**Project Name: SHOE STORE DATABASE MANAGEMENT SYSTEM**

**Analyzed Application URL: https://www.shoeplanet.pk/**

**Class: BSCS 4**-**A**

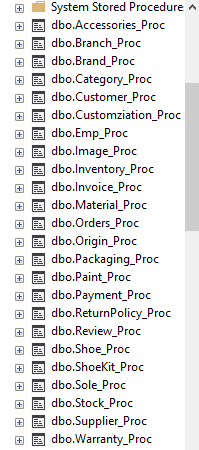
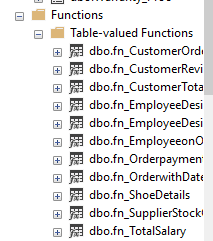
**Group Members**

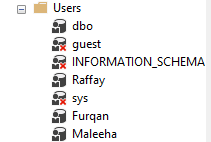
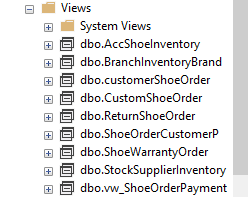
|  |  |  |
| --- | --- | --- |
| Student Name | Enrollment | Viva Marks |
| MALEEHA WAQAR | 02-134211-001 |  |
| RAFFAY AHMED KHAN | 02-134211-069 |  |
| FURQAN ZAKA UD DIN | 02-134211-077 |  |

**Project Marks**

|  |  |  |
| --- | --- | --- |
| Head | Performance | Comments |
| Analysis & Report |  |  |
| ERD |  |  |
| Normalization |  |  |
| DDL/DML/Triggers |  |  |
| Stored Proc/ Views/ Stored Functions |  |  |

1. **Analysis – Screenshots of the Application**

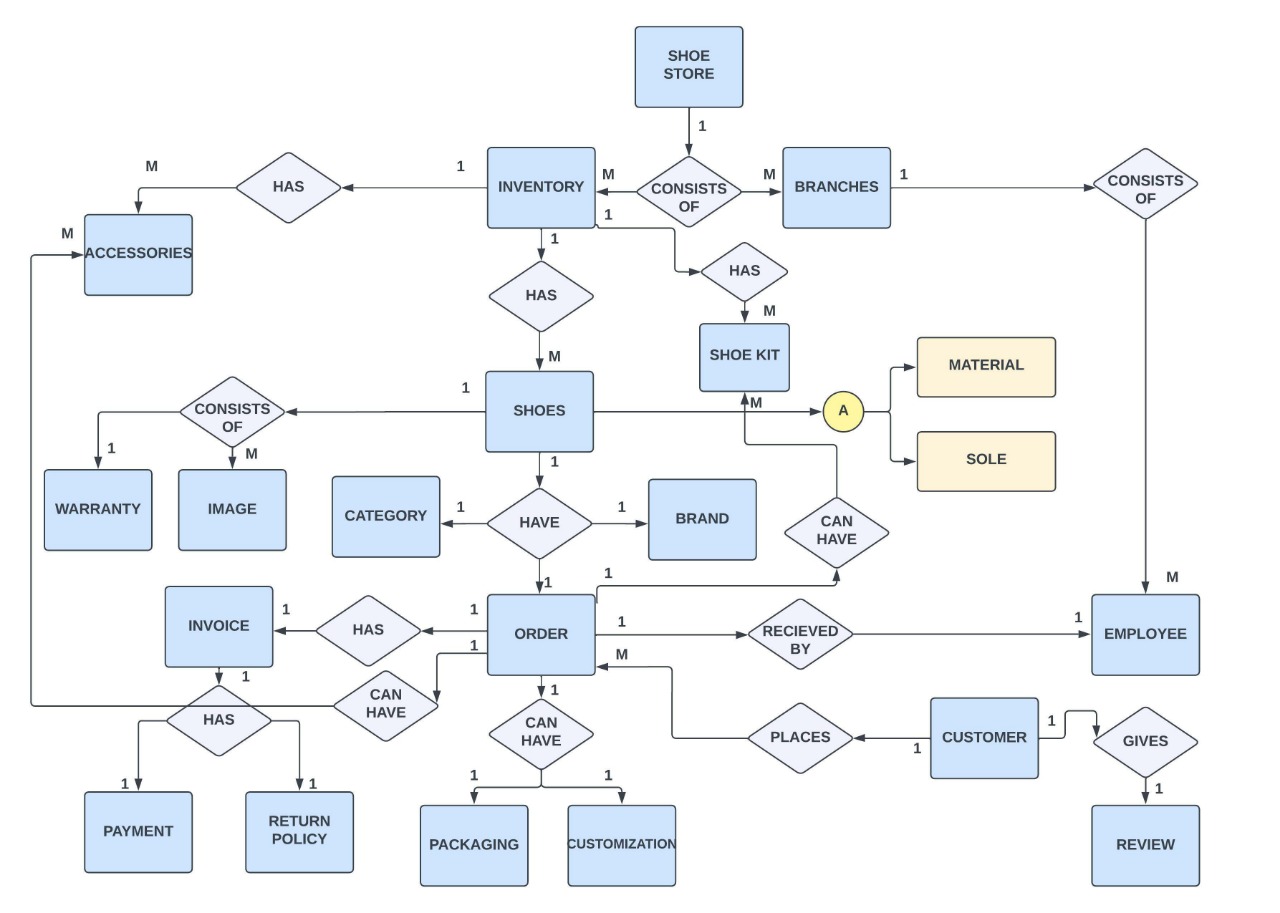
** **

** **

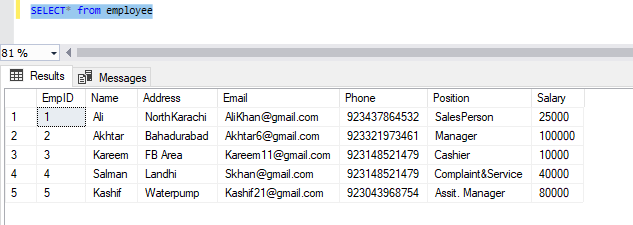
1. **Business Rules for the DBMS**

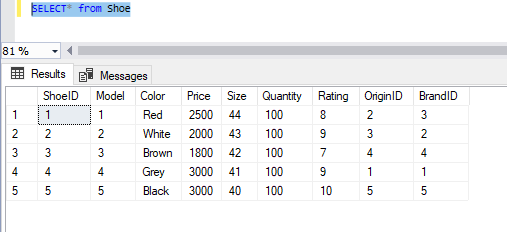
A shoe Store can have many branches that consists of many employes, it also manages many inventories that contains accessories,shoes, shoekits,All shoes must have a unique product code (ShoeID) it should also contain warranty card, images. its brand and category must be defined. a customer can place many order at time. each shoe can only be sold to one customer because it has some unique product code. an employee can manage several customers but only one at a time. after order completion a customer should be given an invoice in which return policy should be present. an order can also consist of customization and packaging (Gift wrap). each customer review should be recorded seperately.

