AMAZON RETAIL DATABASE SYSTEM

Teams For CS6360.002.S20

Database Design class (CS 6360.002) ECSS 2.311.

Team Number -15
Team Size - 2

Ashwin Ramananda –AXR190005
Pattabhi Ramanna Vella – PXV180022

INDEX

The project consists of the following

1> Data requirements for the system.

2> ER diagram for the system includes multiple
- a> one-to-one binary relationships.
- b> one-to-many binary relationships.
- c> many-to-many binary relationships.

3> The ER diagram is mapped into a relational schema. Showing the resulting relational schema: tables, primary keys and foreign keys.

4> Database normalization rules on the tables

5> Relational schema after normalization.

6> Creating tables using appropriate SQL command. Includes primary key and foreign key definitions and triggered actions on foreign keys. NOT NULL constraints and DEFAULT values for the attributes are also defined.

7> PL/SQL: Define two meaningful stored procedures and two triggers.

DATA REQUIREMENTS

- Amazon, one of the largest retailers in the e-commerce industry has millions of brands and billions of products. To build the database for such a system of that huge scale is a challenge. Each product would have a unique productID, name, specs and most of the time these details along with stock availability of the products such as instock and out of stock.

- Each item will have multiple pictures of the products stored in the database to display them to the customers. Each product will have different dimensions depending in the product.

- The db will have thousands of users ranging from a seller, customer and the admin, Each of these entities will have their own login credentials for their unique phone Number or emailID. Most of the time each user will have their own address as well.

- Each item will have its own brand and a unique name as well. Each brand will have a lot of products under its belt and each customer can buy multiple products under different brands.

- The customers can have home,office and multiple addresses as well as there are a lot of types of payments that can be processed, Credit, debit, echeque, venmo, paypal, gift card and each of them will be having an expiry date.

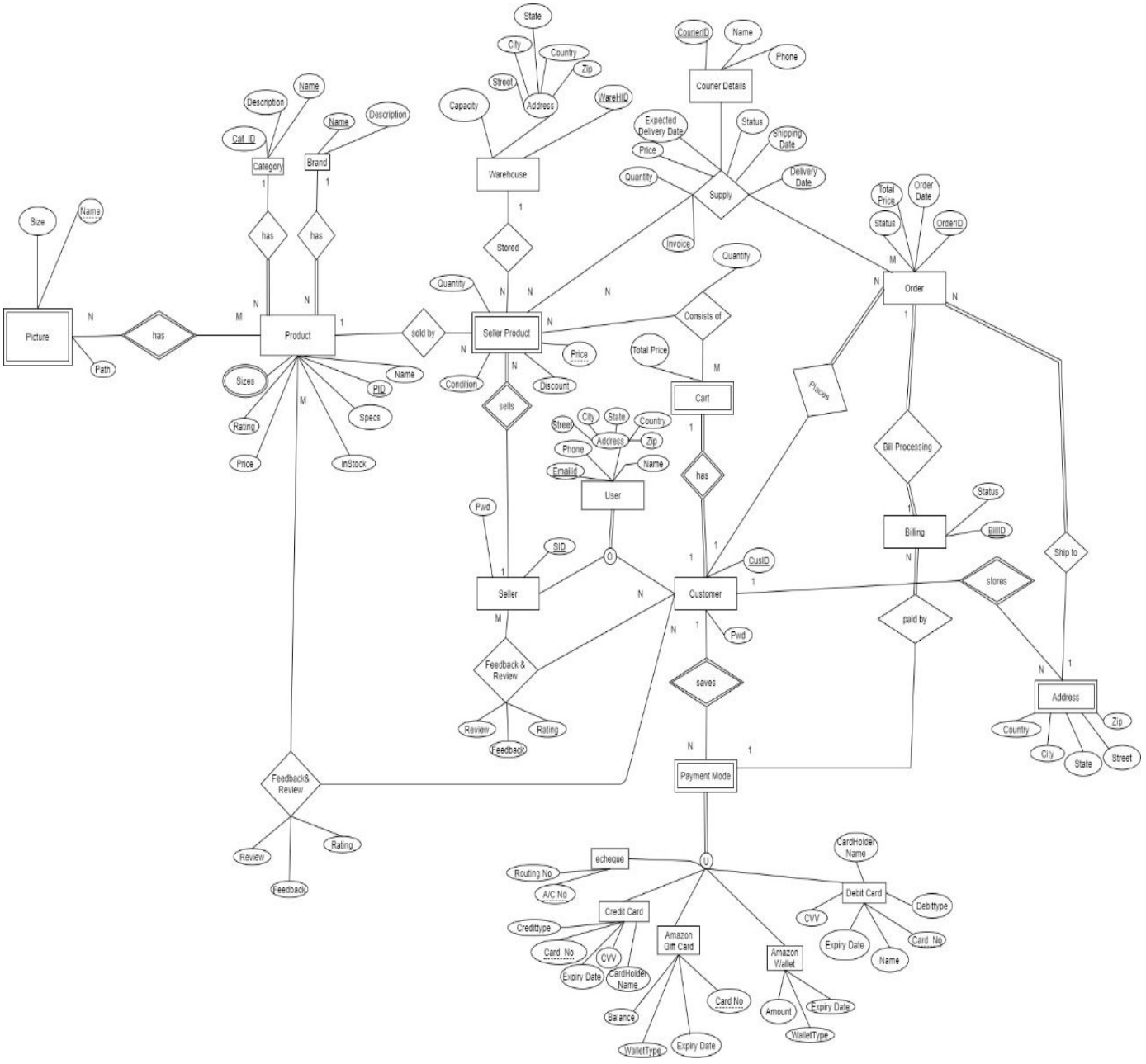
- Each product for example a monitor can be sold by hp, dell, asus or apple, so every product has an option to be sold under multiple sellers. The prices can also vary in the same way. The products are stored in the warehouses of their own sellers most of the time and each warehouse will have an address and a unique location.

