Physical Database Implementation Report

150116048 – Kevser İldeş

150116001 – Çağla Şen

What our project is about?

Our project's target audience is the supermarket owners. They can keep track of stocks with our database system. Our database system will help firstly to the manager of the supermarket with giving ease of administration to him/her, increasing profits. It will secondly help to the staff with many services which are:

- Labeling products
- Stock tracking
- Keeping customer information and giving a bunch of services to them
- Keeping product info which can be vitally important sometimes (for example throwing away the products which are expired, or will be expire soon)
- Keeping employee information
- Keeping the expenses of the company

and the list of beneficial features goes on like that...

Business Rules

- A stock may have many archived prices. If there is an archived price, there should be one and only one stock.
- A rayon should consist of at least one stock and each stock should be in only one rayon.
- Each stock should have a brand and there may be many stocks belong to same brand.
- A stock may have many stock images and if there is a stock image, then there should be exactly one stock.
- Each stock has a category assigned.
- There are different units for each stock. Each unit has unit id and description about which unit it belongs to.
- A stock unit has unit price assigned to it and a barcode. Prices may differ for different locations and so each price has location specifies where to sell.
- For each stock there has to be at least one label that shows id and prices.
- Each stock has vat (KDV in Turkish) that describes name and ratio.
- Companies have at least one category and exactly one balance. A company also has at least one location and of course there can be several different locations assigned to it.

TABLES

STOCKCATEGORY

```sql
CREATE TABLE STOCKCATEGORY(
id int NOT NULL,
scname varchar(25) NOT NULL,
CONSTRAINT StockCategory_PK PRIMARY KEY(id)
);
```

- Scname is the name of category which can be either grocery, butcher, cosmetic or sth. else
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

BRAND

```sql
CREATE TABLE BRAND(
id int NOT NULL,
bname varchar(25) NOT NULL,
CONSTRAINT Brand_PK PRIMARY KEY(id)
);
```

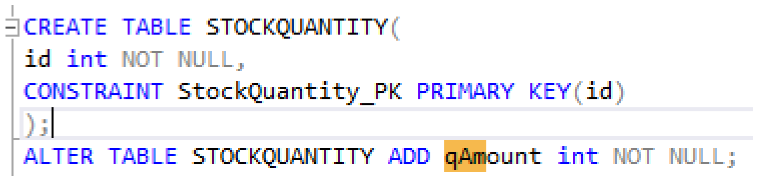
- bname is the name of the brand which can be Ülker, Rexona etc.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

REYON

```sql
CREATE TABLE REYON(
id int NOT NULL,
rname varchar(25) NOT NULL,
CONSTRAINT Reyon_PK PRIMARY KEY(id)
);
```

- rname is the coordinates which can take values like (1,3) which indicates the place of an stock in the stock room. (First corridor, third shelf)
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

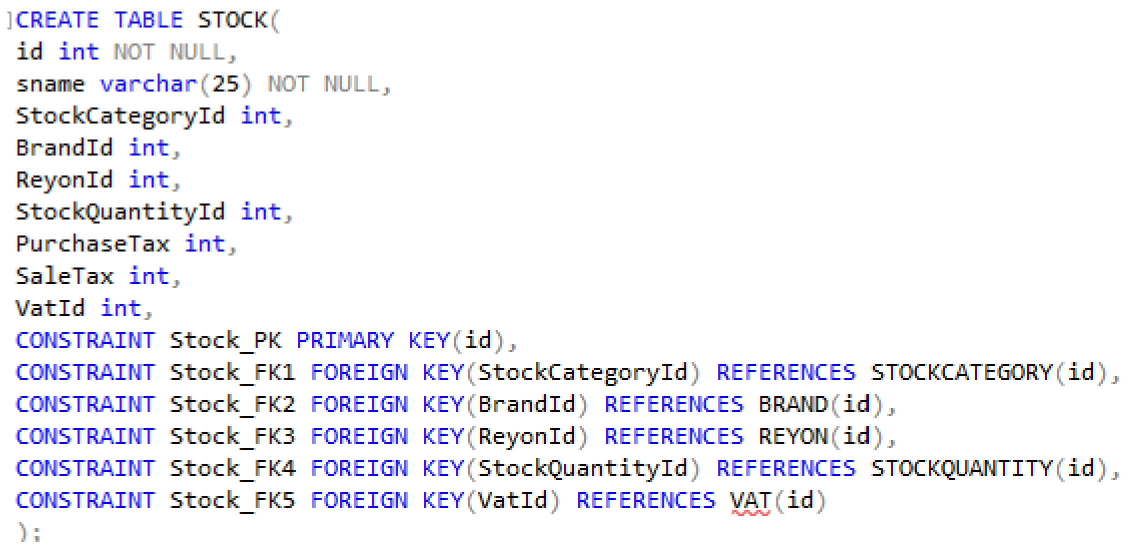
STOCKQUANTITY



```sql
CREATE TABLE STOCKQUANTITY(
id int NOT NULL,
CONSTRAINT StockQuantity_PK PRIMARY KEY(id)
);
ALTER TABLE STOCKQUANTITY ADD qAmount int NOT NULL;
```

- qAmount is the quantity amount of the stock
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

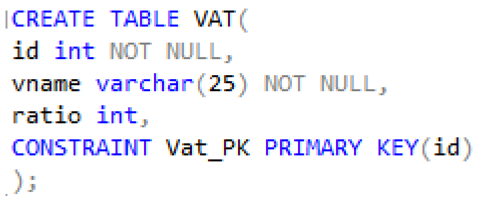
STOCK



```sql
CREATE TABLE STOCK(
id int NOT NULL,
sname varchar(25) NOT NULL,
StockCategoryId int,
BrandId int,
ReyonId int,
StockQuantityId int,
PurchaseTax int,
SaleTax int,
VatId int,
CONSTRAINT Stock_PK PRIMARY KEY(id),
CONSTRAINT Stock_FK1 FOREIGN KEY(StockCategoryId) REFERENCES STOCKCATEGORY(id),
CONSTRAINT Stock_FK2 FOREIGN KEY(BrandId) REFERENCES BRAND(id),
CONSTRAINT Stock_FK3 FOREIGN KEY(ReyonId) REFERENCES REYON(id),
CONSTRAINT Stock_FK4 FOREIGN KEY(StockQuantityId) REFERENCES STOCKQUANTITY(id),
CONSTRAINT Stock_FK5 FOREIGN KEY(VatId) REFERENCES VAT(id)
);
```

- sname is the name of the product which can be “dana döş” or “fresh deo” etc.
- SaleTax is the tax ratio for sale and PurchaseTax is the tax ratio for purchasing a product.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.
- There is foreign key CONSTRAINTS between STOCK and StockCategoryId, BrandId, ReyonId, StockQuantityId, VatId.

