1. Crear un procedimiento totalCompraPr al cual se le envíe como parámetro un número de cliente “desde” y un número de cliente “hasta”. El procedimiento deberá calcular y guardar en la tabla ComprasHist el número de cliente y el monto total comprado de cada cliente. Si el cliente ya existe, registrar en una tabla de errores el número de cliente y el mensaje de error “Cliente ya existente” y continuar con el procesamiento de los demás clientes. Devolver por pantalla un total de control que indique la cantidad de clientes leídos, la cantidad de clientes grabados y la cantidad de erróneos.

DROP table ComprasHist;

Create table ComprasHist (

cliente\_num int,

total decimal(12,2)

);

DROP table ComprasError

Create table ComprasError (

cliente\_num int,

mensaje varchar(20)

);

ALTER procedure totalCompraPr (@clienteDde int, @clienteHta int) As

begin

--

declare compras\_c cursor local for

select c.cliente\_num, sum(precio\_unit \* cantidad) total

from clientes c join facturas f on c.cliente\_num = f.cliente\_num

join facturas\_det d on d.factura\_num = f.factura\_num

where @clienteDde <= c.cliente\_num and c.cliente\_num <= @clienteHta

group by c.cliente\_num;

--

declare @cliente\_num int, @total int

declare @clientesTot int, @clientesOk int, @clientesError int

--

Set @clientesTot=0

set @clientesOk=0

set @clientesError= 0

--

OPEN compras\_c

fetch compras\_c into @cliente\_num, @total

--

while @@FETCH\_STATUS = 0

begin

begin tran

begin try

set @clientesTot += 1

if exists (select 1 from comprasHist where cliente\_num = @cliente\_num)

THROW 50001, 'Cliente repetido', 1

insert into comprasHist (cliente\_num, total)

values (@cliente\_num, @total);

set @clientesOk += 1

end try

begin catch

if ERROR\_NUMBER() != 50001

begin

rollback;

THROW

end

insert into ComprasError (cliente\_num, mensaje)

values (@cliente\_num, ERROR\_MESSAGE());

set @clientesError += 1

end catch

commit

fetch compras\_c into @cliente\_num, @total

end

CLOSE compras\_c

DEALLOCATE compras\_c

print 'Total Clientes:' + cast(@clientesTot as varchar)

print 'Cliente Ok:' + cast(@clientesOk as varchar)

print 'Clientes error:' + cast(@clientesError as varchar)

--

end

1. Seleccionar número de cliente, nombre, monto total comprado, y número, nombre y monto total comprado de un cliente 2, y comparándolos y mostrándolos de a pares. El monto del primer cliente deberá ser mayor al del segundo. Mostrar la consulta ordenada por número de cliente y monto del segundo cliente.

select c1.cliente\_num, c1.nombre, c1.apellido, sum(d1.precio\_unit \* d1.cantidad) total1,

c2.cliente\_num, c2.nombre, c2.apellido, c2.total2

from clientes c1 join facturas f1 on c1.cliente\_num = f1.cliente\_num

join facturas\_det d1 on d1.factura\_num = f1.factura\_num

, (select c2.cliente\_num, c2.nombre, c2.apellido, sum(d2.precio\_unit \* d2.cantidad) total2

from clientes c2 join facturas f2 on c2.cliente\_num = f2.cliente\_num

join facturas\_det d2 on d2.factura\_num = f2.factura\_num

group by c2.cliente\_num, c2.nombre, c2.apellido ) c2

where c1.cliente\_num != c2.cliente\_num

group by c1.cliente\_num, c1.nombre, c1.apellido,

c2.cliente\_num, c2.nombre, c2.apellido, c2.total2

having sum(d1.precio\_unit \* d1.cantidad) > c2.total2

order by 1, total2 desc

1. Se requiere listar para la provincia de Buenos Aires el par de clientes que sean los que suman el mayor monto facturado, con el formato de salida:

'Nombre Provincia', 'Apellido, Nombre', 'Apellido, Nombre', 'Total Solicitado' (\*)

(\*) El total solicitado contendrá la suma de los dos clientes.