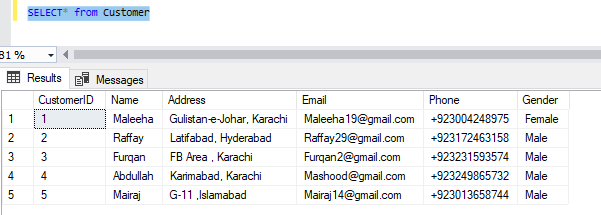
1. **Entity Relationship Diagram**

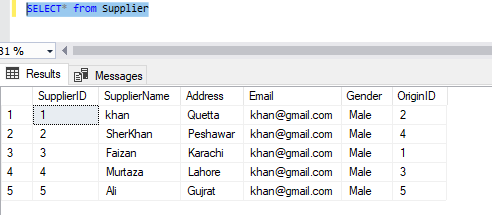
****

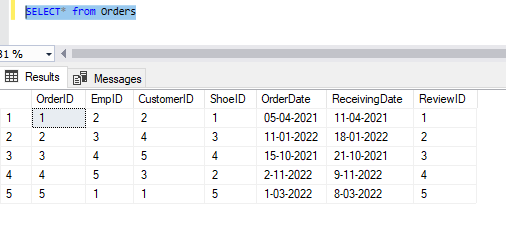
1. **Conceptual to Logical Mapping**
2. **Shoe**(ShoeID, BrandID, Model, Color, Price, size, Quantity, Origin,Rating)
3. **Customer**(CustomerID, Name, Address, Email, Phone, Gender, ReviewID)
4. **Order**(OrderId, EmpID, CustomerId, ShoeID, OrderDate, ReceivingDate)
5. **Employee**(EmpID, Name, Address, Email, Phone, Position)
6. **Inventory**(InventoryID, Loc, StockID, SupplierID, BrandID, ShoeID, Limit, EmpID)
7. **Stock**(StockId, Stockquantity, OrderDate, ReceivingDate, SupplierID, BuyingPrice, SellingPrice)
8. **Supplier**(SupplierID, SupplierName, Address, Email, Gender, OriginID)
9. **Brand**(BrandID, BrandName, OriginID, Model,CategoryID)
10. **Origin**(OriginID, OriginName, PostalCode, Address, Email)
11. **Branch**(StoreID, Location, City, StockID, EmpID, InventoryID)
12. **Accessories**( AccID, ShoeID, AccessoryName, Price, InventoryID,)
13. **ShoeKit**(KitID, Kitname, Price, ItemsIncluded)
14. **Warranty**(WarrantyID, ShoeID, WarrantyPeriod, OrderID, WarrantyEndDate)
15. **Image**(ImageID, ShoeID, ImageURL)
16. **Payment**(PaymentID, PaymentMethod, Amount, Date, OrderID)
17. **Review**(ReviewID,ShoeID,CustomerID, Review, Date)
18. **Invoice**(InoviceID, Date, CutomerID, OrderID, Amount,ShoeID)
19. **Category**(CategoryID, CategoryName, Styles)
20. **Return** **Policy**(ReturnID, ReturningDate, ReturnedDate, Reason, OrderID, ShoeID)
21. **Packaging**(PackagingID, ShoeID, Packagingtype, Recyclabity, Material)
22. **Customization**(CustomizationID, ShoeID, CustomerID, OrderID, CustomizationDetails, Cost, MaterialID, SoleID, PaintID )
23. **Material**(MaterialID, ShoeID, Name, Durablity, BReathablity, Comfort)
24. **Sole**(SoleID, ShoeID, Durablity, Flexiblity, Medicated, Hard/Soft, HeightIncreaser)
25. **Paint**(PaintID, PaintName, ShoeID, CustomizationID)
26. **Normalized Tables up to BCNF (Sql Server Schema Diagram)**
27. **Shoe**(ShoeID, BrandID, Model, Color, Price, size, Quantity, Origin,Rating)
28. **Customer**(CustomerID, Name, Address, Email, Phone, Gender, ReviewID)
29. **Order**(OrderId, EmpID, CustomerId, ShoeID, OrderDate, ReceivingDate)
30. **Employee**(EmpID, Name, Address, Email, Phone, Position)
31. **Inventory**(InventoryID, Loc, StockID, SupplierID, BrandID, ShoeID, Limit, EmpID)
32. **Stock**(StockId, Stockquantity, OrderDate, ReceivingDate, SupplierID, BuyingPrice, SellingPrice)
33. **Supplier**(SupplierID, SupplierName, Address, Email, Gender, OriginID)
34. **Brand**(BrandID, BrandName, OriginID, Model,CategoryID)
35. **Origin**(OriginID, OriginName, PostalCode, Address, Email)
36. **Branch**(StoreID, Location, City, StockID, EmpID, InventoryID)
37. **Accessories**( AccID, ShoeID, AccessoryName, Price, InventoryID,)
38. **ShoeKit**(KitID, Kitname, Price, ItemsIncluded)
39. **Warranty**(WarrantyID, ShoeID, WarrantyPeriod, OrderID, WarrantyEndDate)
40. **Image**(ImageID, ShoeID, ImageURL)
41. **Payment**(PaymentID, PaymentMethod, Amount, Date, OrderID)
42. **Review**(ReviewID,ShoeID,CustomerID, Review, Date)
43. **Invoice**(InoviceID, Date, CutomerID, OrderID, Amount,ShoeID)
44. **Category**(CategoryID, CategoryName, Styles)
45. **Return** **Policy**(ReturnID, ReturningDate, ReturnedDate, Reason, OrderID, ShoeID)
46. **Packaging**(PackagingID, ShoeID, Packagingtype, Recyclabity, Material)
47. **Customization**(CustomizationID, ShoeID, CustomerID, OrderID, CustomizationDetails, Cost, MaterialID, SoleID, PaintID )
48. **Material**(MaterialID, ShoeID, Name, Durablity, BReathablity, Comfort)
49. **Sole**(SoleID, ShoeID, Durablity, Flexiblity, Medicated, Hard/Soft, HeightIncreaser)
50. **Paint**(PaintID, PaintName, ShoeID, CustomizationID)
51. **ShoesCatBrand**(ShoeID, BRandID, OriginID, CategoryID)
52. **InovicePaymentReturn**(InoviceID, PaymentID, ReturnID)
53. **OrderPackageCustomize**(OrderID, CustomerID, PAckagingID, CustomizationID)
54. **ShoesImageWarranty**(ShoeID, ImageID, WarrantyID)
55. **ReviewCustomer** (CustomerID, KitID, AccID, ReviewID, ShoeID, OrderID)
56. **InventoryShoeBranch**(InventoryID, BranchID, ShoeID, StockID)
57. **Sample Reports (6 samples)**

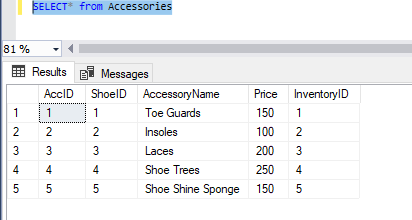
****

****

****

****

****

****

1. **Sample DDL/DML/Triggers/Stored Proc/Views/Stored Functions**

**DDL OPERATIONS:**

create database ShoeStoreDBMS

create table Origin(

OriginID int,

OriginName varchar(25),

PostalCode int,

Address varchar(50),

Email varchar(20),

constraint OriginID\_pk primary key(OriginID))

create table Brand(

BrandID int,

BrandName varchar(20),

OriginID int,

CategoryID int,

constraint BrandID\_pk primary key(BrandID),

constraint OriginID\_fk foreign key(OriginID) references Origin,

constraint CategoryID\_fk foreign key(CategoryID) references Category

)

create table Shoe(

ShoeID int,

Model int,

Color varchar(10),

Price int,

Size int not null,

Quantity int not null,

Rating int,

OriginID int,

BrandID int,

constraint ch\_model check (Model != null),

constraint OriginID1\_fk foreign key(OriginID) references Origin,

constraint ShoeID1\_pk primary key(ShoeID),

constraint BrandID1\_fk foreign key(BrandID) references Brand

)

create table Customer(

CustomerID int,

Name varchar(10) not null,

Address varchar(50) not null,

Email varchar(20),

Phone int not null,

Gender varchar(10),

constraint CustomerID\_pk primary key(CustomerID)

