Amish Bisaria

eBay Database Report

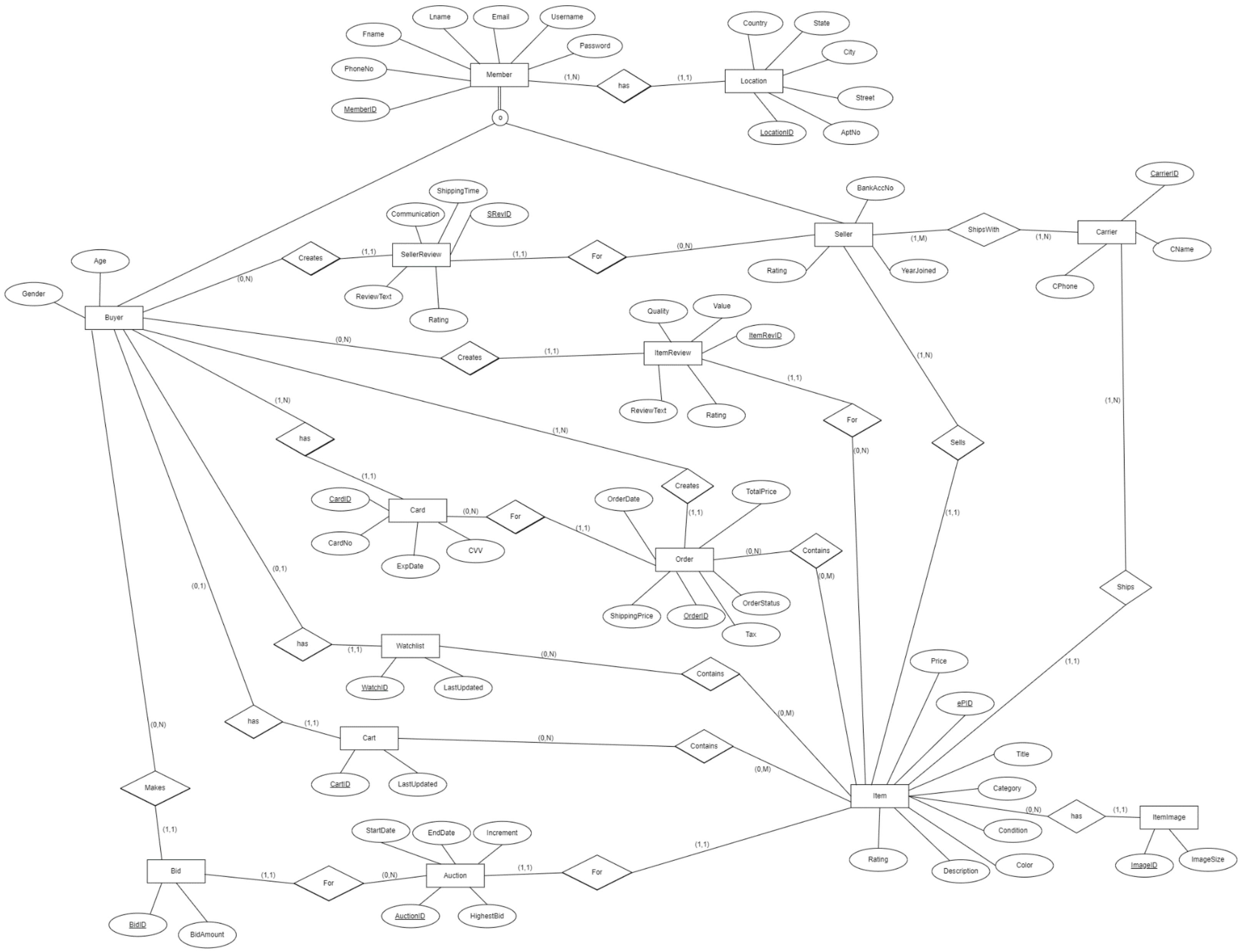
Data Requirements:

Let EBAY_DB be a database system where members can buy, sell, and bid on items. Here are the data requirements for this system.

