CREATE TABLE order\_list (  
id\_order\_list int PRIMARY KEY IDENTITY(1,1),  
id\_catalog int FOREIGN KEY REFERENCES catalog,  
id\_order int FOREIGN KEY REFERENCES orderr,  
quantity tinyint CHECK(quantity>0)  
)  
  
CREATE TABLE prescription (  
id\_prescribtion int PRIMARY KEY IDENTITY(1,1),  
id\_employee int FOREIGN KEY REFERENCES eye\_doctor,  
id\_customer int FOREIGN KEY REFERENCES customer,  
right\_diopter decimal(4,2) CHECK(right\_diopter>=-15 AND right\_diopter<=15),  
left\_diopter decimal(4,2) CHECK(left\_diopter>=-15 AND left\_diopter<=15),  
distance tinyint NOT NULL check(distance>0),  
date date NOT NULL  
)  
  
CREATE TABLE customer (  
id\_customer int PRIMARY KEY IDENTITY(1,1),  
name varchar(20) NOT NULL check(name not like '%[^А-Яа-я]%'),  
surname varchar(20) NOT NULL check(surname not like '%[^А-Яа-я]%'),  
phone\_number varchar(12) NOT NULL UNIQUE CHECK(len(phone\_number) = 11),  
email varchar(30) UNIQUE  
)  
  
create TABLE catalog (  
id\_catalog int PRIMARY KEY IDENTITY(1,1),  
id\_lenses int unique FOREIGN KEY REFERENCES lenses on delete cascade,  
id\_rim int unique FOREIGN KEY REFERENCES rim on delete cascade,  
id\_accessories int unique FOREIGN KEY REFERENCES accessories on delete cascade,  
quantity int NOT NULL check(quantity >= 0),  
CONSTRAINT Only\_One\_ID  
)

CREATE UNIQUE NONCLUSTERED INDEX UQ\_Party\_SamAccountName1  
ON catalog(id\_lenses)  
WHERE id\_lenses IS NOT NULL;

CREATE UNIQUE NONCLUSTERED INDEX UQ\_Party\_SamAccountName2  
ON catalog(id\_rim)  
WHERE id\_rim IS NOT NULL;

CREATE UNIQUE NONCLUSTERED INDEX UQ\_Party\_SamAccountName3  
ON catalog(id\_accessories)  
WHERE id\_accessories IS NOT NULL;  
  
CREATE TABLE orderr (  
id\_order int PRIMARY KEY IDENTITY(1,1),  
id\_employee int FOREIGN KEY REFERENCES shop\_employee,  
id\_prescription int FOREIGN KEY REFERENCES prescription,  
date\_acceptance smalldatetime NOT NULL,  
date\_assignment smalldatetime,  
id\_status tinyint FOREIGN KEY REFERENCES status,  
end\_price float NOT NULL  
)  
  
CREATE TABLE shop\_employee (  
id\_employee int PRIMARY KEY IDENTITY(1,1),  
name varchar(20) NOT NULL check(name not like '%[^А-Яа-я]%'),  
surname varchar(20) NOT NULL check(surname not like '%[^А-Яа-я]%'),  
patronymic varchar(20) check(patronymic not like '%[^А-Яа-я]%'),  
post varchar(20) NOT NULL check(post not like '%[^А-Яа-я]%'),  
phone\_number varchar(12) UNIQUE NOT NULL CHECK(len(phone\_number) = 11),  
passport\_details varchar(50) UNIQUE NOT NULL,  
adress varchar(60) NOT NULL  
)  
  
CREATE TABLE eye\_doctor (  
id\_employee int PRIMARY KEY IDENTITY(1,1),  
name varchar(20) NOT NULL check(name not like '%[^А-Яа-я]%'),  
surname varchar(20) NOT NULL check(surname not like '%[^А-Яа-я]%'),  
patronymic varchar(20) check(patronymic not like '%[^А-Яа-я]%'),  
post varchar(20) NOT NULL check(post not like '%[^А-Яа-я]%'),  
phone\_number varchar(12) NOT NULL UNIQUE CHECK(len(phone\_number) = 11),  
passport\_details varchar(50) NOT NULL UNIQUE,  
adress varchar(60)  
)  
  
CREATE TABLE lenses (  
id\_lenses int PRIMARY KEY IDENTITY(1,1),

Name\_lenses VARCHAR(30),  
diopter decimal(4,2) NOT NULL CHECK(diopter>=-15 AND diopter<=15),  
id\_color int FOREIGN KEY REFERENCES color,  
manufact VARCHAR(30),  
price float NOT NULL  
)  
  
CREATE TABLE rim (  
id\_rim int PRIMARY KEY IDENTITY(1,1),

Name\_rim VARCHAR(30),  
id\_color int FOREIGN KEY REFERENCES color,  
manufact VARCHAR(30),  
price float NOT NULL,  
)  
  
CREATE TABLE accessories (  
id\_accessories int PRIMARY KEY IDENTITY(1,1),

Name\_ accessories VARCHAR(30),  
manufact VARCHAR(30),  
type VARCHAR(20),  
price float NOT NULL  
)  
  
