DATABASE DESIGN PROJECT
CS6360.003

Amazon Database Design

Presented by Amazon-2:

Deepanshu Sharma	DXS190018
Maitri Naresh Sanghvi	MXS180131
Prachi Vats	PXV180021

Table of contents

Introduction

In the digital age, ecommerce has become an integral part of our lives. From the fresh international cuisine, the apps on our phones and even the mattress we sleep on, everything is sold on some form of ecommerce platform. Amazon probably is the largest and the original player of traditional ecommerce category. The website is extremely vast and complex.

Keywords:

Product: A product is any item being sold on amazon which has it's own unique ProductID. Different variants of a product are also considered individual products. E.g. Different colorsof a particular shoe model.

Listing: A listing is a method to represent collections of different variants. So all the different colors and sizes of a particular shoe are encompassed by a single listing.

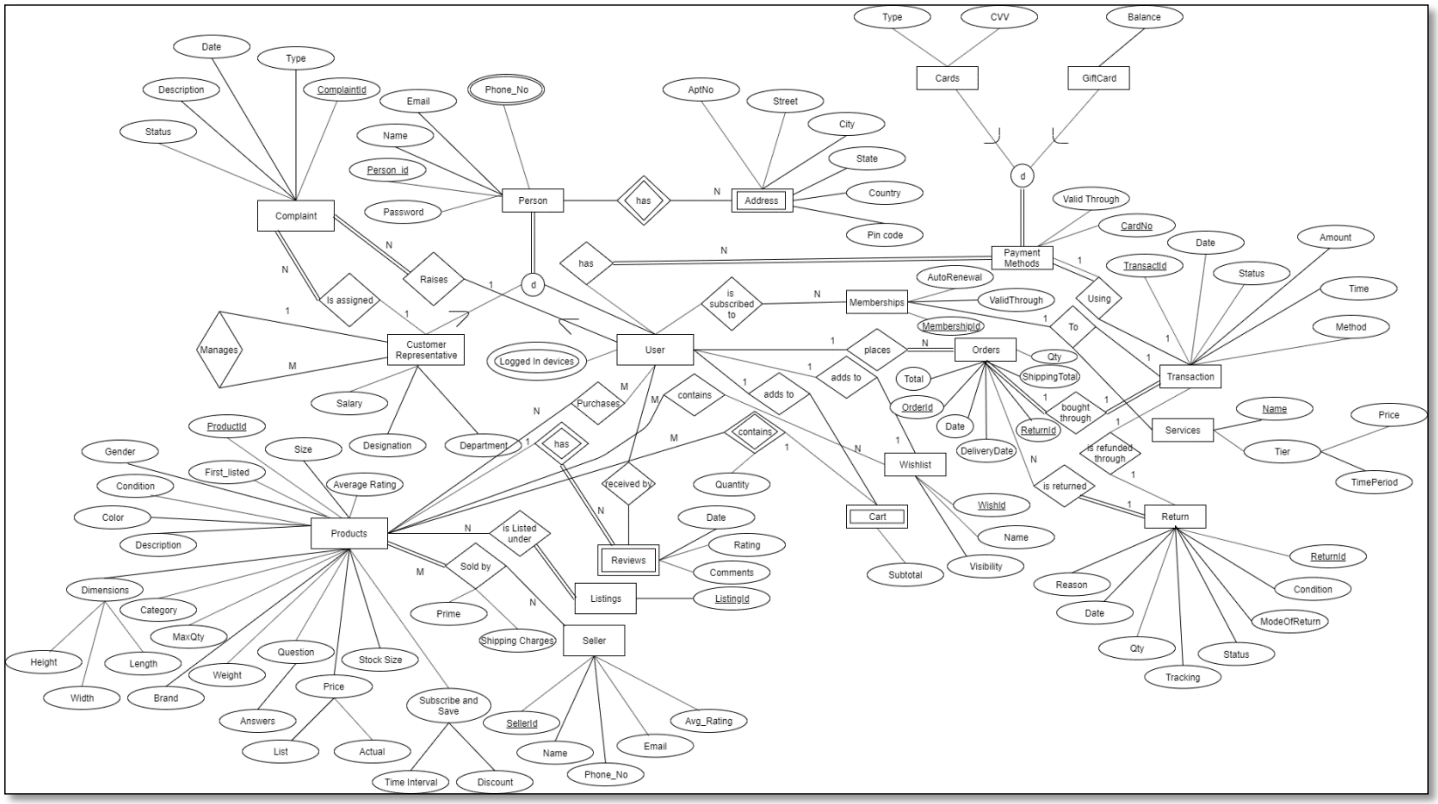
Services: Amazon being so huge, also provides a lot of services with different tiers available. We have also touched upon these to give an idea of things other than the physical products that Amazon sells.

