eBay

CS6360.002

Team 2

<u>Team Members</u>

Asif Sohail Mohammed (axm190041)

Abhishek Ramesh Hosmani (axr190014)

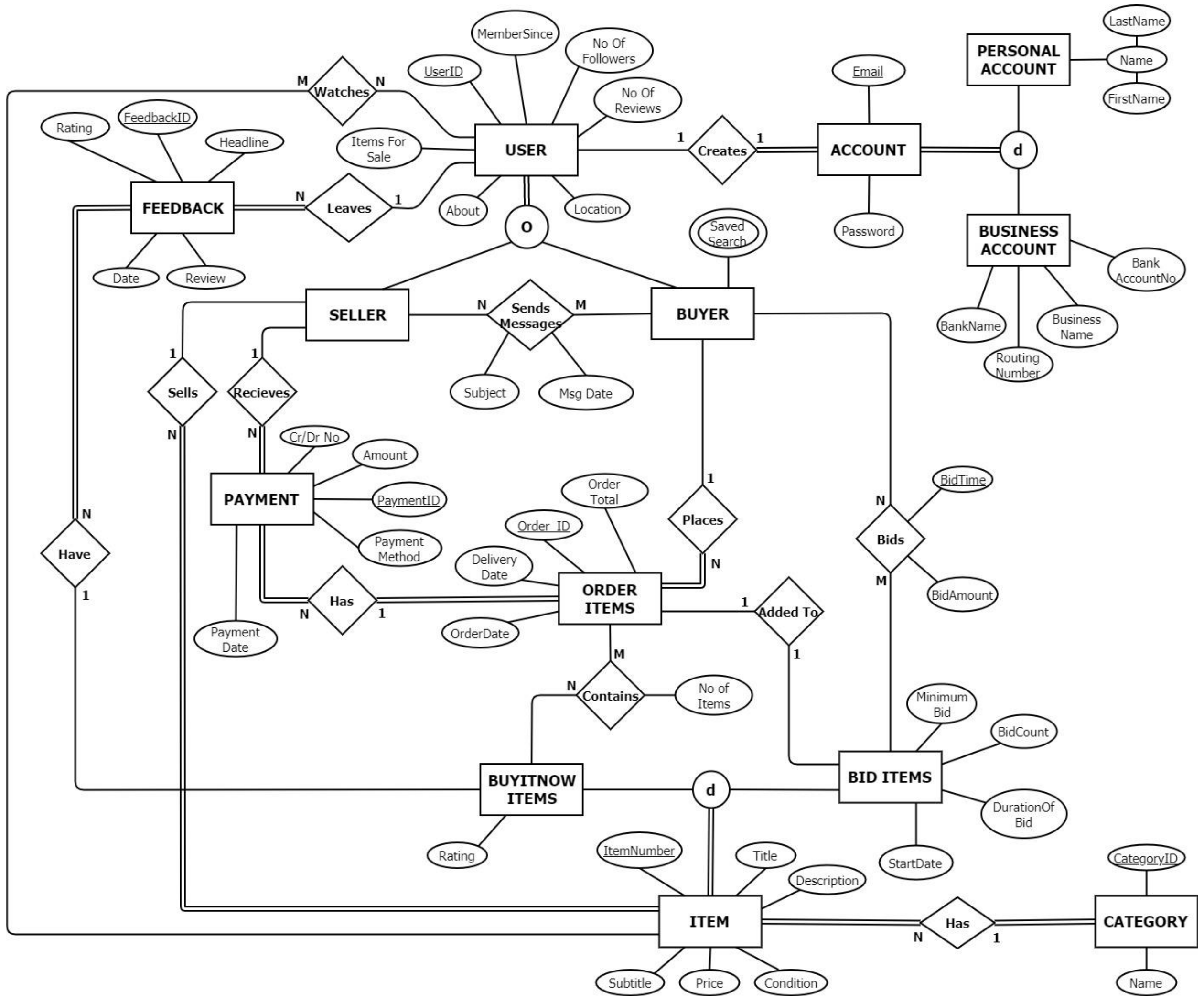
Sayan Guha (sxg190031)

