí tu código con la respuesta

```
GO
CREATE OR ALTER FUNCTION cuentaProductosCategoria(@codCategoria CHAR(2), @minPrecio
DECIMAL(7,2), @maxPrecio DECIMAL(7,2))
RETURNS INT
AS
BEGIN
	DECLARE @salida INT

	SET @salida = (SELECT COUNT(codProducto)
				FROM PRODUCTOS
				WHERE codCategoria = @codCategoria
				AND precio_venta BETWEEN @minPrecio AND @maxPrecio)

	RETURN @salida
END
------------------------------
SELECT codCategoria, dbo.cuentaProductosCategoria(codCategoria, 3.55, 20.50) AS numproductos
	FROM CATEGORIA_PRODUCTOS
	WHERE codCategoria = 'OR'
```

EXAMEN TERCERA EVALUACIÓN

```
------------------------------------------------------------
------------------------------------------------------------
GO
CREATE OR ALTER FUNCTION obtenerCostePedido(@codPedido INT)
RETURNS DECIMAL(9,2)
AS
BEGIN
	DECLARE @salida DECIMAL(9,2)

	SET @salida = (SELECT ISNULL(SUM(precio_unidad * cantidad),0)
				FROM DETALLE_PEDIDOS
				WHERE codPedido = @codPedido)

	RETURN @salida
END
---------------------------
SELECT codPedido, dbo.obtenerCostePedido(codPedido) AS costeTotalPedido
	FROM PEDIDOS
	WHERE codPedido = 1
```

EXAMEN TERCERA EVALUACIÓN

3.- Creación de un procedimiento y llamada (4 puntos)

Crea un **procedimiento** llamado **realizarPago** que reciba como parámetros de entrada: "codCliente", "codFormaPago", "importe_pago" y "codPedido". El procedimiento deberá:

1º Validar que los parámetros son correctos y permiten realizar el procedimiento

2º Insertar un nuevo registro en la tabla de PAGOS. Los campos que tiene la tabla son:

- **codCliente, codFormaPago, importe_pago, codPedido: parámetros de entrada**
- **fechaHora_pago: fecha del sistema**
- **id_transaccion: debe calcularse automáticamente dentro del procedimiento**. Todos ellos siguen la estructura: "*ak-std-NNNNNN*", siendo N un número de 6 cifras relleno con ceros por la izquierda. Por ejemplo, si el último número es el "*ak-std-000026*" el procedimiento deberá obtener el "*ak-std-000027*", y así sucesivamente.

3º Actualizar el campo codEstado del pedido relacionado con el pago a estado 'F' (finalizado) y concatenar al final del campo "comentarios" la cadena "Pago realizado." (respetando lo que hubiera previamente).

Si se consigue realizar todas las acciones del procedimiento sin problemas, **se imprimirá un mensaje dentro del procedimiento indicando que el pago se ha realizado correctamente**.

IMPORTANTE: Considera utilizar TODO lo que hemos visto en clase (incluida la tabulación).

Por último, implementa un script que llame a tu procedimiento con variables y datos de prueba evaluando el valor de retorno de la llamada. Si el procedimiento finaliza con errores, debes impedir que se continúe la ejecución del script tras la llamada. (1 punto).

