eBay: Database Design Final Project

Course: CS6360 (Database Design)
Team Members Name:Prateek

What is eBay?

eBay is an American multinational e-commerce company that deals with customer-to-customer and business-to-customer sales online through their website. The company was founded by Pierre Omidyar in 1995 and is headquartered in San Jose, California. The company looks after the eBay website that serves worldwide and is an online auction and shopping website where customers and businesses buy and sell a wide variety of products. According to eBay, whether you are buying a new or used, plain or luxurious, commonplace or rare, trendy or one-of-a-kind – if it exists in the world, it probably is for sale on eBay. Their mission is to be the world's favourite destination for discovering great value and unique selection. They give their sellers the platform, solutions and support they need to grow their businesses and thrive. eBay believes in measuring their success by their customers' success. Their vision for commerce is one that is enabled by people, powered by technology, and open to everyone.

System Design

eBay is an online shopping portal where people can buy or sell products. Buyers can buy new or used products which are sold by certains users. There are also some products which are sold in auctions. The website displays the number of bids made on the product and the highest bid at the current time. The user can bid as per the rates shown and the winner is decided once the time runs out. Also the eBay system involves various shipping companies which help in taking the products from the sellers to the buyers and these shipping companies have delivery agents which carry these products from the company warehouses to the buyer's address. Also eBay provides the buyer the chance to lodge complaints in case of any difficulties faced and the company keeps certain employees as the customer care representative who resolves these issues and the buyers can then rate the help they got from these representatives. Also the buyers can rate the products received and also the seller can be given ratings which ultimately rates the sellers on different aspects.

Users/Actors

Customers/ Buyer - A member who can view and buy products listed by various sellers, pay for the items purchased,provide reviews to the products bought. He can also initiate refunds for a bought item, raise an issue with a customer service agent who is basically a customer care representative for eBay, and can also bid for items which are listed for auctions on the website.

Seller - A member who can list their products online, receive payments for items sold, prepare products well in time for the shipment company to pickup and deliver. In case the product is not upto the mark, he is also responsible to replace/return the product and accordingly refund the money back to the customer.

Customer Care representatives- Customer service agents who attend to queries and complaints raised by customers, take it up with the respective teams, escalate issues and finally provide resolutions.

Shipment company - Users who handle the orders placed on the web portal and fulfill them using distribution logistics. The company has delivery agents and is responsible for timely and safe delivery of products from sellers to buyers.

Delivery agents - Handles the orders that the company receives and physically travels from the company's warehouses to the customers' shipping addresses to complete the transaction. They also take care of pick-up items for replacement or returns.

Auction Manager - User is responsible for auctioning the products on eBay. He decides which products are to be put up for auction, deals with different bids and finalizes the best bidder for a product. He supervises the entire auction process at eBay.