- The only way to buy a product is to add it to a cart and place and order by making the payment, he can also use a payment gateway. The payment can be confirmed and or rejected, in case it is confirmed the transaction is successful and in case it is rejected the transaction is a failure. Once it is successful the invoice is created with the expected shipping date, expected date of delivery, quantity, price, discount, status and the address.

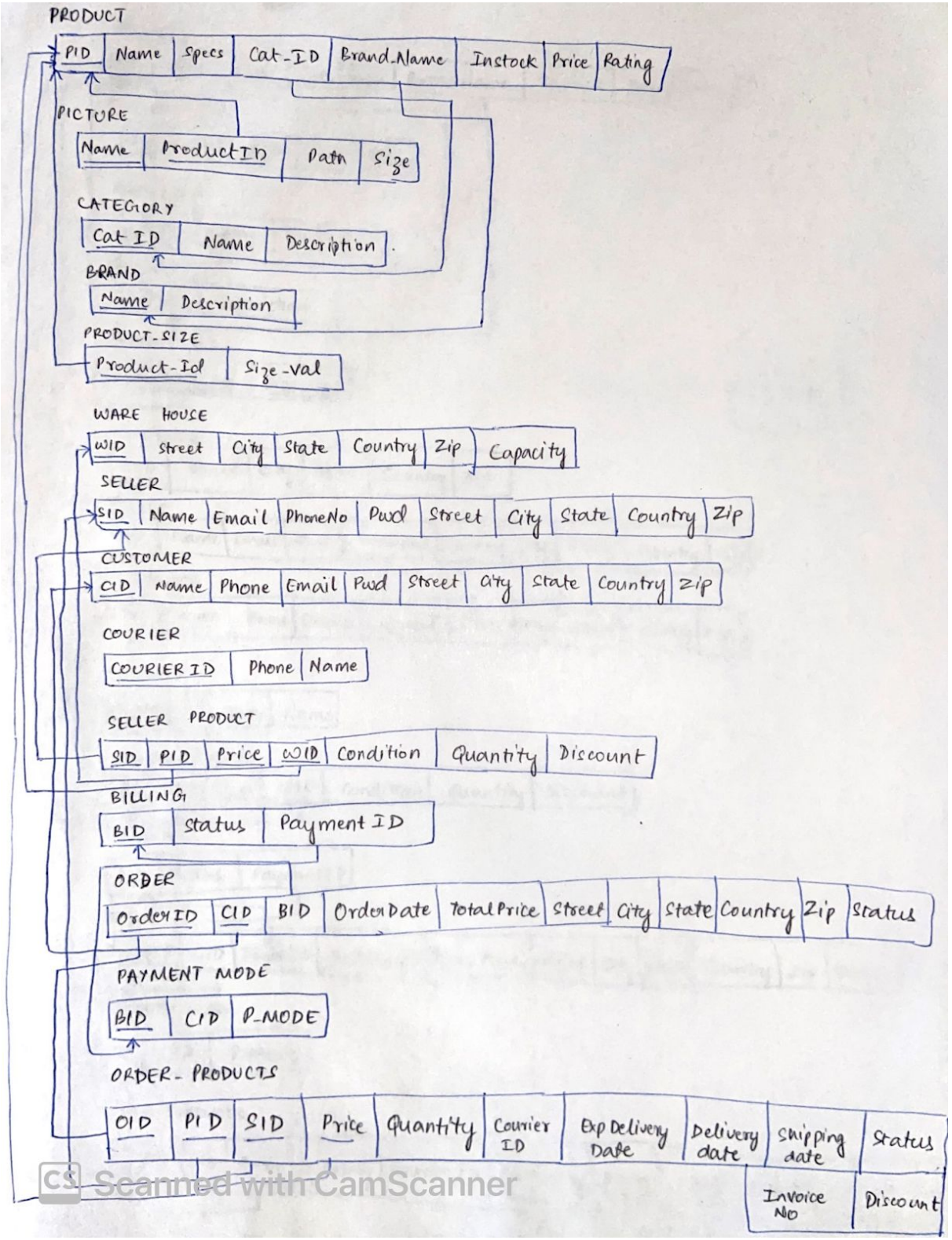
- The products will have a tracking ID and a shipment date. The Courier company like fedEx and DHL will have their own ID and name for example.