EXAMEN TERCERA EVALUACIÓN

Pega aquí tu código con la respuesta

```
GO
CREATE OR ALTER PROCEDURE realizarPago(@codCliente INT, @codFormaPago CHAR(1), @importe_pago
DECIMAL(9,2), @codPedido INT)
AS
BEGIN
	BEGIN TRY
		BEGIN TRAN
			/*Validaciones*/
			IF @codCliente IS NULL
			BEGIN
				PRINT('El codigo de cliente es obligatorio')
				RETURN -1
			END

			IF NOT EXISTS(SELECT codCliente
						FROM CLIENTES
						WHERE codCliente = @codCliente)
			BEGIN
				PRINT('El cliente no existe en la BD')
				RETURN -1
			END

			IF @codFormaPago IS NULL
			BEGIN
				PRINT('La forma de pago es obligatoria')
				RETURN -1
			END

			IF @importe_pago IS NULL
			BEGIN
				PRINT('El importe es obligatorio')
			END

			IF @codPedido IS NULL OR
			NOT EXISTS(SELECT codPedido FROM PEDIDOS WHERE codPedido = @codPedido)
			BEGIN
				PRINT('El Pedido es obligatorio')
				RETURN -1
			END
			/*Insert*/
			DECLARE @numTransaccion CHAR(6)
			SET @numTransaccion = (SELECT TOP(1)RIGHT(id_transaccion,6) + 1
									FROM PAGOS
									ORDER BY id_transaccion DESC)


			INSERT INTO PAGOS (codCliente, id_transaccion, fechaHora_pago,
			importe_pago, codFormaPago, codPedido)
			VALUES (@codCliente, CONCAT('ak-std-','0000', @numTransaccion),
			GETDATE(), @importe_pago, @codFormaPago, @codPedido)
```

EXAMEN TERCERA EVALUACIÓN

```
                /*Update*/
                UPDATE PEDIDOS
                SET codEstado = 'F',
                        comentarios = CONCAT(comentarios, '/ Pago Realizado')
                WHERE codPedido = @codPedido

                /*No se inserta el codEstado porque no existe ese codEstado*/

                PRINT('Todo correcto')
        COMMIT
    END TRY
    BEGIN CATCH
        ROLLBACK
        PRINT CONCAT('Error: ', ERROR_NUMBER(),
                        'Linea: ', ERROR_LINE(),
                        'Mensaje: ', ERROR_MESSAGE(),
                        'Procedure: ', ERROR_PROCEDURE())
    END CATCH
END

SELECT *
    FROM ESTADOS_PEDIDO
--------------------
DECLARE @codCliente INT = 1
DECLARE @codFormaPago CHAR(1) = 'T'
DECLARE @importe DECIMAL(9,2) = 33
DECLARE @codPedido INT = 3
DECLARE @ret INT

EXEC @ret = realizarPago @codCliente, @codFormaPago, @importe, @codPedido

IF @ret <> 0
RETURN

PRINT('el pago se ha realizado correctamente')
```

EXAMEN TERCERA EVALUACIÓN

4.- Gestión de triggers (1 punto)

Crea un trigger llamado **TR_CATEGORIA_PRODUCTOS** que se active **cuando se actualice o se elimine un registro de la tabla CATEGORIA_PRODUCTOS** y cree automáticamente una copia de seguridad del registro modificado/borrado en otra tabla llamada **HIST_CAT_PRODUCTOS** que tenga la misma estructura que la tabla **CATEGORIA_PRODUCTOS** más otro campo llamado **fechaOperación** de tipo fecha/hora. Este campo deberá rellenarse con la fecha del día.

Pega aquí tu código con la respuesta

```
GO
CREATE TABLE HIST_CAT_PRODUCTOS(
codCategoria                CHAR(2),
nombre                      VARCHAR(50) NOT NULL,
descripcion_texto           VARCHAR(100),
descripcion_html            VARCHAR(100),
imagen                      VARCHAR(255),
fechaOperacion              DATETIME NOT NULL
)

GO
CREATE OR ALTER TRIGGER TR_CATEGORIA_PRODUCTOS ON CATEGORIA_PRODUCTOS
AFTER UPDATE
AS
BEGIN
       INSERT INTO HIST_CAT_PRODUCTOS (codCategoria, nombre, descripcion_texto,
       descripcion_html, imagen, fechaOperacion)
       SELECT codCategoria, nombre, descripcion_texto, descripcion_html, imagen, GETDATE()
              FROM deleted
END

---------
UPDATE CATEGORIA_PRODUCTOS
SET nombre = 'Nuevo Nombre'
WHERE codCategoria = 'OR'
```