CREATE TABLE color (  
id\_color int PRIMARY KEY IDENTITY(1,1),  
name\_color varchar(20) NOT NULL,  
)  
  
CREATE TABLE status (  
id\_status tinyint PRIMARY KEY IDENTITY(1,1),  
meaning\_status varchar(20) NOT NULL ,  
)

create procedure [create\_prescription]

(@id\_employee int, @id\_customer int, @right\_diopter decimal(4,2),

@left\_diopter decimal(4,2), @distance tinyint, @date date)

as

begin

insert into prescription(id\_employee, id\_customer, right\_diopter,

left\_diopter, distance, date)

values(@id\_employee, @id\_customer, @right\_diopter,

@left\_diopter, @distance, @date)

End

create procedure [create\_order]

(@id\_employee int, @id\_prescription int, @date\_acceptance date,

@date\_assignment date, @id\_status int, @end\_price float)

as

begin

insert into orderr(id\_employee, id\_prescription, date\_acceptance,

date\_assignment, id\_status, end\_price)

values(@id\_employee, @id\_prescription, @date\_acceptance,

@date\_assignment, @id\_status, @end\_price)

End

create procedure [create\_customer]

(@name varchar(20), @surname varchar(20),

@phone\_number varchar(12), @email varchar(30))

as

begin

insert into customer(name, surname,

phone\_number, email)

values(@name, @surname,

@phone\_number, @email)

End

~~CREATE PROCEDURE ItogStoimost~~

~~@orderID int~~

~~AS~~

~~BEGIN~~

~~DECLARE @totalPrice float;~~

~~SELECT @totalPrice =~~

~~COALESCE(SUM(l.price \* ol.quantity), 0) +~~

~~COALESCE(SUM(r.price \* ol.quantity), 0) +~~

~~COALESCE(SUM(a.price \* ol.quantity), 0)~~

~~FROM order\_list ol~~

~~LEFT JOIN catalog c ON ol.id\_catalog = c.id\_catalog~~

~~LEFT JOIN lenses l ON c.id\_lenses = l.id\_lenses~~

~~LEFT JOIN rim r ON c.id\_rim = r.id\_rim~~

~~LEFT JOIN accessories a ON c.id\_accessories = a.id\_accessories~~

~~WHERE ol.id\_order = @orderID;~~

~~SELECT @totalPrice as TotalPrice;~~

~~UPDATE orderr~~

~~SET end\_price = @totalPrice~~

~~WHERE id\_order = @orderID;~~

~~END;~~

~~EXEC ItogStoimost @orderID = 5;~~

~~CREATE TRIGGER end\_price\_auto~~

~~ON order\_list~~

~~FOR INSERT~~

~~AS~~

~~BEGIN~~

~~select inserted.id\_order~~

~~from inserted~~

~~EXEC ItogStoimost id\_order~~

~~END~~

CREATE TRIGGER end\_price\_auto

ON order\_list

FOR INSERT

AS

BEGIN

DECLARE @totalPrice float;

SELECT @totalPrice =

COALESCE(SUM(l.price \* ol.quantity), 0) +

COALESCE(SUM(r.price \* ol.quantity), 0) +

COALESCE(SUM(a.price \* ol.quantity), 0)

FROM order\_list ol

LEFT JOIN catalog c ON ol.id\_catalog = c.id\_catalog

LEFT JOIN lenses l ON c.id\_lenses = l.id\_lenses

LEFT JOIN rim r ON c.id\_rim = r.id\_rim

LEFT JOIN accessories a ON c.id\_accessories = a.id\_accessories

WHERE ol.id\_order = (SELECT id\_order FROM INSERTED);

SELECT @totalPrice as TotalPrice;

UPDATE orderr

SET end\_price = @totalPrice

WHERE id\_order = (SELECT id\_order FROM INSERTED);

END

CREATE TRIGGER end\_price\_delete\_auto

ON order\_list

after delete

AS

BEGIN

DECLARE @totalPrice float;

SELECT @totalPrice =

COALESCE(SUM(l.price \* ol.quantity), 0) +

COALESCE(SUM(r.price \* ol.quantity), 0) +

COALESCE(SUM(a.price \* ol.quantity), 0)

FROM order\_list ol

LEFT JOIN catalog c ON ol.id\_catalog = c.id\_catalog

LEFT JOIN lenses l ON c.id\_lenses = l.id\_lenses

LEFT JOIN rim r ON c.id\_rim = r.id\_rim

LEFT JOIN accessories a ON c.id\_accessories = a.id\_accessories

WHERE ol.id\_order in (SELECT id\_order FROM deleted);

SELECT @totalPrice as TotalPrice;

UPDATE orderr

SET end\_price = @totalPrice

WHERE id\_order in (SELECT id\_order FROM deleted);

END

~~CREATE TRIGGER status\_check~~

~~ON orderr~~

~~instead of update~~

~~AS~~

~~BEGIN~~

~~IF (select id\_status from inserted) = 3~~

~~begin~~

~~update orderr~~

~~set date\_assignment = GETDATE(), id\_status = (select id\_status from inserted)~~