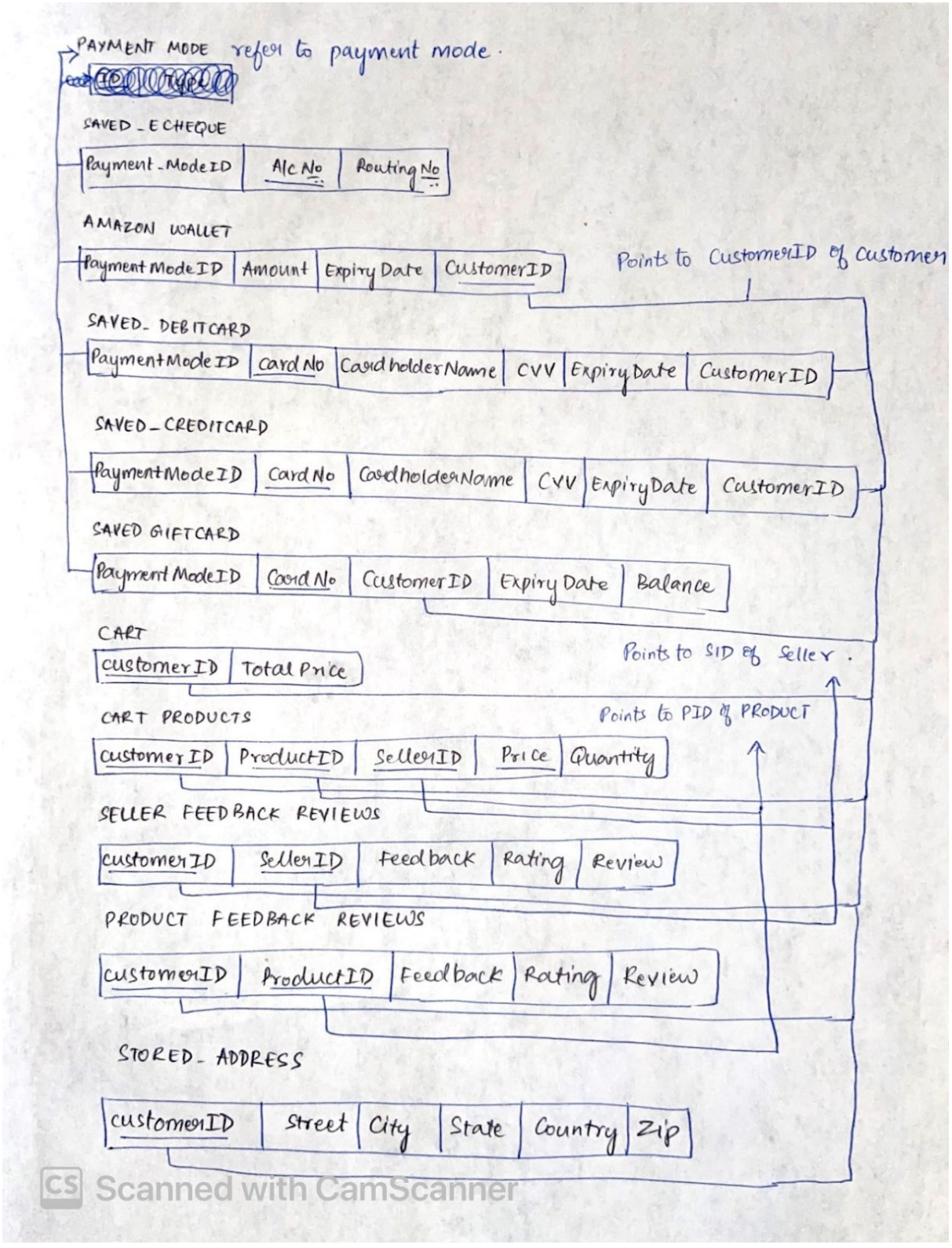
- The final part will be the customer rating to be displayed on the site, each product will be rated by multiple customers and reviews will be given on the UI.

