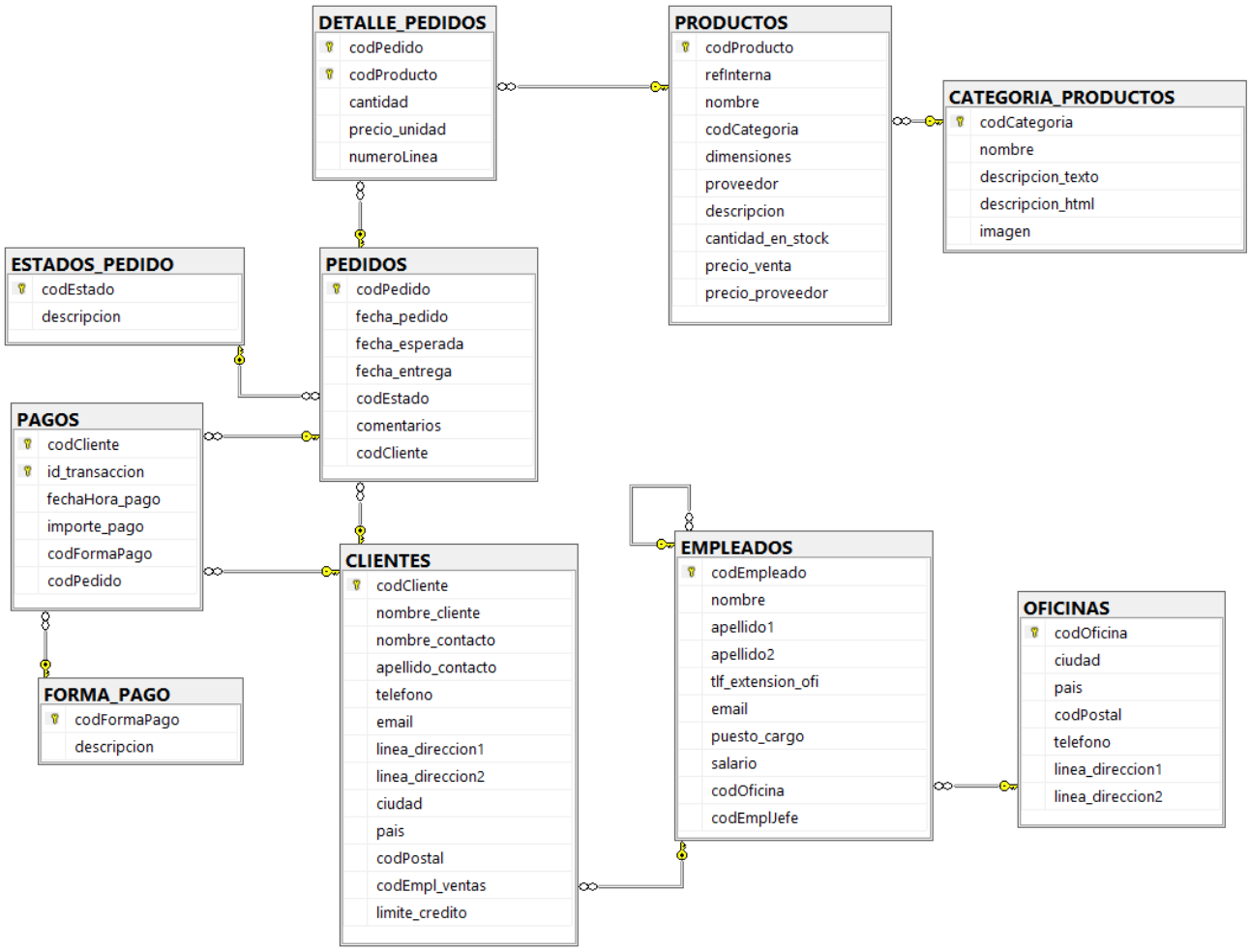
EXAMEN TERCERA EVALUACIÓN

Nombre: _____

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

```
DECLARE @cont INT = 0
    DECLARE @max INT = (SELECT COUNT(1) FROM OFICINAS)
    DECLARE @salida VARCHAR(MAX)

    WHILE (@cont < @max)
    BEGIN
        DECLARE @codOficina CHAR(6)
        DECLARE @ciudad VARCHAR(40)
        DECLARE @numEmpleados INT

        SELECT @codOficina = codOficina, @ciudad = ciudad
          FROM OFICINAS
         ORDER BY codOficina ASC
        OFFSET @cont ROWS
         FETCH NEXT 1 ROWS ONLY

        SELECT @numEmpleados = COUNT(1)
          FROM EMPLEADOS
         WHERE codOficina = @codOficina

        PRINT CONCAT('La oficina ', @codOficina, ' ubicada en la ciudad ', @ciudad,', tiene un
total de ', @numEmpleados,' empleados')
        SET @cont += 1
    END
```

