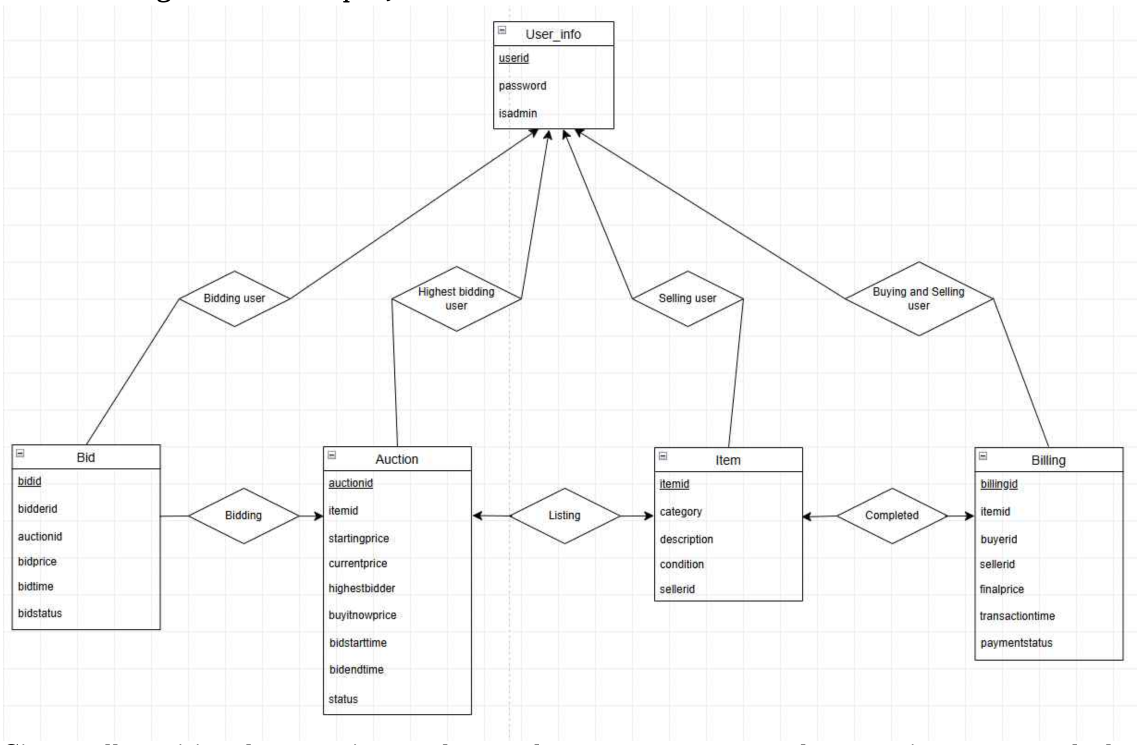
# Introduction to Database_SWE3003_41 Project1 Report

2021312738 소프트웨어학과 김서환

## 1) Introduction

Auction in Project1 is an auction system where a seller lists an item for selling, and bidders can either pay the buy-it-now-price to purchase it immediately or place a bidprice to compete with other bidders. After the bidendtime (the auction end time), the bidder who has offered the highest price wins the auction. The seller then receives the final selling price minus a 5% commission fee.

The ER-Diagram for the project is as follows.



Since all entities have primary keys, they are represented as entity sets, and the relationships between these entity sets are shown.

## 2) Demonstration Video Description

There is a brief interruption in the middle, so I kindly ask for your understanding. The attached project demonstration video proceeds in the following order:

1. Sign-up, user, and admin login menu – Demonstration of their operations and outputs.

2. Sell Item, Check Sell Status, Buy Item, Check Buy Status, Check Account – Basic operation and output process.

3. Bid Closing Time Check Scenarios (as required in the PDF) – 2nd case : Demonstration and output of the system displaying the message "Bid Ended." when a user attempts to place a bid after the bidding period has ended.