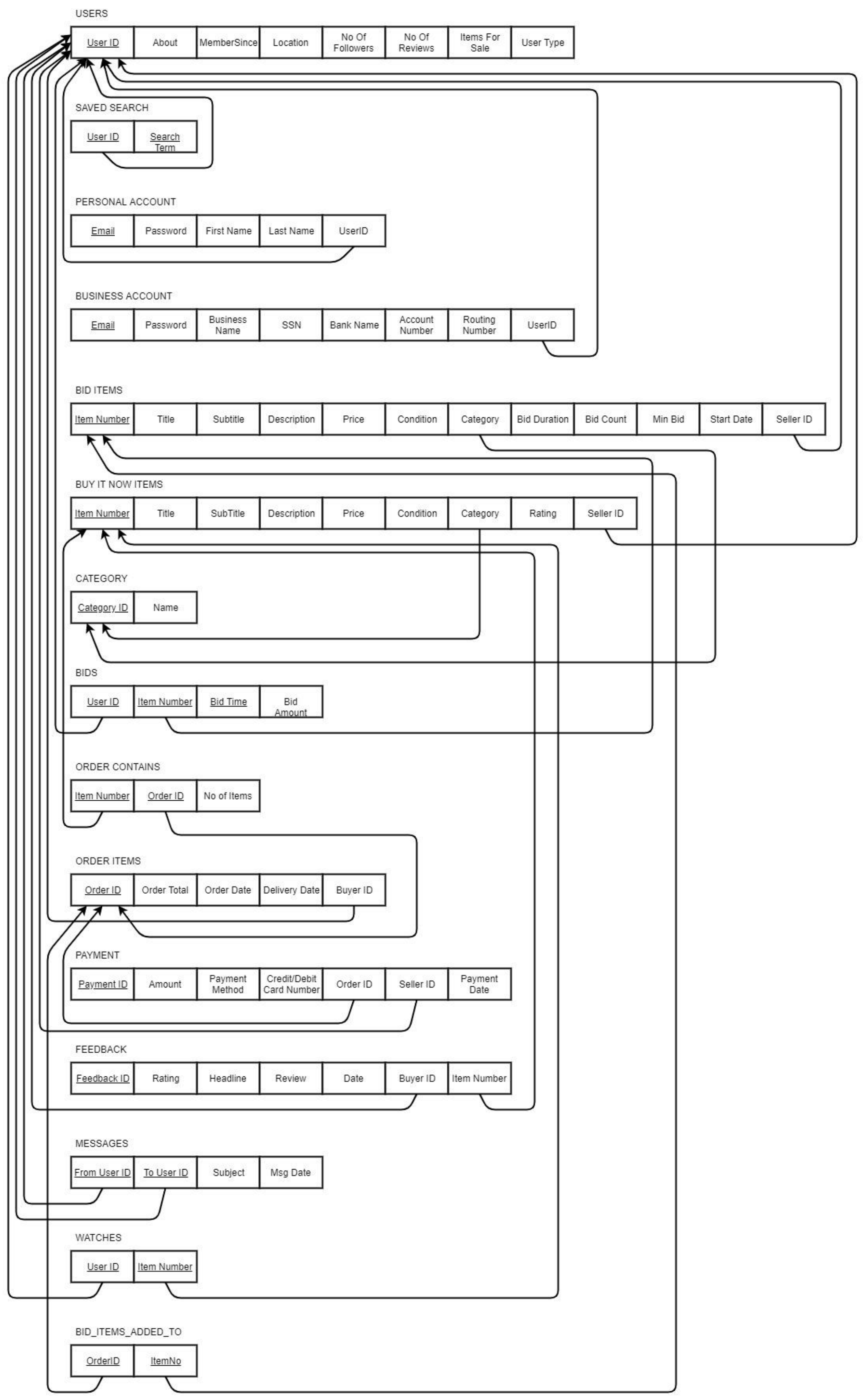
Requirements

- eBay has USERS, every user has only one ACCOUNT with eBay. Each ACCOUNT has a unique   user ID, about the user, member since (when user has created the account), location of the user, number of followers, number of reviews, number of items for sale. Also, every ACCOUNT will be associated with one user.
- Every ACCOUNT is categorized into two types, PERSONAL_ACCOUNT or BUSINESS_ACCOUNT.
- PERSONAL_ACCOUNT has details about the user's first name and last name.
- BUSINESS ACCOUNT has legal business name, bank account number, bank name, and routing number.
- Every user in USERS can be both BUYER or SELLER at the same time. BUYER can search for anything and can save any number of search terms.
- BUYER/SELLER can send messages to SELLER/BUYER respectively. Every message has subject and date the message has been sent.
- eBay has ITEMS, every item has unique item number, title which can't be n, subtitle, description, price of the item and condition (New, Used, Open box).
- Every item has a CATEGORY to which it belongs to and every CATEGORY has at least one ITEM.
- Every ITEM is divided in to either BUY_IT_NOW_ITEM or BID_ITEM. Buyer can buy BUY_IT_NOW_ITEMS whenever he wants to buy but to buy BID_ITEM buyer has to successfully win a bid.
- BID_ITEMS have minimum bid amount, no of current bids on the item, duration of the bid (which can be 1, 3, 5, 7 day(s)), start date when bid is created by the seller.
- BUY_IT_NOW_ITEMS have rating and which will be calculated based on the FEEDBACK received from the BUYER.
- BUYER can place bid on multiple BID_ITEMS and on each BID_ITEM multiple byers can place bid. Each bid has bid amount and bid time.
- When the bid time expires BID_ITEM should be bought by BUYER who bids the highest amount. BID_ITEM will be added to the ORDER and PAYMENT should be done. Not all BID_ITEM will have bids and such items will remain unsold.
- BUYER can place multiple ORDERS, which has unique order ID, order total, order date and expected delivery date.
- Every ORDER contains either at least one BUY_IT_NOW_ITEM and count of items in the order or exactly one BID_ITEM. Each BUY_IT_NOW_ITEMS can be part of multiple ORDERS while each BID_ITEM will be part only one ORDER.

