INF-202 Assignment 3

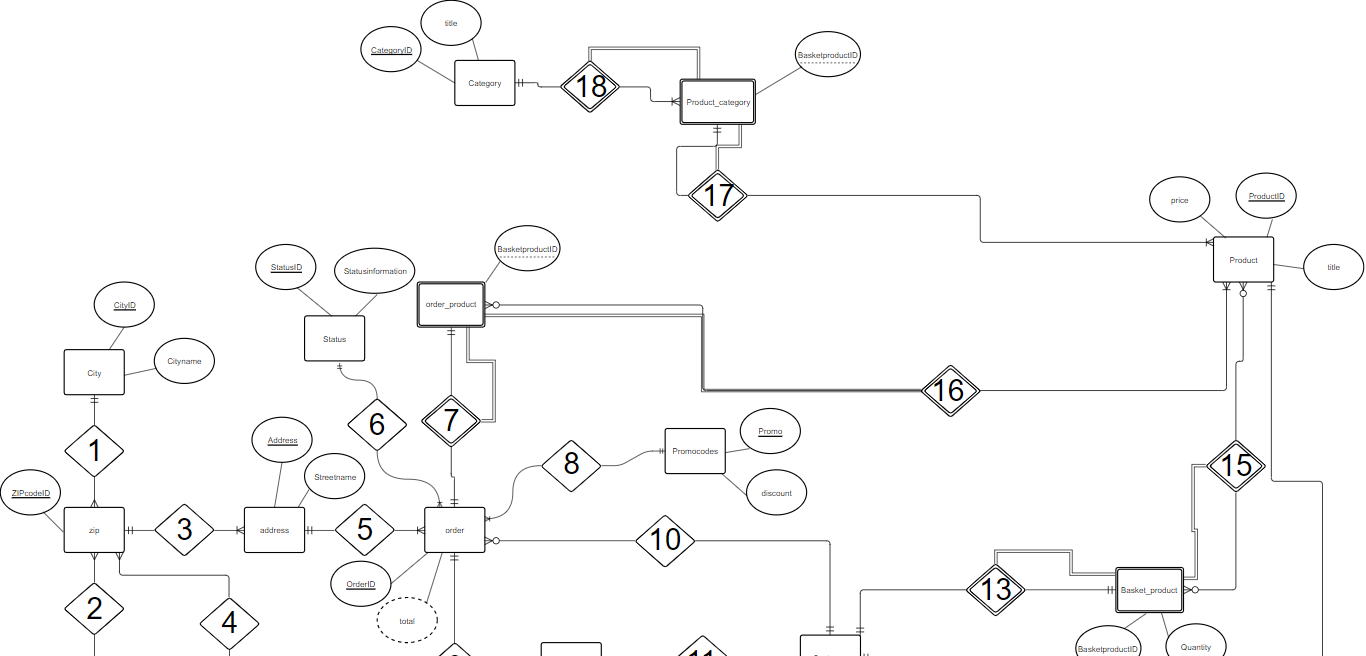
Kadyr Temirlan [12-p]; Alymov Almaz [14-p]; Zhuandyk Sultan [14-p]; Bakytov Yerbol [12-p];

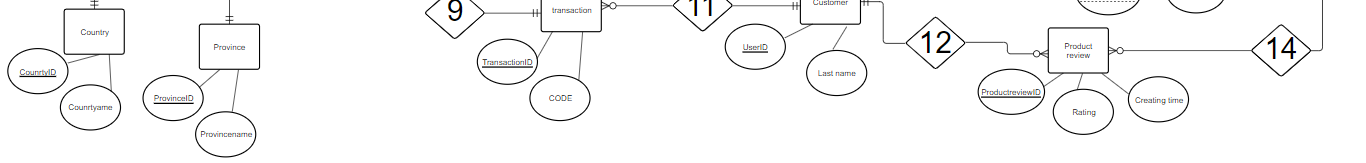
**Task #1**

Database should keep information about Online Shop Store. Shop Store holds records of customers which makes transaction for order. Product has Basket, so Customer can put product to this basket of product and peek it anytime. If Customer decided to buy this product it will store like order. Customer can choose any productCategory from categories. After choosing product Customer can see the quantity of this product in orderProduct. During transaction can used promocodes which can be entered by user. Also product has its own productReview where customer can leave comments. For finall order creation assigned its status and address. Also for addres assigned its zipcode, province, country and city.

**Task #2**

Sorry teacher, please enlarge screen a little bit and everything will be ok (small distance here because of 2 different screenshots).

