T-SQL

views, procedures, triggers

Variables

```
declare @i int, @j int
declare @first varchar(50)
declare @last varchar(50)

set @i = 10
set @j = @i + 5
set @first = 'john'
select @last = 'gold'

print @i
print @j
print @first
print @last
```

IF

```
declare @i int

set @i = 1

if @i > 2
  print 'i is > 2'
else
  print 'i is <= 2'
```

IF c.d.

```
declare @i int

set @i = 1

if @i > 2
begin
  print 'i is > 2'
  select * from employees
end
else
begin
  print 'i is <= 2'
  select * from orders
end
```

WHILE

```
declare @j int = 1

while @j <= 5
begin
  print @j
  set @j = @j + 1
end
```

WHILE c.d.

```
declare @j int = 1

while @j <= 5
begin
  print @j
  select lastname from employees where employeeid = @j
  set @j = @j + 1
end
```

Variables c.d.

```
declare @first varchar(50)
declare @last varchar(50)

set @first = (select firstname from employees where employeeid = 1)
print @first

select @last = lastname from employees where employeeid = 1
print @last

select @first = firstname, @last = lastname  from employees where employeeid = 2
print @first + ' ' + @last
```

Variables c.d.

- What if `select` returns several rows?

```
set @first = (select firstname from employees where employeeid > 1)
print @first
```

> Error
> [S0001][512] Line 3: Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=, <, <= , >, >= or when the subquery is used as an expression.

```
select @last = lastname from employees where employeeid > 1
print @last
```

-The variable will adopt the value from the last row processed by the command `select * from employees where employeeid > 1`

CURSOR

```
declare @id int
declare @last varchar(50)

declare emp_cursor cursor
    for select employeeid, lastname from employees
open emp_cursor
fetch next from emp_cursor into @id, @last

close emp_cursor
deallocate emp_cursor

print @id
print @last
```

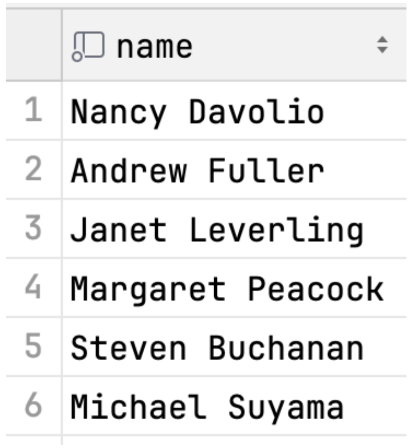
CURSOR c.d.

```
declare @sum decimal(10,2) = 0, @qty decimal(10,2), @price decimal(10,2)

declare od_cursor cursor
    for select unitprice, quantity from [order details] where orderid = 10250
open od_cursor

fetch next from od_cursor into @qty, @price

if @@FETCH_STATUS <> 0
    print 'empty set'

while @@FETCH_STATUS = 0
begin
    print cast(@qty as varchar) + ', ' + cast(@price as varchar)
    set @sum = @sum + @qty * @price
    fetch next from od_cursor into @qty, @price
end

close od_cursor
deallocate od_cursor

print @sum
```

Views

```
create view emp_names
as
select firstname + ' ' + lastname as name
from employees
```

```
select * from emp_names
```



	name
1	Nancy Davolio
2	Andrew Fuller
3	Janet Leverling
4	Margaret Peacock
5	Steven Buchanan
6	Michael Suyama

Views c.d.

- Modification

```
alter view emp_name
as
select employeeid, firstname + ' ' + lastname as name
from employees
```

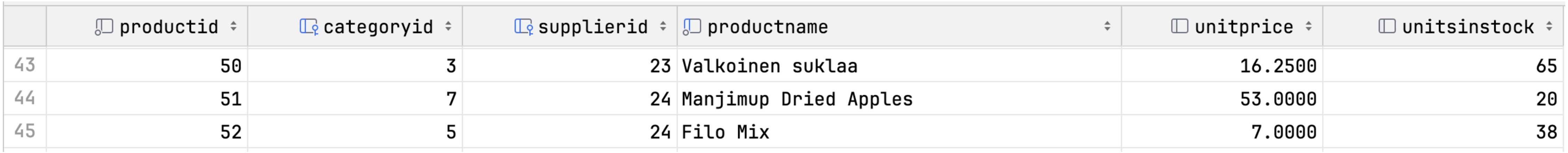
- Removal

```
drop view emp_name
```

Views c.d.

```
create view product
as
select productid, categoryid, supplierid, productname, unitprice, unitsinstock
from products
where discontinued = 0
```

```
select * from product
```

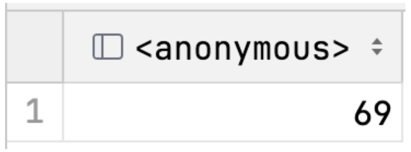


	productid	categoryid	supplierid	productname	unitprice	unitsinstock
43	50	3	23	Valkoinen suklaa	16.2500	65
44	51	7	24	Manjimup Dried Apples	53.0000	20
45	52	5	24	Filo Mix	7.0000	38

Views c.d.