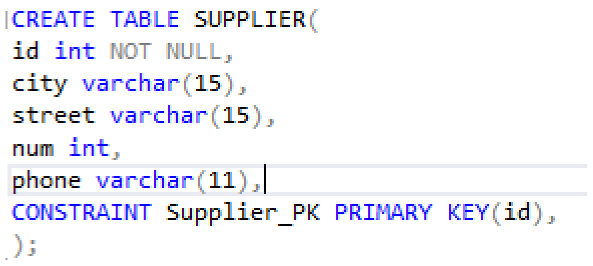
VAT



```
CREATE TABLE VAT(
id int NOT NULL,
vname varchar(25) NOT NULL,
ratio int,
CONSTRAINT Vat_PK PRIMARY KEY(id)
);
```

- vname is the tax ratio name of the product which can be either vat0 vat1 vat8 or vat18 for the ratios %0, %1, %8 or %18.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

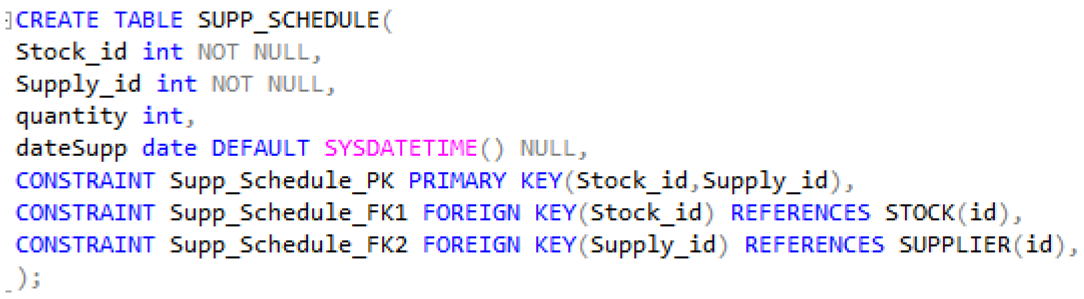
SUPPLIER



```
CREATE TABLE SUPPLIER(
id int NOT NULL,
city varchar(15),
street varchar(15),
num int,
phone varchar(11),
CONSTRAINT Supplier_PK PRIMARY KEY(id),
);
```

- City,street and num are the resident city, street and number of the supplier.
- Phone is the phone number of the supplier.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.
-

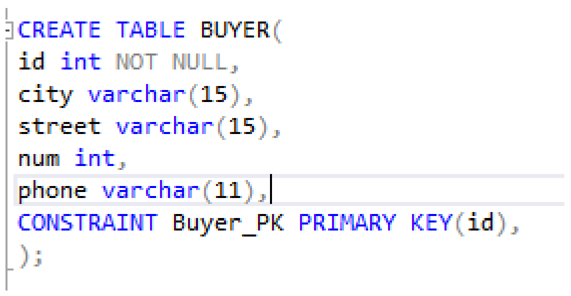
SUPPLY_SCHEDULE



```
CREATE TABLE SUPP_SCHEDULE(
Stock_id int NOT NULL,
Supply_id int NOT NULL,
quantity int,
dateSupp date DEFAULT SYSDATETIME() NULL,
CONSTRAINT Supp_Schedule_PK PRIMARY KEY(Stock_id,Supply_id),
CONSTRAINT Supp_Schedule_FK1 FOREIGN KEY(Stock_id) REFERENCES STOCK(id),
CONSTRAINT Supp_Schedule_FK2 FOREIGN KEY(Supply_id) REFERENCES SUPPLIER(id),
);
```

- Quantity is the quantity amount of the supplied stock
- dateSupp is the date which indicates when this supply event is done.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.
- There is foreign key CONSTRAINTS between SUPPLY_SCHEDULE and Stock_id, Supply_id.

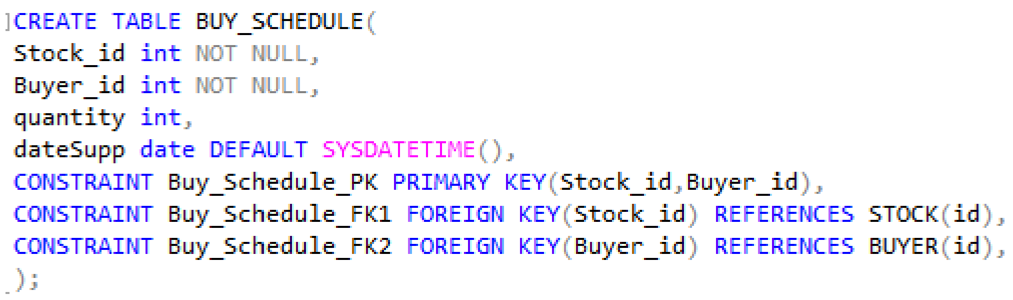
BUYER



```sql
CREATE TABLE BUYER(
id int NOT NULL,
city varchar(15),
street varchar(15),
num int,
phone varchar(11),
CONSTRAINT Buyer_PK PRIMARY KEY(id),
);
```

- City, street and num are the resident city, street and number of the buyer.
- Phone is the phone number of the buyer.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

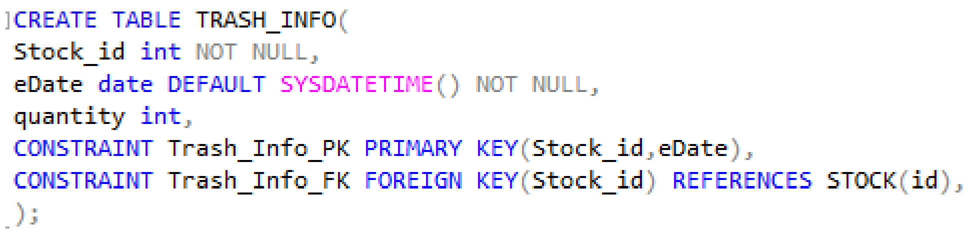
BUY_SCHEDULE



```sql
CREATE TABLE BUY_SCHEDULE(
Stock_id int NOT NULL,
Buyer_id int NOT NULL,
quantity int,
dateSupp date DEFAULT SYSDATETIME(),
CONSTRAINT Buy_Schedule_PK PRIMARY KEY(Stock_id,Buyer_id),
CONSTRAINT Buy_Schedule_FK1 FOREIGN KEY(Stock_id) REFERENCES STOCK(id),
CONSTRAINT Buy_Schedule_FK2 FOREIGN KEY(Buyer_id) REFERENCES BUYER(id),
);
```

- Quantity is the quantity amount of the purchased stock.
- dateBuyed is the date which indicates when this purchase event is done.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.
- There is foreign key CONSTRAINTS between SUPPLY_SCHEDULE and Stock_id, Buyer_id.