- PAYMENT is done for every ORDER and vice versa. PAYMENT for each ORDER can be made in multiple methods (credit/debit card and gift card).
- PAYMENT has unique payment ID, payment method, amount, credit or debit card number and date of payment.
- SELLER receives multiple payments but every PAYMENT corresponds to only one SELLER.
- USER can watch multiple ITEMS to keep track of the price or bid price of an item. Each ITEM can be watched by multiple USERS.
- Once the BUYER places ORDER, BUYER can give feedback to the BUY_IT_NOW_ITEMS he bought. FEEDBACK will have unique feedback ID, rating, headline about the item and date of feedback.
- BUY_IT_NOW_ITEMS will have rating which will be calculated from the FEEDBACK given by the BUYER. Each item can have multiple FEEDBACKS and every FEEDBACK is associated with a BUY_IT_NOW_ITEM.

ER Diagram and Relational Schema




USERS
User ID | About | MemberSince | Location | No Of Followers | No Of Reviews | Items For Sale | User Type
SAVED SEARCH
User ID | Search Term
PERSONAL ACCOUNT
Email | Password | First Name | Last Name | UserID
BUSINESS ACCOUNT
Email | Password | Business Name | SSN | Bank Name | Account Number | Routing Number | UserID
BID ITEMS
Item Number | Title | Subtitle | Description | Price | Condition | Category | Bid Duration | Bid Count | Min Bid | Start Date | Seller ID
BUY IT NOW ITEMS
Item Number | Title | SubTitle | Description | Price | Condition | Category | Rating | Seller ID
CATEGORY
Category ID | Name
BIDS
User ID | Item Number | Bid Time | Bid Amount
ORDER CONTAINS
Item Number | Order ID | No of Items
ORDER ITEMS
Order ID | Order Total | Order Date | Delivery Date | Buyer ID
PAYMENT
Payment ID | Amount | Payment Method | Credit/Debit Card Number | Order ID | Seller ID | Payment Date
FEEDBACK
Feedback ID | Rating | Headline | Review | Date | Buyer ID | Item Number
MESSAGES
From User ID | To User ID | Subject | Msg Date
WATCHES
User ID | Item Number
BID_ITEMS_ADDED_TO
OrderID | ItemNo

Normalization

All the tables are already in 3NF, normalization is not required.