4. Bid Closing Time Check Scenarios (as required in the PDF) – 3rd case : When ‘Check Status of your Item Listed’, ‘Check Status of your Bid’, or ‘Check your Account’ is selected, demonstration of the correct transaction of items whose bidding period has ended.

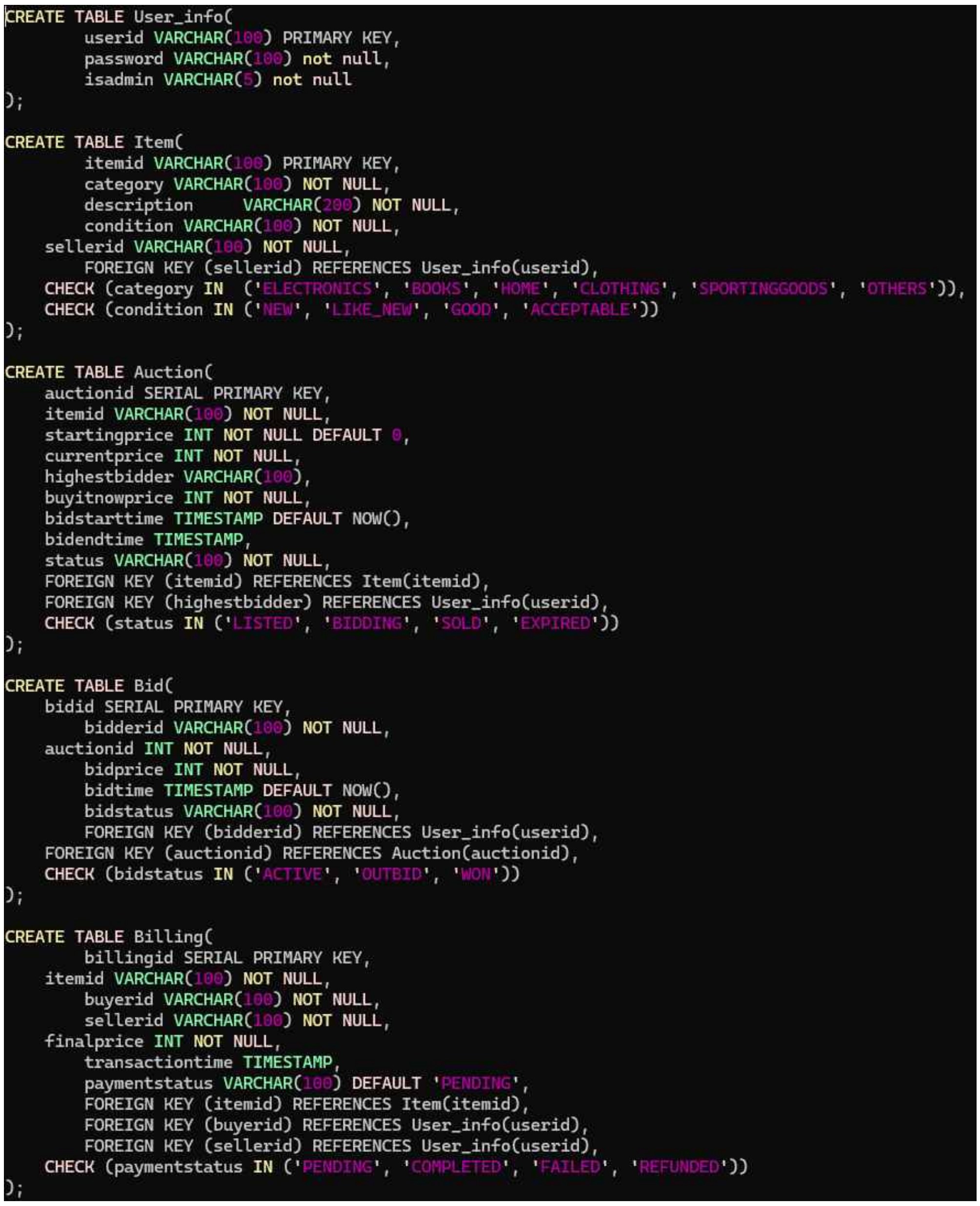
5. Bid Closing Time Check Scenarios (as required in the PDF) – 1st case : When ‘Buy Item’ is selected, demonstration of how items whose bidding period has ended are not displayed, based on current time comparison.