ER DIAGRAM

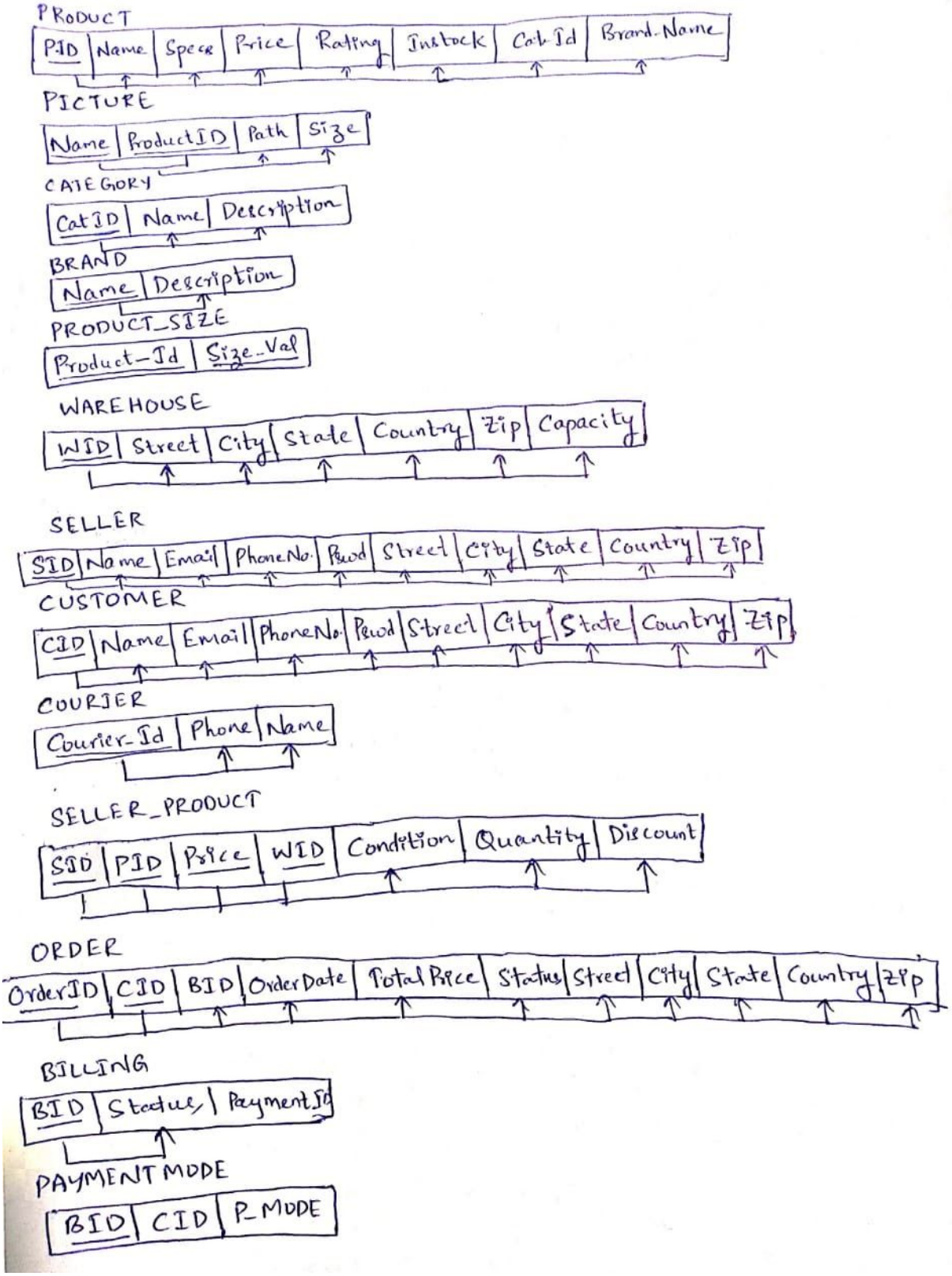

State
City
Country
Zip
Street
Address
Capacity
WareHID
Warehouse
CourierID
Name
Phone
Courier Details
Expected Delivery Date
Status
Shipping Date
Price
Quantity
Supply
Delivery Date
Invoice
Total Price
Order Date
Status
OrderID
Order
Cat_ID
Category
Description
Name
Brand
Description
Size
Name
Picture
Path
has
Product
sold by
Seller Product
Stored
Quantity
Consists of
Cart
Total Price
Sizes
Name
PID
Specs
inStock
Price
Rating
Condition
Discount
sells
Pictures
Bill Processing
Billing
Status
BillID
Ship to
stores
paid by
Address
Country
City
State
Street
Zip
User
Emailid
Phone
Pwd
SID
Seller
CusID
Customer
saves
Feedback & Review
Review
Rating
Feedback
Payment Mode
echeque
Routing No
A/C No
Credit Card
Credittype
Card_No
CVV
Expiry Date
CardHolder Name
Amazon Gift Card
Balance
WalletType
Card No
Expiry Date
Amazon Wallet
Amount
WalletType
Expiry Date
Debit Card
CardHolder Name
Debittype
CVV
Expiry Date
Card_No
Name

Relational schema.