)

create table Orders(

OrderID int,

EmpID int,

CustomerID int,

ShoeID int,

OrderDate varchar(25) not null,

ReceivingDate varchar(25) not null,

ReviewID int,

constraint OrderID\_pk primary key(OrderID),

constraint ReviewID\_fk foreign key(ReviewID) references Review

)

create table Employee(

EmpID int,

Name varchar(25) not null,

Address varchar(50) not null ,

Email varchar(25),

Phone int not null,

Position varchar(25) not null,

constraint EmpID\_pk primary key(EmpID)

)

create table Inventory(

InventoryID int,

Loc varchar(20),

StockID int,

SupplierID int,

BrandID int,

ShoeID int,

Limit int,

EmpID int,

constraint InventoryID\_pk primary key(InventoryID),

constraint StockID2\_fk foreign key(StockID) references Stock,

constraint SupplierID1\_fk foreign key(SupplierID) references Supplier,

constraint BrandID2\_fk foreign key(BrandID) references Brand,

constraint ShoeID2\_fk foreign key(ShoeID) references Shoe,

constraint EmpID2\_fk foreign key(EmpID) references Employee

)

create table Stock(

StockID int,

Quantity int,

OrderDate varchar(30),

RecevingDate varchar(25),

SupplierID int,

BuyingPrice int,

SellingPrice int,

constraint StockID\_pk primary key(StockID),

constraint SupplierID\_fk foreign key(SupplierID) references Supplier

)

create table Supplier(

SupplierID int,

SupplierName varchar(25) not null,

Address varchar(25) not null,

Email varchar(25),

Gender varchar(10),

OriginID int,

constraint SupplierID2\_pk primary key(SupplierID),

constraint OriginID2\_fk foreign key(OriginID) references Origin

)

create table Branch(

BranchID int,

Loc varchar(20) not null,

City varchar(10) not null,

StockID int,

EmpID int,

InventoryID int,

constraint BranchID\_pk primary key(BranchID),

constraint StockID\_fk foreign key(StockID) references Stock,

constraint EmpID\_fk foreign key(EmpID) references Employee,

constraint InventoryID\_fk foreign key(InventoryID) references Inventory

)

create table Accessories (

AccID int, ShoeID int, AccessoryName varchar(30) not null, Price int, InventoryID int

constraint prime\_Acc primary key (AccID)

constraint foreign\_Acc foreign key references Shoe (ShoeID),

constraint foriegn\_Acc1 foreign key(InventoryID) references Inventory

)

Create table ShoeKit(

KitID int, KitName varchar(30) not null, Price int, ItemsIncluded varchar(50)

constraint prime\_ShoeKit primary key (KitID)

)

create table Warranty (

WarrantyID int, ShoeID int, WarrantyPeriod varchar(20) , OrderID int, WarrantyEndDate Varchar(20)

constraint Prime\_Warranty primary key (WarrantyID)

constraint foreign\_Warranty foreign key(ShoeID) references Shoe,

constraint foreign\_Warranty1 foreign key(OrderID) references Orders

)

create table Image(

ImageID int, ShoeID int, ImageURL varchar(80)

constraint prime\_Image primary key (ImageID)

constraint foreign\_Image foreign key(ShoeID) references Shoe

)

create Table Payment(

PaymentID int, PaymentMethod varchar(20) not null, Amount int not null, Date varchar(20) not null, OrderID int

constraint prime\_Payment primary key (PaymentID)

constraint foreign\_Payment foreign key(OrderID) references Orders

)

create table Review(

ReviewID int,

ShoeID int,

CustomerID int,

Review varchar(50),

Date varchar(25),

constraint ShoeID\_fk foreign key(ShoeID) references Shoe,

constraint CustomerID\_fk foreign key(CustomerID) references Customer,

constraint ReviewID\_pk primary key(ReviewID)

)

create table Invoice(

InvoiceID int,

Date varchar(25) not null,

CustomerID int,

OrderID int,

Amount int,

ShoeID int,

constraint InvoiceID3\_pk primary key(InvoiceID),

constraint CustomerID3\_fk foreign key(CustomerID) references Customer,

constraint OrderID3\_fk foreign key(OrderID) references Orders,

constraint ShoeID3\_fk foreign key(ShoeID) references Shoe

)

create table Category(

CategoryID int,

CategoryName varchar(20) not null,

Style varchar(20),

constraint CategoryID\_pk primary key(CategoryID)

)

create table ReturnPolicy(

ReturnID int,

ReturningDate varchar(20) not null,

ReturnedDate varchar(20) not null,

Reason varchar(30) not null,

OrderID int,

ShoeID int,

constraint ReturnID4\_pk primary key(ReturnID),

constraint OrderID4\_fk foreign key(OrderID) references Orders,

constraint ShoeID4\_fk foreign key(ShoeID) references Shoe

)

create table Packaging(

PackagingID int,

ShoeID int,

Packagingtype varchar(15),

Recyclablity varchar(10),

Material varchar(15)

constraint PackagingID\_pk primary key(PackagingID)

)

create table Customization (

customizationID int,

ShoeID int,

CustomerID int,

OrderID int,

CustomizationDetails varchar(150),

cost int, MaterialID int, SoleID int, PaintID int

constraint prime\_Customization primary key (customizationID)

constraint foreign\_customization foreign key references Shoe (ShoeID),

constraint foreign\_customization1 foreign key(customerID) references customer ,

constraint foreign\_customization2 foreign key(orderID) references Orders,

constraint foreign\_customization3 foreign key(MaterialID) references Material,

constraint foreign\_customization4 foreign key(SoleID) references Sole,

constraint foreign\_customization5 foreign key(PaintID) references Paint

)

create table Material (

MaterialID int, ShoeID int,

Name varchar(50), Durability varchar(10), Breathability varchar(10), Comfort varchar(10),

constraint prime\_Material primary key (MaterialID),

constraint foreign\_Material foreign key(ShoeID) references Shoe,

constraint ch\_mat check( Name != null)

)

create table Sole (

SoleID int, ShoeID int, Durability varchar (10), Flexibility varchar(10),

Medicated varchar(10), HardOrSoft varchar(10), HeightIncreaser varchar(10),

constraint Sole\_pk primary key (SoleID),

constraint foreign\_Sole foreign key (ShoeID) references Shoe

)

create table Paint(

PaintID int, PaintName varchar(50), ShoeID int,

constraint prime\_Paint primary key (PaintID),

constraint foreign\_Paint foreign key (ShoeID) references Shoe,

)

create table ShoesCategoryBrand (

ShoeID int, BrandID int, OriginID int, CategoryID int

constraint foreign\_SCB foreign key(ShoeID) references Shoe ,

constraint foreign\_SCB1 foreign key(BrandID) references Brand,

constraint foreign\_SCB2 foreign key(OriginID) references Origin,

constraint foreign\_SCB3 foreign key(CategoryID) references Category

)

create table InvoicePaymentReturn (

InvoiceID int, PaymentID int, ReturnID int

constraint foreign\_IPR foreign key(InvoiceID) references Invoice,

constraint foreign\_IPR1 foreign key(PaymentID) references Payment ,

constraint foreign\_IPR2 foreign key(ReturnID) references ReturnPolicy

)

create table OrderPackageCustomize (

OrderID int, CustomerID int, PackagingID int, CustomizationID int

constraint foreign\_OPC foreign key(OrderID) references Orders,

constraint foreign\_OPC1 foreign key(CustomerID) references Customer,

constraint foreign\_OPC2 foreign key(PackagingID) references Packaging,

constraint foreign\_OPC3 foreign key(CustomizationID) references Customization

)

create table ShoeImageWarranty (

ShoeID int, ImageID int, WarrantyID int

constraint foreign\_SIW foreign key references Shoe(ShoeID),

constraint foreign\_SIW1 foreign key(ImageID) references Image,

constraint foreign\_SIW2 foreign key(WarrantyID) references Warranty

)

create table ReviewCustomer (

CustomerID int, KitID int, AccID int, ReviewID int, ShoeID int, OrderID int

constraint foreign\_RC foreign key(CustomerID) references Customer,

constraint foreign\_RC1 foreign key(KitID) references ShoeKit,

constraint foreign\_RC2 foreign key(AccID) references Accessories,

constraint foreign\_RC3 foreign key(ReviewID) references Review,

constraint foreign\_RC4 foreign key(ShoeID) references Shoe,

constraint foreign\_RC5 foreign key(OrderID) references Orders

)

create table InventoryShoeBranch (

InventoryID int, BranchID int, ShoeID int, StockID int

constraint foreign\_ISB foreign key(InventoryID) references Inventory,

constraint foreign\_ISB1 foreign key(BranchID) references Branch,

constraint foreign\_ISB2 foreign key(ShoeID) references Shoe,

constraint foreign\_ISB3 foreign key(StockID) references Stock

)