SELECT TOP 1 provincia\_desc, c1.nombre+', '+c1.apellido, c2.nombre+', '+c2.apellido

, totcli1 + totcli2 'Total'

FROM provincias p JOIN clientes c1 ON (p.provincia\_cod = c1.provincia\_cod)

JOIN clientes c2 ON (p.provincia\_cod = c2.provincia\_cod)

JOIN (SELECT f1.cliente\_num, SUM(precio\_unit\*cantidad) totcli1

FROM facturas f1 JOIN facturas\_det d1

ON (f1.factura\_num = d1.factura\_num)

GROUP BY f1.cliente\_num) totc1

ON (c1.cliente\_num = totc1.cliente\_num)

JOIN (SELECT f2.cliente\_num, SUM(precio\_unit\*cantidad) totcli2

FROM facturas f2 JOIN facturas\_det d2

ON (f2.factura\_num = d2.factura\_num)

GROUP BY f2.cliente\_num) totc2

ON (c2.cliente\_num = totc2.cliente\_num)

WHERE c1.cliente\_num > c2.cliente\_num AND p.provincia\_desc = 'Buenos Aires'

ORDER BY 4 DESC

4.

Seleccionar los clientes de mayor monto comprador por cada provincia.

SELECT provincia\_cod, c.cliente\_num, sum(precio\_unit\*cantidad) total

from clientes c join facturas o on c.cliente\_num = o.cliente\_num

join facturas\_det i on o.factura\_num = i.factura\_num

WHERE c.cliente\_num = (SELECT top 1 c2.cliente\_num

from clientes c2 join facturas o2 on c2.cliente\_num = o2.cliente\_num

join facturas\_det i2 on o2.factura\_num = i2.factura\_num

WHERE c2.provincia\_cod = c.provincia\_cod

group by c2.cliente\_num

order by sum(i2.precio\_unit\*i2.cantidad) DESC)

GROUP BY provincia\_cod, c.clientE\_num

ORDER by 1, 3 desc, 2

5.

Crear un trigger que ante cambios (inserts, borrados, modificaciones) en las filas factura\_detalle de una factura en particular, mantenga un atributo montoTOTAL con el monto correcto (p x q) en dicha factura.

ALTER trigger mantieneTotal on facturas\_det

AFTER INSERT, DELETE, UPDATE AS

begin

--

update facturas

set montoTotal = (select sum(d.precio\_unit \* d.cantidad)

from facturas\_det d

where d.factura\_num = f.factura\_num)

from facturas f

where f.factura\_num in (select distinct(coalesce(i.factura\_num, d.factura\_num))

from inserted i FULL OUTER join deleted d

on i.factura\_num = d.factura\_num and

i.renglon = d.renglon)

--

end

6.

Crear un trigger que ante compras efectuadas por los clientes, valide lo siguiente:

Que los fabricantes de CABA o Buenos Aires puedan vender a clientes de todo el país excepto a los de Tierra del Fuego.

Que los demás fabricantes de otras provincias solo puedan vender mercaderías a clientes de las mismas provincias.

Asuma que las operaciones son de una misma factura que puede tener varios renglones.

ALTER trigger controlaProvinciaVta on facturas\_det

AFTER INSERT AS

begin

--

Declare @nro\_factura int, @cliente\_num int, @provincia\_cli char(2)

Declare @provincia\_fab char(2)

Select distinct @provincia\_cli = c.provincia\_cod

from INSERTED I JOIN facturas f ON i.factura\_num = f.factura\_num

JOIN clientes c ON c.cliente\_num = f.cliente\_num;

Declare detalle\_cur CURSOR LOCAL FOR

Select f.provincia\_cod

from INSERTED I JOIN productos p ON i.producto\_cod = p.producto\_cod

JOIN fabricantes f ON f.fabricante\_cod = p.fabricante\_cod

OPEN detalle\_cur

FETCH detalle\_cur INTO @provincia\_fab

While @@FETCH\_STATUS=0

begin

if @provincia\_fab NOT IN ('CF', 'BA')

begin

if @provincia\_fab != @provincia\_cli

THROW 50001, 'Venta invalida', 1

end

else

if @provincia\_cli = 'TF'

THROW 50001, 'Venta invalida', 1

--

FETCH detalle\_cur INTO @provincia\_fab

end

CLOSE Detalle\_cur

DEALLOCATE detalle\_cur

end