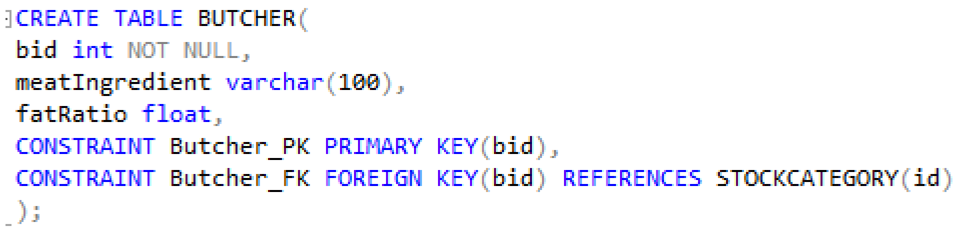
TRASH_INFO



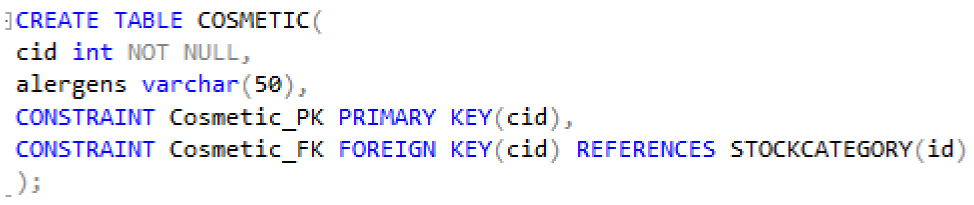
```
CREATE TABLE TRASH_INFO(
Stock_id int NOT NULL,
eDate date DEFAULT SYSDATETIME() NOT NULL,
quantity int,
CONSTRAINT Trash_Info_PK PRIMARY KEY(Stock_id,eDate),
CONSTRAINT Trash_Info_FK FOREIGN KEY(Stock_id) REFERENCES STOCK(id),
);
```

- If there is an expired product then we will create an trash info record.
- eDate is the expiration date of the expired product.
- Quantity is the quantity of the expired product.
- There is foreign key CONSTRAINTS between TRASH_INFO and Stock_id.

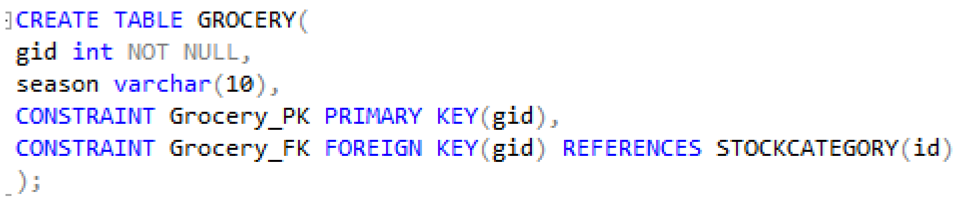
BUTCHER, COSMETIC AND GROCERY



```
CREATE TABLE BUTCHER(
bid int NOT NULL,
meatIngredient varchar(100),
fatRatio float,
CONSTRAINT Butcher_PK PRIMARY KEY(bid),
CONSTRAINT Butcher_FK FOREIGN KEY(bid) REFERENCES STOCKCATEGORY(id)
);
```



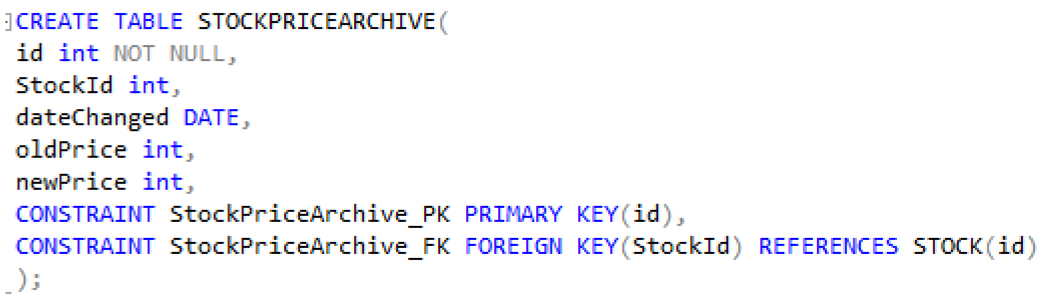
```
CREATE TABLE COSMETIC(
cid int NOT NULL,
alergens varchar(50),
CONSTRAINT Cosmetic_PK PRIMARY KEY(cid),
CONSTRAINT Cosmetic_FK FOREIGN KEY(cid) REFERENCES STOCKCATEGORY(id)
);
```



```
CREATE TABLE GROCERY(
gid int NOT NULL,
season varchar(10),
CONSTRAINT Grocery_PK PRIMARY KEY(gid),
CONSTRAINT Grocery_FK FOREIGN KEY(gid) REFERENCES STOCKCATEGORY(id)
);
```

- There is discriminative attributes in these tables as fatratio and meatIngredient in the BUTCHER and allergens in the COSMETIC etc.
- There is foreign key CONSTRAINTS between these tables and StockCategory_id.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

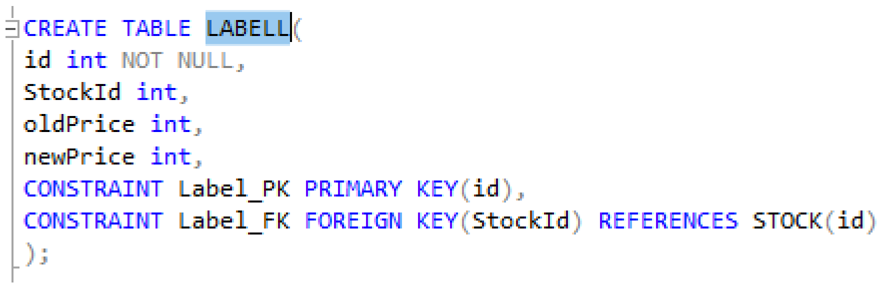
STOCKPRICEARCHIVE



```
CREATE TABLE STOCKPRICEARCHIVE(
id int NOT NULL,
StockId int,
dateChanged DATE,
oldPrice int,
newPrice int,
CONSTRAINT StockPriceArchive_PK PRIMARY KEY(id),
CONSTRAINT StockPriceArchive_FK FOREIGN KEY(StockId) REFERENCES STOCK(id)
);
```

- This table's records indicates the old and new prices of an stock.
- There is dateChanged attribute to indicate when this price is changed.

LABELL



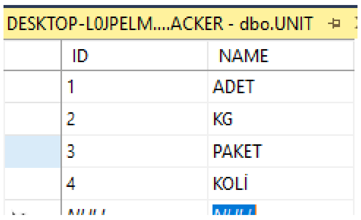
```
CREATE TABLE LABELL(
id int NOT NULL,
StockId int,
oldPrice int,
newPrice int,
CONSTRAINT Label_PK PRIMARY KEY(id),
CONSTRAINT Label_FK FOREIGN KEY(StockId) REFERENCES STOCK(id)
);
```

- **Label** is the information paper attached on the shelving. It can also give information about the older price to show that there is sale on that product.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.
- There is foreign key CONSTRAINTS between LABEL and Stock_id.