****

****

***About relationships:***

1. 1 city can have: min infinite zipcodes / max infinite zipcodes. 1 zipcode can have: min 1 city/ max 1 city ;
2. 1 country can have: min infinite zipcodes / max infinite zipcodes. 1 zipcode can have: min 1 country/ max 1 country ;
3. 1 zipcode can have: min 1 address / max infinite address. 1 address can have: min 1 zipcode/ max 1 zipcode;
4. 1 zipcode can have: min 1 province / max 1 province. 1 province can have: min infinite zipcodes/ max infinite zipcodes;
5. 1 order can have: min 1 address / max 1 address. 1 address can have: min 1 order/ max infinite orders;
6. 1 status can have: min 1 orders / max infinite orders. 1 order can have: min 1 status/ max 1 status;
7. 1 orderproduct can have: min 1 order / max 1 order. 1 order can have: min 1 orderproduct/ max 1 orderproduct;
8. 1 promocode can have: min 1 order / max infinite orders. 1 order can have: min 1 promocode/ max 1 promocode;
9. 1 order can have: min 1 transaction / max 1 transaction. 1 transaction can have: min 1 order / max 1 order;
10. 1 order can have: min 1 customer / max 1 customer. 1 customer can have: min 0 order / max infinite order;
11. 1 transaction can have: min 1 customer / max 1 customer. 1 customer can have: min 0 transaction / max infinite transactions;
12. 1 customer can have: min 0 productreview / max 1 productreview. 1 productreview can have: min 1 customer / max 1 customer;
13. 1 customer can have: min 1 basketproduct / max 1 basketproduct. 1 basketproduct can have: min 1 customer / max 1 customer;
14. 1 productreview can have: min 1 product / max 1 product. 1 product can have: min 0 productreview / max infinite productreview;
15. 1 basketproduct can have: min 0 product / max infinite product. 1 product can have: min 0 basketproduct / max infinite basketproduct;
16. 1 orderproduct can have: min 1 product / max infinite product. 1 product can have: min 0 orderproduct / max infinite orderproduct;
17. 1 productcategory can have: min 1 product / max infinite product. 1 product can have: min 1 productcategory / max 1 productcategory;
18. 1 productcategory can have: min 1 category/ max 1 category. 1 category can have: min 1 productcategory / max infinite productcategory;

**Basket\_Product, Orders\_Product, Product\_Category** can’t be uniquely identified by its attributes alone. They have partial participation because they can’t exist without their strong entity to which they are attached.

Our database is in 3NF (attributes only depend on primary keys and not on other attributes), in Boyce-Codd (because key attributes do not depend on non-key attributes), in 4NF (because there are no multivalued dependencies) ;

Boyce-Codd and 4NF not finished because we have a limit of 15 tables. Example: type of product and tax of the order etc.

***Entity and their attributes:***

1) *Product* (ProductID, Title, Type, Price, Quantity, CreatedDate, UpdatedDate, Discount, SaleStart, SaleEnd, SKU, MetaTitle, URL);

2) *Customer* (UserID, FirstName, MiddleName, LastName, Phone, Email, Password, Gender, CreatedDate, LastLoginDate, Age);

3) *Orders* (OrderID, UserID, StatusID, Promo, AddressID, FirstName, MiddleName, LastName, TotalOfProducts, ProductDiscount, Tax, Shipping, TotalDiscount, Total, Phone, Email, City, State, Country, CreatedTime, UpdatedTime, Details);

4) *Orders*\_*Product* (OrderProductID, ProductID, OrderID, Quantity);

5) *Transactions* (TransactionID, UserID, OrderID, StatusID, Code, Type, Mode, Createdtime, Updatedtime, Details);

6) *Product*\_Review (ProductReviewID, ProductID, Rating, CreatedTime, Comment);

7) *Category* (CategoryID, Title, Details);

8) *Product*\_*Category* (ProductID, CategoryID);

9) *Status* (StatusID, Status\_Information);

10) *Promocodes* (Promo, Sale\_Information);

11) *Basket*\_*Product* (BasketProductId, ProductID, UserID, Quantity);

12) *Address* (AddressID, UserID, ZIPCode, Street);

13) *ZIP* (ZIPCode, ProvinceID, CityID, CountryID);

14) *Country* (CountryID, CountryName);

15) *Province* (ProvinceID, ProvinceName);

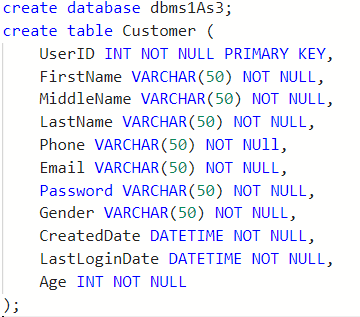
16) *City* (CityID, CityName);

**Task #3**

We create database called “dbms1As3” and start create our tables with all constraints. Overall we have 16☺ tables.