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

```
CREATE OR ALTER FUNCTION cuentaProductosCategoria (@codCategoria CHAR(2), @minPrecio
DECIMAL(11,2), @maxPrecio DECIMAL(11,2))
RETURNS INT
AS
BEGIN
    RETURN (SELECT COUNT(1)
              FROM PRODUCTOS
             WHERE precio_venta >= @minPrecio
               AND precio_venta <= @maxPrecio
               AND codCategoria = @codCategoria)
END

GO

CREATE OR ALTER FUNCTION obtenerCostePedido (@codPedido INT)
RETURNS DECIMAL(11,2)
BEGIN
    RETURN (SELECT SUM(precio_unidad * cantidad)
              FROM DETALLE_PEDIDOS
             WHERE codPedido = @codPedido)
END
GO

SELECT *
  FROM PRODUCTOS
 WHERE codProducto = dbo.cuentaProductosCategoria ('OR', 5, 10)

SELECT *
  FROM PAGOS
 WHERE importe_pago = dbo.obtenerCostePedido(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

```
CREATE OR ALTER PROCEDURE realizarPago(@codCliente INT, @codFormaPago CHAR, @importe_pago
DECIMAL(11,2), @codPedido INT)
AS
BEGIN
    DECLARE @errorCodigos INT
    SELECT @errorCodigos = codCliente FROM CLIENTES WHERE codCliente = @codCliente
    IF(@codCliente IS NULL OR @errorCodigos IS NULL)
    BEGIN
        RETURN -1
    END

    DECLARE @errorFormaPago CHAR
    SELECT @errorFormaPago = codFormaPago FROM FORMA_PAGO WHERE codFormaPago = @codFormaPago
    IF(@codFormaPago IS NULL OR @errorFormaPago IS NULL)
    BEGIN
        RETURN -2
    END

    IF(@importe_pago IS NULL OR @importe_pago < 0)
    BEGIN
        RETURN -3
    END

    SET @errorCodigos = NULL
    SELECT @errorCodigos = codPedido FROM PEDIDOS WHERE codPedido = @codPedido
    IF(@codPedido IS NULL OR @errorCodigos IS NULL)
    BEGIN
        RETURN -4
    END

    DECLARE @idTransaccion VARCHAR(15) = 'ak-std-000000'
    DECLARE @numIdTransaccion INT = 0

    SELECT @numIdTransaccion = CAST(SUBSTRING(id_transaccion,8,8) AS INT)+1
      FROM PAGOS
     ORDER BY id_transaccion DESC
    OFFSET 0 ROWS
     FETCH NEXT 1 ROWS ONLY

    SET @idTransaccion = CONCAT(SUBSTRING(@idTransaccion, 0, 14-LEN(@numIdTransaccion)),
@numIdTransaccion)

    BEGIN TRY
    BEGIN TRAN
        INSERT INTO PAGOS
        VALUES (@codCliente, @idTransaccion, GETDATE(), @importe_pago, @codFormaPago, @codPedido)

        UPDATE PEDIDOS
           SET codEstado = 'F',
               comentarios = CONCAT(comentarios, CHAR(10), 'Pago realizado.')
```

EXAMEN TERCERA EVALUACIÓN

```
        WHERE codPedido = @codPedido
    COMMIT
    END TRY
    BEGIN CATCH
        ROLLBACK
        PRINT('Error al realizar el pago')
        RETURN -10
    END CATCH

    PRINT ('Pago realizado con exito')
    RETURN 1
END

GO

DECLARE @exit VARCHAR(MAX)
EXEC @exit = realizarPago 8, 'B', 98.20, 9
PRINT @exit
```

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

```
CREATE OR ALTER TRIGGER TR_CATEGORIA_PRODUCTOS ON CATEGORIA_PRODUCTOS
INSTEAD OF UPDATE, DELETE
AS
BEGIN
    INSERT INTO HIST_CAT_PRODUCTOS
    SELECT *, GETDATE() AS fechaOperacion
      FROM deleted
END
```