UNIT

```
CREATE TABLE UNIT(
id int NOT NULL,
uname varchar(25) NOT NULL,
CONSTRAINT Unit_PK PRIMARY KEY(id)
);
```

- Unit table is the table that holds units and their keys



DESKTOP-L0JPELM....ACKER - dbo.UNIT

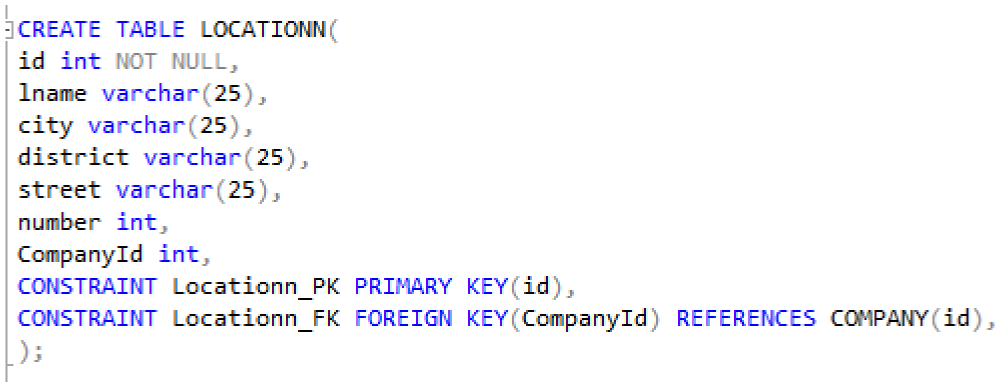
ID	NAME
1	ADET
2	KG
3	PAKET
4	KOLİ
NULL	NULL

STOCKUNIT

```
CREATE TABLE STOCKUNIT(
id int NOT NULL,
StockId int,
UnitId int,
ratio int,
referenceId int,
CONSTRAINT StockUnit_PK PRIMARY KEY(id),
CONSTRAINT StockUnit_FK1 FOREIGN KEY(StockId) REFERENCES STOCK(id),
CONSTRAINT StockUnit_FK2 FOREIGN KEY(UnitId) REFERENCES UNIT(id),
CONSTRAINT StockUnit_FK3 FOREIGN KEY(referenceId) REFERENCES STOCKUNIT(id)
);
```

- This table indicates the unit of an particular stock.
- For example let’s say there is a product called “12’li Ülker Çikolatalı Gofret” and “Ülker Çikolatalı Gofret” then the ratio of “12’li Ülker Çikolatalı Gofret” will be 12.
- There is foreign key CONSTRAINTS between STOCKUNIT and Stock_id, UnitId, referenceId.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

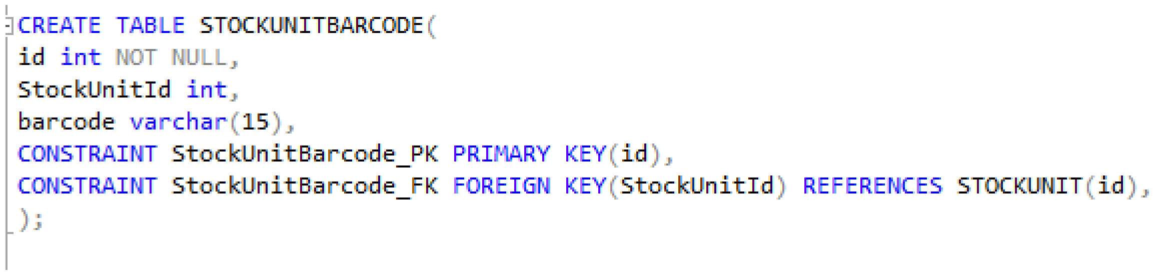
LOCATION



```
CREATE TABLE LOCATIONN(
id int NOT NULL,
lname varchar(25),
city varchar(25),
district varchar(25),
street varchar(25),
number int,
CompanyId int,
CONSTRAINT Locationn_PK PRIMARY KEY(id),
CONSTRAINT Locationn_FK FOREIGN KEY(CompanyId) REFERENCES COMPANY(id),
);
```

- It is the location of the company
- There is foreign key CONSTRAINT between LOCATION and CompanyId.

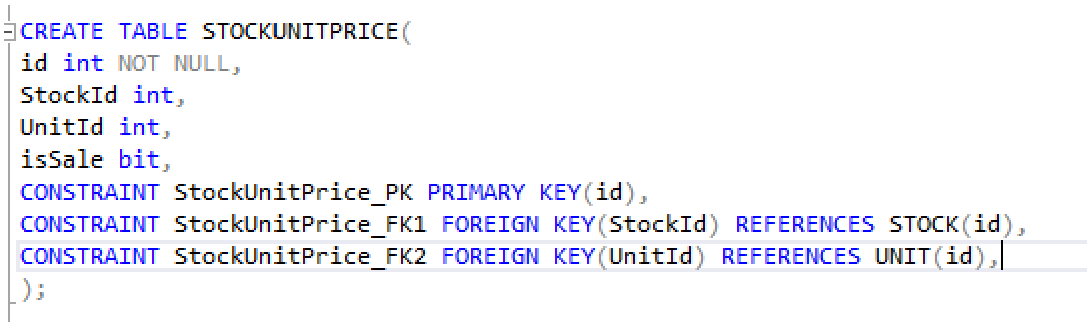
STOCKUNITBARCODE



```
CREATE TABLE STOCKUNITBARCODE(
id int NOT NULL,
StockUnitId int,
barcode varchar(15),
CONSTRAINT StockUnitBarcode_PK PRIMARY KEY(id),
CONSTRAINT StockUnitBarcode_FK FOREIGN KEY(StockUnitId) REFERENCES STOCKUNIT(id),
);
```

- It is the barcode of the stock.
- There is foreign key CONSTRAINT between STOCKUNITBARCODE and StockUnitID.