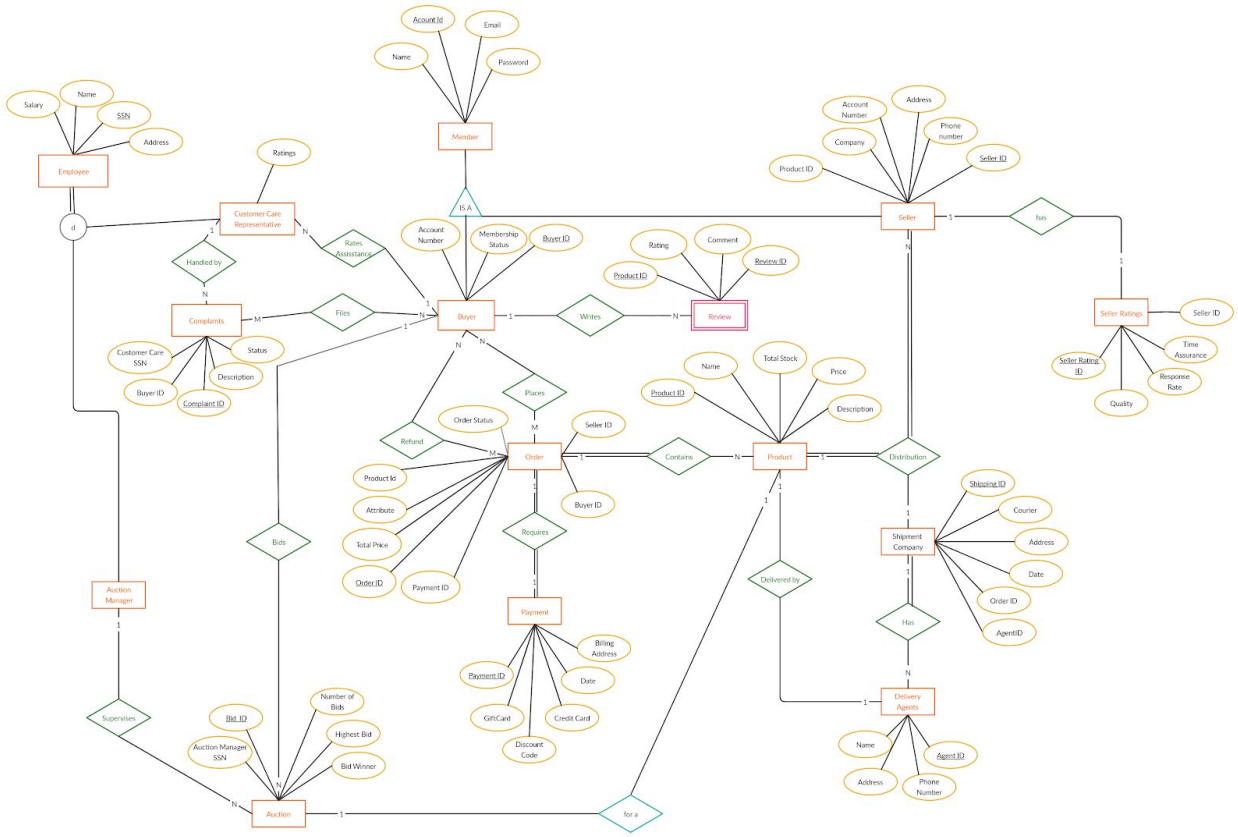
Working of the System

Online Purchase System - This system allows a customer to browse through the products, prepare the cart, become a member on the website and move for checkout. The customer then pays via a preferred mode of payment. If the payment is successful, the order is placed.

Auction System - Here, if the customer wants, he can bid for the products that are available for auction. The auction manager decides on who wins the bid and then the product goes to the chosen customer who can proceed to pay.

Delivery system - Once an order is placed, the seller is notified along with the shipment company. Once the order is ready from the seller's side, the shipment company takes charge and delivers it to the customer's given address.