- The website has members. A member can be a buyer or a seller or both. Each member has a unique member id, first name, last name, phone number, e-mail address, location, username, password. A buyer also has age and gender.
- Location has a unique location id, country, state, city, street, and apartment number.
- A buyer can create reviews for sellers and items. A seller review has a unique id, review text, rating, communication score, and shipping time score. An item review has a unique id, review text, rating, quality, and value.
- A seller sells items and must have a bank account number to receive their profits. Each seller also has a seller rating based on how much the customers like what they have sold. Each seller also has the year that they joined ebay.
- Each buyer has a credit card with a unique card id, card number, expiry date, and cvv. The card will be used for orders. Orders contain items and have a unique order ID, order status, order date, shipping price, tax, and total price.
- When buyers make orders from the sellers, sellers place shipment for delivery carriers. Sellers ship their items with carriers that have a unique carrier id, carrier name, and phone number.
- Buyers have a watchlist and cart that contain items. The watchlist has a unique watchlist id and the date last updated. The cart has a unique cart id and the date last updated.
- Items have a unique eBay product ID (ePID), item title, item images, item description, category, condition, price, description, rating, and color. Items also have item images that have a unique image id and an image size.
- Sales can be either through a fixed price or an auction.
- Auctions have a unique auction ID, highest bid, minimum bidding increment, the start date of the auction, and the end date of the auction.
- Buyers can bid on items. Each bid price has a unique bid id and a bid amount. The bidder at the end of the auction with the highest bid price is declared the winner and a transaction between buyer and seller may then proceed.