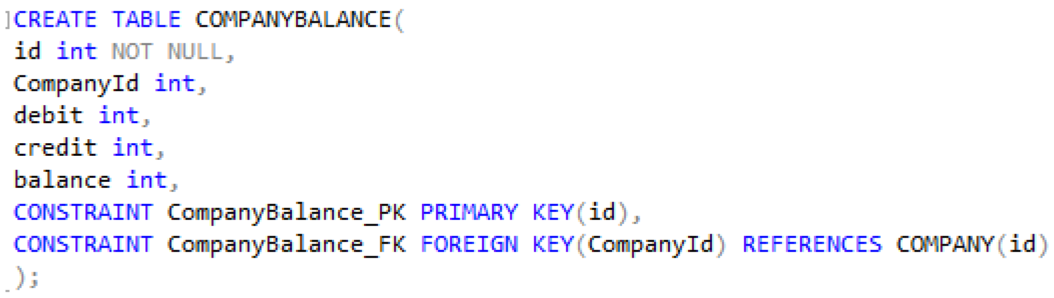
STOCKUNITPRICE



```
CREATE TABLE STOCKUNITPRICE(
id int NOT NULL,
StockId int,
UnitId int,
isSale bit,
CONSTRAINT StockUnitPrice_PK PRIMARY KEY(id),
CONSTRAINT StockUnitPrice_FK1 FOREIGN KEY(StockId) REFERENCES STOCK(id),
CONSTRAINT StockUnitPrice_FK2 FOREIGN KEY(UnitId) REFERENCES UNIT(id),
);
```

- It indicates the price of the stock unit based on the stock and it’s unit type.
- There is foreign key CONSTRAINTS between STOCKUNITPRICE and StockID, UnitID.

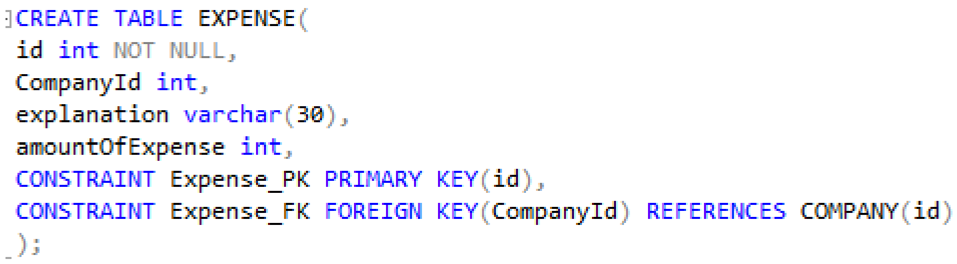
COMPANYBALANCE



```sql
CREATE TABLE COMPANYBALANCE(
id int NOT NULL,
CompanyId int,
debit int,
credit int,
balance int,
CONSTRAINT CompanyBalance_PK PRIMARY KEY(id),
CONSTRAINT CompanyBalance_FK FOREIGN KEY(CompanyId) REFERENCES COMPANY(id)
);
```

- This table indicates the balance of an company.

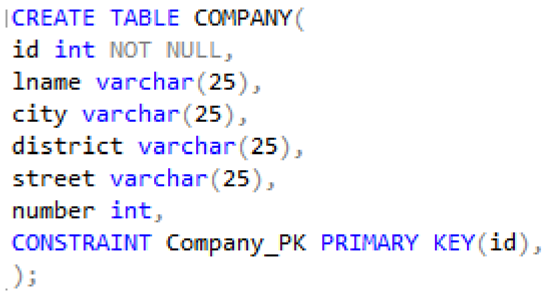
EXPENSE



```sql
CREATE TABLE EXPENSE(
id int NOT NULL,
CompanyId int,
explanation varchar(30),
amountOfExpense int,
CONSTRAINT Expense_PK PRIMARY KEY(id),
CONSTRAINT Expense_FK FOREIGN KEY(CompanyId) REFERENCES COMPANY(id)
);
```

- This table indicates an expense which can be for example: “Electricity bill 68 TL”
- There is foreign key CONSTRAINTS between EXPENSE and CompanyId.

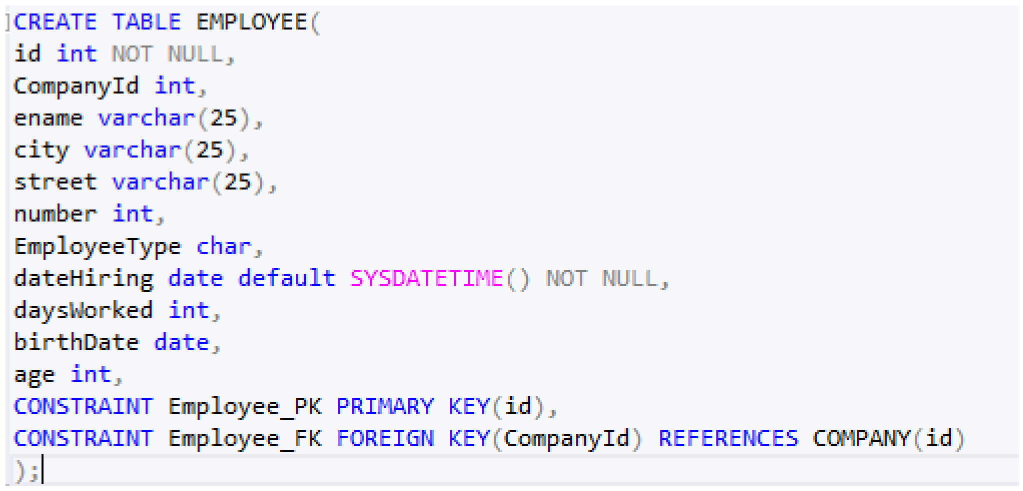
COMPANY



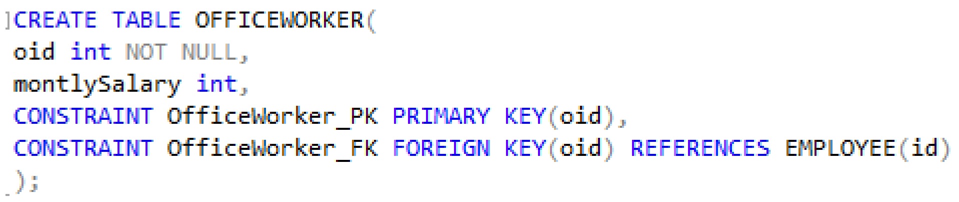
```sql
CREATE TABLE COMPANY(
id int NOT NULL,
lname varchar(25),
city varchar(25),
district varchar(25),
street varchar(25),
number int,
CONSTRAINT Company_PK PRIMARY KEY(id),
);
```

- It is simply the supermarket.
- It has address fields like city, district , street etc.
- Id must be the primary key so we added PRIMARY KEY CONSTRAINT to it.