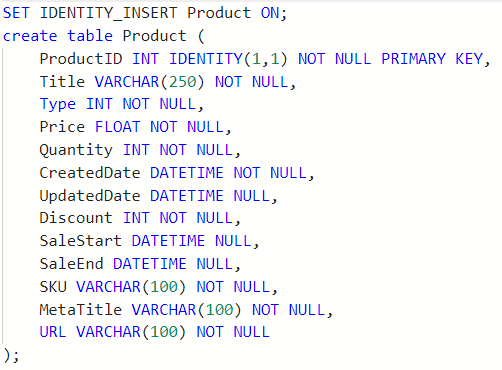
So first one **Customer** and **Product** tables here, for Product table we have Identity(1, 1) we will use it later for parametrized stored procedures. ‘Set Identity\_Insert On/Off’ is for inserting values to this table without procedure;

Create and Insert values for **Customer** table:

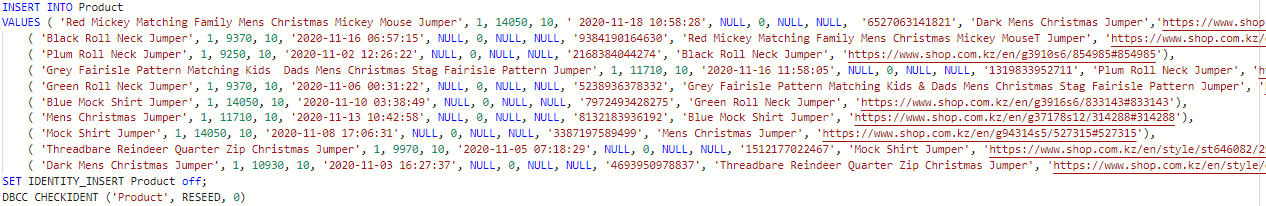
****

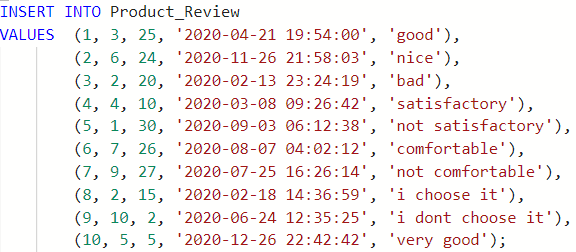
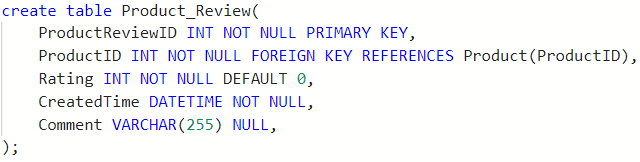
****

**Product** table:

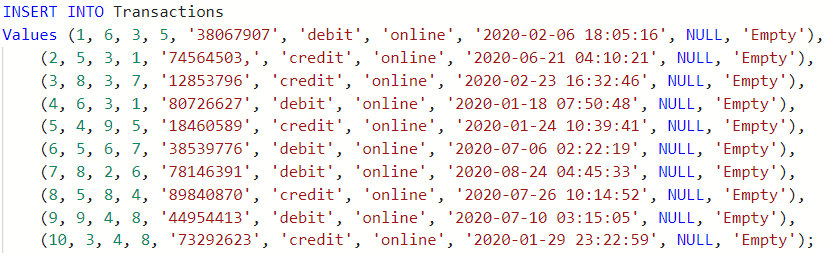
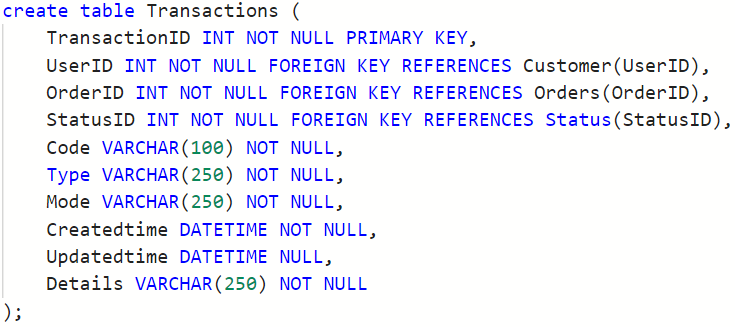
****

Insert values for **Product** table: (DBCC CHECKIDENT… we use it because we have some problem with ordering, so with it our *ProductId* ordering automatically start from 1 till 10). We think that Problem is because of Identity for Primary key, so we decided to fix it like this;

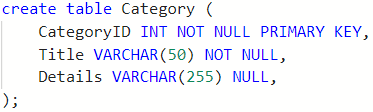
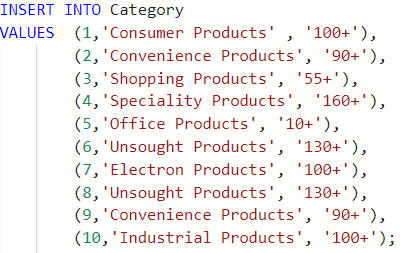
**** Create and Insert values for **Product\_Review**:

****

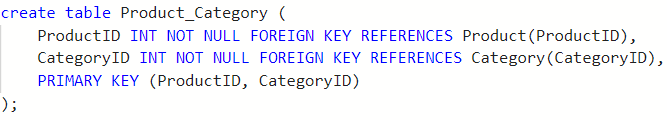
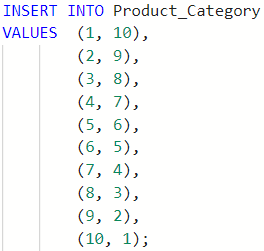
Create and Insert values for **Transactions**



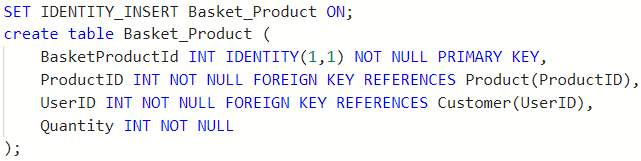
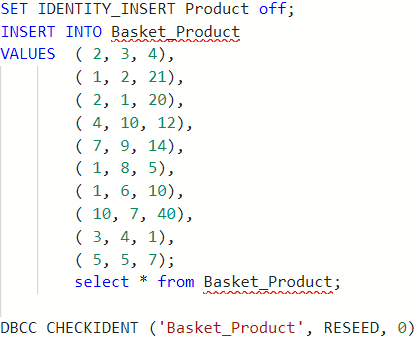
Create and Insert values for **Category** table:

** **

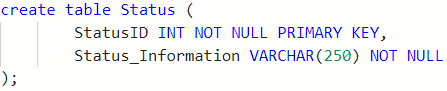
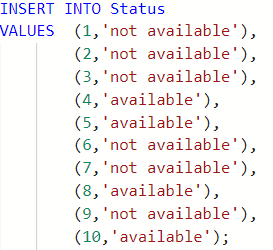
Create and Insert values for **Product\_Category** table:

** **

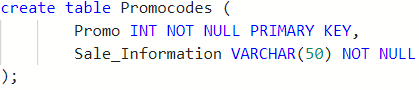
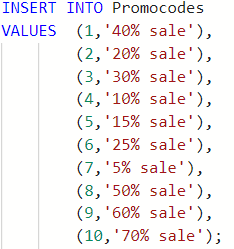
**Basket\_Product** table and Insert values to this table: (Here also using DBCC CHECKIDENT(like for Product table) for automaticaly ordering *BasketProductId* from 1 till 10);

** **

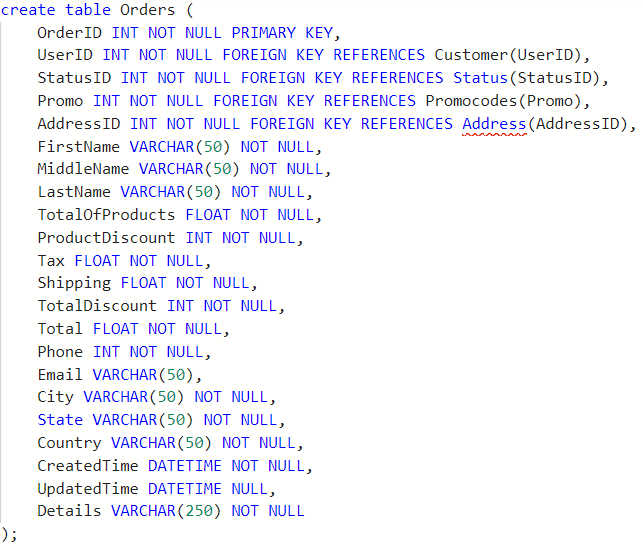
Create and Insert values for **Status** table:

** **

Create and Insert values for **Promocodes** table:

** **

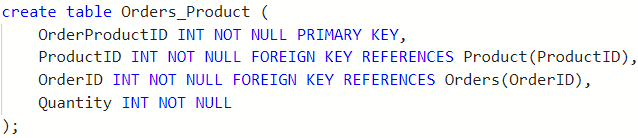
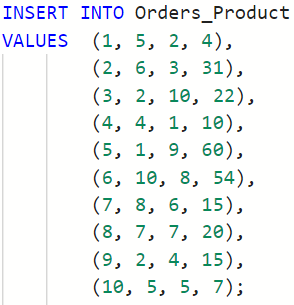
Create and Insert values for **Orders** table:

****

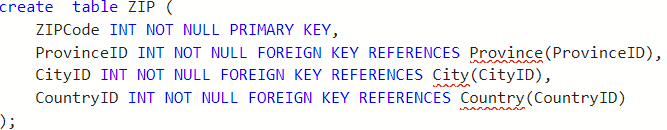
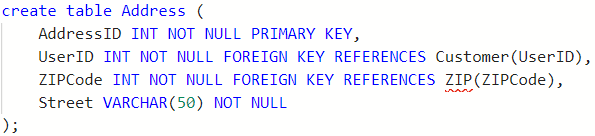
The total sum of order also we can derived using totalOfProducts, promocode, totalDiscount, tax and shipping. But we decided just to insert it ourselves.