SQL Tables

```
DROP TABLE USERS;

CREATE TABLE USERS (

user_id                     varchar(15)         PRIMARY KEY,

about                   varchar(50),

member_since                date                NOT NULL,

user_location           varchar(20)         NOT NULL,

no_of_followers         int                 DEFAULT 0,

items_for_sale          int                 DEFAULT 0,

user_type               varchar(6)          NOT NULL CHECK(user_type in ('Buyer',
'Seller'))

);
```

```
DROP TABLE PERSONAL_ACCOUNT;
CREATE TABLE PERSONAL_ACCOUNT (
email                   varchar(30)        PRIMARY KEY,
password                varchar(20)        NOT NULL CHECK(LENGTH(password)>6),
first_name              varchar(20)        NOT NULL,
last_name               varchar(20)        NOT NULL,
user_id                         varchar(15),
FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE
);



DROP TABLE BUSINESS_ACCOUNT;
CREATE TABLE BUSINESS_ACCOUNT (
email                   varchar(25)        PRIMARY KEY,
password                varchar(20)        CHECK(length(password)>6),
business_name                   varchar(20)        NOT NULL,
ssn                     char(10)           NOT NULL,
bank_name                varchar(20)       NOT NULL,
account_number          varchar(11)        NOT NULL,
routing_number          varchar(8)         NOT NULL,
user_id                         varchar(15),
FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE
);
```

```
DROP TABLE SAVED_SEARCH;

CREATE TABLE SAVED_SEARCH (

user_id                                         varchar(15),

search_term                     varchar(50)               NOT NULL,

FOREIGN KEY(user_id) REFERENCES users (user_id) ON DELETE CASCADE,

PRIMARY KEY(user_id, search_term)

);




DROP TABLE CATEGORY;

CREATE TABLE CATEGORY (

category_id                     char(5)                   PRIMARY KEY,

name                            varchar(20)               NOT NULL

);
```

```
DROP TABLE BID_ITEMS;

CREATE TABLE BID_ITEMS (

Item_number                       char(12)            PRIMARY KEY,

Title                             varchar(30)         NOT NULL,

subtitle                                 varchar(20),

description                       varchar(50),

bid_price                         numeric(10,2)       NOT NULL,

condition                         varchar(15)         NOT NULL,

category                          char(5),

bid_duration                      int                 NOT NULL,

bid_count                         int            DEFAULT 0,

min_bid                           numeric(10,2)       NOT NULL,

start_date                        timestamp      DEFAULT SYSDATE,

seller_id                         varchar(15),

CHECK(condition in('Used', 'New', 'Refurbished', 'Openbox')),

CHECK(bid_duration in(1,3,5,7)),

FOREIGN KEY(category) REFERENCES category (category_id) ON DELETE SET NULL,

FOREIGN KEY(seller_id) REFERENCES USERS (user_id) ON DELETE CASCADE

);
```

```
DROP TABLE BUY_IT_NOW_ITEMS;

CREATE TABLE BUY_IT_NOW_ITEMS (

Item_number                 char(12)          PRIMARY KEY,

Title                       varchar(30)       NOT NULL,

subtitle                          varchar(20),

description                 varchar(50),

price                       numeric(10,2)     NOT NULL,

condition                   varchar(15)       NOT NULL,

category                    char(5),

rating                      numeric(2,1)      DEFAULT 0,

seller_id                   varchar(15),

CHECK(condition in('Used', 'New', 'Refurbished', 'Openbox')),

FOREIGN KEY(category) REFERENCES category (category_id) ON DELETE SET NULL,

FOREIGN KEY(seller_id) REFERENCES USERS (user_id) ON DELETE CASCADE

);
```

```
DROP TABLE BIDS;

CREATE TABLE BIDS (

user_id                                 varchar(15),

item_no                     char(12),

bid_amount                  numeric(10,2)          NOT NULL,

bid_time                    timestamp              DEFAULT SYSDATE,

FOREIGN KEY(user_id) REFERENCES users (user_id) ON DELETE CASCADE,

FOREIGN KEY(item_no) REFERENCES bid_items (item_number) ON DELETE CASCADE,

PRIMARY KEY(user_id, item_no, bid_time)

);




DROP TABLE ORDER_ITEMS;

CREATE TABLE ORDER_ITEMS (

order_id                    char(10)                       PRIMARY KEY,

order_total                 numeric(10,2),

order_date                  date                           DEFAULT SYSDATE,

delivery_date               date,

buyer_id                    varchar(15),

FOREIGN KEY(buyer_id) REFERENCES users (user_id) ON DELETE CASCADE

);
```

```
DROP TABLE ORDER_CONTAINS;

CREATE TABLE ORDER_CONTAINS (

item_no                         char(12),

order_no                        char(10),

no_of_items                     int,

FOREIGN KEY(item_no) REFERENCES buy_it_now_items (item_number),

FOREIGN KEY(order_no) REFERENCES order_items (order_id),

PRIMARY KEY(item_no, order_no)

);




DROP TABLE PAYMENT;

CREATE TABLE PAYMENT (

payment_id                          char(10)                PRIMARY KEY,

amount                                      numeric(10,2)           NOT NULL,

payment_method                      varchar(20)             NOT NULL,

cr_dr_card_number                   char(16),

payment_date                        date                    DEFAULT SYSDATE,

order_id                            char(10),

seller_id                           varchar(15),

FOREIGN KEY(order_id) REFERENCES order_items (order_id) ON DELETE CASCADE,

FOREIGN KEY(seller_id) REFERENCES USERS (user_id) ON DELETE CASCADE

);
```

```
DROP TABLE FEEDBACK;
CREATE TABLE FEEDBACK (
feedback_id                 char(8)             PRIMARY KEY,
rating                      numeric(2,1)        NOT NULL,
headline                    varchar(20)         NOT NULL,
review                      varchar(50),
feedback_date                       date                DEFAULT SYSDATE,
buyer_id                    varchar(15),
item_no                     char(12),
CHECK(rating >= 0.0 AND rating <= 5.0),
FOREIGN KEY(buyer_id) REFERENCES users (user_id) ON DELETE CASCADE,
FOREIGN KEY(item_no) REFERENCES buy_it_now_items (item_number) ON DELETE CASCADE
);
```

```
DROP TABLE MESSAGES;
CREATE TABLE MESSAGES (
from_user_id                varchar(15),
to_user_id                  varchar(15),
subject                     varchar(50)         NOT NULL,
msg_date                    date                DEFAULT SYSDATE,
FOREIGN KEY(from_user_id) REFERENCES users (user_id) ON DELETE CASCADE,
FOREIGN KEY(to_user_id) REFERENCES users (user_id) ON DELETE CASCADE
);
```

```
DROP TABLE WATCHES;

CREATE TABLE WATCHES (

user_id                                                   varchar(15),

item_no                                         char(12),

FOREIGN KEY(user_id) REFERENCES users (user_id) ON DELETE CASCADE,

FOREIGN KEY(item_no) REFERENCES buy_it_now_items (item_number) ON DELETE CASCADE,

FOREIGN KEY(item_no) REFERENCES bid_items (item_number) ON DELETE CASCADE

);




DROP TABLE BID_ITEMS_ADDED_TO;

CREATE TABLE BID_ITEMS_ADDED_TO (

Order_id                                        char(10),

item_no                                         char(12),

FOREIGN KEY(order_id) REFERENCES order_items (order_id) ON DELETE CASCADE,

FOREIGN KEY(item_no) REFERENCES bid_items (item_number) ON DELETE CASCADE

);
```