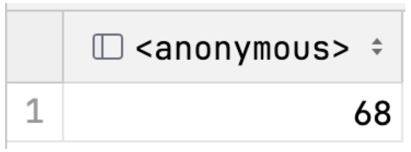
```sql
create view avail_product
as
select productid, categoryid, supplierid, productname, unitprice, unitsinstock
from product
where unitsinstock > 0
```

```sql
select count(*) from product
```



	<anonymous>
1	69

```sql
select count(*) from avail_product
```

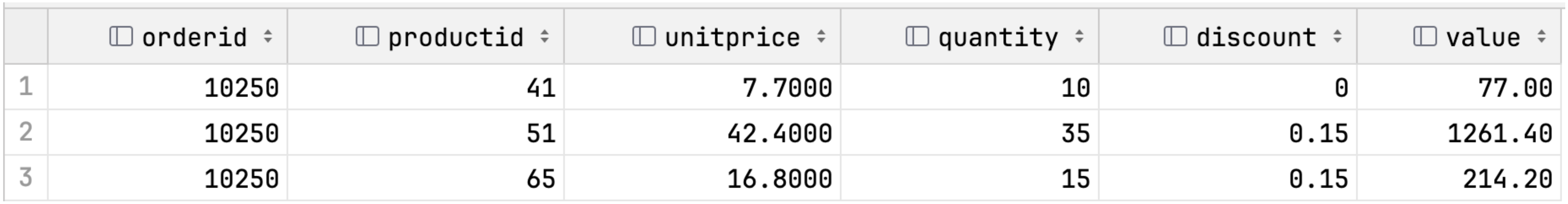


	<anonymous>
1	68

Views c.d.

```sql
create view order_detail
as
select orderid, productid, unitprice, quantity, discount,
       cast(unitprice * quantity * (1-discount) as decimal(10,2)) as value
from [Order Details]
```

```sql
select * from order_detail
where orderid = 10250
```

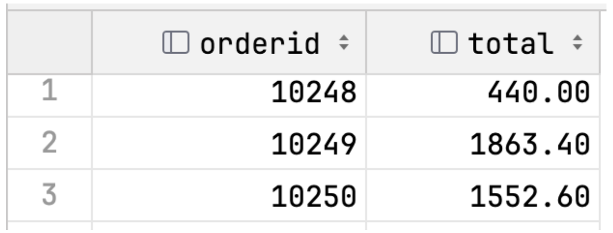


	orderid	productid	unitprice	quantity	discount	value
1	10250	41	7.7000	10	0	77.00
2	10250	51	42.4000	35	0.15	1261.40
3	10250	65	16.8000	15	0.15	214.20

Views c.d.

```
create view order_total_1
as
select orderid, cast(sum(unitprice * quantity * (1-discount)) as decimal(10,2)) as total
from [Order Details]
group by orderid
```

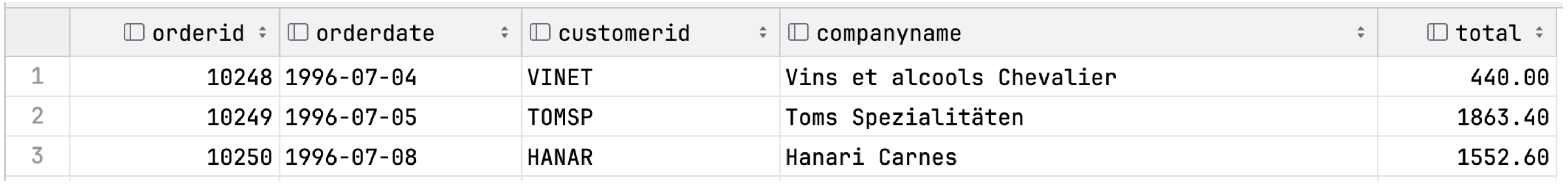
```
create view order_total_2
as
select orderid, sum(value) as total
from order_detail
group by orderid
```



	orderid	total
1	10248	440.00
2	10249	1863.40
3	10250	1552.60

Views c.d.

```
alter view order_total_3
as
select o.orderid, cast(o.orderdate as date) orderdate, o.customerid, c.companyname,
       sum(value) as total
from order_detail od join orders o on od.orderid = o.orderid
join customers c on o.customerid = c.customerid
group by o.orderid, o.orderdate,  o.customerid, c.companyname
```



	orderid	orderdate	customerid	companyname	total
1	10248	1996-07-04	VINET	Vins et alcools Chevalier	440.00
2	10249	1996-07-05	TOMSP	Toms Spezialitäten	1863.40
3	10250	1996-07-08	HANAR	Hanari Carnes	1552.60

Procedures

```
create proc p_customer_order_total
@customerid char(5)
as
begin
    if not exists (select * from customers where customerid = @customerid)
       throw 50001, 'No customer with such id', 1

    select orderid, orderdate, total
    from order_total_3
    where customerid = @customerid
end
```

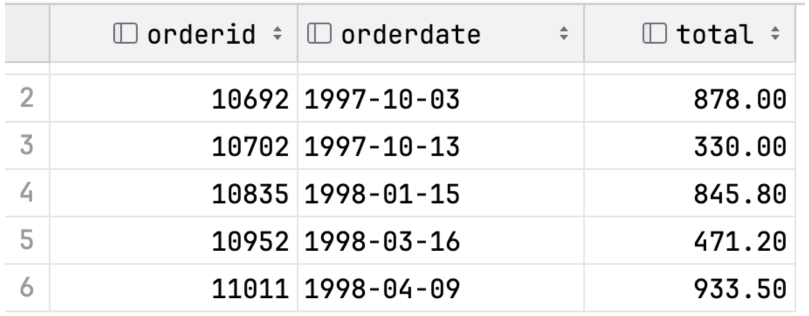
- `alter proc` - modification
- `drop proc` - removal