**DML OPERATIONS:**

insert into Origin

values (1,'Vietnam',245150,'plot 4 street 7','Vshoes@hotmail.com'),

(2,'China',200000,'plot 18 street 1','Cshoes@hotmail.com'),

(3,'Combodia',20870,'plot 3 street 7','Comshoes@hotmail.com'),

(4,'Pakistan',75950,'plot 49 street 2','Fource@gmail.com'),

(5,'India',75987,'plot 6 street 10','batashoes@gmail.com')

insert into Category

values (1,'Sneakers','High-Tops'),

(2,'Dress Shoes','Oxford'),

(3,'Runnimg Shoes','Athletic'),

(4,'Boots','Chelsea Boots'),

(5,'Casual Shoes','Lows')

ALTER TABLE Customer

ALTER COLUMN Phone VARCHAR(20);

insert into Customer

values (1,'Maleeha','Gulistan-e-Johar, Karachi','Maleeha19@gmail.com','+923004248975','Female'),

(2,'Raffay','Latifabad, Hyderabad','Raffay29@gmail.com','+923172463158','Male'),

(3,'Furqan','FB Area , Karachi','Furqan2@gmail.com','+923231593574','Male'),

(4,'Abdullah','Karimabad, Karachi','Mashood@gmail.com','+923249865732','Male'),

(5,'Mairaj','G-11 ,Islamabad','Mairaj14@gmail.com','+923013658744','Male')

ALTER TABLE Employee

ALTER COLUMN Phone VARCHAR(20);

insert into Employee

values (1,'Ali','NorthKarachi','Alikhan@gmail.com','+923437864532','SalesPerson'),

(2,'Akhtar','Bahadurabad','Akhtar6@gmail.com','+923321973461','Manager'),

(3,'Kareem','FB Area','Kareem11@gmail.com','+923148521479','Cashier'),

(4,'Salman','Landhi','Skhan@gmail.com','+923249876542','Complaint&Service'),

(5,'Kashif','Waterpump','Kashif21@gmail.com','+923043968754','Assit. Manager')

insert into Brand

values(1,'Adidas',1,3),

(2,'Nike',2,1),

(3,'Converse',3,5),

(4,'Zara',5,2),

(5,'Sputnik',4,4)

insert into Shoe

values

(1,1,'Red',2500,44,100,8,2,3),

(2,2,'White',2000,43,100,9,3,2),

(3,3,'Brown',1800,42,100,7,4,4),

(4,4,'Grey',3000,41,100,9,1,1),

(5,5,'Black',3000,40,100,10,5,5)

insert into Paint values

(1,'Montana white Acrylic',2),

(2,'Newton red Acrylic',1),

(3,'James brown Acrylic',3),

(4,'Young black Acrylic',5),

(5,'Wolf grey Acrylic',4)

insert into Review values

(1,2,1,'Very Comfortable','12-04-2021'),

(2,4,3,'Great Quality','19-01-2022'),

(3,5,4,'Superb Quality and amazing fit','22-10-2021'),

(4,3,2,'Very Nice material','10-11-2022'),

(5,1,5,'Loved them','9-03-2022')

insert into Supplier values

(1,'khan','Quetta','khan@gmail.com','Male',2),

(2,'SherKhan','Peshawar','khan@gmail.com','Male',4),

(3,'Faizan','Karachi','khan@gmail.com','Male',1),

(4,'Murtaza','Lahore','khan@gmail.com','Male',3),

(5,'Ali','Gujrat','khan@gmail.com','Male',5)

insert into Orders values

(1,2,2,1,'05-04-2021','11-04-2021',1),

(2,3,4,3,'11-01-2022','18-01-2022',2),

(3,4,5,4,'15-10-2021','21-10-2021',3),

(4,5,3,2,'2-11-2022','9-11-2022',4),

(5,1,1,5,'1-03-2022','8-03-2022',5)

alter table Packaging

alter column Material varchar(25);

insert into Packaging values

(1,4,'Corrugated Box','YES','corrugated paper'),

(2,3,'Corrugated Box','YES','corrugated paper'),

(3,2,'Corrugated Box','YES','corrugated paper'),

(4,1,'Corrugated Box','YES','corrugated paper'),

(5,5,'Corrugated Box','YES','corrugated paper')

insert into Payment values

(1,'Cash on Delivery',2600,'11-04-2021',1),

(2,'Cash on Delivery',1900,'18-01-2022',2),

(3,'Bank Transfer',3000,'21-10-2021',3),

(4,'Cash on Delivery',2100,'9-11-2022',4),

(5,'Bank Trasnfer',3000,'8-03-2022',5)

insert into ReturnPolicy values

(1,'19-01-2022','22-01-2022','Not my size',3,4),

(2,'09-03-2022','12-03-2022','Sent wrong Article',5,5)

insert into ShoeKit values

(1,'Sneaker Care',300,'cleaning solution, scrub brush, microfiber cloth'),

(2,'Boot Savvy',350,'boot wax, brush, cloth'),

(3,'Shoe Spa',500,' leather cleaner,leather conditioner, polish'),

(4,'Sole-ful Solutions',450,'shoe polish, brush, cloth, leather conditioner'),

(5,'Sole Savvy',250,'rubber patches, glue, brush, sandpaper')

insert into ShoesCategoryBrand values

(1,3,1,3),

(2,2,2,1),

(3,4,3,5),

(4,1,5,2),

(5,5,4,4)

insert into Sole values

(1,1,'YES','MEDIUM','YES','SOFT','1 INCH'),

(2,2,'YES','YES','YES','SOFT','2 INCH'),

(3,3,'YES','NO','YES','SOFT','2 INCH'),

(4,4,'YES','YES','YES','SOFT','2 INCH'),

(5,5,'YES','NO','YES','SOFT','4 INCH')

insert into Stock values

(1,100,'01-11-2020','01-12-2020',1,1000,2500),

(2,100,'02-02-2021','01-03-2021',2,1000,3000),

(3,100,'10-05-2021','01-06-2021',3,1200,3000),

(4,100,'11-07-2021','01-08-2021',4,800,2500),

(5,100,'01-09-2022','01-10-2022',5,500,2000)

insert into Warranty values

(1,1,'6 Months',1,'05-10-2021'),

(2,2,'6 Months',3,'11-07-2022'),

(3,3,'6 Months',4,'15-04-2022'),

(4,4,'6 Months',2,'02-05-2023'),

(5,5,'6 Months',5,'01-09-2022')

insert into Inventory values

(1,'Quetta',1,1,3,1,100,1),

(2,'Peshawar',1,1,2,2,100,2),

(3,'Karachi',1,1,4,3,100,3),

(4,'Lahore',1,1,1,4,100,4),

(5,'Gujrat',1,1,5,5,100,5)