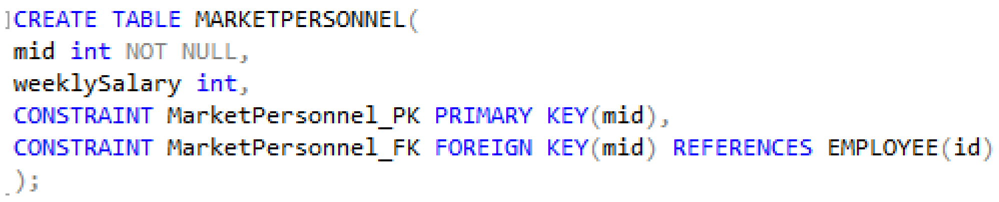
EMPLOYEE, OFFICEWORKER AND MARKETPERSONNEL



```sql
CREATE TABLE EMPLOYEE(
id int NOT NULL,
CompanyId int,
ename varchar(25),
city varchar(25),
street varchar(25),
number int,
EmployeeType char,
dateHiring date default SYSDATETIME() NOT NULL,
daysWorked int,
birthDate date,
age int,
CONSTRAINT Employee_PK PRIMARY KEY(id),
CONSTRAINT Employee_FK FOREIGN KEY(CompanyId) REFERENCES COMPANY(id)
);
```



```sql
CREATE TABLE OFFICEWORKER(
oid int NOT NULL,
montlySalary int,
CONSTRAINT OfficeWorker_PK PRIMARY KEY(oid),
CONSTRAINT OfficeWorker_FK FOREIGN KEY(oid) REFERENCES EMPLOYEE(id)
);
```

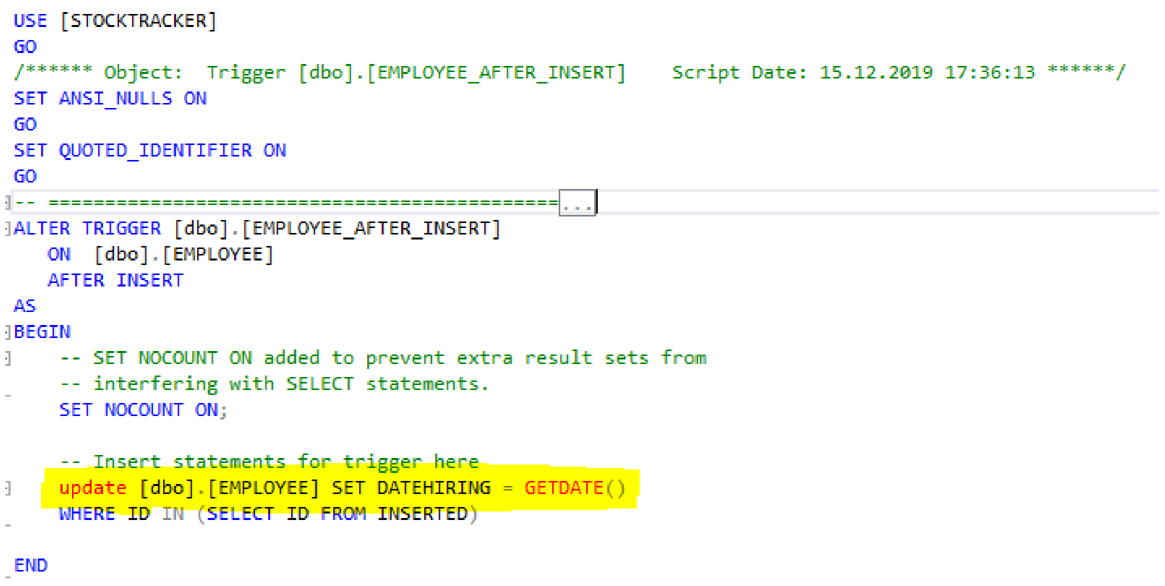


```sql
CREATE TABLE MARKETPERSONNEL(
mid int NOT NULL,
weeklySalary int,
CONSTRAINT MarketPersonnel_PK PRIMARY KEY(mid),
CONSTRAINT MarketPersonnel_FK FOREIGN KEY(mid) REFERENCES EMPLOYEE(id)
);
```

- These tables indicates the personnel that works in a particular company.
- Employee has many fields that describes him/her like the address or daysworked.
- OfficeWorker has monthlySalary and marketPersonnel has weeklySalary.

TRIGGERS

EMPLOYEE_AFTER_INSERT trigger sets the DATEHIRING time as the current time if this field is entered as null while entering the new EMPLOYEE record.

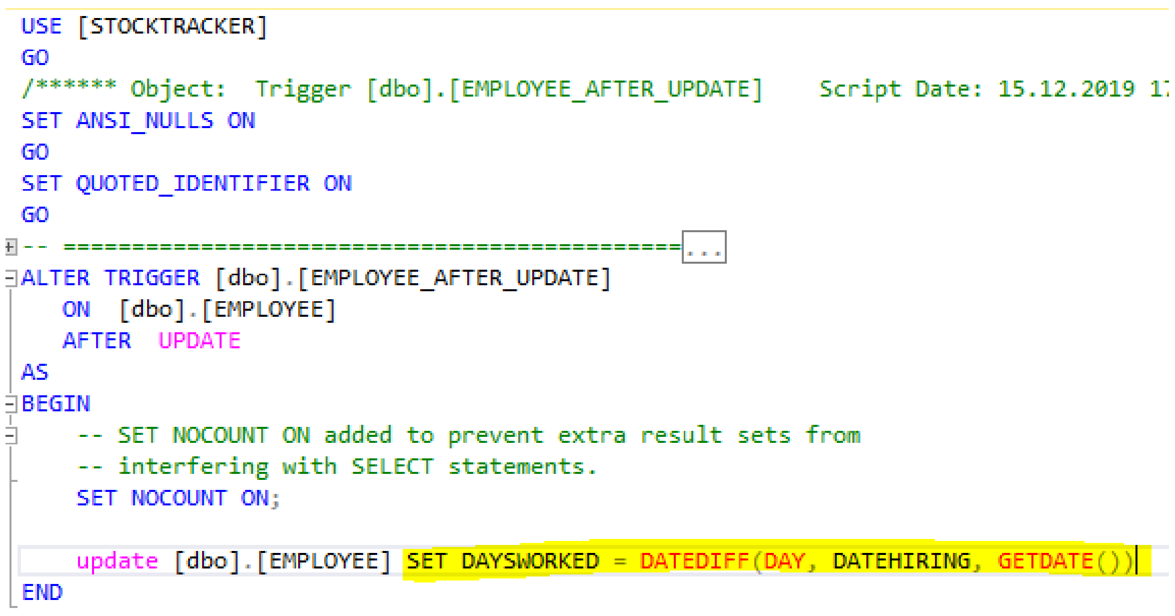


```sql
USE [STOCKTRACKER]
GO
/****** Object:  Trigger [dbo].[EMPLOYEE_AFTER_INSERT]    Script Date: 15.12.2019 17:36:13 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================...
ALTER TRIGGER [dbo].[EMPLOYEE_AFTER_INSERT]
   ON  [dbo].[EMPLOYEE]
   AFTER INSERT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    update [dbo].[EMPLOYEE] SET DATEHIRING = GETDATE()
    WHERE ID IN (SELECT ID FROM INSERTED)

END
```