6. Handling Bidding and Payments (as required in the PDF) – When the bidding period ends (i.e., the bid closing time has passed), demonstration of how the highest bid is determined as the winning bid, and the bidder who placed it is determined as the buyer, who then pays the final price. (Both scenarios that are only one bidder, and multiple bidders are demonstrated)

7. Admin menu features – Operation and output for each of the following: Sold Items per Category, Account Balance for Seller, Seller Ranking, and Buyer Ranking

(Because of the size of the picture, I will explain it on the next page.)

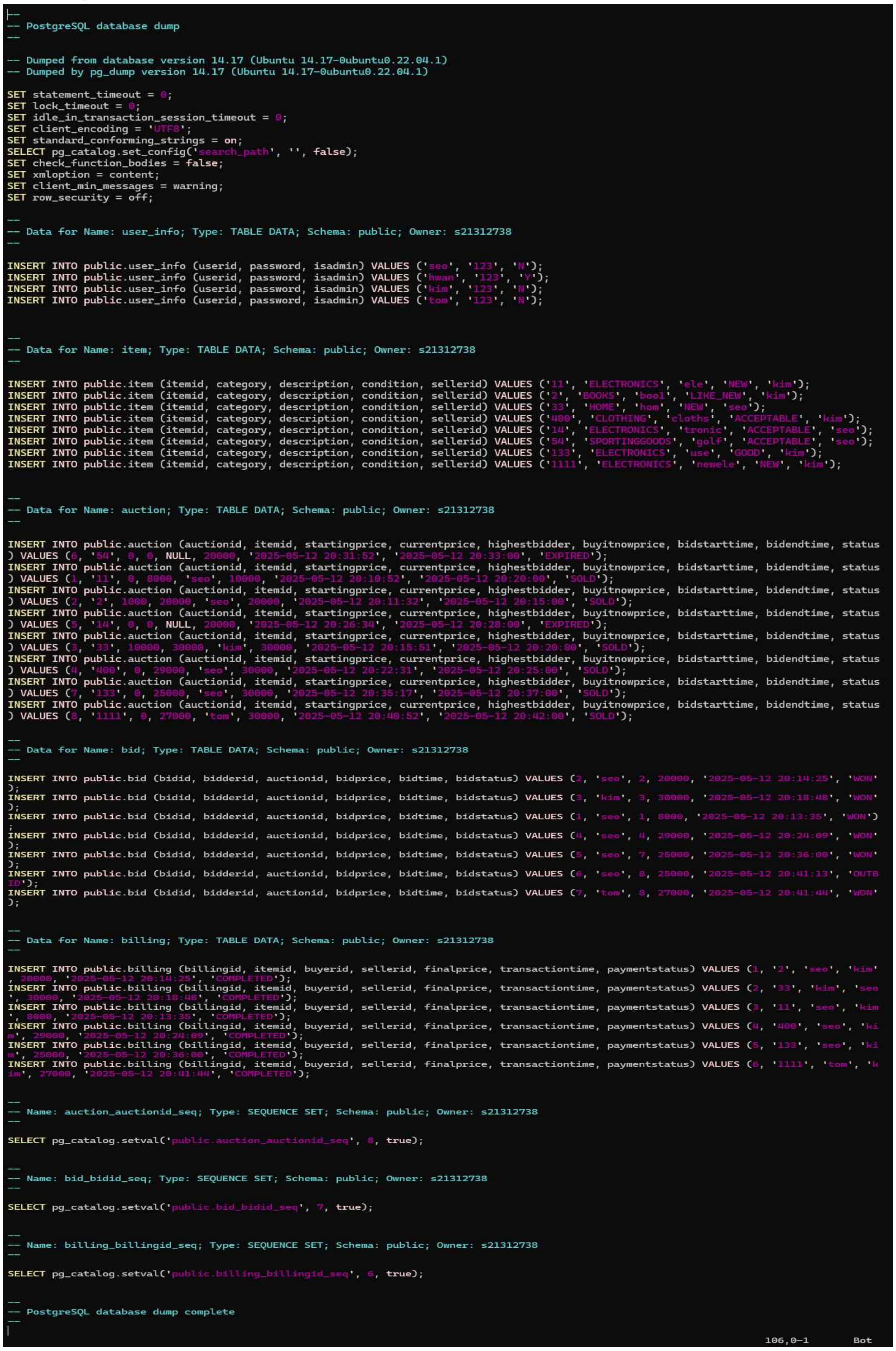
3) ddl.sql (file defining the schema of the table) and data.sql (file after applying all the processes shown in the demonstration video.)
The time inserted in data.sql is the timestamp from the time of the demonstration.
<ddl.sql>



```sql
CREATE TABLE User_info(
        userid VARCHAR(100) PRIMARY KEY,
        password VARCHAR(100) not null,
        isadmin VARCHAR(5) not null
);

CREATE TABLE Item(
        itemid VARCHAR(100) PRIMARY KEY,
        category VARCHAR(100) NOT NULL,
        description     VARCHAR(200) NOT NULL,
        condition VARCHAR(100) NOT NULL,
    sellerid VARCHAR(100) NOT NULL,
        FOREIGN KEY (sellerid) REFERENCES User_info(userid),
    CHECK (category IN  ('ELECTRONICS', 'BOOKS', 'HOME', 'CLOTHING', 'SPORTINGGOODS', 'OTHERS')),
    CHECK (condition IN ('NEW', 'LIKE_NEW', 'GOOD', 'ACCEPTABLE'))
);

CREATE TABLE Auction(