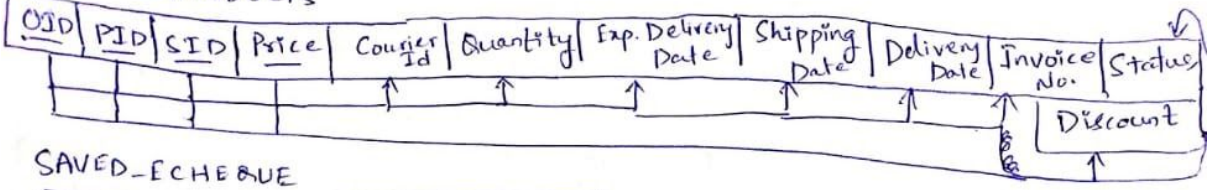




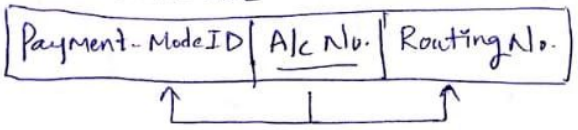
NORMALIZATION AND FUNCTIONAL DEPENDENCIES



ORDER_PRODUCTS



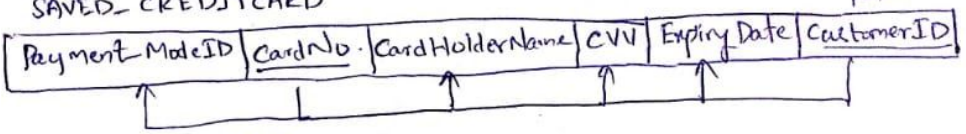
SAVED_ECHEQUE



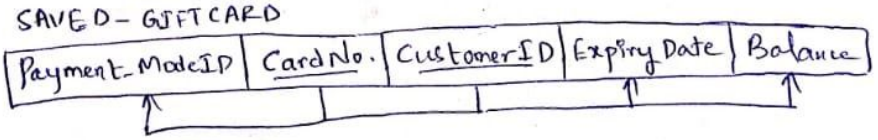
AMAZON WALLET

SAVED_DEBITCARD

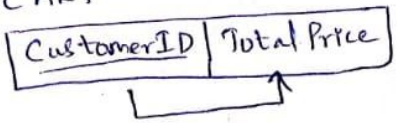
SAVED_CREDITCARD



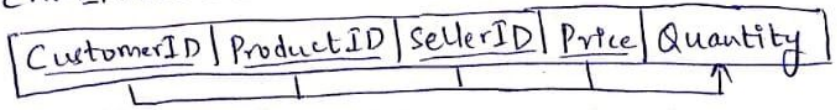
SAVED_GIFTCARD



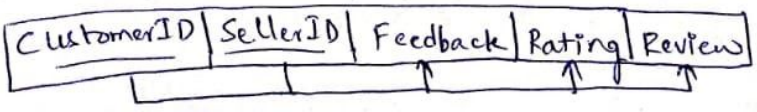
CART

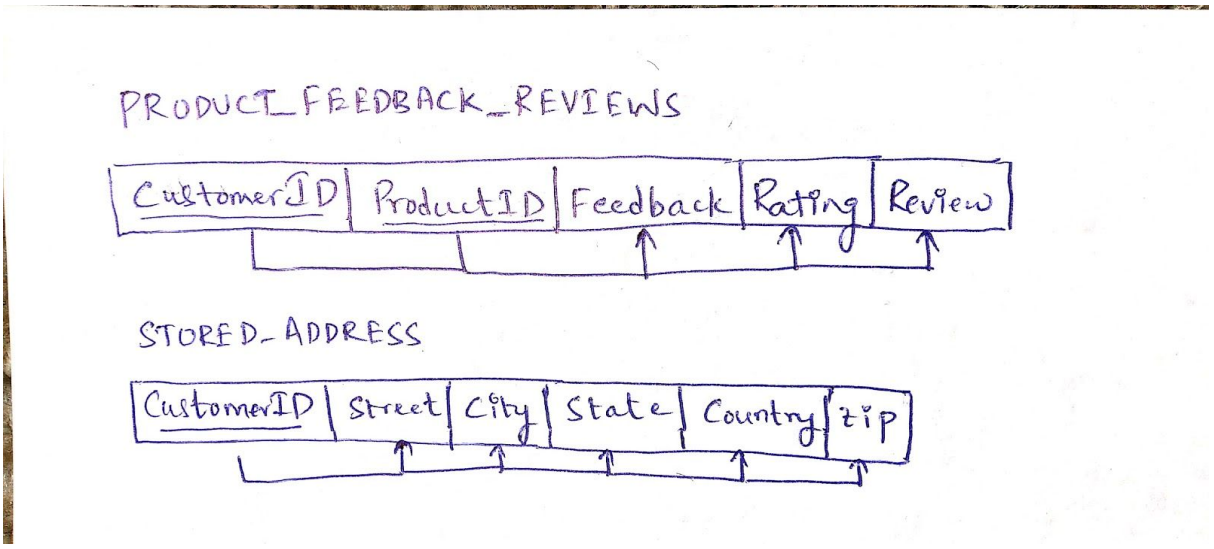


CART_PRODUCTS



SELLER_FEEDBACK_REVIEWS





The above relational schema is already in 3rd Normal Form(3NF). The functional dependencies in the above relational schema don't violate 1NF rules as none of the attributes are multivalued or composite and none of the relations are nested relations. So, the above relational schema is in 1st Normal Form(1NF). They also don't violate 2NF rules as all non-prime attributes are fully dependent on the primary key. So, the above relational schema is in 2nd Normal Form(2NF). They also don't violate the 3NF rules as no non-prime attribute is transitively dependent on the primary key. Hence, the above relational schema is in 3rd Normal Form(3NF).