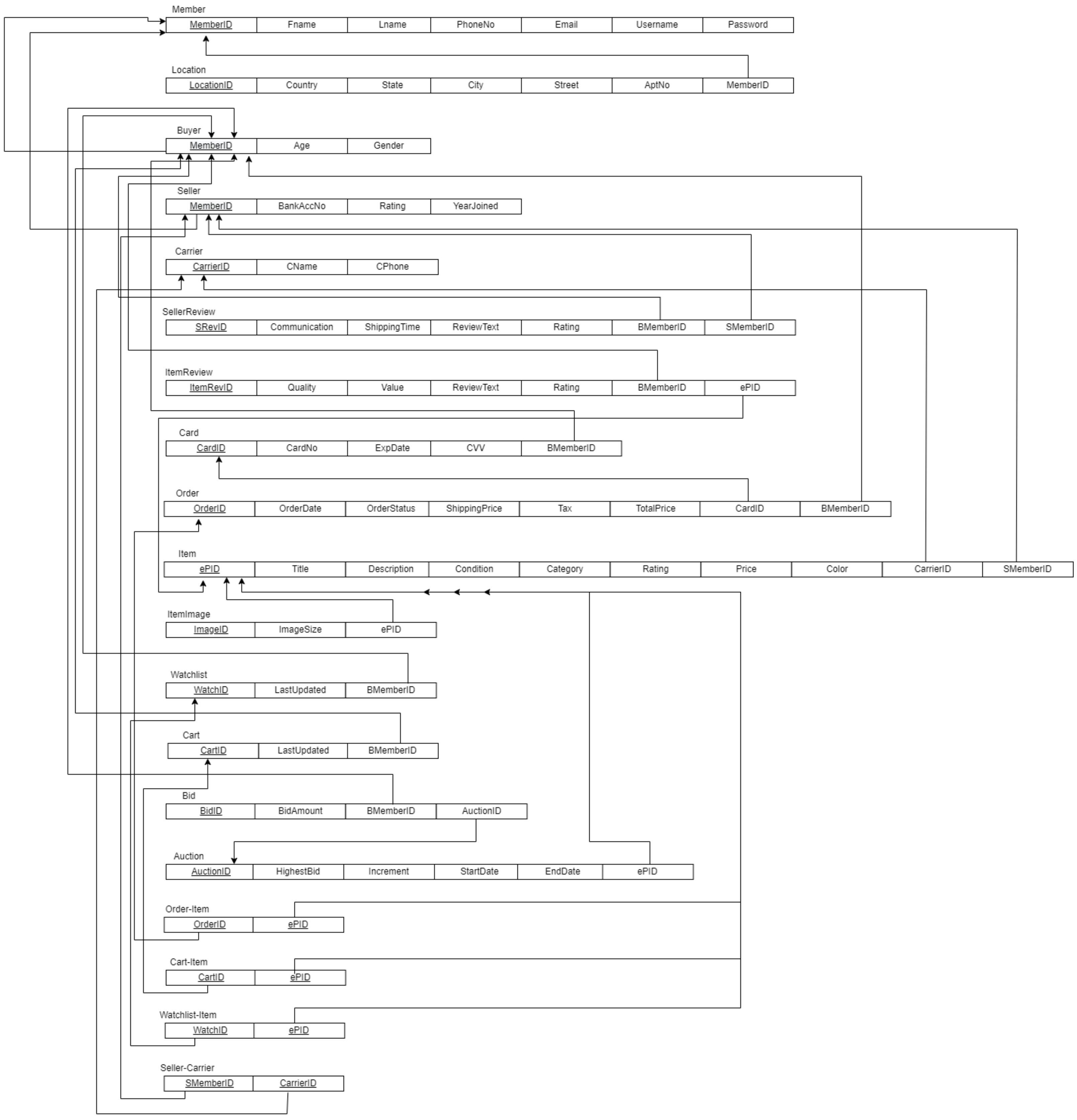
Amish Bisaria

EER Diagram:



Amish Bisaria

Map ER Diagram to Relational Schema:


Member
MemberID | Fname | Lname | PhoneNo | Email | Username | Password
Location
LocationID | Country | State | City | Street | AptNo | MemberID
Buyer
MemberID | Age | Gender
Seller
MemberID | BankAccNo | Rating | YearJoined
Carrier
CarrierID | CName | CPhone
SellerReview
SRevID | Communication | ShippingTime | ReviewText | Rating | BMemberID | SMemberID
ItemReview
ItemRevID | Quality | Value | ReviewText | Rating | BMemberID | ePID
Card
CardID | CardNo | ExpDate | CVV | BMemberID
Order
OrderID | OrderDate | OrderStatus | ShippingPrice | Tax | TotalPrice | CardID | BMemberID
Item
ePID | Title | Description | Condition | Category | Rating | Price | Color | CarrierID | SMemberID
ItemImage
ImageID | ImageSize | ePID
Watchlist
WatchID | LastUpdated | BMemberID
Cart
CartID | LastUpdated | BMemberID
Bid
BidID | BidAmount | BMemberID | AuctionID
Auction
AuctionID | HighestBid | Increment | StartDate | EndDate | ePID
Order-Item
OrderID | ePID
Cart-Item
CartID | ePID
Watchlist-Item
WatchID | ePID
Seller-Carrier
SMemberID | CarrierID

Amish Bisaria

Normalization:
The relational schema does not violate the 1st, 2nd and 3rd normal forms and is already normalized into 3NF. This is because there are no composite attributes, multivalued attributes, and nested relations. Also, every non-prime attribute in the relations is fully functionally dependent on the primary key. Lastly, no non-prime attribute in the relations is transitively dependent on the primary key.