insert into Image values

(1,1,'https://source.Thriftfource.com/400x300'),

(2,2,'https://source.Thriftfource.com/400x300'),

(3,3,'https://source.Thriftfource.com/400x300'),

(4,4,'https://source.Thriftfource.com/400x300'),

(5,5,'https://source.Thriftfource.com/400x300')

insert into Branch values

(1,'Pakistan','karachi',1,1,1),

(2,'Pakistan','Lahore',2,2,2),

(3,'Pakistan','Hyderabad',3,3,3),

(4,'Pakistan','Islamabad',4,4,4),

(5,'Pakistan','Quetta',5,5,5)

insert into Accessories values

(1,1,'Toe Guards',150,1),

(2,2,'Insoles',100,2),

(3,3,'Laces',200,3),

(4,4,'Shoe Trees',250,4),

(5,5,'Shoe Shine Sponge',150,5)

insert into InventoryShoeBranch values

(1,1,1,1),

(2,2,2,2),

(3,3,3,3),

(4,4,4,4),

(5,5,5,5)

insert into Invoice values

(1,'05-04-2021',2,1,2600,1),

(2,'11-01-2022',4,2,1900,3),

(3,'15-10-2021',5,3,3000,4),

(4,'02-11-2022',3,4,2100,2),

(5,'01-03-2022',1,5,3000,5)

insert into InvoicePaymentReturn values

(1,1,1),

(2,2,2)

insert into Material values

(1,1,'Canvas','YES','YES','YES'),

(2,2,'Suede','YES','YES','YES'),

(3,3,'Leather','YES','NO','YES'),

(4,4,'Mesh','YES','YES','YES'),

(5,5,'Leather','YES','NO','YES')

insert into ShoeImageWarranty values

(1,1,1),

(2,2,2),

(3,3,3),

(4,4,4),

(5,5,5)

insert into Customization values

(1,1,1,1,'Neon Green and Black High-Top Sneakers with Silver Embroidery',1200,1,1,1),

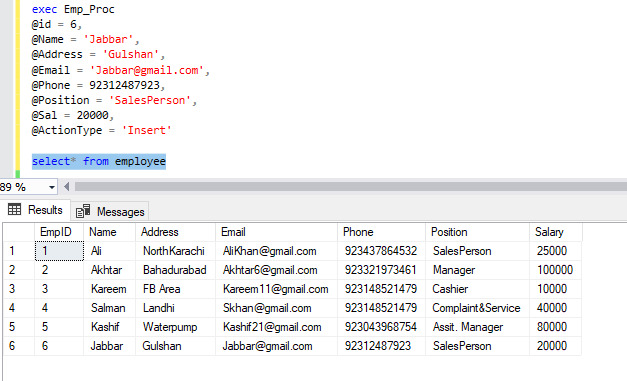
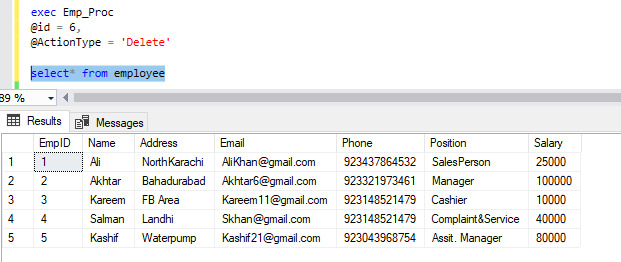
(2,2,2,2,' Blue and White Low-Top Sneakers with Metallic Rose Gold Trim',1500,2,2,2),

(3,3,3,3,' Gray High-Top Sneakers with Hot Pink Stripes and Glitter Laces',1150,3,3,3),

(4,4,4,4,'White and Black Mid-Top Sneakers with Animal Print Accents',1300,4,4,4),

(5,5,5,5,'Red and Black Low-Top Sneakers with Contrast Stitching and Studs',1200,5,5,5)

**PROCEDURES:**

****

Create Procedure Origin\_Proc

(

@Id INTEGER,

@OriginName VARCHAR(20) = NULL,

@PostalCode int = NULL,

@Address varchar(50) = NULL,

@Email VARCHAR(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Origin(OriginID, OriginName, PostalCode, Address, Email) values( @Id, @OriginName, @PostalCode, @Address, @Email)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Origin

END

IF @ActionType = 'Update'

BEGIN

UPDATE Origin SET

OriginName = @OriginName,

PostalCode = @PostalCode,

Address = @Address,

Email = @Email

WHERE OriginID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Origin Where OriginID = @Id

END

END

Create Procedure Brand\_Proc

(

@Id INTEGER,

@BrandName VARCHAR(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Brand(BrandID, BrandName) values( @Id, @BrandName)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Brand

END

IF @ActionType = 'Update'

BEGIN

UPDATE Brand SET

BrandName = @BrandName

WHERE BrandID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Brand Where BrandID = @Id

END

END

Create Procedure Shoe\_Proc

(

@Id INTEGER,

@Model int = NULL,

@Color varchar(10) = NULL,

@Price int = NULL,

@Size int = NULL,

@Quantity int = NULL,

@Rating int = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Shoe(ShoeID, Model, Color, Price, Size, Quantity, Rating) values( @Id, @Model, @Color, @Price, @Size, @Quantity, @Rating)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Shoe

END

IF @ActionType = 'Update'

BEGIN

UPDATE Shoe SET

Model = @Model,

Color = @Color,

Price = @Price,

Size = @Size,

Quantity = @Quantity,

Rating = @Rating

WHERE ShoeID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Shoe Where ShoeID = @Id

END

END

Create Procedure Customer\_Proc

(

@Id INTEGER,

@Name varchar(10) = NULL,

@Address varchar(50) = NULL,

@Email varchar(20) = NULL,

@Phone varchar(20) = NULL,

@Gender varchar(10) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Customer(CustomerID, Name, Address, Email, Phone, Gender) values( @Id, @Name, @Address, @Email, @Phone, @Gender)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Shoe

END

IF @ActionType = 'Update'

BEGIN

UPDATE Customer SET

Name = @Name,

Address = @Address,

Email = @Email,

Phone = @Phone,

Gender = @Gender

WHERE CustomerID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Customer Where CustomerID = @Id

END

END

Create Procedure Orders\_Proc

(

@Id INTEGER,

@OrderDate varchar(25) = NULL,

@ReceivngDate varchar(25) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Orders(OrderID, OrderDate, ReceivingDate) values( @Id, @OrderDate, @ReceivngDate)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Shoe

END

IF @ActionType = 'Update'

BEGIN

UPDATE Orders SET

OrderDate = @OrderDate,

ReceivingDate = @ReceivngDate

WHERE ShoeID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Orders Where OrderID = @Id

END

END

Create Procedure Emp\_Proc(

@Id INTEGER,

@Name varchar(25) = NULL,

@Address varchar(50) = NULL,

@Email varchar(20) = NULL,

@Phone varchar(20) = NULL,

@Position varchar(20) = NULL,

@Sal varchar(20) = NULL,

@ActionType nvarchar(20) = '' )

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Employee(EmpID, Name, Address, Email, Phone, Position, Salary) values( @Id, @Name, @Address, @Email, @Phone, @Position, @Sal)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Employee

END

IF @ActionType = 'Update'

BEGIN

UPDATE Employee SET

Name = @Name,

Address = @Address,

Email = @Email,

Phone = @Phone,

Position = @Position,

Salary = @Sal

WHERE EmpID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Employee Where EmpID = @Id

END

END

exec Emp\_Proc

@id = 6,

@ActionType = 'Delete'

select\* from employee

Create Procedure Inventory\_Proc

(

@Id INTEGER,

@loc varchar(10) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Inventory(InventoryID, Loc) values( @Id, @loc)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Shoe

END

IF @ActionType = 'Update'