Procedures c.d.

```sql
exec p_customer_order_total 'ala'
```

> Error
> [S0001][50001] Line 6: No customer with such id

```sql
exec p_customer_order_total 'ALFKI'
```



	orderid	orderdate	total
2	10692	1997-10-03	878.00
3	10702	1997-10-13	330.00
4	10835	1998-01-15	845.80
5	10952	1998-03-16	471.20
6	11011	1998-04-09	933.50

```sql
exec p_customer_order_total 'PARIS'
```

- emoty result set

Procedures c.d.

```
create proc p_customer_order_total_2
@customerid char(5), @start_date date, @end_date date
as
begin
    if @start_date > @end_date
       throw 50001, 'wrong date rangel', 1
    if not exists (select * from customers where customerid = @customerid)
       throw 50001, 'No customer with such id', 1

    select orderid, orderdate, total
    from order_total_3
    where customerid = @customerid
          and orderdate >= @start_date
          and orderdate <= @end_date
end
```

```
exec p_customer_order_total_2 'ALFKI', '1997-01-01', '1997-12-31'
```

Functions

```
create function f_customer_order_total (@customerid char(5))
returns table
as return (
    select orderid, orderdate, customerid, total
    from order_total_3
    where customerid = @customerid
)
```

- `alter func` - modyfikacja
- `drop func` - usunięcie

Functions c.d.

```sql
select * from f_customer_order_total('ALFKI')
```

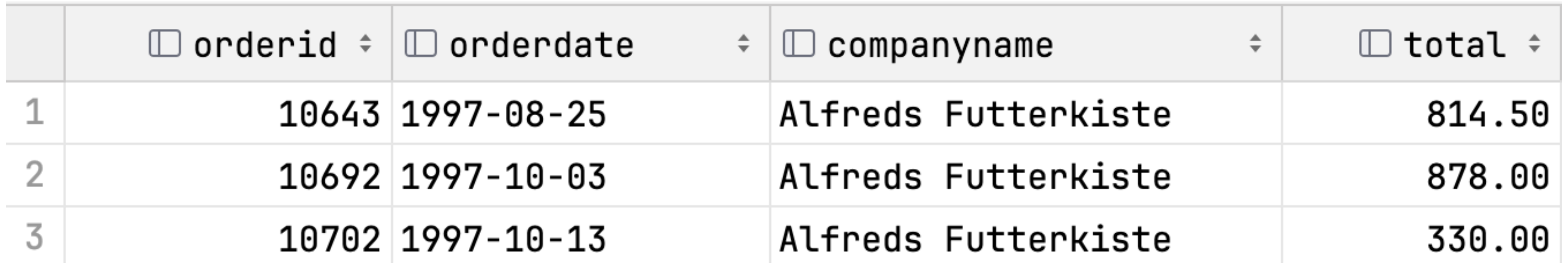
	orderid	orderdate	customerid	total
1	10643	1997-08-25	ALFKI	814.50
2	10692	1997-10-03	ALFKI	878.00
3	10702	1997-10-13	ALFKI	330.00

```sql
select sum(total) from f_customer_order_total('ALFKI')
```

	<anonymous>
1	4273.00

Functions c.d.

```
select f.orderid, f.orderdate, c.companyname, f.total
from f_customer_order_total('ALFKI') f
join customers c on f.customerid = c.customerid
```



	orderid	orderdate	companyname	total
1	10643	1997-08-25	Alfreds Futterkiste	814.50
2	10692	1997-10-03	Alfreds Futterkiste	878.00
3	10702	1997-10-13	Alfreds Futterkiste	330.00

- Attention:
    - such statement could be ineffective

Functions c.d.

```
create function f_customer_order_total_2(@customerid char(5))
returns @result table (orderid int, orderdate date, customerid char(5), total decimal(10,2))
as
begin
   insert @result
   select orderid, orderdate, customerid, total
   from order_total_3
   where customerid = @customerid

   return
end
```

Scalar functions

```sql
create function f_max(@a int, @b int)
returns int
as
begin
  declare @r int
  if @a > @b set @r = @a else set @r = @b
  return @r
end
```

```sql
select dbo.f_max(1,5)
```

Scalar functions c.d.

```
create function f_customer_name(@customerid char(5))
returns varchar(100)
as
begin
    declare @companyname varchar(100)

    select @companyname = companyname
    from customers
    where customerid = @customerid

    return @companyname
end
```

```
select orderid, customerid, dbo.f_customer_name(customerid)
from orders
order by orderid
```

Scalar functions c.d.

```sql
create function order_detail_total(@orderid int)
returns decimal(10,2)
as
begin
    declare @total decimal(10,2)

    select @total = sum(unitprice * quantity * (1-discount))
    from [Order Details]
    where orderid = @orderid

    return @total
end
```

```sql
select orderid, freight + dbo.order_detail_total(orderid)
from orders
```

Procedures

```sql
create procedure add_order
@customerid char(5),
@employeeid int,
@requireddate date
as
begin
    declare @orderdate date = getdate()
    insert orders(customerid,employeeid,orderdate,requireddate)
    values(@customerid,@employeeid,@orderdate,@requireddate)
end
```

Procedures c.d.

```
declare @requireddate date = dateadd(day, 7, getdate())

exec add_order 'ALFKI', 1, @requireddate
```

- OK

```
declare @requireddate date = dateadd(day, 7, getdate())

exec add_order 'ala', 1, @requireddate
```

> Error
> [23000][547] Line 8: The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Orders_Customers". The conflict occurred in database "Northwind_m", table "dbo.Customers", column 'CustomerID'.