SQL Tables:

```
--TABLES
CREATE TABLE Member (
  MemberID INTEGER NOT NULL,
  Fname VARCHAR(20) NOT NULL,
  Lname VARCHAR(20) NOT NULL,
  PhoneNo CHAR(10),
  Email VARCHAR(30) NOT NULL,
  Username VARCHAR(30) NOT NULL,
  Password VARCHAR(30) NOT NULL,
  PRIMARY KEY (MemberID)
)

CREATE TABLE Location (
  LocationID INTEGER NOT NULL,
  Country VARCHAR(20) NOT NULL,
  State VARCHAR(20) NOT NULL,
  City VARCHAR(20) NOT NULL,
  Street VARCHAR(20) NOT NULL,
  AptNo INTEGER NOT NULL,
  MemberID INTEGER NOT NULL,
  PRIMARY KEY (LocationID)
)

CREATE TABLE Buyer (
  MemberID INTEGER NOT NULL,
  Age INTEGER,
  Gender CHAR(1),
  PRIMARY KEY (MemberID)
)

CREATE TABLE Seller (
  MemberID INTEGER NOT NULL,
  BankAccNo INTEGER NOT NULL,
  Rating INTEGER NOT NULL,
  YearJoined INTEGER NOT NULL,
  PRIMARY KEY (MemberID)
)

CREATE TABLE Carrier (
  CarrierID INTEGER NOT NULL,
  CName  VARCHAR(20) NOT NULL,
  CPhone CHAR(10) NOT NULL,
  PRIMARY KEY (CarrierID)
)

CREATE TABLE SellerReview (
  SRevID INTEGER NOT NULL,
  Communication INTEGER,
  ShippingTime INTEGER,
  ReviewText VARCHAR(1000) NOT NULL,
  Rating INTEGER NOT NULL,
  BMemberID INTEGER NOT NULL,
  SMemberID INTEGER NOT NULL,
  PRIMARY KEY (SRevID)
)

CREATE TABLE ItemReview (
  ItemRevID INTEGER NOT NULL,
  Quality INTEGER,
  ItemValue INTEGER,
  ReviewText VARCHAR(1000) NOT NULL,
  Rating INTEGER NOT NULL,
  BMemberID INTEGER NOT NULL,
  ePID INTEGER NOT NULL,
  PRIMARY KEY (ItemRevID)
)

CREATE TABLE Card (
  CardID INTEGER NOT NULL,
  CardNo INTEGER NOT NULL,
  ExpDate DATE NOT NULL,
  CVV INTEGER NOT NULL,
  BMemberID INTEGER NOT NULL,
  PRIMARY KEY (CardID)
)

CREATE TABLE Order (
  OrderID INTEGER NOT NULL,
  OrderDate DATE NOT NULL,
  OrderStatus VARCHAR(10) NOT NULL,
  ShippingPrice DECIMAL(10,2) NOT NULL,
  Tax DECIMAL(10,2) NOT NULL,
  TotalPrice DECIMAL(10,2) NOT NULL,
  CardID INTEGER NOT NULL,
  BMemberID INTEGER NOT NULL,
  PRIMARY KEY (OrderID)
)

CREATE TABLE Item (
  ePID INTEGER NOT NULL,
  Title VARCHAR(30) NOT NULL,
  Description VARCHAR(1000) NOT NULL,
  Condition VARCHAR(10),
  Category VARCHAR(10),
  Rating INTEGER NOT NULL,
  Price DECIMAL(10,2) NOT NULL,
  Color VARCHAR(10),
  CarrierID INTEGER NOT NULL,
  SMemberID INTEGER NOT NULL,
  PRIMARY KEY (ePID)
)

CREATE TABLE ItemImage (
  ImageID INTEGER NOT NULL,
  ImageSize INTEGER,
  ePID INTEGER NOT NULL,
  PRIMARY KEY (ImageID)
)

CREATE TABLE Watchlist (
  WatchID INTEGER NOT NULL,
  LastUpdated DATE NOT NULL,
  BMemberID INTEGER NOT NULL,
  PRIMARY KEY (WatchID)
)

CREATE TABLE Cart (
  CartID INTEGER NOT NULL,
  LastUpdated DATE NOT NULL,
  BMemberID INTEGER NOT NULL,
  PRIMARY KEY (CartID)
)

CREATE TABLE Bid (
  BidID INTEGER NOT NULL,
  BidAmount DECIMAL(10,2) NOT NULL,
  BMemberID INTEGER NOT NULL,
  AuctionID INTEGER NOT NULL,
  PRIMARY KEY (BidID)
)

CREATE TABLE Auction (
  AuctionID INTEGER NOT NULL,
  HighestBid DECIMAL(10,2) NOT NULL,
  Increment DECIMAL(10,2) NOT NULL,
  StartDate DATE NOT NULL,
  EndDate DATE NOT NULL,
  ePID INTEGER NOT NULL,
  PRIMARY KEY (AuctionID)
)

CREATE TABLE Order_Item (
  OrderID INTEGER NOT NULL,
  ePID INTEGER NOT NULL,
  PRIMARY KEY (OrderID, ePID)
)

CREATE TABLE Cart_Item (
  CartID INTEGER NOT NULL,
  ePID INTEGER NOT NULL,
  PRIMARY KEY (CartID, ePID)
```

```
)

CREATE TABLE Watchlist_Item (
  WatchID INTEGER NOT NULL,
  ePID INTEGER NOT NULL,
  PRIMARY KEY (WatchID, ePID)
)

CREATE TABLE Seller_Carrier (
  SMemberID INTEGER NOT NULL,
  CarrierID INTEGER NOT NULL,
  PRIMARY KEY (SMemberID, CarrierID)
)

--FOREIGN KEYS
ALTER TABLE Location ADD CONSTRAINT fklocate1 FOREIGN KEY(MemberID)
REFERENCES Member(MemberID)
  ON DELETE CASCADE;
ALTER TABLE Buyer ADD CONSTRAINT fkbuyer1 FOREIGN KEY(MemberID)
REFERENCES Member(MemberID)
  ON DELETE CASCADE;
ALTER TABLE Seller ADD CONSTRAINT fkseller1 FOREIGN KEY(MemberID)
REFERENCES Member(MemberID)
  ON DELETE CASCADE;
ALTER TABLE SellerReview ADD CONSTRAINT fksellrev1 FOREIGN KEY(BMemberID)
REFERENCES Buyer(MemberID)
  ON DELETE CASCADE;
ALTER TABLE SellerReview ADD CONSTRAINT fksellrev2 FOREIGN KEY(SMemberID)
REFERENCES Seller(MemberID)
  ON DELETE CASCADE;
ALTER TABLE ItemReview ADD CONSTRAINT fksellrev1 FOREIGN KEY(BMemberID)
REFERENCES Buyer(MemberID)
  ON DELETE CASCADE;
ALTER TABLE ItemReview ADD CONSTRAINT fksellrev2 FOREIGN KEY(ePID)
REFERENCES Item(ePID)
  ON DELETE CASCADE;
ALTER TABLE Card ADD CONSTRAINT fkcard1 FOREIGN KEY(BMemberID)
REFERENCES Buyer(MemberID)
  ON DELETE CASCADE;
```