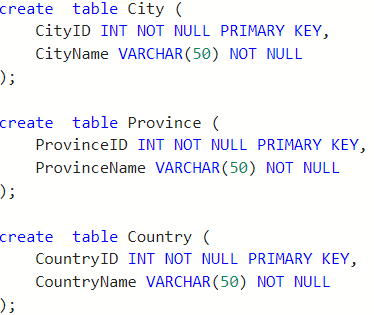
****

Create and Insert values for **Orders\_Product** table:

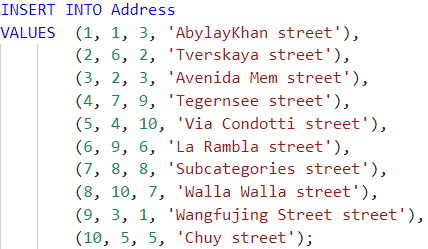
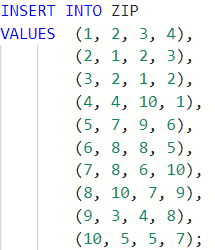
**** 

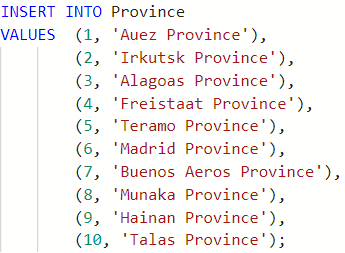
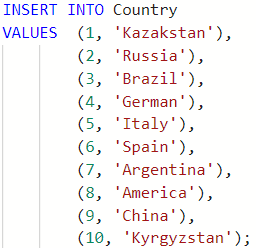
Here is **Address** table which is *foreign key* in **Orders** table, so we create separately **Address** table, then **ZIP** table which is *foreign key* in **Address** table, **City**, **Country**, **Province** tables which is also foreign keys in **ZIP** table. All tables are connected via keys. Also we decided to do this for normalization.

****

****

**INSERT VALUES to this tables:**

**** 

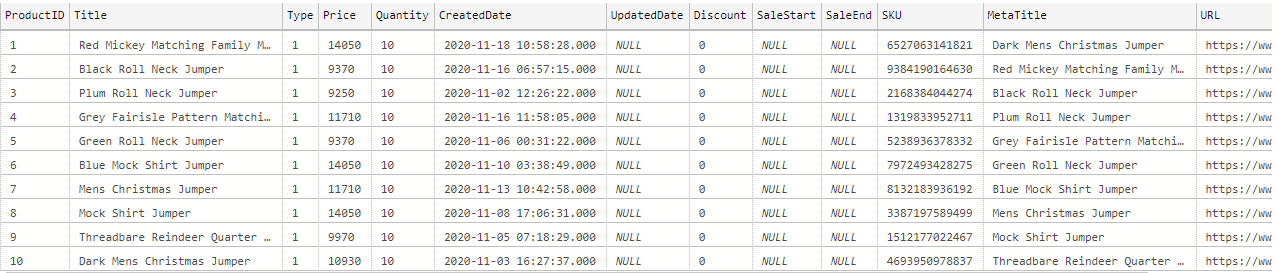
****

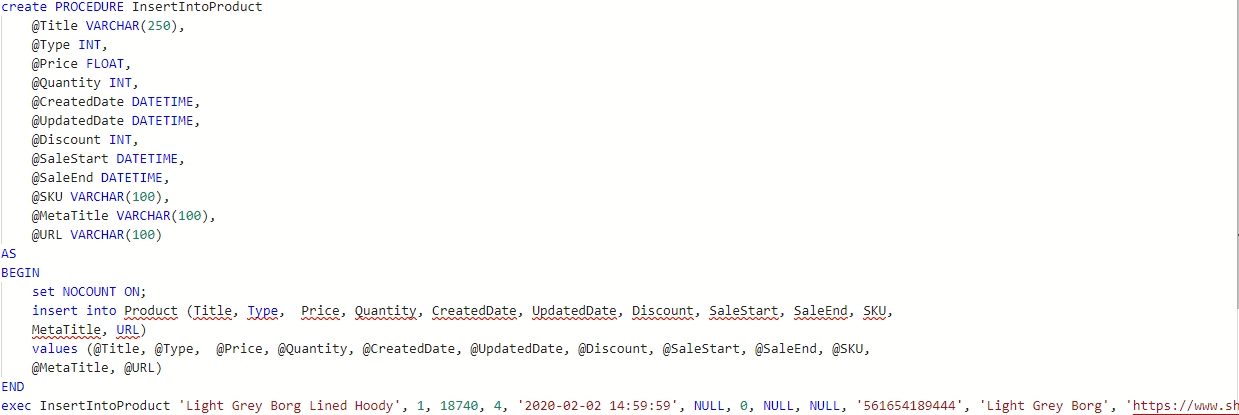
-----------------------------------------------------------------------------------------------------------------------------------------------------------

So about procedures: why we decided to do it for this two tables. For **Basket\_Product** table use insert and delete procedures because User can add products and also delete this products from Basket. Update procedure is for updating quantity in this basket. For **Product** table we use insert because we can add new product for example, and delete procedure is because user can delete this product for example if it is not exist. Update procedure is because we can update price for products.