Procedures c.d.

```
alter procedure add_order
@customerid char(5), @employeeid int, @requireddate date
as
begin
    if not exists (select * from customers where customerid = @customerid)
       throw 50001, 'No customer with such id', 1

    declare @orderdate date = getdate()
    insert orders(customerid,employeeid,orderdate,requireddate)
    values(@customerid,@employeeid,@orderdate,@requireddate)
end
```

```
exec add_order 'ala', 1, @requireddate
```

> Error
> [S0001][50001] Line 6: No customer with such id

Procedures c.d.

```
create procedure add_detail
@orderid int, @productid int, @quantity int, @discount decimal(3,2)
as
begin
    declare @unitprice decimal(10,2)

    select @unitprice = unitprice from products where productid = @productid

    insert [Order Details](orderid, productid, unitprice, quantity, discount)
    values(@orderid, @productid, @unitprice, @quantity, @discount)
end
```

Procedures c.d.

```
exec add_detail 11081, 1, 10, 0.12
```

- OK (with assumption, that exists order with id = `11081`)

```
exec add_detail 11081, 999, 10, 0.12
```

> Error
> [23000][515] Line 10: Cannot insert the value NULL into column 'UnitPrice', table 'Northwind_m.dbo.Order Details'; column does not allow nulls. INSERT fails.

- other problems
  - product might exist, but can be discontinued
  - there could be no product in the stock
  - after adding product ot the order, the value of `unitsonstock` should be decreased

Procedury c.d.

```
alter procedure add_detail
@orderid int, @productid int, @quantity int, @discount decimal(3,2)
as
begin
    declare @unitprice decimal(10,2)

    select @unitprice = unitprice from avail_product where productid = @productid

    insert [Order Details](orderid, productid, unitprice, quantity, discount)
    values(@orderid, @productid, @unitprice, @quantity, @discount)
end
```

- use of the `avail_product` view

```
exec add_detail 11081, 31, 10, 0.12
```

- product with id = 31 is not available, the `null` value will be assigned to the `@unitprice` variable

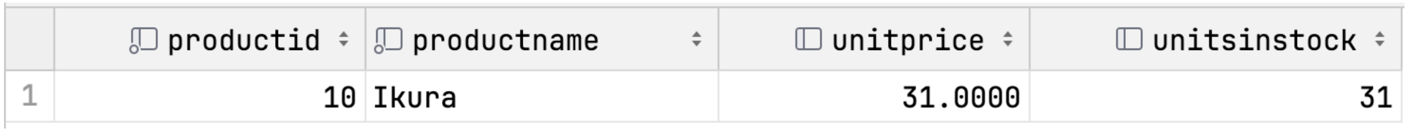
> Error
> [23000][515] Line 10: Cannot insert the value NULL into column 'UnitPrice', table 'Northwind_m.dbo.Order Details'; column does not allow nulls. INSERT fails.

Procedures c.d.

```
alter procedure add_detail
@orderid int, @productid int, @quantity int, @discount decimal(3,2)
as
begin
    begin try
       begin transaction

       if not exists (select * from avail_product
                      where productid = @productid and avail_product.unitsinstock >= @quantity)
           throw 5003, 'No such product', 1

       declare @unitprice decimal(10,2)
       select @unitprice = unitprice from avail_product where productid = @productid

       insert [Order Details](orderid, productid, unitprice, quantity, discount)
       values(@orderid, @productid, @unitprice, @quantity, @discount)

       update products
       set unitsinstock = unitsinstock - @quantity
       where productid = @productid

       commit
    end try
    begin catch
        rollback        ;
        throw
    end catch
```

Procedures c.d.

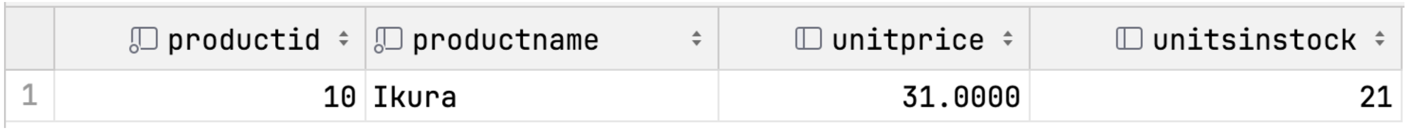
```
select productid, productname, unitprice, unitsinstock
from avail_product where productid = 10
```



	productid	productname	unitprice	unitsinstock
1	10	Ikura	31.0000	31

```
exec add_detail 11081, 10, 10, 0.12
```

```
select productid, productname, unitprice, unitsinstock
from avail_product where productid = 10
```



	productid	productname	unitprice	unitsinstock
1	10	Ikura	31.0000	21

Procedures c.d.

```
declare @requireddate date = dateadd(day, 7, getdate())

exec add_order 'ALFKI', 1, @requireddate

exec add_detail @@identity, 10, 25, 0.12
```