Procedures

1. Procedure to calculate the item rating based on user feedback which includes rating and updates the item rating in item table.

```
set serveroutput ON;
create or replace procedure calculate_feedback(itemno IN feedback.item_no%TYPE,
currentRating IN feedback.rating%TYPE) AS
thisRating buy_it_now_items.rating%TYPE;
rating_sum numeric(5,1);
row_count numeric(5,1);

CURSOR calculate_rating IS
select rating from feedback where item_no = itemno;
BEGIN
        OPEN calculate_rating;
                rating_sum := 0.0;
        LOOP
                FETCH calculate_rating into thisRating;
                EXIT WHEN calculate_rating%NOTFOUND;
     rating_sum := rating_sum + thisRating;
        END LOOP;
  row_count := calculate_rating%ROWCOUNT;
  dbms_output.put_line(rating_sum || row_count);
  UPDATE buy_it_now_items set rating = (rating_sum+currentRating)/(row_count+1)
where item_number = itemno;
        CLOSE calculate_rating;
END;
```

2. Procedure that will raise application error when buyer tries to bid for an expired item. If not, it will make sure that the bid amount is not less than minimum amount you can bid and if it is less it will raise application error.

```
set serveroutput ON;
create or replace procedure bid_expiry(itemno IN bid_items.item_number%type,
currentBid bids.bid_amount%TYPE) AS
  thisEnddate bid_items.start_date%TYPE;
  thisMinBid bid_items.min_bid%TYPE;
BEGIN
        select start_date + bid_duration into thisEnddate from bid_items where
item_number = itemno;
  select min_bid into thisMinBid from bid_items where item_number = itemno;
  IF sysdate > thisEnddate THEN
    Raise_Application_Error(-20000, 'The item you are trying to bid has ended.');
  END IF;
  IF currentBid < thisMinBid THEN
    Raise_Application_Error(-20000, 'You have to bid atleast ' || thisMinBid);
  END IF;
END;
```