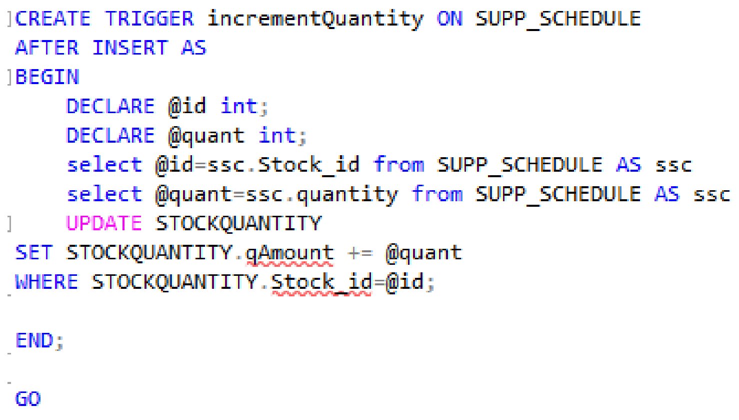
EMPLOYEE_AFTER_UPDATE trigger sets the DAYSWORKED field as the difference of current time and DATEHIRING when there is an update in the employee table.



```sql
USE [STOCKTRACKER]
GO
/****** Object:  Trigger [dbo].[EMPLOYEE_AFTER_UPDATE]    Script Date: 15.12.2019 17
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================...
ALTER TRIGGER [dbo].[EMPLOYEE_AFTER_UPDATE]
   ON  [dbo].[EMPLOYEE]
   AFTER  UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    update [dbo].[EMPLOYEE] SET DAYSWORKED = DATEDIFF(DAY, DATEHIRING, GETDATE())
END
```

incrementQuantity trigger increments quantity of an product when there is a supply schedule.

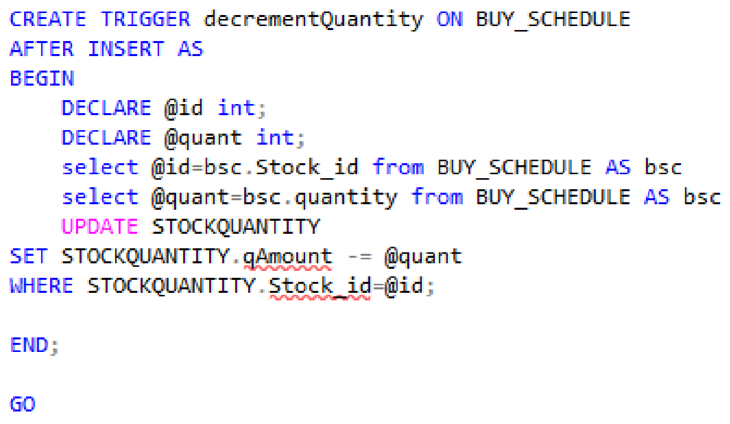


```
CREATE TRIGGER incrementQuantity ON SUPP_SCHEDULE
AFTER INSERT AS
BEGIN
    DECLARE @id int;
    DECLARE @quant int;
    select @id=ssc.Stock_id from SUPP_SCHEDULE AS ssc
    select @quant=ssc.quantity from SUPP_SCHEDULE AS ssc
    UPDATE STOCKQUANTITY
SET STOCKQUANTITY.qAmount += @quant
WHERE STOCKQUANTITY.Stock_id=@id;

END;

GO
```

decrementQuantity trigger decrements quantity of an product when there is a buy schedule.

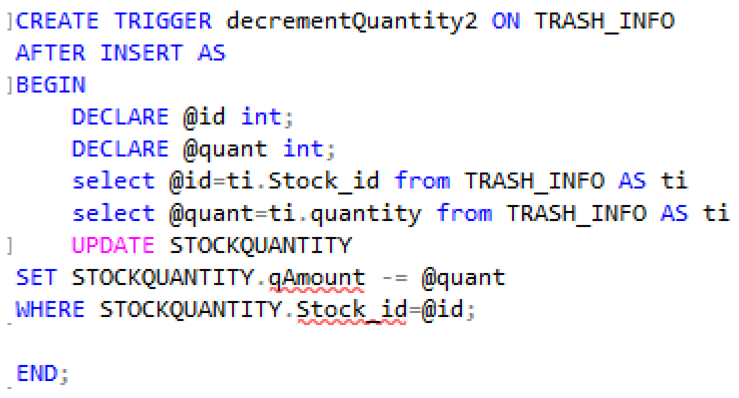


```
CREATE TRIGGER decrementQuantity ON BUY_SCHEDULE
AFTER INSERT AS
BEGIN
    DECLARE @id int;
    DECLARE @quant int;
    select @id=bsc.Stock_id from BUY_SCHEDULE AS bsc
    select @quant=bsc.quantity from BUY_SCHEDULE AS bsc
    UPDATE STOCKQUANTITY
SET STOCKQUANTITY.qAmount -= @quant
WHERE STOCKQUANTITY.Stock_id=@id;

END;

GO
```

decrementQuantity2 trigger decrements quantity of an product when there is an expired product go into the trash.

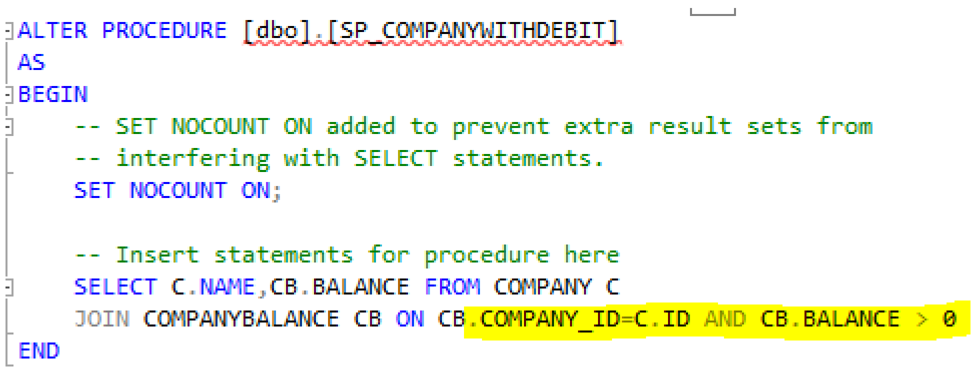


```
CREATE TRIGGER decrementQuantity2 ON TRASH_INFO
AFTER INSERT AS
BEGIN
    DECLARE @id int;
    DECLARE @quant int;
    select @id=ti.Stock_id from TRASH_INFO AS ti
    select @quant=ti.quantity from TRASH_INFO AS ti
    UPDATE STOCKQUANTITY
SET STOCKQUANTITY.qAmount -= @quant
WHERE STOCKQUANTITY.Stock_id=@id;

END;
```