> Error
> [S0001][50003] Line 10: No such product

- Of course, the `add_detail` procedure still requires refinement
  - e.g. adding error messages
  - e.g. increasing the value of `unitsonorder`

Procedures c.d.

```sql
create procedure add_order2
alter procedure add_order2
@customerid char(5), @employeeid int, @requireddate date,
@productid int, @quantity int, @discount decimal(3,2)
as
begin
   begin try
       begin transaction

       exec add_order 'ALFKI', 1, @requireddate

       exec add_detail @@identity, @productid, @quantity, @discount

       commit
   end try
   begin catch
       if @@trancount > 1
          rollback
       ;throw
   end catch
end
```

Procedures c.d.

```sql
declare @requireddate date = dateadd(day, 7, getdate())

exec add_order2 'ALFKI', 1, @requireddate, 10, 25, 0.12
```

> Error
> [S0001][50003] Line 10: No such product

- there are not enough units of the product with id = `10` in the stock

```sql
declare @requireddate date = dateadd(day, 7, getdate())

exec add_order2 'ALFKI', 1, @requireddate, 11, 5, 0.12
```

- OK

Triggery

Triggers are called automatically `after` data modification

- `insert`
- `update`
- `delete`

Trigger `instead of`

Access to changed values with tables

- `inserted`
- `deleted`

Triggers - example

```
create table test (
    id int identity primary key,
    val1 int,
    val2 varchar(10)
)
```

```
create trigger test_tr on test
    after insert, update, delete
as
begin
   select 'deleted', * from deleted

   select 'inserted', * from inserted
end
```

Triggers c.d.

```
enable trigger test_tr on test;
```

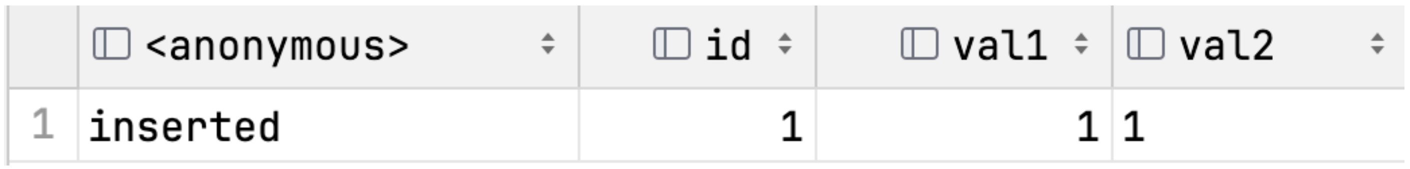
```
disable trigger test_tr on test;
```

```
drop trigger test_tr;
```

Triggers - insert

```
insert test (val1, val2)
values (1,'1')
```

- `deleted` table
    - empty
- `inserted` table

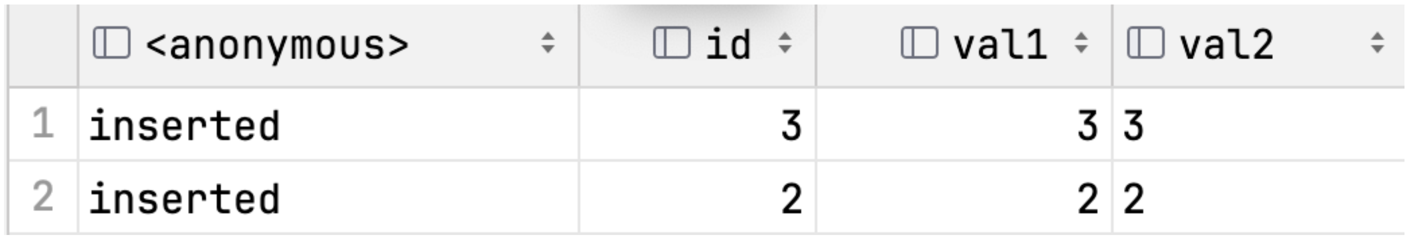


	<anonymous>	id	val1	val2
1	inserted	1	1	1

Triggers - insert c.d.

```sql
insert test (val1, val2)
values (2,'2'),
       (3,'3')
```