BEGIN

UPDATE Inventory SET

Loc = @loc

WHERE InventoryID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Inventory Where InventoryID = @Id

END

END

Create Procedure Stock\_Proc

(

@Id INTEGER,

@Quantity int = NULL,

@Orderdate varchar(50) = NULL,

@receivngDate varchar(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Stock(StockID, Quantity, OrderDate, RecevingDate) values( @Id, @Quantity, @Orderdate, @receivngDate)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Stock

END

IF @ActionType = 'Update'

BEGIN

UPDATE Stock SET

Quantity = @Quantity,

OrderDate = @Orderdate,

RecevingDate = @receivngDate

WHERE StockID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Stock Where StockID = @Id

END

END

Create Procedure Supplier\_Proc

(

@Id INTEGER,

@Name varchar(10) = NULL,

@Address varchar(50) = NULL,

@Email varchar(20) = NULL,

@Gender varchar(10) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Supplier(SupplierID, SupplierName, Address, Email, Gender) values( @Id, @Name, @Address, @Email, @Gender)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Supplier

END

IF @ActionType = 'Update'

BEGIN

UPDATE Supplier SET

SupplierName = @Name,

Address = @Address,

Email = @Email,

Gender = @Gender

WHERE SupplierID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Supplier Where SupplierID = @Id

END

END

Create Procedure Branch\_Proc

(

@Id INTEGER,

@Loc varchar(10) = NULL,

@City varchar(50) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Branch(BranchID, Loc, City) values( @Id, @Loc, @City)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Branch

END

IF @ActionType = 'Update'

BEGIN

UPDATE Branch SET

Loc = @Loc,

City = @City

WHERE BranchID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Branch Where BranchID = @Id

END

END

Create Procedure Accessories\_Proc

(

@Id INTEGER,

@Name varchar(10) = NULL,

@Price int = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Accessories(AccID, AccessoryName, Price) values( @Id, @Name, @Price)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Accessories

END

IF @ActionType = 'Update'

BEGIN

UPDATE Accessories SET

AccessoryName = @Name,

Price = @Price

WHERE AccID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Accessories Where AccID = @Id

END

END

Create Procedure ShoeKit\_Proc

(

@Id INTEGER,

@Name varchar(10) = NULL,

@Price int = NULL,

@Items varchar(50) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into ShoeKit(KitID, KitName, Price, ItemsIncluded) values( @Id, @Name, @Price, @Items)

END

IF @ActionType = 'Select'

BEGIN

Select \* from ShoeKit

END

IF @ActionType = 'Update'

BEGIN

UPDATE ShoeKit SET

KitName = @Name,

Price = @Price,

ItemsIncluded = @Items

WHERE KitID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from ShoeKit Where KitID = @Id

END

END

Create Procedure Warranty\_Proc

(

@Id INTEGER,

@Period varchar(20) = NULL,

@enddate varchar(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Warranty(WarrantyID, WarrantyPeriod, WarrantyEndDate) values( @Id, @Period, @enddate)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Warranty

END

IF @ActionType = 'Update'

BEGIN

UPDATE Warranty SET

WarrantyPeriod = @Period,

WarrantyEndDate = @enddate

WHERE WarrantyID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Warranty Where WarrantyID = @Id

END

END

Create Procedure Image\_Proc

(

@Id INTEGER,

@Url varchar(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Image(ImageID, ImageURL ) values( @Id, @Url)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Image

END

IF @ActionType = 'Update'

BEGIN

UPDATE Image SET

ImageURL = @Url

WHERE ImageID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Image Where ImageID = @Id

END

END

Create Procedure Payment\_Proc

(

@Id INTEGER,

@Method varchar(20) = NULL,

@amount int = NULL,

@Date varchar(20),

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Payment(PaymentID, PaymentMethod, Amount, Date) values( @Id, @Method, @amount, @Date)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Payment

END

IF @ActionType = 'Update'

BEGIN

UPDATE Payment SET

PaymentMethod = @Method,

Amount = @amount

WHERE PaymentID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Payment Where PaymentID = @Id

END

END

Create Procedure Review\_Proc

(

@Id INTEGER,

@Review varchar(50) = NULL,

@Date varchar(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Review(ReviewID, Review, Date) values( @Id, @Review, @Date)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Review

END

IF @ActionType = 'Update'

BEGIN

UPDATE Review SET

Review = @Review,

Date = @Date

WHERE ReviewID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Review Where ReviewID = @Id

END

END

Create Procedure Invoice\_Proc

(

@Id INTEGER,

@Date varchar(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Invoice(InvoiceID, Date) values( @Id, @Date)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Invoice

END

IF @ActionType = 'Update'

BEGIN

UPDATE Warranty SET

@Date = @Date

WHERE WarrantyID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Invoice Where InvoiceID = @Id

END

END

Create Procedure Category\_Proc

(

@Id INTEGER,

@Name varchar(20) = NULL,

@Style varchar(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Category(CategoryID, CategoryName, Style) values( @Id, @Name, @Style)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Category

END

IF @ActionType = 'Update'

BEGIN

UPDATE Category SET

CategoryName = @Name,

@Style = @Style

WHERE CategoryID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Category Where CategoryID = @Id

END

END

Create Procedure ReturnPolicy\_Proc

(

@Id INTEGER,

@ReturningDate varchar(20) = NULL,

@ReturnedDate varchar(20) = NULL,

@Reason varchar(50) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into ReturnPolicy(ReturnID, ReturningDate, ReturnedDate, Reason) values( @Id, @ReturningDate, @ReturningDate, @Reason)

END

IF @ActionType = 'Select'

BEGIN

Select \* from ReturnPolicy

END

IF @ActionType = 'Update'

BEGIN

UPDATE ReturnPolicy SET

ReturningDate = @ReturningDate,

ReturnedDate = @ReturnedDate,

Reason = @Reason

WHERE ReturnID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from ReturnPolicy Where ReturnID = @Id

END

END

Create Procedure Packaging\_Proc

(

@Id INTEGER,

@Type varchar(20) = NULL,

@Recycle varchar(20) = NULL,

@Material varchar(20) = Null,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Packaging(PackagingID, Packagingtype, Recyclablity, Material) values( @Id, @Type, @Recycle, @Material)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Packaging

END

IF @ActionType = 'Update'

BEGIN

UPDATE Packaging SET

Packagingtype = @Type,

Recyclablity = @Recycle,

Material = @Material

WHERE PackagingID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Packaging Where PackagingID = @Id

END

END

Create Procedure Customziation\_Proc

(

@Id INTEGER,

@Details varchar(50) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Customization(customizationID, CustomizationDetails) values( @Id, @Details)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Customization

END

IF @ActionType = 'Update'

BEGIN

UPDATE Customization SET

CustomizationDetails = @Details

WHERE customizationID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Customization Where customizationID = @Id

END

END

Create Procedure Warranty\_Proc

(

@Id INTEGER,

@Period varchar(20) = NULL,

@enddate varchar(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Warranty(WarrantyID, WarrantyPeriod, WarrantyEndDate) values( @Id, @Period, @enddate)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Warranty

END

IF @ActionType = 'Update'

BEGIN

UPDATE Warranty SET

WarrantyPeriod = @Period,

WarrantyEndDate = @enddate

WHERE WarrantyID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Warranty Where WarrantyID = @Id

END

END

Create Procedure Material\_Proc

(

@Id INTEGER,

@Name varchar(20) = NULL,

@Durable varchar(20) = NULL,

@Breathe varchar(10) = null,

@Comfort varchar(10) = null,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Material(MaterialID, Durability, Breathability, Comfort) values( @Id, @Durable, @Breathe, @Comfort)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Material

END

IF @ActionType = 'Update'