Subscribe & Save:A lot of high frequency products (in terms of sales) are available as a subscription to the user. Thus, the user can just subscribe to it with a frequency best suited to them. This provides both convenience to the user and higher sales to Amazon. E.g. User subscribes to toothpaste to be delivered every month.

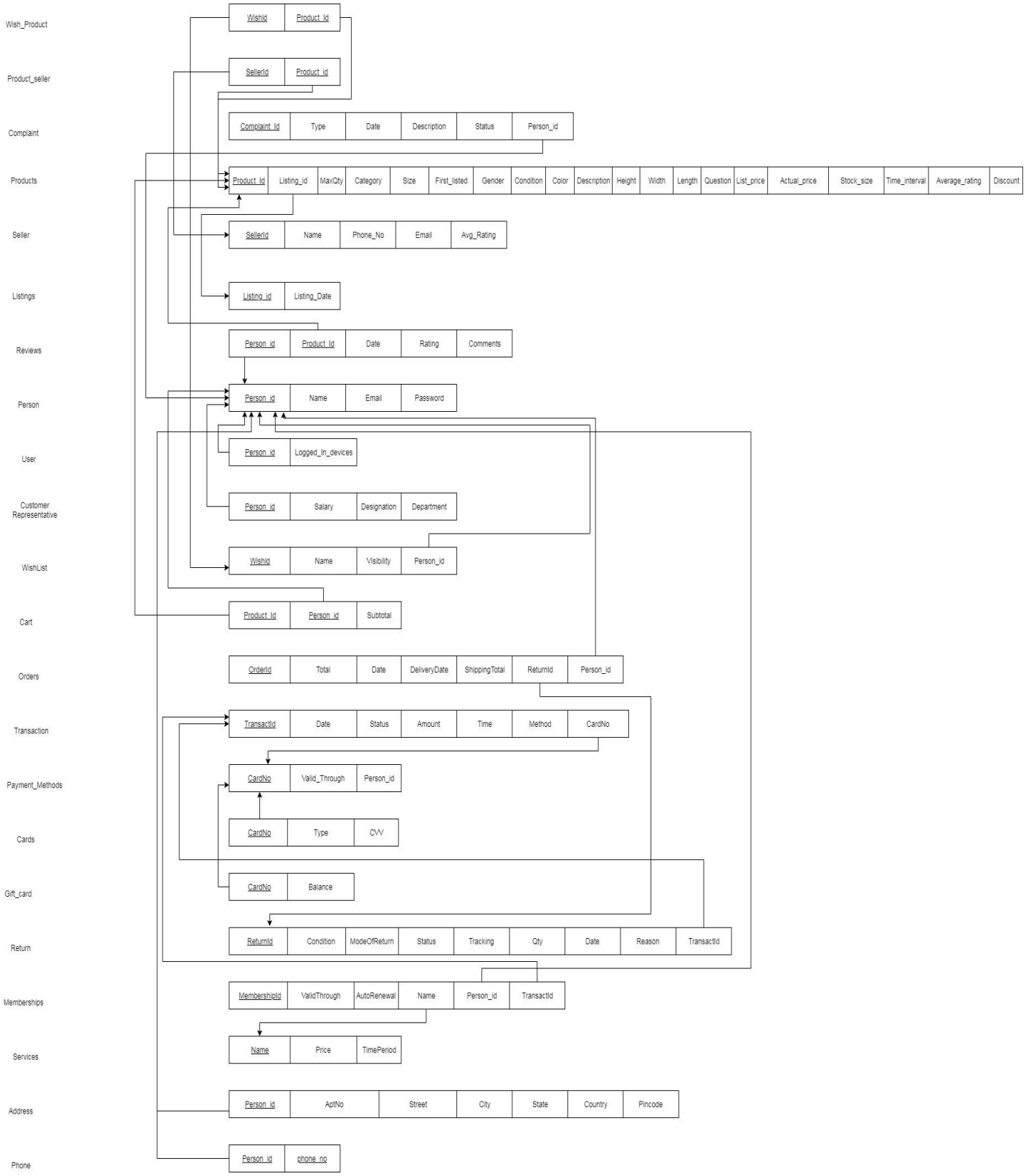
Assumptions:

1. Each Person will either categorize in User or customer_representative.
2. User places order and can return order within 20 days of buy.
3. User can either use gift card or credit card for purchase.
4. Each Product has Listing
5. Seller can sell multiple products.
6. There can multiple logged in devices for one user.

ER Diagram:


Type
CVV
Balance
Cards
GiftCard
Date
Type
Description
ComplaintId
Email
Phone_No
AptNo
Street
City
Status
Name
State
Person_id
Person
has
Address
Country
Password
Pin code
Complaint
Valid Through
CardNo
Date
Amount
Payment Methods
has
Transactid
Status
Raises
is assigned
is subscribed to
Memberships
AutoRenewal
Time
ValidThrough
Using
Manages
Customer Representative
Logged In devices
User
MembershipId
To
Method
places
Orders
Qty
Salary
adds to
Total
Transaction
ShippingTotal
Purchases
contains
OrderId
bought through
ReturnId
Date
Name
Price
ProductId
Designation
Department
has
Services
Tier
Gender
Size
DeliveryDate
is refunded through
Wishlist
First_listed
Quantity
is returned
TimePeriod
Condition
Average Rating
received by
Wishid
Color
Cart
Return
Products
is Listed under
Date
Name
Description
Rating
Reviews
Visibility
ReturnId
Dimensions
Sold by
Comments
Subtotal
Reason
Condition
Category
Listings
ListingId
Date
ModeOfReturn
Prime
MaxQty
Shipping Charges
Seller
Qty
Height
Length
Question
Stock Size
Status
Weight
Price
Tracking
Width
Brand
Subscribe and Save
Answers
SellerId
Avg_Rating
List
Actual
Name
Email
Time Interval
Discount
Phone_No

Relational schema:



Normalization:

The following are the functional dependencies that exist in the relational schema:

1. **Complaint** { ComplaintId -> Type,Date, Description,Status,}
2. **Products**{ ProductId-> Max_QTY, Category, Size, First_listed , Gender, Condition, color, Description, Height, length, question,List_price, Actual_price, Stock_size, Time_interval, Average_rating, Discount}
3. **Seller**{ SellerId-> Name, Phone_no, Email, Avg_Rating}
4. **Listings**{ ListingId -> Listing_Date}
5. **Reviews** { Reviews -> Person_id, Product_id -> Date, rating,Comments}
6. **Person**{ Person -> Person_id, Name, Email, Password}
7. **User**{ Person_id -> Logged_in_devices}
8. **Customer_representative** { Person_id -> Salary, Designation, Department}
9. **WishList** { WishId -> Name, Visbilty, Person_id}
10. **Cart** {Product_id, Person_id -> Subtotal}
11. **Orders**{ OrderId -> Total, Date, DeliverDate, ShippingTotal, Return_id}
12. **Transaction** { TransactId -> date, Status, Amount, Time, Method}
13. **Payment_Methods** { Card_no -> Valid_Through, Person_id}
14. **Cards** { Card_no -> Type, CVV}
15. **Gift_Card** { Card_no -> Balance}
16. **Return** { ReturnId -> Condition, ModeofReturn, Status, Tracking, QTY, Date, Reason, TransactId}
17. **Memberships** { MermbershipId -> ValidThrough, AutoRenewal, Name, Person_id, TransactId}
18. **Services** { Name - > Price, TimePeriod}
19. **Address**{ Person_id -> AptNo, Street, City, State, Country, Pincode}