ER DIAGRAM

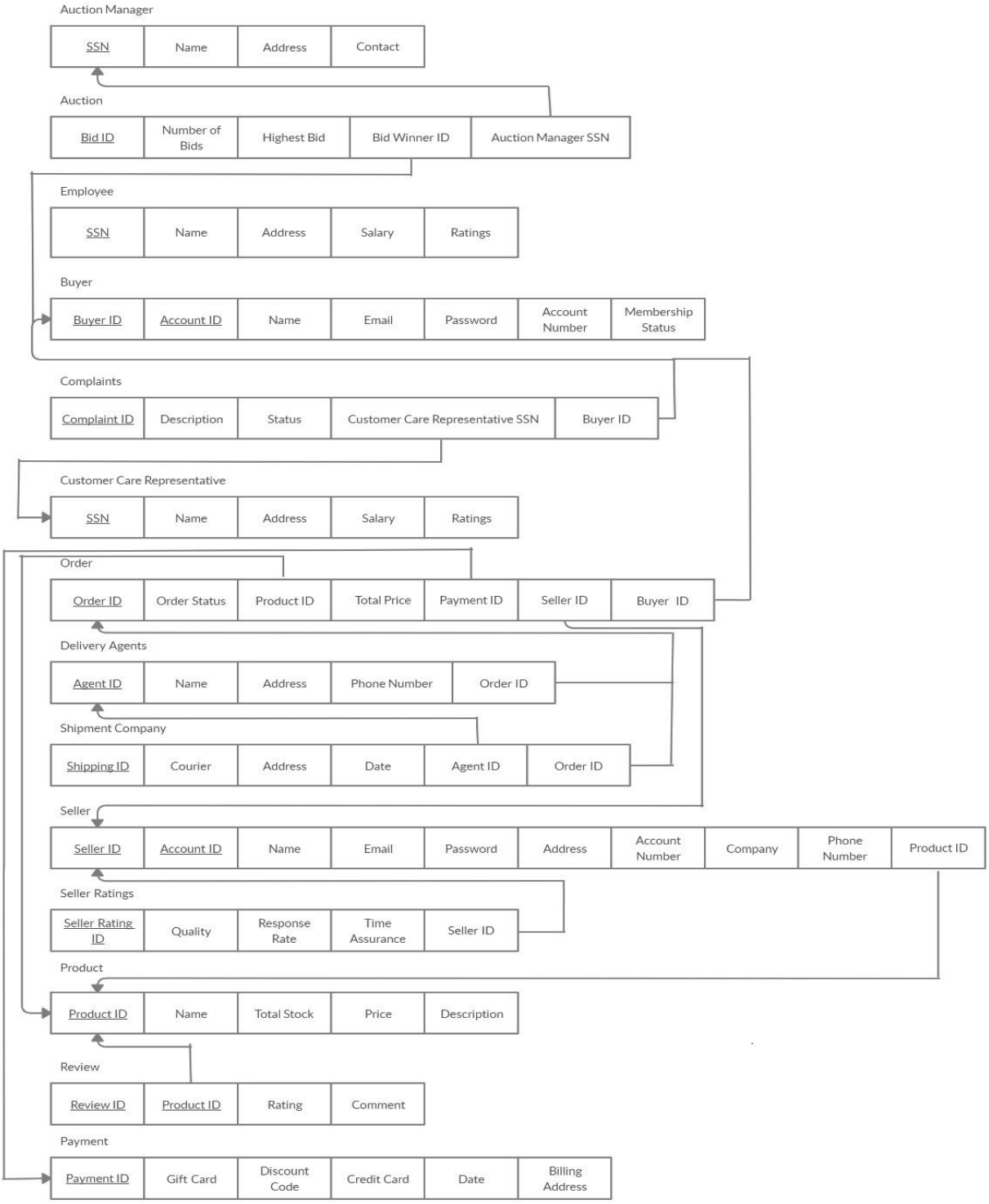

Acount Id
Email
Name
Password
Member
IS A
Salary
Name
SSN
Address
Employee
d
Ratings
Customer Care Representative
Rates Assistance
Handled by
Complaints
Files
Customer Care SSN
Status
Description
Buyer ID
Complaint ID
Account Number
Membership Status
Buyer ID
Buyer
Writes
Rating
Comment
Review ID
Product ID
Review
Account Number
Address
Phone number
Company
Product ID
Seller ID
Seller
has
Seller Ratings
Seller ID
Seller Rating ID
Time Assurance
Response Rate
Quality
Places
Refund
Order Status
Seller ID
Order
Product Id
Attribute
Total Price
Order ID
Payment ID
Buyer ID
Requires
Payment
Payment ID
Billing Address
Date
GiftCard
Credit Card
Discount Code
Contains
Name
Total Stock
Price
Product ID
Description
Product
Distribution
Bids
Auction Manager
Supervises
Bid ID
Auction Manager SSN
Number of Bids
Highest Bid
Bid Winner
Auction
for a
Delivered by
Shipment Company
Shipping ID
Courier
Address
Date
Order ID
AgentID
Has
Delivery Agents
Name
Agent ID
Address
Phone Number