SQL COMMANDS FOR THE DATABASE

```
CREATE TABLE Customer(
CID Integer PRIMARY KEY,
Name Varchar(300) NOT NULL,
Email Varchar(200) UNIQUE NOT NULL,
Password Varchar(30) NOT NULL,
DOB Date,
Street Varchar(200),
City Varchar(100),
State Varchar(100),
Country Varchar(100),
Zip Varchar(10),
PhoneNo Varchar(20)
);
```

```
CREATE TABLE Seller(
SID Integer PRIMARY KEY,
Name Varchar(300) NOT NULL,
Email Varchar(200) UNIQUE NOT NULL,
Password Varchar(30) NOT NULL,
Street Varchar(200),
City Varchar(100),
State Varchar(100),
Country Varchar(100),
Zip Varchar(20),
PhoneNo Varchar(20)
);

CREATE TABLE Stored_Address(
Street Varchar(300),
City Varchar(100),
State Varchar(100),
Country Varchar(100),
Zip Varchar(10),
CID Integer,
PRIMARY KEY (CID, Street, City, State, Country, Zip),
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE );

CREATE TABLE PaymentMode(
ID Integer Primary Key,
Type Varchar(15)
);

CREATE TABLE AmazonWallet(
PaymentModeID Integer,
CID Integer,
Amount Integer,
ExpiryDate Date NOT NULL,
PRIMARY KEY (CID),
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE,
FOREIGN KEY (PaymentModeID) REFERENCES PaymentMode(ID) ON DELETE
CASCADE );

CREATE TABLE Saved_DebitCard(
PaymentModeID Integer,
CardNo Varchar(16) NOT NULL,
CID Integer,
```

```
CardHolderName Varchar(200) NOT NULL,
CardType Varchar(100) NOT NULL,
ExpiryDate Date NOT NULL,
CVV Integer NOT NULL,
PRIMARY KEY (CardNo, CID),
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE,
FOREIGN KEY (PaymentModeID) REFERENCES PaymentMode(ID) ON DELETE
CASCADE
);

CREATE TABLE Saved_CreditCard(
PaymentModeID Integer,
CardNo Varchar(16) NOT NULL,
CID Integer,
CardHolderName Varchar(300) NOT NULL,
CardType Varchar(100) NOT NULL,
ExpiryDate Date NOT NULL,
CVV Integer NOT NULL,
PRIMARY KEY (CardNo, CID),
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE,
FOREIGN KEY (PaymentModeID) REFERENCES PaymentMode(ID) ON DELETE
CASCADE );

CREATE TABLE Saved_GiftCard(
PaymentModeID Integer,
CardNo Varchar(16) NOT NULL,
CID Integer,
CardHolderName Varchar(300) NOT NULL,
Balance Integer,
ExpiryDate Date NOT NULL,
PRIMARY KEY (CardNo, CID),
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE,
FOREIGN KEY (PaymentModeID) REFERENCES PaymentMode(ID) ON DELETE
CASCADE );

CREATE TABLE Category(
CatID Varchar(20) PRIMARY KEY,
Name Varchar(200) NOT NULL,
Description Varchar(500)
);

CREATE TABLE Product(
```

```
PID Integer PRIMARY KEY,
CatID Varchar(20),
Name Varchar(200) NOT NULL,
BrandName Varchar(100),
Specs Varchar(300),
price Integer CHECK (price > 0),
Rating Varchar(10),
InStock Char(1) DEFAULT 'N' CHECK (inStock in ('Y', 'N')),
FOREIGN KEY (CatID) REFERENCES Category(CatID) ON DELETE CASCADE );

CREATE TABLE Warehouse(
WID Integer Primary Key,
Street Varchar(300),
City Varchar(100),
State Varchar(100),
Country Varchar(100),
Zip Varchar(10)
);

CREATE TABLE Product_Size(
ProductID Integer,
SizeVal Varchar(20),
PRIMARY KEY (SizeVal, ProductID),
FOREIGN KEY (ProductID) REFERENCES Product(PID) ON DELETE CASCADE );

CREATE TABLE Picture(
Name Varchar(200),
PID Integer,
PicPath Varchar(300) NOT NULL,
Size Varchar(100) NOT NULL,
PRIMARY KEY (Name, PID),
FOREIGN KEY (PID) REFERENCES Product(PID) ON DELETE CASCADE );

CREATE TABLE Cart(
CustomerID Integer PRIMARY KEY,
TotalPrice Integer,
