-- scenario # 1

create procedure archiveorders @minimalage int

as

begin

begin transaction

set identity\_insert archivedorders on

insert into archivedorders (orderid, customerid, employeeid, orderdate,

requireddate, shippeddate, shipvia, freight, shipname, shipaddress, shipcity,

shipregion, shippostalcode, shipcountry, archivedata)

select \*, getdate()

from orders

where datediff(yy, orderdate, getdate()) > @minimalage

insert into archivedorderdetails

select od.\*

from [order details] as od

join orders as o on o.orderid=od.orderid

where datediff(yy, orderdate, getdate()) > @minimalage

delete from [order details]

where orderid in (

select orderid

from orders

where datediff(yy, orderdate, getdate()) > @minimalage

)

delete from orders

where datediff(yy, orderdate, getdate()) > @minimalage

set identity\_insert archivedorders off

commit

end

execute archiveorders 18

select \* from archivedorders

select \* from orders

-- scenario # 2

alter procedure rabaty

as

begin

declare @klient nchar(5)

declare @product int

declare @liczbazamowien int

declare k cursor local for select customerid from customers

open k

fetch next from k into @klient

while @@fetch\_status=0

begin

declare p cursor local for (

select productid

from [order details] as od

join orders as o on o.orderid=od.orderid

where o.customerid=@klient

)

open p

fetch next from p into @product

while @@fetch\_status=0

begin

set @liczbazamowien = (

select count(\*)

from orders as o

join [order details] as od on od.orderid=o.orderid

where o.customerid=@klient and od.productid=@product

)

if @liczbazamowien=0

begin

update [order details]

set discount=0

from [order details] as od

join orders as o on od.orderid=o.orderid

where o.customerid=@klient and od.productid=@product

end

if @liczbazamowien in (1,2)

begin

update [order details]

set discount=0.05

from [order details] as od

join orders as o on od.orderid=o.orderid

where o.customerid=@klient and od.productid=@product

end

if @liczbazamowien=3

begin

update [order details]

set discount=0.1

from [order details] as od

join orders as o on od.orderid=o.orderid

where o.customerid=@klient and od.productid=@product

end

if @liczbazamowien>=4

begin

update [order details]

set discount=0.2

from [order details] as od

join orders as o on od.orderid=o.orderid

where o.customerid=@klient and od.productid=@product

end

fetch next from p into @product

end

close p

deallocate p

fetch next from k into @klient

end

close k

deallocate k

end

execute rabaty

select \* from [Order Details]

-- scenario # 1

create trigger insertdiscount on [order details] after insert

as

begin

update [order details] set discount=0.1234

where orderid=(

select orderid

from inserted

) and productid=(

select productid

from inserted

)

end

-- albo raczej tak:

alter trigger insertdiscount on [order details] after insert

as

begin

execute rabaty

end

-- sprawdzenie:

delete from [order details]

where orderid=10248

insert into [order details] (orderid, productid) values(10248,56)

select \* from [order details]

-- a teraz dla insertow wielu wierszy:

create trigger insertdiscountwiele on [order details] after insert

as

begin

declare @ordid int

declare @prodid int

declare i cursor local for select orderid, productid from inserted

open i

fetch next from i into @ordid, @prodid

while @@fetch\_status=0

begin

update [order details] set discount=0.1234

where orderid=@ordid and productid=@prodid

fetch next from i into @ordid, @prodid

end

close i

deallocate i

end

-- sprawdzenie:

delete from [order details]

where orderid=10248 and orderid=10248

insert into [order details] (orderid, productid) values(10248,56), (10248,55), (10249,1)

select \* from [order details]

-- a tak, to juz w ogole najlepiej:

alter trigger insertdiscountwiele on [order details] after insert

as

begin

declare @oid int

declare @pid int

declare @liczzam int

declare i cursor local for select orderid, productid from inserted

open i

fetch next from i into @oid, @pid

while @@fetch\_status=0

begin

set @liczzam = (

select count(\*)

from [order details]

where orderid=@oid and productid=@pid

)

if @liczzam=1 or @liczzam=2

begin

update [order details] set discount=0.05

where orderid=@oid and productid=@pid

end

else if @liczzam=3

begin

update [order details] set discount=0.1

where orderid=@oid and productid=@pid

end

else if @liczzam>4

begin

update [order details] set discount=0.2

where orderid=@oid and productid=@pid

end

else

begin

update [order details] set discount=0

where orderid=@oid and productid=@pid

end

fetch next from i into @oid, @pid

end

close i

deallocate i

end

delete from [order details]

where orderid in (10248,10249)

insert into [order details] (orderid, productid) values(10248,56), (10248,55), (10249,1)

select \* from [order details]

-- cos innego:

alter trigger insertdiscount2 on [order details] instead of insert

as

begin

declare @oid int

declare @pid int

declare @liczzam int

declare @up int

declare @q int

set @oid=(select orderid from inserted)

set @pid=(select productid from inserted)

set @up=(select unitprice from inserted)

set @q=(select quantity from inserted)

set @liczzam=(

select count(\*)

from [order details] as od

join orders as o on o.orderid=od.orderid

where o.customerid=(

select customerid

from orders

where orderid=@oid

) and od.productid=@pid

)

if @liczzam=1 or @liczzam=2

begin

insert into [order details] (orderid, productid, unitprice, quantity, discount)

values (@oid, @pid, @up, @q, 0.05)

end

-- tu trzeba jeszcze dopisac ten milion innych warunkow...

end

-- zad4 (w nawiązaniu do zadania 3)

-- a) Dodaj kolumnę Bonus typu money w tab. Employees.

-- b) Napisz wyzwalacz, który po dodaniu nowego zamówienia zaktualizuje wysokość premii,

-- przy czym wartość premii powinna zostać zmieniona tylko w przypadku, gdy zachodzi

-- taka konieczność

--

CREATE TRIGGER updBonus ON Orders AFTER INSERT

AS

BEGIN

DECLARE @empl int

DECLARE @lzam int

DECLARE k1 CURSOR LOCAL FOR SELECT employeeID FROM inserted

OPEN k1

FETCH NEXT FROM k1 INTO @empl

WHILE @@FETCH\_STATUS=0

BEGIN

SET @lzam=(SELECT COUNT(\*) FROM Employees e

JOIN Orders o ON o.EmployeeID=e.EmployeeID

WHERE e.EmployeeID=@empl)

IF @lzam=1 -- pamietac o pracowniku, ktory nie mial zamowien !!!

BEGIN

UPDATE Employees SET Bonus=100

WHERE EmployeeID=@empl

END

ELSE IF @lzam=51

BEGIN

UPDATE Employees SET Bonus=150

WHERE EmployeeID=@empl

END

ELSE IF @lzam=101

BEGIN

UPDATE Employees SET Bonus=250

WHERE EmployeeID=@empl

END

FETCH NEXT FROM k1 INTO @empl

END

CLOSE k1

DEALLOCATE k1

END