1. Escribir una sentencia SELECT que devuelva el número de factura, fecha de emisión y el nombre del día de la semana en español de la fecha de emisión de todas las facturas que no han sido pagadas.

   Nota: `SET @DIA = datepart(weekday,@fecha)`
   Devuelve en la variable @DIA el nro. de día de la semana , comenzando con 1 Domingo hasta 7 Sábado.

2. Escribir un Select que devuelva código de fabricante y todos los Productos que fabrica separados entre sí por coma (**,**). Asimismo mostrar en otra columna si el fabricante posee ventas o no (S o N). Utilizar una función para resolver la columna de ***productos***. Tiene libertad para resolver la de ***posee ventas*** como crea conveniente. Ejemplo de la salida

Fabricante	**Productos**	**Posee Ventas**
ALAS	Tornillos, Tuercas, Remaches	S
CASA	Silla	N
DOTO	Llave inglesa	S

3. Crear la tabla temporal ComprasClientes con los campos cliente_num (int, pk), CantFacturas(int), MontoTotal (decimal(12,2)).
   Crear un procedimiento SumaCompras que inserte (si no existe) o modifique el registro de la tabla ComprasClientes con la siguiente información:
   CantFacturas: Cantidad de órdenes de cada cliente (no nulo).
   MontoTotal: Monto total comprado por el cliente (p x q)

   Ejecute el procedimiento.

4. Crear un procedimiento actualizaPrecios que cree una tabla temporal ListaPrecios con los campos producto_cod, producto_desc, precio_unit.
   a. Inserte registros en la tabla temporal para cada producto de la tabla de Productos tal que
   b. Si el producto no tuvo ventas, copiar el producto con el precio actual
   c. Si el producto tuvo ventas y
      Si el precio unitario del producto es menor a 150 aumentar el precio en un 10%,
      Si no, el precio es mayor o igual a 150, aumentar el precio en un 20%

5. Crear una tabla temporal FabricantesNuevos sin datos pero con los mismos campos que la tabla de Fabricantes. Insertar 4 o 5 fabricantes en la tabla temporal creada.
   Crear un procedimiento insertaFabNuevos que
   a. Inserte en la tabla de Fabricantes los nuevos fabricantes informados en la tabla temporal.
   b. Valide que el fabricante no existe, en caso contrario es un error.
   c. Valide que el tiempo de entrega no es negativo.
   d. Valide que la provincia informada exista.
   e. En caso de haber un error. Deshacer **todas** las operaciones de **todos** los fabricantes.

6. Crear un procedimiento insertaFabNuevos2 que realice lo mismo que solicitado en el ejercicio anterior pero en caso de error solo deshaga las operaciones correspondientes al fabricante actual que se está procesando y continúe procesando los siguientes.

7. Crear un procedimiento borrarFactura que reciba un número de factura por parámetro y realice la eliminación de la misma junto con su detalle. Deberá manejar toda la operación en una transacción y manejar excepciones ante algún error que ocurra.
   Asimismo, el procedimiento deberá guardar en una tabla de auditoria auditFC los siguientes datos: factura_num, fecha_emision, cliente_num, cantidad de renglones, Monto Total (p x q), cant_productos (q).
   Ante un error deberá almacenar en una tablas erroresFC, parámetro ingresado, fecha ejecución, código y mensaje del error ocurrido (int y VARCHAR(50) respectivamente).

1.

```
SELECT factura_num, fecha_emision,
       dbo.fx_dia_semana(fecha_emision) dia
 FROM facturas f
 WHERE f.fecha_pago IS NULL


CREATE FUNCTION Fx_DIA_SEMANA
            (@FECHA  DATETIME)
    RETURNS VARCHAR (20) AS
--
BEGIN
    DECLARE @DIA INT
    DECLARE @RETORNO VARCHAR(20)

    SET @DIA = datepart(weekday,@fecha)

    SET @RETORNO = case when @dia = 1 then 'Domingo'
               when @dia = 2 then 'lunes'
                     when @dia = 3 then 'Martes'
                     when @dia = 4 then 'Miercoles'
                     when @dia = 5 then 'Jueves'
                  when @dia = 6 then 'Viernes'
                  else 'Sábado'
            END
RETURN @RETORNO
END
```