- `deleted` table
  - empty
- `inserted` table

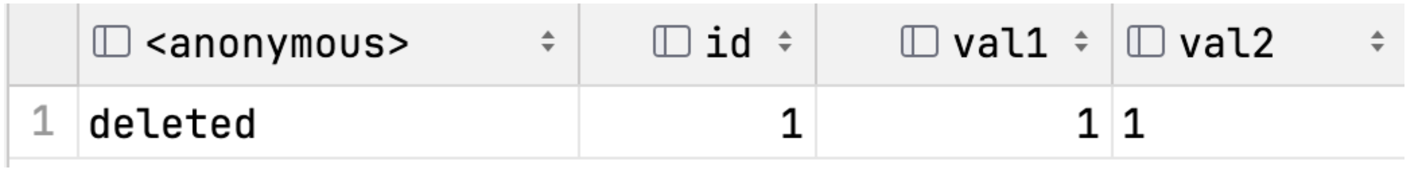


	<anonymous>	id	val1	val2
1	inserted	3	3	3
2	inserted	2	2	2

Triggers - update

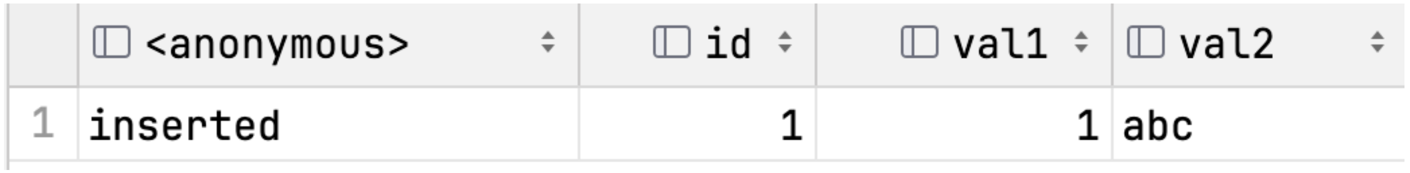
```sql
update test
set val2 = 'abc'
where id = 1
```

- `deleted` table



	<anonymous>	id	val1	val2
1	deleted	1	1	1

- `inserted` table

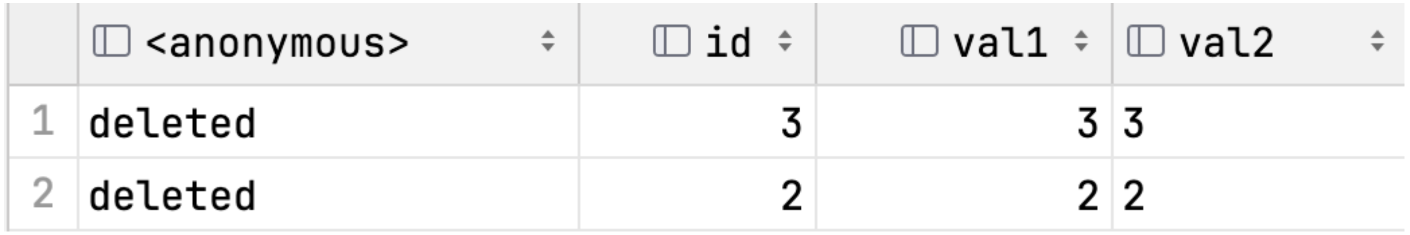


	<anonymous>	id	val1	val2
1	inserted	1	1	abc

Triggers- update c.d.

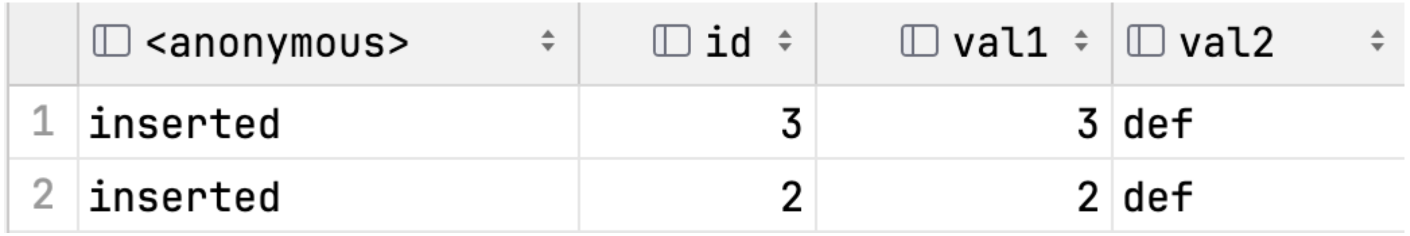
```sql
update test
set val2 = 'def'
where id > 1
```

- `deleted` table



	<anonymous>	id	val1	val2
1	deleted	3	3	3
2	deleted	2	2	2

- `inserted` table

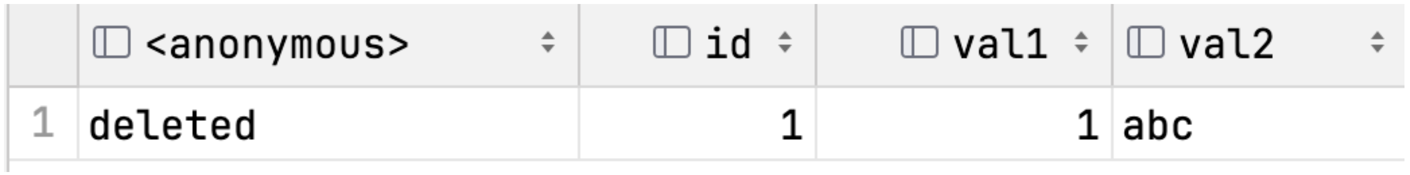


	<anonymous>	id	val1	val2
1	inserted	3	3	def
2	inserted	2	2	def

Triggers - delete

```
delete test
where id = 1
```

`deleted` table



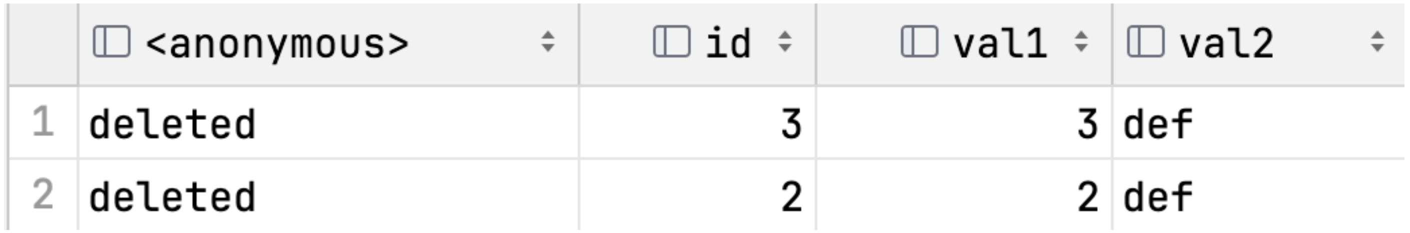
	<anonymous>	id	val1	val2
1	deleted	1	1	abc