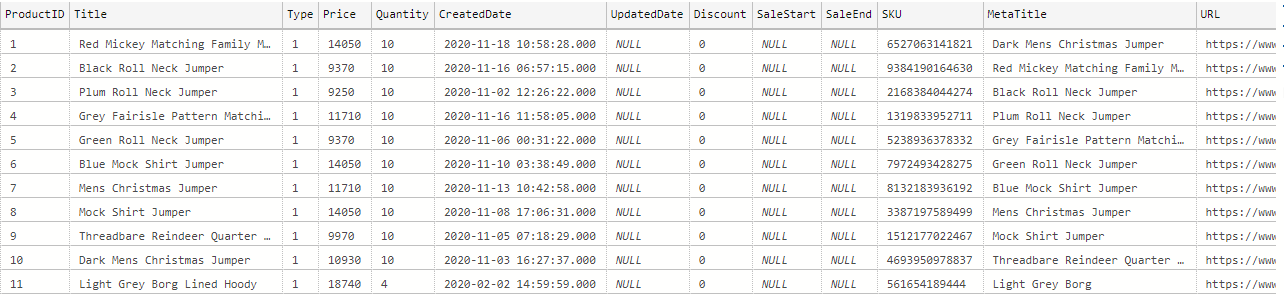
***Example1 to Insert Procedure:***

Before executing this procedure our **Product** table have 10 records;



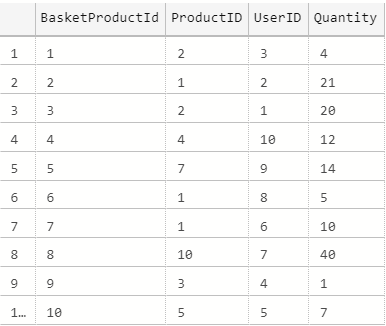
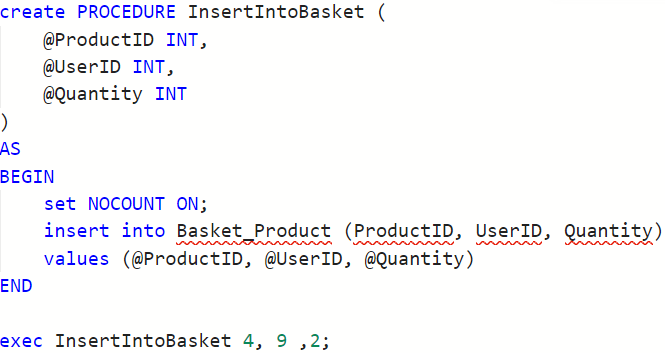


After executing procedure will append new row with new values and ordering will continue because we use Identity(1, 1) for Primary Key;

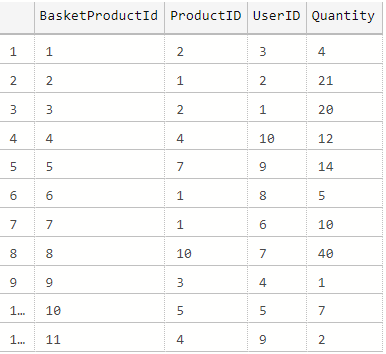


***Example2 to Insert Procedure:***

Before executing this procedure our **Basket\_Product** table have 10 records;

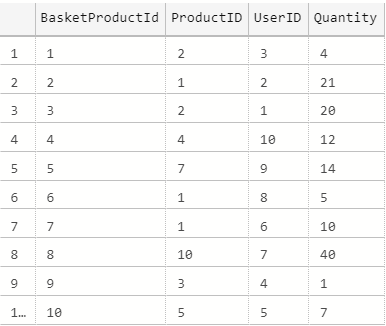
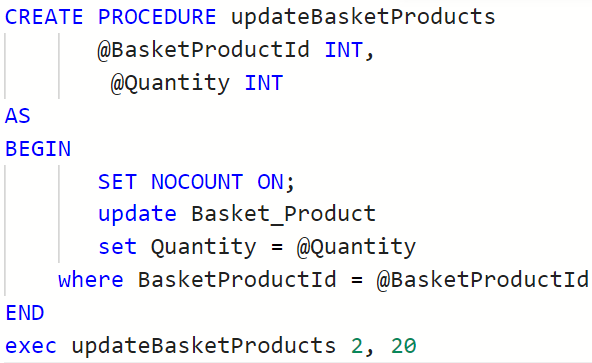
 