~~where id\_order = (select id\_order from inserted)~~

~~end~~

~~IF (select id\_status from inserted) = 4~~

~~begin~~

~~delete order\_list~~

~~from inserted~~

~~where order\_list.id\_order = inserted.id\_order~~

~~update orderr~~

~~set date\_assignment = GETDATE(), id\_status = (select id\_status from inserted)~~

~~where id\_order = (select id\_order from inserted)~~

~~end~~

~~ELSE~~

~~update orderr~~

~~set id\_status = (select id\_status from inserted)~~

~~where id\_order = (select id\_order from inserted)~~

~~END~~

CREATE TRIGGER status\_check

ON orderr

for update

AS

BEGIN

IF (select id\_status from inserted) = 3

begin

update orderr

set date\_assignment = GETDATE()

where id\_order = (select id\_order from inserted)

end

IF (select id\_status from inserted) = 4

begin

delete order\_list

from inserted

where order\_list.id\_order = inserted.id\_order

end

END

CREATE TRIGGER remove\_catalog

ON order\_list

instead of INSERT

AS

BEGIN

IF EXISTS (

SELECT c.id\_catalog

FROM catalog c

JOIN inserted i ON c.id\_catalog = i.id\_catalog

WHERE c.quantity - i.quantity < 0

)

BEGIN

THROW 50001, 'Недостаточно товара в каталоге.', 1;

END

ELSE

BEGIN

UPDATE catalog

SET quantity = c.quantity - i.quantity

FROM catalog c

JOIN inserted i ON c.id\_catalog = i.id\_catalog;

insert into order\_list

SELECT inserted.id\_catalog, inserted.id\_order, inserted.quantity

from inserted

END

END;

create function [dbo].[cusomer\_id]

(@Name varchar(30), @Surname varchar(30))

returns int

as

begin

declare @id\_customer int

set @id\_customer = (select id\_customer

from customer

where name = @Name and surname = @Surname

)

return @id\_customer

end

create function [dbo].[customer\_prescription]

(@Name varchar(30), @Surname varchar(30))

returns int

as

begin

declare @id\_prescription int

set @id\_prescription = (select top(1) id\_prescribtion

from prescription

where id\_customer = (select id\_customer

from customer

where name = @name and surname = @Surname)

order by date desc

)

return @id\_prescription

end

~~create function [dbo].[order\_list\_orderr\_id](@id\_order int)~~

~~returns @order\_list table~~

~~(~~

~~id\_order int,~~

~~name varchar(30),~~

~~surname varchar(30),~~

~~id\_catalog int,~~

~~product\_name varchar(30),~~

~~product\_manufact varchar(30),~~

~~quantity int~~

~~)~~

~~as~~

~~begin~~

~~insert @order\_list~~

~~select order\_list.id\_order, customer.name, customer.surname, order\_list.id\_catalog, rim.name\_rim, rim.manufact, order\_list.quantity~~

~~from customer, order\_list, rim~~

~~return~~

~~end~~

~~go~~

create function [dbo].[order\_list\_orderr\_id](@id\_order int)

returns @order\_list table

(

id\_order int,

name varchar(30),

surname varchar(30),

id\_catalog int,

product\_name varchar(30),

product\_manufact varchar(30),

quantity int

)

as

begin

insert @order\_list

select order\_list.id\_order, customer.name, customer.surname, order\_list.id\_catalog, rim.name\_rim, rim.manufact, order\_list.quantity

from orderr join prescription on orderr.id\_prescription = prescription.id\_prescribtion join customer on customer.id\_customer = prescription.id\_customer join order\_list

on order\_list.id\_order = orderr.id\_order join catalog on order\_list.id\_catalog = catalog.id\_catalog join rim on catalog.id\_rim = rim.id\_rim

where order\_list.id\_order = @id\_order

insert @order\_list

select order\_list.id\_order, customer.name, customer.surname, order\_list.id\_catalog, lenses.name\_lenses, lenses.manufact, order\_list.quantity

from orderr join prescription on orderr.id\_prescription = prescription.id\_prescribtion join customer on customer.id\_customer = prescription.id\_customer join order\_list

on order\_list.id\_order = orderr.id\_order join catalog on order\_list.id\_catalog = catalog.id\_catalog join lenses on catalog.id\_lenses = lenses.id\_lenses

where order\_list.id\_order = @id\_order

insert @order\_list

select order\_list.id\_order, customer.name, customer.surname, order\_list.id\_catalog, accessories.name\_accessories, accessories.manufact, order\_list.quantity

from orderr join prescription on orderr.id\_prescription = prescription.id\_prescribtion join customer on customer.id\_customer = prescription.id\_customer join order\_list

on order\_list.id\_order = orderr.id\_order join catalog on order\_list.id\_catalog = catalog.id\_catalog join accessories on catalog.id\_accessories = accessories.id\_accessories

where order\_list.id\_order = @id\_order

return

end

go

select \*

from dbo.[order\_list\_orderr\_id](15)