Tables:

```
CREATE DATABASE Amazon;

Use Amazon

CREATE TABLE `amazon`.`TblUserssss` (
 `sdasd` VARCHAR(55) NOT NULL ,
 `sada` INT(5) NULL )
 ENGINE = InnoDB;

CREATE TABLE Wish_Product (
WishIDVARCHAR(12) NOT NULL ,
Product_IdVARCHAR(12) NOT NULL ,
PRIMARY KEY(WishID, Product_Id),
FOREIGN KEY  (WishID) REFERENCES WishList(WishID),
FOREIGN KEY  (Product_Id) REFERENCES Products(Product_Id) ON DELETE CASCADE ON UPDATE CASCADE) ;

CREATE TABLE Product_seller (
SellerIdVARCHAR(12) NOT NULL ,
Product_IdVARCHAR(12) NOT NULL ,
PRIMARY KEY(SellerId),
FOREIGN KEY  (SellerId) REFERENCES Seller(SellerId) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY  (Product_Id) REFERENCES Products(Product_Id) ON DELETE CASCADE ON UPDATE CASCADE) ;

CREATE TABLE Complaint (
Complaint_IdVARCHAR(12) NOT NULL ,
Type VARCHAR(12) NOT NULL ,
```

```
Date DATE NOT NULL ,

Description VARCHAR(200) NOT NULL ,

Status VARCHAR(12) NOT NULL ,

Person_idVARCHAR(12) NOT NULL ,

PRIMARY KEY(Complaint_Id),

FOREIGN KEY  (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE
CASCADE) ;


CREATE TABLE Products (

Product_IdVARCHAR(12) NOT NULL ,

Listing_idVARCHAR(12),

Max_qty INT NOT NULL ,

Category VARCHAR(12) NOT NULL ,

Size VARCHAR(5) NOT NULL ,

First_Listed DATE NOT NULL ,

Gender VARCHAR(6),

ItemConditionVARCHAR(12) NOT NULL ,

ColorVARCHAR(12) NOT NULL ,

Description VARCHAR(200) NOT NULL ,

Dimension VARCHAR(10) NOT NULL ,

Question VARCHAR(12) NOT NULL ,

List_PriceVARCHAR(12) NOT NULL ,

Actual_PriceVARCHAR(12) NOT NULL ,

Stock_size INT NOT NULL ,

Time_IntervalVARCHAR(12) NOT NULL ,

Average_ratingDECIMAL(2,1) NOT NULL ,

DISCOUNT DECIMAL(3,2) NOT NULL ,

PRIMARY KEY(Product_Id),

FOREIGN KEY  (Listing_id) REFERENCES Listings(Listing_id) ON DELETE CASCADE ON UPDATE
CASCADE) ;
```

```
CREATE TABLE Seller (
SellerIdVARCHAR(12) NOT NULL ,
Name VARCHAR(12) NOT NULL ,
Phone_noVARCHAR(12) NOT NULL ,
Email VARCHAR(20) NOT NULL ,
Average_ratingDECIMAL(2,1),
PRIMARY KEY(SellerId)) ;

CREATE TABLE Listings (
Listing_idVARCHAR(12) NOT NULL ,
PRIMARY KEY(Listing_id)) ;

CREATE TABLE Reviews (
Person_idVARCHAR(12) NOT NULL ,
Product_IdVARCHAR(12) NOT NULL ,
review_date DATE NOT NULL ,
Rating INT NOT NULL  CHECK (Rating > 0 AND Rating <=5),
Comments VARCHAR(200),
PRIMARY KEY(Person_id,Product_Id),
FOREIGN KEY  (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE
CASCADE,
FOREIGN KEY  (Product_Id) REFERENCES Products(Product_Id) ON DELETE CASCADE ON UPDATE
CASCADE) ;