BEGIN

UPDATE Material SET

Durability = @Durable,

Breathability = @Breathe,

Comfort = @Comfort

WHERE MaterialID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Material Where MaterialID = @Id

END

END

Create Procedure Sole\_Proc

(

@Id INTEGER,

@Durable varchar(20) = NULL,

@Flexible varchar(10) = null,

@Medicate varchar(10) = null,

@HardSoft varchar(10) = null,

@Height varchar(10) = null,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Sole(SoleID, Durability, Flexibility, HardOrSoft, HeightIncreaser) values( @Id, @Durable, @Flexible, @HardSoft, @Height)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Sole

END

IF @ActionType = 'Update'

BEGIN

UPDATE Sole SET

Durability = @Durable,

Flexibility = @Flexible,

HardOrSoft = @HardSoft,

HeightIncreaser = @Height

WHERE SoleID = @Id

END

IF @ActionType = 'Delete'

BEGIN

Delete from Sole Where SoleID = @Id

END

END

Create Procedure Paint\_Proc

(

@Id INTEGER,

@Name varchar(20) = NULL,

@ActionType nvarchar(20) = ''

)

AS

BEGIN

IF @ActionType = 'Insert'

BEGIN

Insert into Paint(PaintID, PaintName) values( @Id, @Name)

END

IF @ActionType = 'Select'

BEGIN

Select \* from Paint

END

IF @ActionType = 'Update'

BEGIN

UPDATE Paint SET

PaintName = @Name

WHERE PaintID = @Id

END

IF @ActionType = 'Delete'

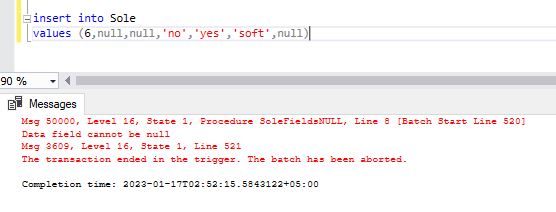
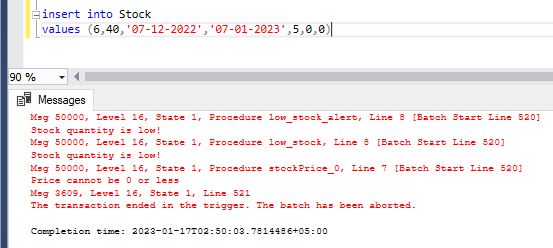
BEGIN

Delete from Paint Where PaintID = @Id

END

END;

**TRIGGERS:**

****

CREATE TRIGGER prevent\_zero\_price

ON shoe

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE price <= 0)

BEGIN

ROLLBACK;

RAISERROR ('Price cannot be set to 0 or less.', 16, 1);

END

END;

CREATE TRIGGER low\_stock\_alert

ON stock

AFTER INSERT

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE quantity < 50)

BEGIN

RAISERROR('Stock quantity is low!', 16, 1)

END

END

CREATE TRIGGER Accessorie\_price\_0

ON accessories

AFTER INSERT,UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Price<=0)

BEGIN

RAISERROR('Price cannot be set 0 or less!', 16, 1)

END

END

CREATE TRIGGER Customization\_zero\_price

ON customization

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE cost< = 0)

BEGIN

ROLLBACK;

RAISERROR ('Price cannot be set to 0 or less!', 16, 1);

END

END;

CREATE TRIGGER Emp\_Sal\_0

ON employee

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Salary< = 0)

BEGIN

ROLLBACK;

RAISERROR ('Salary cannot be set to 0 or less!', 16, 1);

END

END;

CREATE TRIGGER Emp\_Sal\_00

ON employee

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Salary< = 0)

BEGIN

ROLLBACK;

RAISERROR ('Salary cannot be set to 0 or less!', 16, 1);

END

END;

CREATE TRIGGER Invoice\_amount\_0

ON invoice

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Amount<=0)

BEGIN

ROLLBACK;

RAISERROR ('wrong amount entered!', 16, 1);

END

END;

CREATE TRIGGER tr\_PreventNullNames ON origin

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE OriginName IS NULL or address is null or email is null)

BEGIN

RAISERROR('Name field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER PreventNullNames\_packaging ON packaging

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE PackagingID is null or ShoeID is null or Packagingtype is null or Material is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER PreventNullNames\_paint ON paint

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE paintID is null or PaintName is null or ShoeID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER Payment\_am\_0 ON payment

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Amount<=0)

BEGIN

RAISERROR('Amount cannot be 0 or less', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER PaymentFieldsNULL ON payment

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE PaymentID is null or PaymentMethod is null or date is null or OrderID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER returnFieldsNULL ON returnpolicy

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE ReturnID is null or ReturningDate is null or ReturnedDate is null or Reason is null or OrderID is null or ShoeID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER reviewFieldsNULL ON review

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE ReviewID is null or ShoeID is null or CustomerID is null or Review is null or Date is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER reviewCustFieldsNULL ON reviewcustomer

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE ReviewID is null or ShoeID is null or CustomerID is null or KitID is null or AccID is null or OrderID is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER ShoeFieldsNULL ON shoe

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE model is null or ShoeID is null or color is null or size is null or Quantity is null or Rating is null or OriginID is null or BrandID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER ShoeImageFieldsNULL ON shoeImageWarranty

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE ShoeID is null or ShoeID is null or imageid is null or WarrantyID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER ShoekitPrice\_0 ON shoekit

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Price<=0 )

BEGIN

RAISERROR('Price cannot be 0 or less', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER ShoekitFieldsNULL ON shoekit

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE KitID is null or KitName is null or ItemsIncluded is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER ShoeCatFieldsNULL ON shoesCategoryBrand

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE ShoeID is null or BrandID is null or OriginID is null or CategoryID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER SoleFieldsNULL ON sole

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE ShoeID is null or SoleID is null or HeightIncreaser is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER stockPrice\_0 ON stock

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE BuyingPrice<=0 or SellingPrice<=0 )

BEGIN

RAISERROR('Price cannot be 0 or less', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER lowStock ON stock

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Quantity<=50 )

BEGIN

RAISERROR('Low Stock!', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER StockFieldsNULL ON stock

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE StockID is null or OrderDate is null or RecevingDate is null or SupplierID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER SupplierFieldsNULL ON supplier

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE SupplierID is null or SupplierName is null or Address is null or Email is null or Gender is null or OriginID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER WarrantyFieldsNULL ON warranty

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE WarrantyID is null or ShoeID is null or WarrantyPeriod is null or OrderID is null or WarrantyEndDate is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER OrderFieldsNULL ON orders

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE OrderID is null or ShoeID is null or EmpID is null or CustomerID is null or OrderDate is null or ReceivingDate is null or ReviewID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER OrderPackFieldsNULL ON orderPackageCustomize

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE OrderID is null or CustomerID is null or PackagingID is null or CustomizationID is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER MaterialFieldsNULL ON Material

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE MaterialID is null or ShoeID is null or name is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER InVFieldsNULL ON InvoicePaymentReturn

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE InvoiceID is null or PaymentID is null or ReturnID is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER InFieldsNULL ON Invoice

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE InvoiceID is null or date is null or CustomerID is null or OrderID is null or ShoeID is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER InVshoeBFieldsNULL ON InventoryShoeBranch

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE InventoryID is null or BranchID is null or ShoeID is null or StockID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER InventoryFieldsNULL ON Inventory

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE InventoryID is null or Loc is null or ShoeID is null or StockID is null or SupplierID is null or BrandID is null or EmpID is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER ImageFieldsNULL ON Image

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE ImageID is null or ShoeID is null or ImageURL is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER empFieldsNULL ON employee