Amish Bisaria

```
ALTER TABLE Order ADD CONSTRAINT fkorder1 FOREIGN KEY(CardID)
REFERENCES Card(CardID)
  ON DELETE CASCADE;
ALTER TABLE Order ADD CONSTRAINT fkorder2 FOREIGN KEY(BMemberID)
REFERENCES Buyer(MemberID)
  ON DELETE CASCADE;
ALTER TABLE Item ADD CONSTRAINT fkitem2 FOREIGN KEY(CarrierID)
REFERENCES Carrier(CarrierID)
  ON DELETE CASCADE;
ALTER TABLE Item ADD CONSTRAINT fkitem2 FOREIGN KEY(SMemberID)
REFERENCES Seller(MemberID)
  ON DELETE CASCADE;
ALTER TABLE ItemImage ADD CONSTRAINT fkitemimage1 FOREIGN KEY(ePID)
REFERENCES Item(ePID)
  ON DELETE CASCADE;
ALTER TABLE WatchList ADD CONSTRAINT fkwatch1 FOREIGN KEY(BMemberID)
REFERENCES Buyer(MemberID)
  ON DELETE CASCADE;
ALTER TABLE Cart ADD CONSTRAINT fkcart1 FOREIGN KEY(BMemberID)
REFERENCES Buyer(MemberID)
  ON DELETE CASCADE;
ALTER TABLE Bid ADD CONSTRAINT fkbid1 FOREIGN KEY(BMemberID)
REFERENCES Buyer(MemberID)
  ON DELETE CASCADE;
ALTER TABLE Bid ADD CONSTRAINT fkbid2 FOREIGN KEY(AuctionID) REFERENCES
Auction(AuctionID)
  ON DELETE CASCADE;
ALTER TABLE Auction ADD CONSTRAINT fkauction1 FOREIGN KEY(ePID)
REFERENCES Item(ePID)
  ON DELETE CASCADE;
ALTER TABLE Order_Item ADD CONSTRAINT fkorderitem1 FOREIGN KEY(OrderID)
REFERENCES Order(OrderID)
  ON DELETE CASCADE;
ALTER TABLE Order_Item ADD CONSTRAINT fkorderitem2 FOREIGN KEY(ePID)
REFERENCES Item(ePID)
  ON DELETE CASCADE;
ALTER TABLE Cart_Item ADD CONSTRAINT fkcartitem1 FOREIGN KEY(CartID)
REFERENCES Cart(CartID)
  ON DELETE CASCADE;
```