Stored Procedures

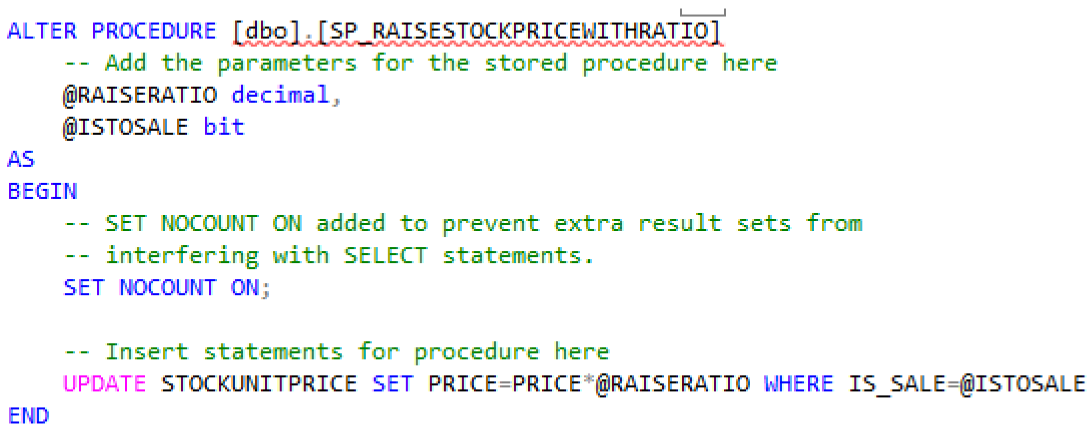
CompanyWithDebit lists the companies which have debits



```
ALTER PROCEDURE [dbo].[SP_COMPANYWITHDEBIT]
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT C.NAME,CB.BALANCE FROM COMPANY C
    JOIN COMPANYBALANCE CB ON CB.COMPANY_ID=C.ID AND CB.BALANCE > 0
END
```

RaiseStockPriceRatio raises the products prices with an given ratio

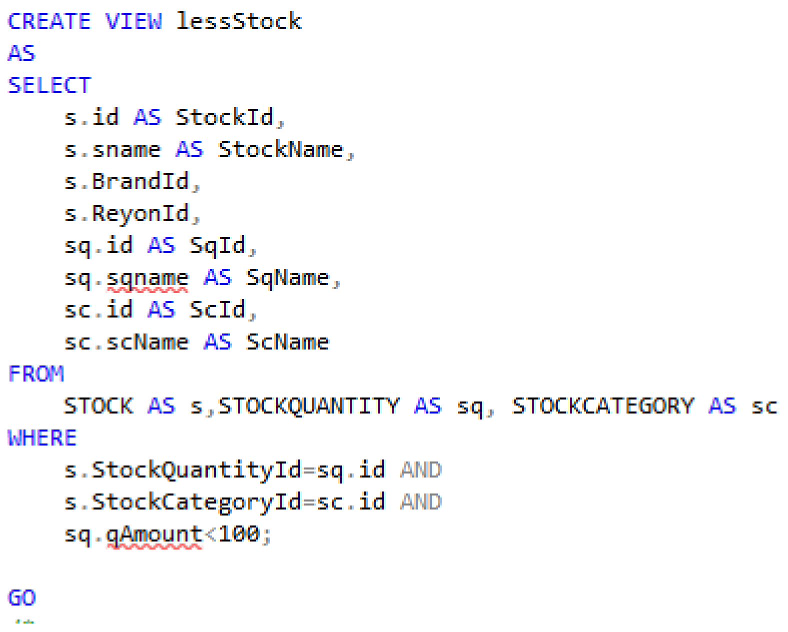


```
ALTER PROCEDURE [dbo].[SP_RAISESTOCKPRICEWITHRATIO]
    -- Add the parameters for the stored procedure here
    @RAISERATIO decimal,
    @ISTOSALE bit
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    UPDATE STOCKUNITPRICE SET PRICE=PRICE*@RAISERATIO WHERE IS_SALE=@ISTOSALE
END
```

Views

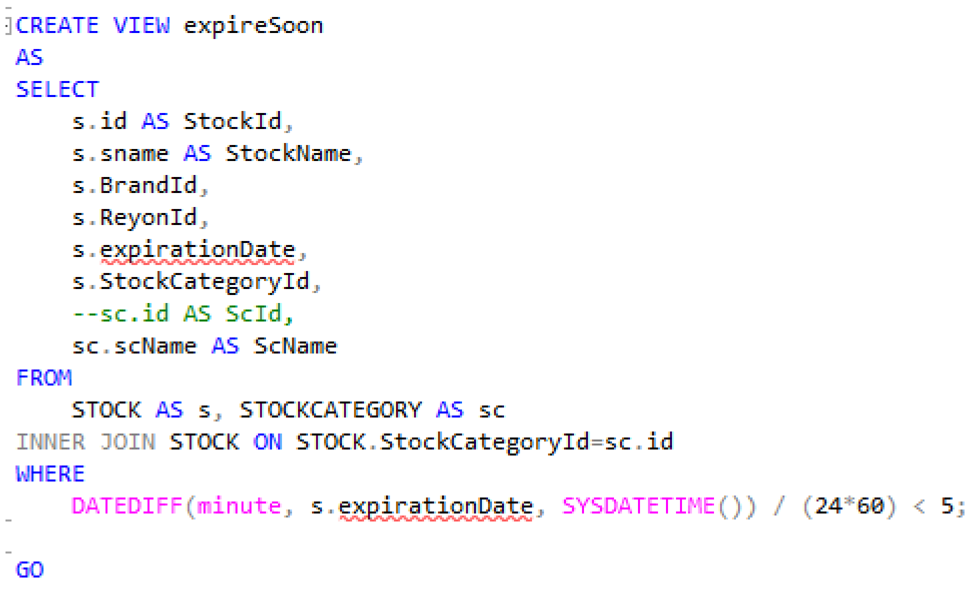
lessStock view lists the products which has amount less than 100.



```sql
CREATE VIEW lessStock
AS
SELECT
    s.id AS StockId,
    s.sname AS StockName,
    s.BrandId,
    s.ReyonId,
    sq.id AS SqId,
    sq.sqname AS SqName,
    sc.id AS ScId,
    sc.scName AS ScName
FROM
    STOCK AS s,STOCKQUANTITY AS sq, STOCKCATEGORY AS sc
WHERE
    s.StockQuantityId=sq.id AND
    s.StockCategoryId=sc.id AND
    sq.qAmount<100;

GO
```

expireSoon view lists the products which will be expire in less than 5 days.



```sql
CREATE VIEW expireSoon
AS
SELECT
    s.id AS StockId,
    s.sname AS StockName,
    s.BrandId,
    s.ReyonId,
    s.expirationDate,
    s.StockCategoryId,
    --sc.id AS ScId,
    sc.scName AS ScName
FROM
    STOCK AS s, STOCKCATEGORY AS sc
INNER JOIN STOCK ON STOCK.StockCategoryId=sc.id
WHERE
    DATEDIFF(minute, s.expirationDate, SYSDATETIME()) / (24*60) < 5;

GO
```