CREATE TABLE Person (
Person_idVARCHAR(12) NOT NULL ,
Name VARCHAR(30) NOT NULL ,
```

```
Email VARCHAR(200),
Password VARCHAR(200),
PRIMARY KEY(Person_id)) ;


CREATE TABLE Userdevices (
Person_idVARCHAR(12) NOT NULL ,
DeviceNameVARCHAR(200),
PRIMARY KEY(Person_id),
FOREIGN KEY  (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE
CASCADE
) ;

CREATE TABLE Customer_rep (
Person_idVARCHAR(12) NOT NULL ,
salary DECIMAL(10,2) NOT NULL ,
Designation VARCHAR(200),
Department VARCHAR(20),
PRIMARY KEY(Person_id),
FOREIGN KEY  (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE
CASCADE) ;


CREATE TABLE WishList (
WishIDVARCHAR(12) NOT NULL ,
Name VARCHAR(12) NOT NULL ,
Visibility VARCHAR(12) NOT NULL ,
Person_idVARCHAR(12) NOT NULL ,
PRIMARY KEY(WishID),
FOREIGN KEY  (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE
CASCADE) ;
```

```
CREATE TABLE Cart (
Product_IdVARCHAR(12) NOT NULL ,
Person_idVARCHAR(12) NOT NULL ,
Subtotal DECIMAL(10,2) NOT NULL ,
PRIMARY KEY(Product_Id),
FOREIGN KEY  (Product_Id) REFERENCES Products(Product_Id) ON DELETE CASCADE ON UPDATE
CASCADE) ;
```

```
CREATE TABLE Orders (
OrderIdVARCHAR(12) NOT NULL ,
OrderDate DATE NOT NULL ,
DeliveryDate DATE NOT NULL ,
Total DECIMAL(10,2) NOT NULL ,
ShippingTotalDECIMAL(10,2) NOT NULL ,
ReturnId  VARCHAR(12) NOT NULL ,
Person_id  VARCHAR(12) NOT NULL ,
PRIMARY KEY(OrderId),
FOREIGN KEY  (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE
CASCADE,
FOREIGN KEY  (ReturnId) REFERENCES Returns(ReturnId) ON DELETE CASCADE ON UPDATE
CASCADE) ;
```

```
CREATE TABLE Transaction (
TidVARCHAR(12) NOT NULL ,
TDate DATE NOT NULL ,
Status  VARCHAR(12) NOT NULL ,
Amount DECIMAL(10,2),
```

```
Time TIME NOT NULL ,

Method VARCHAR(10),

CardNoVARCHAR(20) NOT NULL ,

PRIMARY KEY(Tid),

FOREIGN KEY  (CardNo) REFERENCES PaymentMethods(CardNo)ON DELETE CASCADE ON UPDATE
CASCADE) ;


CREATE TABLE Cards (

CardNoVARCHAR(20) NOT NULL ,

Type  VARCHAR(10) NOT NULL ,

CVV VARCHAR(3) ,

PRIMARY KEY(CardNo),

FOREIGN KEY  (CardNo) REFERENCES PaymentMethods(CardNo) ON DELETE CASCADE ON UPDATE
CASCADE) ;


CREATE TABLE PaymentMethods (

CardNoVARCHAR(20) NOT NULL ,

ValidThrough DATE NOT NULL ,

Person_idVARCHAR(12) NOT NULL ,

PRIMARY KEY(CardNo)) ;



CREATE TABLE GiftCards (

CardNoVARCHAR(20) NOT NULL ,

Balance DECIMAL(8,2) NOT NULL ,

PRIMARY KEY(CardNo),

FOREIGN KEY  (CardNo) REFERENCES PaymentMethods(CardNo) ON DELETE CASCADE ON UPDATE
CASCADE) ;




CREATE TABLE Returns (
```