2.

```
SELECT f.fabricante_cod, dbo.Fx_productos(f.fabricante_cod) as Productos,
       (case WHEN exists (select 1 from fabricantes f2 JOIN productos p
                                            ON f2.fabricante_cod = p.fabricante_cod
                                                                                JOIN
facturas_det d
                                                                                    ON
p.producto_cod = d.producto_cod
                            where f2.fabricante_cod = f.fabricante_cod) THEN 'SI'
                      ELSE 'No'
             end) 'Posee Ventas'
  FROM fabricantes f


DROP FUNCTION Fx_productos
CREATE FUNCTION Fx_productos (@fabricante_cod VARCHAR(5)) RETURNS VARCHAR (100) AS
BEGIN

    DECLARE @RETORNO VARCHAR(100)
    DECLARE @producto_desc VARCHAR(20)

    DECLARE CUR_productos CURSOR FOR  SELECT producto_desc
                                        FROM productos
                                       WHERE fabricante_cod = @fabricante_cod
    SET @RETORNO = ''
    OPEN CUR_productos
    FETCH NEXT FROM CUR_productos INTO @producto_desc
    WHILE (@@FETCH_STATUS = 0)
```

```
    BEGIN
           SET @RETORNO = @RETORNO + @producto_desc + ', '
              FETCH NEXT FROM CUR_productos INTO @producto_desc
    END
    CLOSE CUR_productos
    DEALLOCATE CUR_productos

       if @retorno != ''
        SET @RETORNO = SUBSTRING(@RETORNO, 1, LEN(@RETORNO) - 1)

    RETURN @RETORNO
END
```

3.

```
create table #ComprasClientes
  (cliente_num int primary key,
   CantFacturas int,
   MontoTotal decimal(12,2)
   );

drop procedure sumaCompras

create procedure sumaCompras as
begin

   declare @cliente_num int, @cantidad int, @monto decimal(12,2)

   declare cur_compras CURSOR FOR select cliente_num,
                                    count(distinct f.factura_num) cantidad,
                                      sum(cantidad*precio_unit)
                                from facturas f join facturas_det d
                                    on f.factura_num = d.factura_num
                                  where f.cliente_num is NOT NULL
                                          group by cliente_num;
    OPEN cur_compras
    FETCH NEXT FROM cur_compras into @cliente_num, @cantidad, @monto

    WHILE (@@FETCH_STATUS = 0)
    BEGIN
        if exists (select 1 from #comprasClientes where cliente_num = @cliente_num)
             update #comprasClientes
                         set cantFacturas = @cantidad,
                                montoTotal = @monto
              where cliente_num = @cliente_num;
        else
             Insert into #comprasClientes (cliente_num, cantFacturas, montoTotal)
                                    values (@cliente_num, @cantidad, @monto);
        FETCH NEXT FROM cur_compras into @cliente_num, @cantidad, @monto
    END
    CLOSE CUR_compras
    DEALLOCATE CUR_compras
end

Execute sumaCompras
select * from #comprasClientes
```

4.

```
drop procedure actualizaPrecios

create procedure actualizaPrecios as
begin
   DROP table ##ListaPrecios
   create table ##ListaPrecios (
                                    producto_cod smallint,
                                    producto_desc varchar(30),
                                    precio_unit decimal(12,2)
               );

       insert into ##ListaPrecios (producto_cod, producto_desc, precio_unit)
              (select p.producto_cod, p.producto_desc,
                     case when exists (select 1 from facturas_det d
                                         where d.producto_cod = p.producto_cod) then
                          CASE when p.precio_unit < 150 then p.precio_unit * 1.1
                               else p.precio_unit * 1.2
                           end
                      else
                           p.precio_unit
                      end nuevo_precio
                from productos p           );
end

execute actualizaPrecios

select * from ##ListaPrecios
```