3. Procedure to calculate next minimum bid based on the price of latest bid and update the minimum bid in BID_ITEMS table.

```
set serveroutput ON;
create or replace procedure update_minbid(itemno IN bids.item_no%TYPE, bidPrice IN
bids.bid_amount%TYPE) AS
  minBidPrice bid_items.min_bid%TYPE;
  currentBidprice bid_items.bid_price%TYPE;

BEGIN
        currentBidprice := bidPrice;
        minBidPrice := 0.0;
        IF currentBidprice >= 0.01 AND currentBidprice <= 0.99 THEN
                minBidPrice := currentBidprice + 0.05;
  ELSIF currentBidprice >= 1.00 AND currentBidprice <= 4.99 THEN
    minBidPrice := currentBidprice + 0.25;
  ELSIF currentBidprice >= 5.00 AND currentBidprice <= 24.99 THEN
    minBidPrice := currentBidprice + 0.50;
  ELSIF currentBidprice >= 25.00 AND currentBidprice <= 99.99 THEN
    minBidPrice := currentBidprice + 1.00;
  ELSIF currentBidprice >= 100.00 AND currentBidprice <= 249.99 THEN
    minBidPrice := currentBidprice + 2.50;
  ELSIF currentBidprice >= 250.00 AND currentBidprice <= 499.99 THEN
    minBidPrice := currentBidprice + 5.00;
  ELSIF currentBidprice >= 500.00 AND currentBidprice <= 999.99 THEN
    minBidPrice := currentBidprice + 10.00;
  ELSE
    minBidPrice := currentBidprice + 25.00;
  END IF;
        UPDATE bid_items set min_bid = minBidPrice where item_number = itemno;
    UPDATE bid_items set bid_count = bid_count + 1 where item_number = itemno;
END;
```

4. Procedure to determine the bid winner when the bid expires and the item will be added to orders. After item has been ordered item will be removed from BID_ITEMS table along with its bid history in BIDS table.

```
set serveroutput ON;
create or replace procedure winning_bid(itemno IN bid_items.item_number%type) AS
  thisEnddate bid_items.start_date%TYPE;
  thisMaxBid bids.bid_amount%TYPE;
  thisBuyer bids.user_id%TYPE;
  thisOrderId order_items.order_id%TYPE;
BEGIN
        select start_date + bid_duration into thisEnddate from bid_items where
item_number = itemno;
  select max(bid_amount) into thisMaxBid from bids where item_no = itemno;
  select user_id into thisBuyer from bids where bid_amount = thisMaxBid and item_no
= itemno;
  select round(DBMS_Random.Value(0000000001,9999999999)) into thisOrderID from
dual;
  IF sysdate > thisEnddate THEN
    INSERT INTO order_items values(thisOrderId, thisMaxBid, sysdate, sysdate + 3,
thisBuyer);
    INSERT INTO bid_items_added_to values(thisOrderId, itemno);
    DELETE FROM BID_ITEMS where item_number = itemno;
    DELETE FROM BIDS where item_no = itemno;
  END IF;
END;
```

Triggers

1. Trigger that will calculate average rating of items based on user feedback and will be triggered whenever buyer gives feedback to an item i.e. after user inserts new data into feedback table. Trigger action will call the corresponding procedure (procedure 1).

```
CREATE or REPLACE TRIGGER UPDATE_RATING
AFTER INSERT ON FEEDBACK
FOR EACH ROW
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
          calculate_feedback(:NEW.item_no, :NEW.rating);
   COMMIT;
END;
```

2. Trigger that will check whether the bid is expired or not and also if your bid price is greater than minimum price that you can bid. It will be triggered whenever bidder bids for an item. Trigger action will call the corresponding procedure (procedure 2).

```
CREATE or REPLACE TRIGGER CHECK_EXPIRY
BEFORE INSERT ON BIDS
FOR EACH ROW
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
          BID_EXPIRY(:NEW.ITEM_NO, :NEW.bid_amount);
          COMMIT;
END;
```

3. Trigger that will calculate the minimum bid and update it in the BID_ITEMS table and will be triggered whenever bidder bids for an item i.e. after user inserts new data into bids table. Trigger action will call the corresponding procedure (procedure 3).

```
CREATE OR REPLACE TRIGGER update_bid_details
AFTER INSERT ON bids
FOR EACH ROW
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   update_minbid(:NEW.item_no, :NEW.bid_amount);
   COMMIT;
END;
```