Interactions between Different Entities

Each customer care representative can handle N number of complaints.
Each Auction Manager can supervise N number of auctions.
Each user can rate assistance for N number of customer care representatives.
Each user can bid in N number of auctions.
Each auction is for a single product.
Each order requires a single payment.
N buyers can file M complaints.
N buyers can place M number of orders.
N buyers can also ask for a refund for M number of orders.
Each buyer can write N number of reviews.
Each order contains N number of products.
Each product is delivered by a single delivery Agent.
Each product is handled by a single shipping company.
Each product can be sold by N number of different sellers.
Each shipping company can have N number of delivery agents.
Each seller has one single Seller Rating.

It is also evident from the ER diagram that each and every order made also needs a payment to be done. Hence both these quantities participate totally. Thus a double line in their relationship shows a total participation from their ends.
Also each product that has been sold as part of an order needs to be sold by some sellers. Similarly, each seller must have some items sold. Hence both of these entities also show total participation.
Finally, each shipping company will have some delivery agents has it also depicts a total participation.

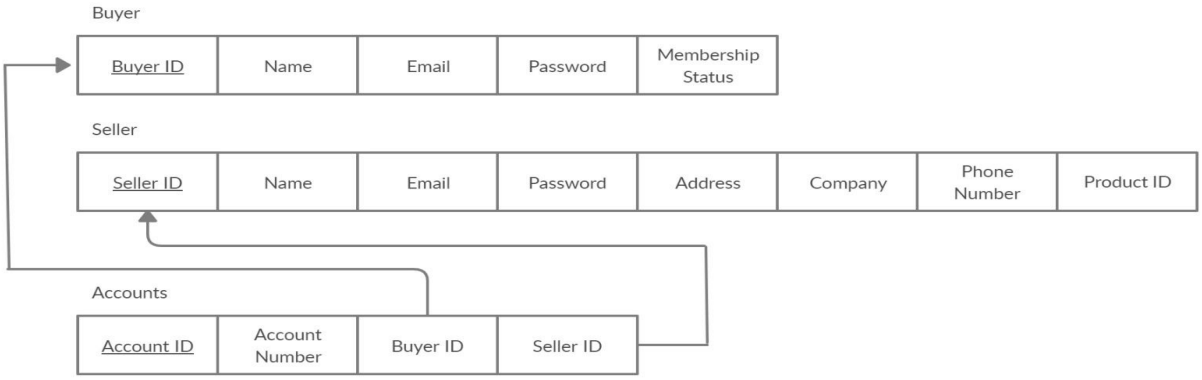
Converting the ER diagram to Relational Schema



Normalization

Converting to 1NF: A relational schema is said to be in 1NF if it does not have any attribute which can have more than one value for the same row. If there does exist such a column we break the multi-valued column to single-valued columns. Since, in our relational schema we do not have any attribute that can hold multiple values for a transaction, we can say that our relational schema is in 1NF form.

Converting to 2NF: A relational schema is said to be in 2NF form if it is in 1NF form and also if there are no partial dependency in the schema. A partial dependency is one in which an attribute depends on the part of the primary key and not on the complete key for its value. In our case such an example exists in the BUYER and SELLER table where the Account Number is dependent on the AccountID and not on the Buyer or Seller ID. Hence we split the BUYER and SELLER tables to form a third ACCOUNTS table which contains the information of the Account numbers and Account ID and also if that account is an account of the buyer or seller or both (in some cases a buyer can also be a seller). The affected tables are shown below.



Converting to 3NF: For a relational schema to be in 3NF it should satisfy two conditions. Firstly, it should be in 2NF and secondly there should be no transitive indepence in the schema. A transitive independence in one in which a non-key attribute depends on another non-key attribute. In our case, there is one scenario in the ORDERS table where the TOTAL_COST is dependent on the non-key attribute PRODUCT_ID. Hence, we can create a new table which will have these details which include the PRODUCT_ID and the quantity order based on which the TOTAL_COST can be calculated. The affected tables are shown below.