FOREIGN KEY (CustomerID) REFERENCES Customer(CID) ON DELETE CASCADE );

CREATE TABLE Cart_Products(
CID Integer,
PID Integer,
SID Integer,
```

```
price Integer,
Quantity Integer DEFAULT 1,
PRIMARY KEY (SID, PID, CID),
FOREIGN KEY (SID, PID, price) REFERENCES Seller_Product(SID, PID, price) ON
DELETE CASCADE,
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE );

CREATE TABLE Seller_Product(
SID Integer,
PID Integer,
WID Integer,
price Integer CHECK (price > 0),
Quantity Integer DEFAULT 0,
Condition Varchar(200),
Discount float DEFAULT 0.00,
PRIMARY KEY (SID, PID, price),
FOREIGN KEY (SID) REFERENCES Seller(SID) ON DELETE CASCADE,
FOREIGN KEY (PID) REFERENCES Product(PID) ON DELETE CASCADE,
FOREIGN KEY (WID) REFERENCES Warehouse(WID) ON DELETE CASCADE );

CREATE TABLE Orders(
OrderID Integer Primary Key,
CID Integer,
BID Integer UNIQUE NOT NULL,
OrderDate Date,
TotalPrice Integer,
Street Varchar(300),
City Varchar(100),
State Varchar(100),
Country Varchar(100),
Zip Varchar(15),
Status Varchar(30) DEFAULT 'PENDING PAYMENT',
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE,
FOREIGN KEY (CID, Street, City, State, Country, Zip) REFERENCES Address(CID,
Street, City, State, Country, Zip),
FOREIGN KEY (BID) REFERENCES BILLING(BID) );

CREATE TABLE Seller_Feedback_Reviews(
CID Integer,
SID Integer,
Rating Integer CHECK (Rating > 0 AND Rating < 10),
Review Varchar(500),
```

```
Feedback Varchar(500),
PRIMARY KEY (SID, CID),
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE,
FOREIGN KEY (SID) REFERENCES Seller(SID) ON DELETE CASCADE
);

CREATE TABLE Product_Feedback_Reviews(
CID Integer,
PID Integer,
Rating Integer CHECK (Rating > 0 AND Rating < 10),
Review Varchar(500),
Feedback Varchar(500),
PRIMARY KEY (PID, CID),
FOREIGN KEY (CID) REFERENCES Customer(CID) ON DELETE CASCADE,
FOREIGN KEY (PID) REFERENCES Product(PID) ON DELETE CASCADE
);

CREATE TABLE Courier(
CourierID Integer Primary Key,
Name Varchar(100),
Phone Varchar(15)
);

CREATE TABLE BILLING(
BID Integer Primary Key,
Status Varchar(30) DEFAULT 'PENDING',
PaymentModeID Integer,
FOREIGN KEY (PaymentModeID) REFERENCES PaymentMode(ID) );

CREATE TABLE Order_Products(
OrderID Integer,
PID Integer,
SID Integer,
CourierID Integer,
price Integer,
Quantity Integer DEFAULT 1,
ShippingDate Date,
expDeliveryDate Date,
deliveryDate Date,
Status Varchar(100) DEFAULT 'PENDING PAYMENT',
InvoiceNo Integer UNIQUE NOT NULL,
Discount float DEFAULT 0.00,
```

```
PRIMARY KEY (SID, PID, price, OrderID),
FOREIGN KEY (SID, PID, price) REFERENCES Seller_Product(SID, PID, price),
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID) ON DELETE CASCADE,
FOREIGN KEY (CourierID) REFERENCES Courier(CourierID),
FOREIGN KEY (PID, SizeVal) REFERENCES Product_Size(PID, SizeVal) );
```

PL/SQL:

TRIGGER 1:

1> Each product might have varying discounts from time to time, so we create a trigger for change in discount of a product.

```
CREATE TABLE Product_discount_log (
pid       VARCHAR (9),
New_discount NUMBER,
Old_discount NUMBER,
Log_date   DATE
);

create or replace TRIGGER Changed_Discount
AFTER UPDATE OF discount ON product
FOR EACH ROW
BEGIN
INSERT INTO Product_discount_log (pid, New_discount, Old_discount, Log_Date)
VALUES (:new.pid, :new.discount, :old.discount, SYSDATE);
END;
```