After executing procedure will append new row with new values and ordering will continue because we use Identity(1, 1) for Primary Key;

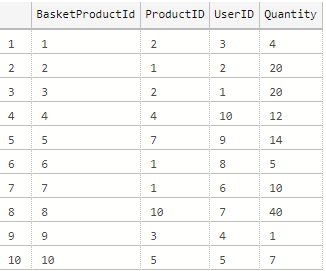


***Example1 to Update Procedure:***

Before executing this procedure in **Basket\_Product** table value for Quantity was ‘21’ for *BasketProductId* = 2;

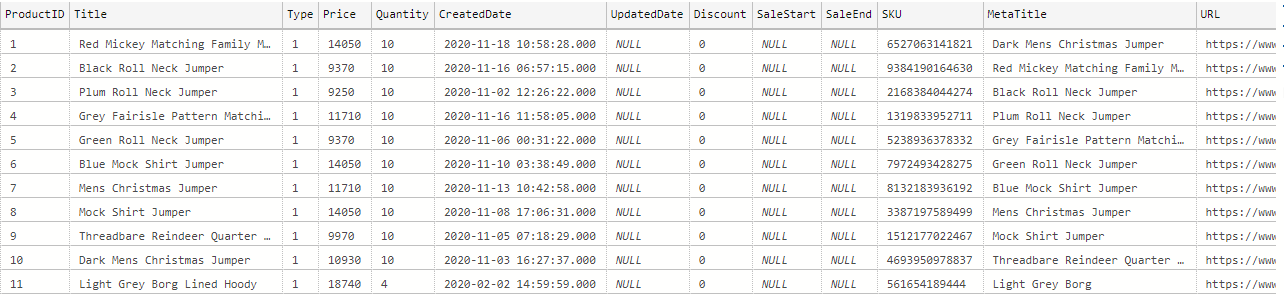
 

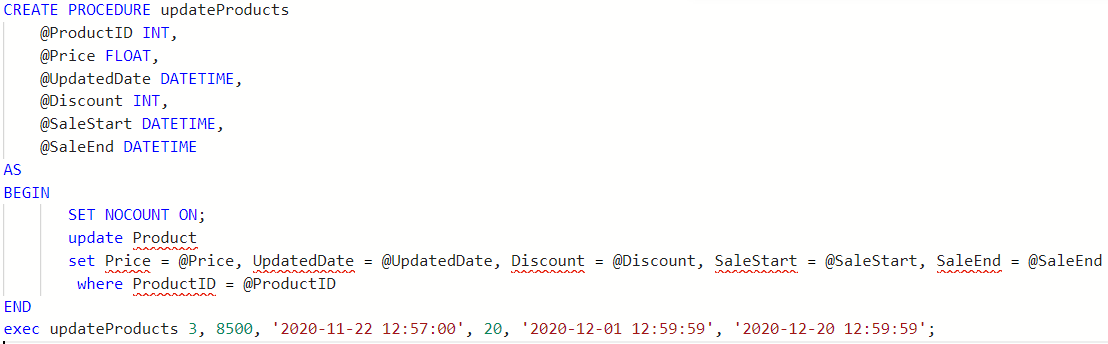
After executing procedure my Quantity value for *BasketProductId* = ‘2’ will update to ‘20’;



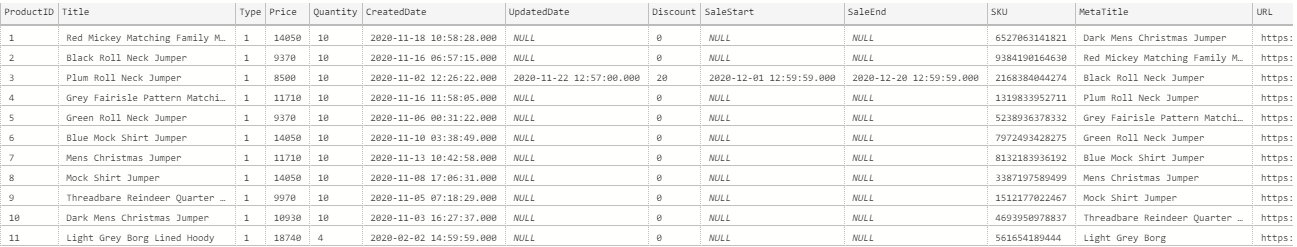
***Example2 to Update Procedure:***

Before executing this procedure in **Product** table value for *Price*, *UpdatedDate*, *Discount*, *SaleStart*, *SaleEnd* for ProductId = 3 was *9250*, *Null*, *0*, *Null*, *Null* respectively;