Order

Order ID	Order Status	Product ID	Payment ID	Seller ID	Buyer ID

Total Cost

Product ID	Quantity	Total Price

Final relational schema in 3NF: The final relational schema is shown below.

SQL commands for implementing the tables and their outputs

```
create table Auction_Manager(
SSN INTEGER,Name Varchar(20),
 Address Varchar(20),
Contact integer,
 primary key(SSN));
```

```
Name    Null?    Type
------- -------- ------------
SSN     NOT NULL NUMBER(38)
NAME             VARCHAR2(20)
ADDRESS          VARCHAR2(20)
CONTACT          NUMBER(38)
```

```
create table EMPLOYEE(
SSN integer,
Name varchar(20),
Address varchar(20),
salary INTEGER,
ratings INTEGER,
primary key(SSN));
```

```
Table EMPLOYEE created.

Name    Null?    Type
------- -------- ------------
SSN     NOT NULL NUMBER(38)
NAME             VARCHAR2(20)
ADDRESS          VARCHAR2(20)
SALARY           NUMBER(38)
RATINGS          NUMBER(38)
```

```
create table BUYER(
BuyerID INTEGER,
name VARCHAR(20),
email VARCHAR(20),
password VARCHAR(8) not null,
membership_status varchar(20),
primary key(BuyerID));
```

```
Name              Null?    Type
----------------- -------- ------------
BUYERID           NOT NULL NUMBER(38)
NAME                       VARCHAR2(20)
EMAIL                      VARCHAR2(20)
PASSWORD          NOT NULL VARCHAR2(8)
MEMBERSHIP_STATUS          VARCHAR2(20)
```

create table **AUCTION**(
BidID INTEGER,
Number_of_Bids integer,
Highest_Bid INTEGER,
Bid_winner_ID integer,
Auction_manager_SSN integer,
primary KEY (BidID),
FOREIGN KEY(Auction_manager_SSN) REFERENCES Auction_manager(SSN) on delete set null,
FOREIGN KEY(Bid_winner_ID) REFERENCES BUYER(BuyerID) on delete set null);

```
Name                Null?    Type
------------------- -------- ----------
BIDID               NOT NULL NUMBER(38)
NUMBER_OF_BIDS               NUMBER(38)
HIGHEST_BID                  NUMBER(38)
BID_WINNER_ID                NUMBER(38)
AUCTION_MANAGER_SSN          NUMBER(38)
```

create table **Customer_Care_representative**(
SSN INTEGER,
EName VARCHAR(20),
address VARCHAR(20),
salary INTEGER,
ratings INTEGER,
primary KEY (SSN));

```
Name    Null?    Type
------- -------- ------------
SSN     NOT NULL NUMBER(38)
ENAME            VARCHAR2(20)
ADDRESS          VARCHAR2(20)
SALARY           NUMBER(38)
RATINGS          NUMBER(38)
```

create table **COMPLAINTS** (
ComplaintID INTEGER,
complaint_description VARCHAR(20),
Status VARCHAR(20),
Customer_Care_SSN INTEGER not null,
Buyer_ID integer not null, primary key(ComplaintID),
foreign key(Customer_care_SSN) REFERENCES Customer_Care_Representative(SSN),
foreign key(buyer_ID) references BUYER(buyerID) on delete set null);

```
Name                 Null?    Type
-------------------- -------- ------------
COMPLAINTID          NOT NULL NUMBER(38)
COMPLAINT_DESCRIPTION         VARCHAR2(20)
STATUS                        VARCHAR2(20)
CUSTOMER_CARE_SSN    NOT NULL NUMBER(38)
BUYER_ID             NOT NULL NUMBER(38)
```

create table **PRODUCT**(
ProductID INTEGER,
ProductName VARCHAR(20),
Stock INTEGER, price INTEGER not null,
Product_description varchar(30) not null,
primary key(ProductID));

```
Name                Null?    Type
------------------- -------- ------------
PRODUCTID           NOT NULL NUMBER(38)
PRODUCTNAME                  VARCHAR2(20)
STOCK                        NUMBER(38)
PRICE               NOT NULL NUMBER(38)
PRODUCT_DESCRIPTION NOT NULL VARCHAR2(40)
```