- `inserted` table
  - empty

Triggers - delete c.d.

```
delete test
where id > 1
```

- `deleted` table



	<anonymous>	id	val1	val2
1	deleted	3	3	def
2	deleted	2	2	def

- `inserted` table
  - empty

Triggers - example

the new field `status` added ro the `orders` table

- N - new
- P - paid
- C - canceled

```
alter table dbo.orders add
    status char(1) not null constraint df_orders_status default 'N'

go

alter table dbo.orders add constraint
    ck_orders check (status in ('N','P','C'))
```

Triggers - example c.d.

`orders_log` table

- tracking changes in order status

```
create table orders_log (
    logid int identity primary key,
    orderid int not null,
    status char(1),
    mod_date datetime not null constraint df_orders_log_mod_date default getdate()
)
```

Triggers - example c.d.

```sql
create trigger orders_status_change on orders
    after insert, update
as
begin
    insert orders_log(orderid,status)
    select orderid,status from inserted
end
```

Triggers - example c.d.

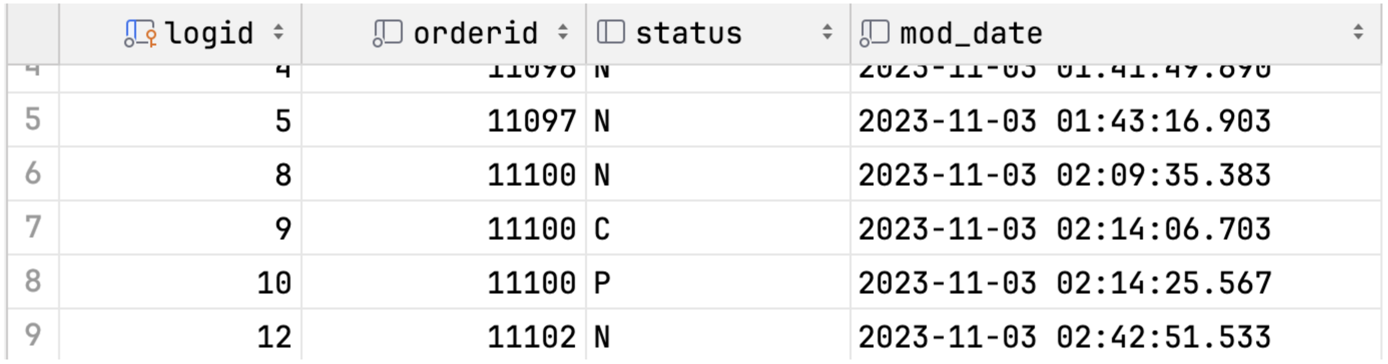
```sql
declare  @requireddate date = dateadd(day, 7, getdate())

exec add_order 'ALFKI', 1, @requireddate
```

```sql
update orders
set status = 'C'
where orderid = 11100
```

```sql
update orders
set status = 'P'
where orderid = 11100
```

Triggers - example c.d.

```sql
select * from orders_log
```



	logid	orderid	status	mod_date
4	4	11096	N	2023-11-03 01:41:49.890
5	5	11097	N	2023-11-03 01:43:16.903
6	8	11100	N	2023-11-03 02:09:35.383
7	9	11100	C	2023-11-03 02:14:06.703
8	10	11100	P	2023-11-03 02:14:25.567
9	12	11102	N	2023-11-03 02:42:51.533

Triggers - example c.d.

```sql
declare @customerid CHAR(5) = 'ALFKI'
declare @employeeid int = 1
declare @orderdate date = getdate()
declare @requireddate date = dateadd(day, 7, getdate())
declare @productid int = 1
declare @quantity int = 1
declare @discount decimal(3,2) = 0.14

begin try
   begin transaction

    insert orders(customerid,employeeid,orderdate,requireddate)
    values(@customerid,@employeeid,@orderdate,@requireddate)

    declare @unitprice decimal(10,2)
    select @unitprice = unitprice from avail_product where productid = @productid

    insert [Order Details](orderid, productid, unitprice, quantity, discount)
    values(scope_identity(), @productid, @unitprice, @quantity, @discount)

    update products
    set unitsinstock = unitsinstock - @quantity
    where productid = @productid

    commit
end try
begin catch
    if @@trancount > 1
          rollback
    ;throw
end catch
```

Triggers - example c.d.

- It is necessary to use `scope_identity()` to get the value of the autogenerated `id`
  - `@@identity` will not return the correct value
    - A trigger containing the `insert` to the `orders_log` was called
      - `@@identity` will return the ID value generated for `ordrs_log` table
- https://learn.microsoft.com/en-us/sql/t-sql/functions/scope-identity-transact-sql?view=sql-server-ver16

T# Triggers - example c.d.