TRIGGER 2:

Trigger for order confirmed or failed for the status column in the billing table to update the status when the order is confirmed or has failed at payment gateway

```
create or replace TRIGGER OrderStatus AFTER UPDATE OF status ON BILLING
FOR EACH ROW
DECLARE
status Transaction.status%TYPE;
thisOID Orders.OID%TYPE; thisOrderProduct Order_Products%ROWTYPE;

CURSOR orderProducts IS
SELECT * FROM Order_Products WHERE OID = thisOID FOR UPDATE;
```

```
BEGIN
status := :NEW.STATUS;
if(status = 'SUCCESS') then
update orders set status = ‘CONFIRMED' where BID = :NEW.BID;
select OID into thisOID from orders where BID = :NEW.BID;
OPEN orderProducts; LOOP
FETCH orderProducts INTO thisOrderProduct;
EXIT WHEN (orderProducts%NOTFOUND);
UPDATE Order_Products SET status = 'CONFIRMED' WHERE CURRENT OF
orderProducts;
END LOOP;
CLOSE orderProducts; end if;

if(status = 'FAILED') then
update orders set status = 'FAILED' where BID = :NEW.BID;
select OID into thisOID from orders where BID = :NEW.BID;
OPEN orderProducts;
LOOP
FETCH orderProducts INTO thisOrderProduct;
EXIT WHEN (orderProducts%NOTFOUND);
UPDATE Order_Products SET status = 'FAILED' WHERE CURRENT OF orderProducts;
END LOOP;
CLOSE orderProducts; end if;
END;
update transaction SET status='SUCCESS' where id=201;
```

TRIGGER 3

Updating the status of order_products column, the following trigger is executed for updating the quantity of seller_products and also updating the isStock field in product table.

```
create or replace TRIGGER productStock AFTER UPDATE OF status ON Order_Products
FOR EACH ROW
DECLARE
thisStatus Order_Products.status%TYPE; oldStatus Order_Products.status%TYPE;
thisQuantity Order_Products.quantity%TYPE; totalQuantity Seller_Product.quantity%TYPE;
BEGIN
thisStatus := :NEW.STATUS; oldStatus := :OLD.STATUS; thisQuantity :=
:NEW.QUANTITY;
if(thisStatus = 'CONFIRMED' and oldStatus = 'PENDING PAYMENT') then
```

```
update seller_product set quantity = quantity - thisQuantity where PID = :NEW.PID and SID
= :NEW.SID and price = :NEW.price;
select sum(quantity) into totalQuantity from seller_product where PID = :NEW.PID;
if(totalQuantity = 0) then
update product set inStock = 'N' where PID = :NEW.PID; end if;
end if;
if((oldStatus = 'PENDING SHIPMENT' or oldStatus = 'CONFIRMED') OR (thisStatus =
'RETURNED'))
then
update seller_product set quantity = quantity + thisQuantity where PID = :NEW.PID and SID
= :NEW.SID and price = :NEW.price;
select sum(quantity) into totalQuantity from seller_product where PID = :NEW.PID;
if(totalQuantity > 0) then
update product set inStock = 'Y' where PID = :NEW.PID;
end if;
end if;
END;
```

STORED PROCEDURE:

PROCEDURE 1:

This procedure is to return the cumulative rating for a product depending on the product id parameter (customer rating)

```
create or replace PROCEDURE Customer_Rating (Id IN
PRODUCT_RATING.PRODUCTID%TYPE, Customer_rating OUT Float) AS
BEGIN
select avg(rating) into Customer_rating from Product_Feedback_Reviews where PID = ID; END;
set SERVEROUTPUT ON; declare
avgCustomerRating Float;
begin
Customer_Rating (1, avgCustomerRating); dbms_output.put_line(avgCustomerRating); end;
```

PROCEDURE 2

PROCEDURE TO CHANGE THE PRICE OF A CERTAIN PRODUCT SOLD BY A PARTICULAR SELLER:

```
create or replace PROCEDURE Price_Change (pid IN soldItems.pid%TYPE, sid IN soldItems.
%type, newCost int) AS
thisSoldItems soldItems%ROWTYPE;
CURSOR soldItemsCustomer IS
SELECT * FROM soldItems;
BEGIN
OPEN soldItemsCustomer;
LOOP
FETCH soldItemsCustomer INTO thisSoldItems;
EXIT WHEN (soldItems_c%NOTFOUND);
update soldItems set price=newCostwhere sid=sid and pid=pid;
dbms_output.put_line (' Price has been updated');
END LOOP;
CLOSE soldItems_c;
END;
```