create table **REVIEW**(
ReviewID INTEGER,
ProductID INTEGER,
Rating INTEGER,

Customer_Comment VARCHAR(40),
PRIMARY KEY (reviewID, ProductID),
foreign key(productId) REFERENCES product(productID) on delete set null);

```
Name             Null?    Type
---------------- -------- ------------
REVIEWID         NOT NULL NUMBER(38)
PRODUCTID        NOT NULL NUMBER(38)
RATING                    NUMBER(38)
CUSTOMER_COMMENT          VARCHAR2(40)
```

create table **PAYMENT**(
PaymentID integer,
gift_card INTEGER,
discount_code INTEGER,
credit_card INTEGER,
Payment_date date,
billing_address VARCHAR(20),
primary key(paymentID));

```
Name             Null?    Type
---------------- -------- ------------
PAYMENTID        NOT NULL NUMBER(38)
GIFT_CARD                 NUMBER(38)
DISCOUNT_CODE             NUMBER(38)
CREDIT_CARD               NUMBER(38)
PAYMENT_DATE              DATE
BILLING_ADDRESS           VARCHAR2(20)
```

create table **SELLER**(
SellerID integer,
SellerName VARCHAR(20),
email VARCHAR(20),
AccountPassword VARCHAR(20),
Address varchar(20),
Company varchar(20),
PhoneNumber integer,
ProductID integer,
PRIMARY KEY (sellerID),
foreign key (productID) REFERENCES product(productID) on delete set null);

```
Name            Null?    Type
--------------- -------- ------------
SELLERID        NOT NULL NUMBER(38)
SELLERNAME               VARCHAR2(20)
EMAIL                    VARCHAR2(20)
ACCOUNTPASSWORD          VARCHAR2(20)
ADDRESS                  VARCHAR2(20)
COMPANY                  VARCHAR2(20)
PHONENUMBER              NUMBER(38)
PRODUCTID                NUMBER(38)
```

create table **ACCOUNTS** (
AccountNumber integer,
AccountID integer,
BuyerID integer, SellerId integer,
PRIMARY KEY (accountID),
foreign key(sellerID) REFERENCES seller(sellerID) on delete set null,
foreign key (buyerID) references buyer(buyerID) on delete set null);

```
Name          Null?    Type
------------- -------- ----------
ACCOUNTNUMBER          NUMBER(38)
ACCOUNTID     NOT NULL NUMBER(38)
BUYERID                NUMBER(38)
SELLERID               NUMBER(38)
```

create table **Seller_ratings**(
SellerRatingID integer,
quality varchar(20),
response_time varchar(20),
TimeAssurance varchar(20),
SellerID integer,
primary key (sellerRatingID),
foreign key(sellerID) REFERENCES seller(sellerID) on delete set null);

```
Name           Null?    Type
-------------- -------- ------------
SELLERRATINGID NOT NULL NUMBER(38)
QUALITY                 VARCHAR2(20)
RESPONSE_TIME           VARCHAR2(20)
TIMEASSURANCE           VARCHAR2(20)
SELLERID                NUMBER(38)
```

create table **ORDERS**(
orderID integer,
orderStatus varchar(10),
productID integer,
paymentID integer not null,
sellerID integer not null,
buyerID integer not null,
PRIMARY KEY (orderID),
foreign key(productid)references product(productID) on delete set null,
foreign key(paymentID) references payment(paymentID) on delete set null,
foreign key(sellerID) references seller(sellerID) on delete set null,
foreign key(buyerID) references buyer(buyerID) on delete set null);

```
Name        Null?    Type
----------- -------- ------------
ORDERID     NOT NULL NUMBER(38)
ORDERSTATUS          VARCHAR2(10)
PRODUCTID            NUMBER(38)
PAYMENTID   NOT NULL NUMBER(38)
SELLERID    NOT NULL NUMBER(38)
BUYERID     NOT NULL NUMBER(38)
```