ReturnIdVARCHAR(12) NOT NULL ,

ItemConditionVARCHAR(12) NOT NULL ,

ReturnModeVARCHAR(12) NOT NULL ,

Status VARCHAR(12) NOT NULL ,

TrackingCodeVARCHAR(20) NOT NULL ,

Qty INT NOT NULL ,

Date DATE NOT NULL ,

Reason VARCHAR(12) NOT NULL ,

TransactionIdVARCHAR(12) NOT NULL ,

PRIMARY KEY(ReturnId),

FOREIGN KEY (TransactionId) REFERENCES Transaction(Tid) ON DELETE CASCADE ON UPDATE CASCADE) ;

CREATE TABLE Memberships (

Mid VARCHAR(12) NOT NULL ,

ValidThrough DATE NOT NULL ,

Autorenew BOOLEAN NOT NULL ,

SnameVARCHAR(12) NOT NULL ,

Person_idVARCHAR(12) NOT NULL ,

TransactionIdVARCHAR(12) NOT NULL ,

PRIMARY KEY(Mid),

FOREIGN KEY (Sname) REFERENCES Services(Name) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (TransactionId) REFERENCES Transaction(Tid) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE CASCADE) ;

CREATE TABLE Services (

Name VARCHAR(12) NOT NULL ,

Price DECIMAL(8,2) NOT NULL ,

```
TimeInMonths INT NOT NULL ,
PRIMARY KEY(Name)) ;


CREATE TABLE Address (
Person_idVARCHAR(12) NOT NULL ,
AptNOVARCHAR(10) NOT NULL ,
Street VARCHAR(20) NOT NULL ,
City VARCHAR(12) NOT NULL ,
State VARCHAR(12) NOT NULL ,
Country VARCHAR(12) NOT NULL ,
ZIPCodeVARCHAR(10) NOT NULL ,
PRIMARY KEY(Person_id),
FOREIGN KEY  (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE
CASCADE) ;


CREATE TABLE Phone (
Person_idVARCHAR(12) NOT NULL ,
Phone_noVARCHAR(13) NOT NULL ,
PRIMARY KEY(Person_id),
FOREIGN KEY  (Person_id) REFERENCES Person(Person_id) ON DELETE CASCADE ON UPDATE
CASCADE) ;
```

Triggers:

Trigger 1: Restock

Trigger Restock is triggered when the order placed by the user is Successful to reduce the product quantity from the Product Table.

```
Create trigger Restock

After Insert on Return

For Each ROW

Update product

SET StockSize = Old.StockSize + New.Qty

Where ProductID = New.ProductID

And New.Status = “Success”
```

Trigger 2: ValidateReturn

Trigger ValidateReturn is triggered when a user initiates a return. This trigger validates if the user has asked for a return within 30 days of order date. This trigger is created to throw error when return date is greater than 30 days of order date.

```
Create trigger ValidateReturn

After Insert on RETURN

For each row

Update RETURN

set Status= "Failed.Return date greater than 30 days"

where datediff(day,(select date from Order where Returnid=new.Returnid),Return.DATE) <= 30
```

Trigger 3: ReduceStock

Trigger ReduceStock is triggered when the order returned by the user is Successful to increase the product quantity in the Product Table.

```
Create trigger ReduceStock

After Insert on Return
```

For Each ROW

Update product

SET StockSize = Old.StockSize - New.Qty

Where ProductID = New.ProductID

And New.Status = “Success”

Procedures

Procedure 1: Rating_update

Rating_update procedure is created to update the rating of the product, as the user gives reviwes,this procedure collects reviews and calculate rating for a particular product. This procedure is executed after every 30 days. It collects reviews in cursor and then accordingly updates rating for all the products whose reviews have popped up in the past 30 days.

```
create or replace PROCEDURE Rating_Upadte AS

thisProdReview%ROWTYPE;

thisratingNumber :=0;

count Number :=0 ;

CURSOR ReviewProd IS

SELECT Product_id, sum(rating) as rating , count(*) as counter FROM Reviews R, Product P

WHERE P.Product_id = R.Product_id AND R.Date> Today()- 30 group by Product_id;

FOR UPDATE;

BEGIN

OPEN ReviewProd;

LOOP

  FETCH ReviewProd INTO thisProd;

  EXIT WHEN (ReviewProd%NOTFOUND);

  --dbms_output.put_line(thisEmployee.Salary);

thisrating = thisrating + thisprod ;

  UPDATE Product SET Rating = rating / counter

  WHERE CURRENT OF ReviewProd;
```

```
END LOOP;

CLOSE ReviewProd;

END;
```

Procedure 2: Order_history

Order_History procedure is created to display the history of orders of the particular user. Whenever User wants to access order history related to account, user clicks order history button and he gets to see all the orders that he has ordered in the past. This procedure displays all the information of orders that has been done in the past.

```
create or replace PROCEDURE Order_history(Personid IN User.Person_id%TYPE) AS

thisOrderOrders%ROWTYPE;

CURSOR History IS

SELECT O.* FROM Orders O WHERE O.Person_id=Personid;

BEGIN

OPEN History;

LOOP

  FETCH History INTO thisOrder;

  EXIT WHEN (History%NOTFOUND);

dbms_output.put_line(thisOrder.Order_id || thisOrder.Date || thisOrder.DeliveryDate ||
thisOrder.ShippingTotal);

END LOOP;

CLOSE History;

END;
```