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE EmpID is null or name is null or email is null or phone is null or Position is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER customizationFieldsNULL ON customization

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE customizationID is null or ShoeID is null or CustomerID is null or OrderID is null or CustomizationDetails is null or MaterialID is null or SoleID is null or PaintID is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER customernFieldsNULL ON customer

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE name is null or Address is null or CustomerID is null or Email is null or Phone is null or Gender is null )

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER BrandFieldsNULL ON brand

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE BrandID is null or BrandName is null or OriginID is null or CategoryID is null)

BEGIN

RAISERROR('Data field cannot be null', 16, 1);

ROLLBACK TRANSACTION;

END

END;

CREATE TRIGGER ACCFieldsNULL ON accessories

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE accid is null or ShoeID is null or AccessoryName is null or InventoryID is null)

BEGIN

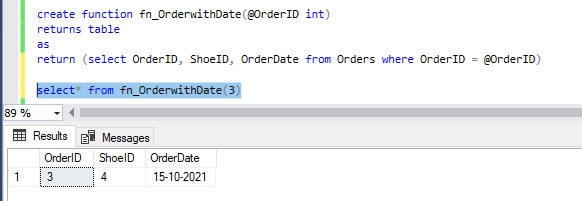
RAISERROR('Data field cannot be null', 16, 1);

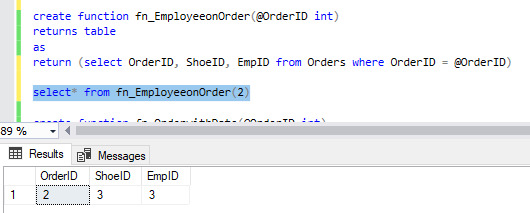
ROLLBACK TRANSACTION;

END

END;

**FUNCTIONS:**

****

****

create function fn\_TotalSalary()

returns table

as

return ( select sum(Salary) as TotalSalary from Employee)

select\* from fn\_TotalSalary()

create function fn\_EmployeeDesignName(@EmpID int)

returns table

as

return (select Name,Position from Employee where EmpID = @EmpID)

select\* from fn\_EmployeeDesignName(2)

create function fn\_EmployeeonOrder(@OrderID int)

returns table

as

return (select OrderID, ShoeID, EmpID from Orders where OrderID = @OrderID)

select\* from fn\_EmployeeonOrder(2)

create function fn\_OrderwithDate(@OrderID int)

returns table

as

return (select OrderID, ShoeID, OrderDate from Orders where OrderID = @OrderID)

select\* from fn\_OrderwithDate(3)

create function fn\_CustomerOrders(@CustomerID int)

returns table

as

return (select CustomerID, OrderID, ORderDate from Orders where CustomerID = @CustomerID)

select\* from fn\_CustomerOrders(4)

create function fn\_CustomerTotalOrders(@CustomerID int)

returns table

as

return (select CustomerID, count(OrderID) as TotalOrders from Orders where CustomerID = @CustomerID

group by CustomerID)

select\* from fn\_CustomerTotalOrders(5)

create function fn\_ShoeDetails(@ShoeID int)

returns table

as

return (select Shoe.ShoeID, CategoryName, BrandName, OriginName from Shoe

inner join ShoesCategoryBrand

on Shoe.ShoeID = ShoesCategoryBrand.ShoeID

inner join Category

on Category.CategoryID = ShoesCategoryBrand.CategoryID

inner join Brand

on Brand.BrandID = ShoesCategoryBrand.BrandID

inner join Origin

on Origin.OriginID = ShoesCategoryBrand.OriginID

and Shoe.ShoeID = @ShoeID)

select\* from fn\_ShoeDetails(2)

create function fn\_CustomerReviews(@CustomerID int)

returns table

as

return (select Customer.CustomerID, Name, ReviewCustomer.ShoeID, Review from Customer

inner join ReviewCustomer

on Customer.CustomerID = ReviewCustomer.CustomerID

inner join Review

on ReviewCustomer.ReviewID = Review.ReviewID

and Customer.CustomerID = @CustomerID)

select\* from fn\_CustomerReviews(3)

create function fn\_SupplierStockOrigin(@SupplierID int)

returns table

as

return (select Supplier.SupplierID,SupplierName,StockId, OriginName

from Supplier

inner join Stock

on Supplier.SupplierID = Stock.SupplierID

inner join Origin

on Supplier.OriginID = Origin.OriginID and

Supplier.SupplierID = @SupplierID)

select\* from fn\_SupplierStockOrigin(2)

create function fn\_Orderpayment(@OrderID int)

returns table

as

return (select Orders.OrderID, PaymentID, PaymentMethod from Orders

inner join Payment

on Orders.OrderID = Payment.OrderID

and Orders.OrderID = @OrderID)

select\* from fn\_OrderPayment(3)

**VIEWS:**

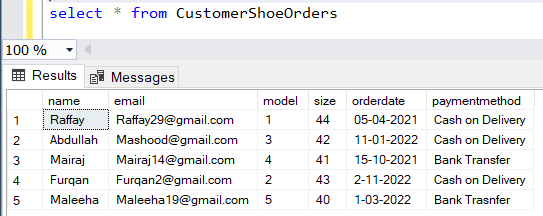
CREATE VIEW vw\_ShoeOrderPayment AS

SELECT s.model,s.size, o.orderID, o.orderDate, p.paymentmethod, p.amount

FROM shoe s

INNER JOIN orders o ON s.shoeID = o.shoeID

INNER JOIN payment p ON o.orderID = p.orderID



create view customerShoeOrder

as

select c.customerID, c.Name, c.Address, s.Model, s.size, p.amount, o.orderID

from customer c inner join

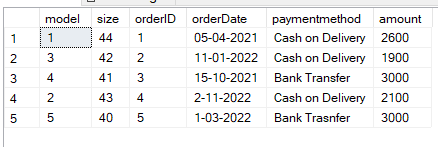
orders o on o.customerid= c.customerid

inner join shoe s on

s.shoeid = o.shoeid

inner join payment p on

p.orderid = o.orderid



create view ReturnShoeOrder

as

select c.name,r.returneddate, r.reason, s.model, o.orderid

from customer c inner join

orders o on o.customerid = c.customerid

inner join shoe s on

s.shoeid = o.shoeid

inner join returnpolicy r

on r.orderid = o.orderid

create view ShoeWarrantyOrder

as

select s.shoeid, s.model, w.warrantyperiod, w.warrantyEndDate, o.orderid

from shoe s inner join warranty w

on s.shoeid = w.shoeid

inner join orders o on

o.orderid = w.orderid

where o.orderid = 2

create view CustomShoeOrder

as

select c.customerid, cz.customizationdetails, cz.cost, p.paintname, m.name, sl.medicated

from customer c inner join

customization cz on cz.customerid = c.customerid

inner join paint p on

p.paintid = cz.paintid

inner join material m on

m.materialid = cz.materialid

inner join sole sl on

sl.soleid= cz.soleid

create view StockSupplierInventory

as

select sup.SupplierName, s.quantity, s.orderdate, s.recevingdate, s.buyingPrice, i.loc

from supplier sup inner join

stock s on s.supplierid = sup.supplierid

inner join inventory i on

i.stockid = s.stockid

where loc= 'Lahore'

create view BranchInventoryBrand

as

select b.brandname, i.limit, bch.loc, bch.city

from brand b inner join inventory i

on i.brandid = b.brandid

inner join branch bch on

bch.inventoryid = i.inventoryid

create view AccShoeInventory

as

select a.AccessoryName, a.price, s.size, s.color, s.quantity, i.limit, i.empid

from accessories a inner join

shoe s on s.shoeid = a.shoeid

inner join inventory i on

i.inventoryid = a.inventoryid

where a.AccessoryName = 'Laces'