create table **delivery_agent**(
AgentID integer,
AgentName varchar(20),
Address varchar(20),
PhoneNumber integer,
OrderID integer,
primary key(AgentID),
foreign key (orderID) REFERENCES orders(orderID) on delete set null);

```
Name        Null?    Type
----------- -------- ------------
AGENTID     NOT NULL NUMBER(38)
AGENTNAME            VARCHAR2(20)
ADDRESS              VARCHAR2(20)
PHONENUMBER          NUMBER(38)
ORDERID              NUMBER(38)
```

create table **shipment_company**(
shippingID integer,
courierName varchar(20),
address varchar(20),
date_of_delivery date,
orderID integer,
agentID integer,
primary key(ShippingID),
foreign key (orderID) references Orders(OrderID) on delete set null,
foreign key(agentID) references delivery_agent(AgentID) on delete set null);

```
Name             Null?    Type
---------------- -------- ------------
SHIPPINGID       NOT NULL NUMBER(38)
COURIERNAME               VARCHAR2(20)
ADDRESS                   VARCHAR2(20)
DATE_OF_DELIVERY          DATE
ORDERID                   NUMBER(38)
AGENTID                   NUMBER(38)
```

create table **total_cost**(
productID integer,
quantity integer,
tax integer,
total_cost integer,
primary key(ProductID),
foreign key(productId) references product(productID) on delete set null);

```
Name       Null?    Type
---------- -------- ----------
PRODUCTID  NOT NULL NUMBER(38)
QUANTITY            NUMBER(38)
TAX                 NUMBER(38)
TOTAL_COST          NUMBER(38)
```

TWO STORED PROCEDURES:

Procedure 1 : **Procedure to delete orders from orders table once it is in completed status.**

In this procedure we check the status of the orders from the orders table. Once the order status is completed we then delete the order.

```
Create or replace
procedure show_completed_orders
as
begin
dbms_output.put_line('The following orders have been completed');
for item IN
(
select orderID from orders where OrderStatus = 'Completed';
)
LOOP
dbms_output.put_line('The order'||item.orderID||'are completed and can be deleted');
```

Procedure 2: **Procedure to increase the salary of all customer care representative by 50USD if they have a perfect rating of 5.**

In this stored procedure we aim to provide increment of USD50 to all the customer care representatives who have a perfect rating of 5.0.

```
Create or Replace
Procedure give_increment
As
Begin
DBMS_OUTPUT.PUT_LINE('The following Customer Care representatives have perfect
rating');
For items IN
(
        Select * from customer_care_representative where rating = 5;
)
LOOP
Update table customer_care_representative set salary = salary+50 where SSN = item.SSN;
DBMS_OUTPUT.PUT_LINE('The Customer Care representatives with SSn' || item.SSN ||'have
been given an increment of USD50');
END LOOP;
END
```

TWO TRIGGERS:

Trigger 1: Trigger whenever a new order has been placed

```
create or replace trigger new_order_placed
AFTER insert
on orders
for each row
begin
   dbms_output.put_line('New Order has been placed');
END;
```

Trigger 2: Removing customer care representatives that have poor ratings

```
create or replace trigger train_representative
AFTER update
on Customer_care_representative
for each row
begin
   select rating from customer_care_representative;
   if(rating<1)
   then
   delete from customer_care_representative where rating<1;
   dbms_output.put_line('Due to poor ratings the customer care representative has been
removed');
END;
```

Conclusion

We started off by collecting the requirements of the system and then studying the different aspects of the system. We started off by writing down the different subsystems and their working. We then made the ER/EER diagram from these requirements and then converted this ER diagram to the relational schema. Later we used the normalization rules to normalise this relational schema to 3NF form. Then we implemented the normalised tables using SQL. We also created 2 stored procedures and 2 triggers thus completing a small model of a larger database model for eBay.