5.

```
drop table #fabricantesNuevos

select *
  into #fabricantesNuevos
  from fabricantes
 where 1=2;

 insert into #fabricantesNuevos (fabricante_cod, fabricante_nom, tiempo_entrega,
provincia_cod)
      values ('XERO', 'Xerox Co', null, 'TF'),
                ('PATO', 'Patogenos Sa', -2, 'TF'),
                      ('GRUN', 'Grundig Srl', 6, 'ZZ'),
                      ('LACO', 'Loco', 6, 'SC'),
                      ('OSLO', 'Otorrino Sca', 2, 'RN')

drop procedure insertaFabNuevos

create procedure insertaFabNuevos as
begin

   declare @fabricante_cod varchar(5), @fabricante_nom varchar(20)
   declare @tiempo_entrega smallint, @provincia_cod char(2)

   declare c_nuevos CURSOR FOR select fabricante_cod, fabricante_nom, tiempo_entrega,
provincia_cod
```

```
                                from #fabricantesNuevos;
      OPEN c_nuevos
    FETCH NEXT FROM c_nuevos into @fabricante_cod, @fabricante_nom,
                                    @tiempo_entrega, @provincia_cod
      BEGIN TRY
    BEGIN TRANSACTION
    WHILE (@@FETCH_STATUS = 0)
    BEGIN
          if exists (select 1 from fabricantes
                        where fabricante_cod = @fabricante_cod)
             begin
                  print @fabricante_cod;
            THROW 50000, 'Error. El fabricante ya existe', 1
             end

          if @tiempo_entrega < 0
            THROW 50001, 'Error. Tiempo de entrega negativo', 2

             if not exists (select 1 from provincias
                            where provincia_cod = @provincia_cod)
            THROW 50002, 'Error. Provincia inexistente', 1

        insert into fabricantes (fabricante_cod, fabricante_nom , tiempo_entrega,
provincia_cod)
                    values (@fabricante_cod, @fabricante_nom, @tiempo_entrega,
@provincia_cod)
         FETCH NEXT FROM c_nuevos into @fabricante_cod, @fabricante_nom,
                                        @tiempo_entrega, @provincia_cod
      END
      CLOSE C_nuevos
    DEALLOCATE C_nuevos
      commit
      end try
      begin catch
            CLOSE C_nuevos
       DEALLOCATE C_nuevos
       rollback
            print 'Nro. Error:' + cast(ERROR_NUMBER() as varchar);
         print 'Mensaje:' +  ERROR_MESSAGE();
         print 'Status:' +  cast(ERROR_STATE() as varchar);
      end catch
end

select * from #fabricantesNuevos

execute insertaFabNuevos


SELECT fabricante_cod, fabricante_nom, tiempo_entrega, provincia_cod
  FROM fabricantes order by 1

-- delete fabricantes
 -- where fabricante_cod in ('XERO', 'PATO', 'GRUN', 'OSLO')


6.
```

```
drop table #fabricantesNuevos

select *
  into #fabricantesNuevos
  from fabricantes
 where 1=2;

 insert into #fabricantesNuevos (fabricante_cod, fabricante_nom, tiempo_entrega,
provincia_cod)
      values ('XERO', 'Xerox Co', null, 'TF'),
                ('PATO', 'Patogenos Sa', -2, 'TF'),
                      ('GRUN', 'Grundig Srl', 6, 'ZZ'),
                      ('LACO', 'Lacolina', 6, 'SC'),
                      ('OSLO', 'Otorrino Sca', 2, 'RN')

drop procedure insertaFabNuevos2

create procedure insertaFabNuevos2 as
begin

   declare @fabricante_cod varchar(5), @fabricante_nom varchar(20)
   declare @tiempo_entrega smallint, @provincia_cod char(2)

   declare c_nuevos CURSOR FOR select fabricante_cod, fabricante_nom, tiempo_entrega,
provincia_cod
                               from #fabricantesNuevos;
       OPEN c_nuevos
    FETCH NEXT FROM c_nuevos into @fabricante_cod, @fabricante_nom,
                                  @tiempo_entrega, @provincia_cod

    WHILE (@@FETCH_STATUS = 0)
    BEGIN
             BEGIN TRY
        BEGIN TRANSACTION
           if exists (select 1 from fabricantes
                      where fabricante_cod = @fabricante_cod)
             THROW 50000, 'Error. El fabricante ya existe', 1


           if @tiempo_entrega < 0
             THROW 50001, 'Error. Tiempo de entrega negativo', 2

              if not exists (select 1 from provincias
                             where provincia_cod = @provincia_cod)
             THROW 50002, 'Error. Provincia inexistente', 1

        insert into fabricantes (fabricante_cod, fabricante_nom , tiempo_entrega,
provincia_cod)
                  values (@fabricante_cod, @fabricante_nom, @tiempo_entrega,
@provincia_cod)

              commit
       end try
           begin catch
            rollback
                 print 'Nro. Error:' + cast(ERROR_NUMBER() as varchar);
              print 'Mensaje:' +  ERROR_MESSAGE();
              print 'Status:' +  cast(ERROR_STATE() as varchar);
```

```
            end catch
               FETCH NEXT FROM c_nuevos into @fabricante_cod, @fabricante_nom,
                                    @tiempo_entrega, @provincia_cod

        END
        CLOSE C_nuevos
    DEALLOCATE C_nuevos
end

select * from #fabricantesNuevos

execute insertaFabNuevos2

SELECT TOP (1000) [fabricante_cod]
      ,[fabricante_nom]
      ,[tiempo_entrega]
      ,[provincia_cod]
  FROM [UADE1].[dbo].[fabricantes]
  order by 1

 delete fabricantes
  where fabricante_cod in ('XERO', 'PATO', 'GRUN', 'OSLO')
```

7.

```
alter procedure borraFactura (@nroFactura int) as
begin
begin try
    begin tran
    insert into auditFC (factura_num, fecha_emision, cliente_num,
                            cantRenglones, montoTotal, cantProductos)
          (select f.factura_num, fecha_emision, cliente_num,
                       count(*), sum(cantidad*precio_unit), sum(cantidad)
                  from facturas f join facturas_det d on f.factura_num = d.factura_num
                  where f.factura_num = @nroFactura
                  group by f.factura_num, fecha_emision, cliente_num);
       --
       delete facturas_det where factura_num = @nroFactura;
       delete facturas where factura_num = @nroFactura;
       commit
       print 'COMMIT'
end try
begin catch
    rollback
       print 'ROLLBACK'
       insert into erroresFC (parametro, fecha, error)
              values (@nroFactura, getDate(), ERROR_MESSAGE())
end catch
end
select * from erroresFC
select * from auditFC

exec borraFactura 9999
```