ATTENTION:

- `add_order2` procedure is not correct now

```
declare @requireddate date = dateadd(day, 7, getdate())

exec add_order2 'ALFKI', 1, @requireddate,
                1, 1, 0.12
```

> Error
> [23000][547] Line 15: The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Order_Details_Orders". The conflict occurred in database "Northwind_m", table "dbo.Orders", column 'OrderID'.

T# Triggers - example c.d.

```
create procedure add_order3
@customerid char(5), @employeeid int, @requireddate date,
@productid int, @quantity int, @discount decimal(3,2)
as
begin
    begin try
        begin transaction

        if not exists (select * from customers where customerid = @customerid)
           throw 50001, 'No customer with such id', 1

        declare @orderdate date = getdate()

        insert orders(customerid,employeeid,orderdate,requireddate)
        values(@customerid,@employeeid,@orderdate,@requireddate)

        declare @orderid int = scope_identity()
        exec add_detail @orderid, @productid, @quantity, @discount

        commit
    end try
    begin catch
        if @@trancount > 1
            rollback
        ;throw
    end catch
end
```

Triggers - example c.d.

```
declare @requireddate date = dateadd(day, 7, getdate())

exec add_order3 'ALFKI', 1, @requireddate,
                1, 1, 0.12
```

- OK

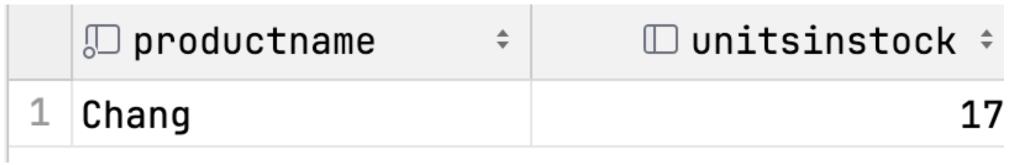
Triggers - example c.d.

- Trigger controls whether the product is available in the stock

```
create trigger order_details_check_stock_insert on [order details]
    after insert
as
begin
     if (select count(*) from inserted) > 1
         throw 50005, 'Only one row at a time', 1

     declare @productid int, @quantity int
     select @productid= productid, @quantity = quantity from inserted

     if not exists (select * from avail_product
                       where productid = @productid and avail_product.unitsinstock >= @quantity)
         throw 50004, 'Product not available', 1
end
```

Triggers - example c.d.

```
select * from avail_product where productid = 2
```



	productname	unitsinstock
1	Chang	17

```
insert [order details](orderid, productid, quantity, unitprice, discount)
values(11102,2,20,10,0.05)
```

> Error
> [S0001][50004] Line 13: Product not available

- lack of enough product in the stock

```
insert [order details](orderid, productid, quantity, unitprice, discount)
values(11102,2,1,10,0.05)
```

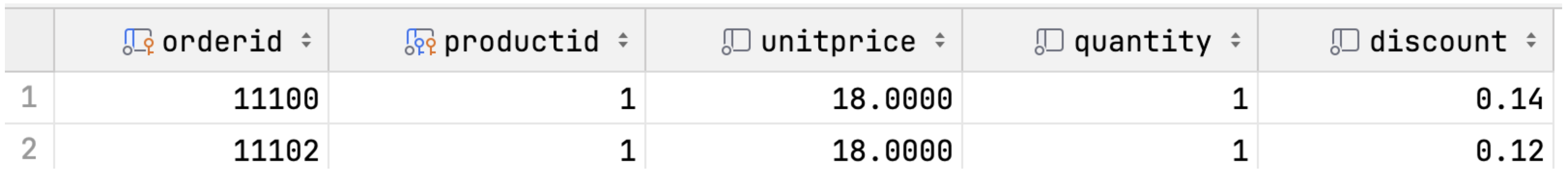
- OK

Triggers - example c.d.

```
alter trigger order_details_check_stock_upd on [order details]
    after update
as
begin
     if (select count(*) from inserted) > 1
         throw 50005, 'Only one row at a time', 1

     declare @productid int, @new_quantity int, @old_quantity int
     select @productid = productid, @new_quantity = quantity from inserted
     select @old_quantity = quantity from inserted

     if not exists (select * from avail_product
                    where productid = @productid and avail_product.unitsinstock >= @new_quantity - @old_quantity)
         throw 50004, 'Product not available', 1
end
```

Triggers - example c.d.

```
select * from [Order Details] where orderid >= 11100
```



	orderid	productid	unitprice	quantity	discount
1	11100	1	18.0000	1	0.14
2	11102	1	18.0000	1	0.12

```
update [Order Details]
set discount = 0.15
where orderid >= 11100
```

> Error
> [S0001][50005] Line 6: Only one row at a time

Triggers - example c.d.

```
create trigger orders_delete_v1 on orders
     after delete
as
begin
     throw 50006, 'order can not be deleted (v1)', 1
end
```

```
delete orders
where orderid >= 11100
```

> Error
> [23000][547] Line 1: The DELETE statement conflicted with the REFERENCE constraint "FK_Order_Details_Orders". The conflict occurred in database "Northwind_m", table "dbo.Order Details", column 'OrderID'.

Triggers - example c.d.

```sql
create trigger orders_delete_v2 on orders
     instead of delete
as
begin
     throw 50006, 'order can not be deleted (v2)', 1
end
```

```sql
delete orders
where orderid >= 11100
```

> Error
> [S0001][50006] Line 5: order can not be deleted (v2)