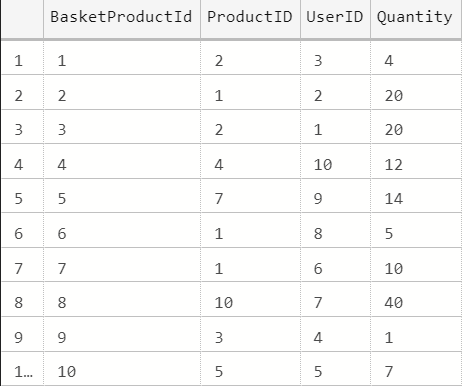
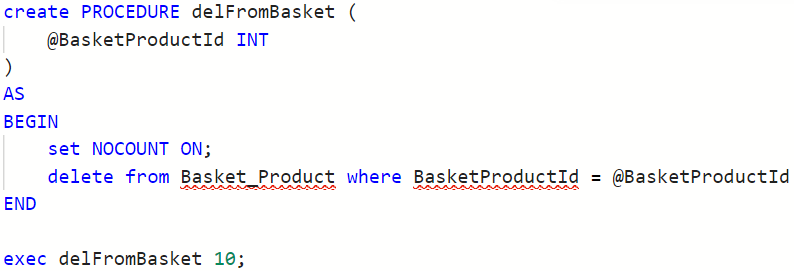


After executing procedure my *Price*, *UpdatetDate*, *Discount*, *SaleStart*, *SaleEnd* for ProductId = 3 update to *8500*, 2020-11-22 12:57:00, *20*, 2020-12-01 12:59:59, 2020-12-20 12:59:59 respectively;

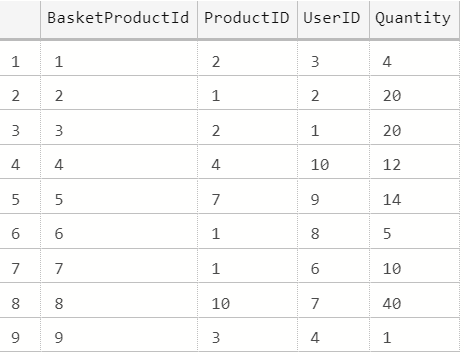


***Example1 to Delete Procedure:***

Before executing this procedure our **Basket\_Product** table have record in row where *BasketProductId* = 10;

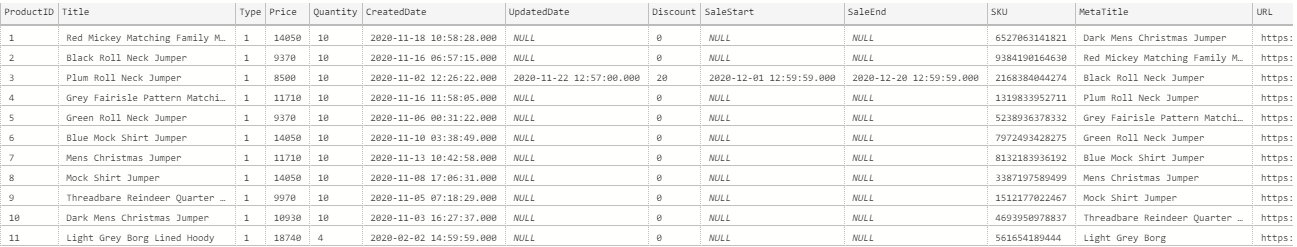
*** ***

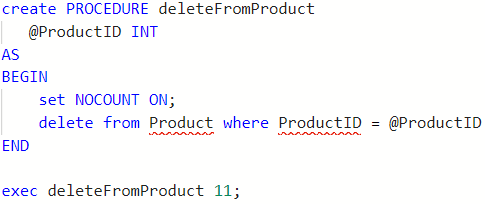
After executing procedure my row with *BasketProductId* = 10 will deleted;

******

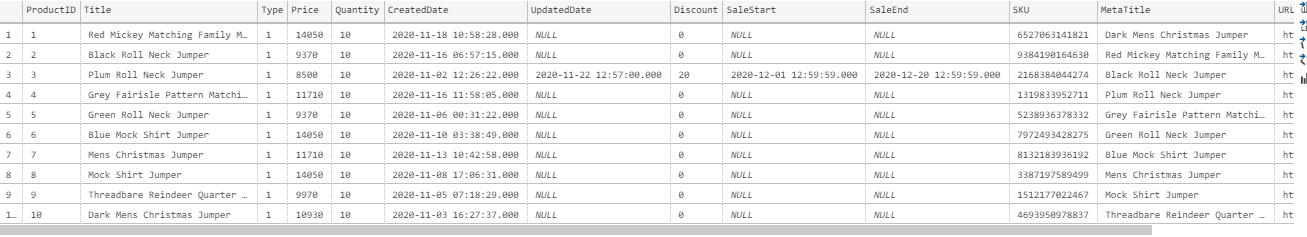
***Example2 to Delete Procedure:***

Before executing this procedure our **Product** table have record in row where *ProductId* = 11;



******

After executing procedure my row with *ProductId* = 11 will deleted;

******