Primary Keys: Cannot be NULL

```
create table Shops
(
	ShopName varchar(50) NOT NULL,
	primary key(ShopName),
)

create table Users
(
	UserId int NOT NULL,
	UserName varchar(50),
	primary key(UserId),
)

create table Employees
(
	EmployeeId int NOT NULL,
	EmployeeName varchar(50),
	EmployeeSalary float CHECK(EmployeeSalary > 0),
	primary key(EmployeeId),
)
```

Assumption:
Salary cannot be less than or equal to zero, assuming the job has a pay regardless how small the amount is.

```
create table Products
(
	ProductName varchar(50) NOT NULL,
	Maker varchar(50),
	Category varchar(50),
	primary key(ProductName),
)
```

```
create table ProductsInShops
(
        PISprice float CHECK(PISprice > 0),
        PISquantity int CHECK(PISquantity > 0),
        ProductName varchar(50) NOT NULL,
        ShopName varchar(50) NOT NULL,
        foreign key (ProductName) references Products(ProductName)
        ON UPDATE CASCADE,
        foreign key (ShopName) references Shops(ShopName)
        ON UPDATE CASCADE,
        primary key (ProductName, ShopName),
)
```

Assumptions: PISprice must be more than 0
PISquantity must be more than 0, if product is in shop there should be more than 1 quantity

```
create table Feedbacks
(
        Rating int CHECK(Rating >= 0 AND Rating <= 5),
        Comment varchar(100) DEFAULT 'NIL',
        FeedbackDateTime datetime,
        ProductName varchar(50) NOT NULL,
        UserId int NOT NULL,
        foreign key (ProductName) references Products(ProductName)
        ON UPDATE CASCADE,
        foreign key (UserId) references Users(UserId)
        ON UPDATE CASCADE,
        primary key(ProductName, UserId),
)
```

Assumptions: Rating must be 0-5
If no comments are added by default it is ‘NIL’

```
create table PriceHistory
(
	PHStartDate date NOT NULL,
	PHEndDate date,
	PHPrice float CHECK (PHPrice >= 0 ),
	ProductName varchar(50) NOT NULL,
	ShopName varchar(50) NOT NULL,
	foreign key (ProductName) references Products(ProductName)
	ON UPDATE CASCADE,
	foreign key (ShopName) references Shops(ShopName)
	ON UPDATE CASCADE,
	primary key (ProductName, ShopName, PHStartDate),
	CONSTRAINT EndMoreThanStart CHECK (PHEndDate>=PHStartDate);
)
```

Assumption: Price cannot be less than or equal to zero, assuming the products have a price on it regardless of the amount.
Price History End Date cannot be less than Price History Start Date. Start is always first followed by the end

```
create table Orders
(
	OrderId int NOT NULL,
	ShippingAddress varchar(100),
	OrderDateTime datetime,
	UserId int,
	foreign key (UserId) references Users(UserId)
	ON DELETE SET NULL
	ON UPDATE CASCADE,
	primary key (OrderId),
)
```

Assumption: OrderDateTime will always be earlier than PIOdeliverydate

```
create table ProductsInOrders
(
        PIOstatus varchar(50) CHECK (PIOstatus = 'Being Processed' or PIOstatus = 'Shipped'
        or PIOstatus = 'Returned' or PIOstatus = 'Delivered'),
        PIOdeliverydate date,
        PIOquantity int,
        PIOprice float CHECK(PIOprice >= 0 ),
        ProductName varchar(50) NOT NULL,
        OrderId int NOT NULL,
        ShopName varchar(50),
        foreign key (ProductName) references Products(ProductName)
        ON UPDATE CASCADE,
        foreign key (OrderId) references Orders(OrderId)
        ON UPDATE CASCADE,
        foreign key (ShopName) references Shops(ShopName)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
        primary key (ProductName, OrderId),
)
```

Assumptions: PIOstatus only has 4 options: Being Processed, Shipped, Returned, Delivered.
PIOdeliverydate must be later than OrderDateTime from Order Table.
There will be a delivery date for each product no matter what the status is.
The tuple of ShopName and Product name must exist in ProductInShops Table.

```
create table Complaints
(
        ComplaintId int NOT NULL,
        ComplaintText varchar(100),
        ComplaintStatus varchar(50) CHECK (ComplaintStatus = 'Pending' or ComplaintStatus =
        'Being Handled' or ComplaintStatus = 'Addressed'),
        ComplaintFiled datetime,
        EmployeeId int,
        UserId int,
        ShopName varchar(50),
        OrderId int,
        HandledDateTime datetime,
        CONSTRAINT HandledMoreThanComplaint CHECK
        (HandledDateTime>=ComplaintFiled);
        CONSTRAINT IfThen CHECK ((ComplaintStatus = 'Pending' AND HandledDateTime IS
        NULL) OR ((ComplaintStatus = 'Being Handled' OR ComplaintStatus = 'Addressed')
        AND HandledDateTime IS NOT NULL));
        foreign key (EmployeeId) references Employees(EmployeeId)
        ON DELETE SET NULL,
        foreign key (UserId) references Users(UserId)
        ON DELETE SET NULL,
        foreign key (ShopName) references Shops(ShopName)
        ON DELETE SET NULL,
        foreign key (OrderId) references Orders(OrderId)
        ON DELETE SET NULL,
        primary key (ComplaintId),
)
```

Assumptions: ComplaintStatus can only have 3 options
HandledDateTime must be later than ComplaintFiled
HandledDateTime can only be NULL if ComplaintStatus is ‘pending’
HandledDateTime cannot be NULL if ComplaintStatus is ‘Being Handled’ or ‘Addressed’

```
CREATE TRIGGER dbo.NoDeliveryDateBeforeOrderDate
ON  dbo.ProductsInOrders
AFTER INSERT, UPDATE AS
IF EXISTS (
        SELECT *
        FROM ProductsInOrders, Orders
        WHERE ProductsInOrders.OrderId = Orders.OrderId
        AND ProductsInOrders.PIOdeliverydate < Orders.OrderDateTime
)
BEGIN
RAISERROR ('The ProductInOrder.DeliveryDate cannot be less than Orders.OrderDateTime',
16, 1);
ROLLBACK TRANSACTION
END
```

```
CREATE TRIGGER dbo.NoUserIdWhoHaveNoOrder
ON  dbo.Feedbacks
AFTER INSERT, UPDATE AS
IF EXISTS (
        SELECT DISTINCT UserId
        FROM Feedbacks
        Except
        SELECT DISTINCT UserId
        FROM Orders
)
BEGIN
RAISERROR ('The UserId inserted must be in Orders Table before inserting into Feedbacks
Table', 16, 1);
ROLLBACK TRANSACTION
END
```

Assumption: UserId in Feedbacks table must also be UserId in Orders Table; User can only provide feedback after they ordered.

```
CREATE TRIGGER dbo.Products_ShopsInPIOandPIS
ON  dbo.ProductsInOrders
AFTER INSERT,UPDATE
AS
IF EXISTS (
        SELECT DISTINCT ShopName, ProductName
        FROM ProductsInOrders
        EXCEPT
        SELECT DISTINCT ShopName, ProductName
        FROM ProductsInShops
)
BEGIN
RAISERROR ('The Tuple (ProductsInOrders.ShopName, ProductsInOrders.ProductName) does
not exist in ProductsinShops', 16, 1);
ROLLBACK TRANSACTION
END
```

Assumption: DeliveryDate cannot be entered before OrderDate is entered; there can only be a delivery date only after a product has been ordered