Amish Bisaria

```
ALTER TABLE Cart_Item ADD CONSTRAINT fkcartitem2 FOREIGN KEY(ePID)
REFERENCES Item(ePID)
  ON DELETE CASCADE;
ALTER TABLE Watchlist_Item ADD CONSTRAINT fkwatchitem1 FOREIGN
KEY(WatchID) REFERENCES WatchList(WatchID)
  ON DELETE CASCADE;
ALTER TABLE Watchlist_Item ADD CONSTRAINT fkwatchitem2 FOREIGN KEY(ePID)
REFERENCES Item(ePID)
  ON DELETE CASCADE;
ALTER TABLE Seller_Carrier ADD CONSTRAINT fksellcarr1 FOREIGN KEY(SMemberID)
REFERENCES Seller(MemberID)
  ON DELETE CASCADE;
ALTER TABLE Seller_Carrier ADD CONSTRAINT fksellcarr2 FOREIGN KEY(CarrierID)
REFERENCES Carrier(CarrierID)
  ON DELETE CASCADE;
```

Amish Bisaria

PL/SQL Procedures and Triggers:

```
--PROCEDURES

/* Procedure that puts a discount on all items from a certain seller.
Useful for sellers that want to give a specific discount on all their items at once.*/

create or replace PROCEDURE Give_Seller_Discount(SellerMemID IN INTEGER, PercentOff
IN DECIMAL(10,2)) AS

thisItem Item%ROWTYPE;

CURSOR SellerItems IS
SELECT I.* FROM Item I WHERE I.SMemberID=SellerMemID FOR UPDATE;

BEGIN
OPEN SellerItems;
LOOP
  FETCH SellerItems INTO thisItem;
  EXIT WHEN (SellerItems%NOTFOUND);
  dbms_output.put_line('Item' || thisItem.ePID || ' has been given a discount.');
  UPDATE Item SET Price = Price * (1-PercentOff)
  WHERE ePID = thisItem.ePID;
END LOOP;
CLOSE SellerItems;
END;
```

Amish Bisaria

```
/* Procedure that gets the highest bid amount for an auction.
Useful for getting the winning bidder of an auction.*/

create or replace PROCEDURE Get_Highest_Bid_Amount(AuctionID IN INTEGER, bestBid
OUT Bid.BidAmount%TYPE) AS

thisBid Bid%ROWTYPE;

CURSOR AuctionBids IS
SELECT B.* FROM Auction A, Bid B WHERE A.AuctionID=B.AuctionID;

BEGIN
bestBid := 0;
OPEN AuctionBids;
LOOP
  FETCH AuctionBids INTO thisBid;
  EXIT WHEN (AuctionBids%NOTFOUND);
  IF (thisBid.BidAmount > bestBid) THEN
    bestBid := thisBid.BidAmount;
  END IF;
END LOOP;
CLOSE AuctionBids;
END;
```

Amish Bisaria

```
--TRIGGERS

/* Trigger that makes sure that item price is valid and shows information of price change when
an item price changes.
Useful to make sure that item price cannot be changed to less than 99 cents.
This is because 99 cents is the minimum amount for a fixed price sale on eBay.*/

create or replace TRIGGER Check_Item_Price_Change
BEFORE UPDATE OF Price ON Item
FOR EACH ROW

DECLARE
  price_change DECIMAL(10,2);
BEGIN
  price_change := :NEW.Price - :OLD.Price;
  IF (:NEW.Price < .99) THEN
    Raise_Application_Error(-20000, 'Item price cannot be less than $0.99.');
  ELSE IF (price_change > 0) THEN
    dbms_output.put('Price of item ' || Item.ePID || ' will increase by ' || price_change || '.');
  ELSE IF (price_change < 0) THEN
    dbms_output.put('Price of item ' || Item.ePID || ' will decrease by ' || price_change || '.');
  END IF;
END;
```

Amish Bisaria

```
/* Trigger that checks if a bid is valid by making sure the auction has not ended.
Useful to make sure bids are not placed on an auction after the end date.*/

create or replace TRIGGER Check_Valid_Bid
BEFORE INSERT OF BidID ON Bid
FOR EACH ROW

DECLARE
  thisAuction Auction%ROWTYPE;

BEGIN
  SELECT A.*
  INTO thisAuction
  FROM Auction
  WHERE :NEW.AuctionID = AuctionID;

  IF (thisAuction.EndDate < SYSDATE) THEN
    Raise_Application_Error(-20001, 'Invalid bid. Auction has already ended.');
  END IF;
END;
```