    auctionid SERIAL PRIMARY KEY,
    itemid VARCHAR(100) NOT NULL,
    startingprice INT NOT NULL DEFAULT 0,
    currentprice INT NOT NULL,
    highestbidder VARCHAR(100),
    buyitnowprice INT NOT NULL,
    bidstarttime TIMESTAMP DEFAULT NOW(),
    bidendtime TIMESTAMP,
    status VARCHAR(100) NOT NULL,
    FOREIGN KEY (itemid) REFERENCES Item(itemid),
    FOREIGN KEY (highestbidder) REFERENCES User_info(userid),
    CHECK (status IN ('LISTED', 'BIDDING', 'SOLD', 'EXPIRED'))
);

CREATE TABLE Bid(
    bidid SERIAL PRIMARY KEY,
        bidderid VARCHAR(100) NOT NULL,
    auctionid INT NOT NULL,
        bidprice INT NOT NULL,
        bidtime TIMESTAMP DEFAULT NOW(),
        bidstatus VARCHAR(100) NOT NULL,
        FOREIGN KEY (bidderid) REFERENCES User_info(userid),
    FOREIGN KEY (auctionid) REFERENCES Auction(auctionid),
    CHECK (bidstatus IN ('ACTIVE', 'OUTBID', 'WON'))
);

CREATE TABLE Billing(
        billingid SERIAL PRIMARY KEY,
    itemid VARCHAR(100) NOT NULL,
        buyerid VARCHAR(100) NOT NULL,
        sellerid VARCHAR(100) NOT NULL,
    finalprice INT NOT NULL,
        transactiontime TIMESTAMP,
        paymentstatus VARCHAR(100) DEFAULT 'PENDING',
        FOREIGN KEY (itemid) REFERENCES Item(itemid),
        FOREIGN KEY (buyerid) REFERENCES User_info(userid),
        FOREIGN KEY (sellerid) REFERENCES User_info(userid),
    CHECK (paymentstatus IN ('PENDING', 'COMPLETED', 'FAILED', 'REFUNDED'))
);
```

(Because of the size of the picture, I will explain it on the next page.)

<data.sql>



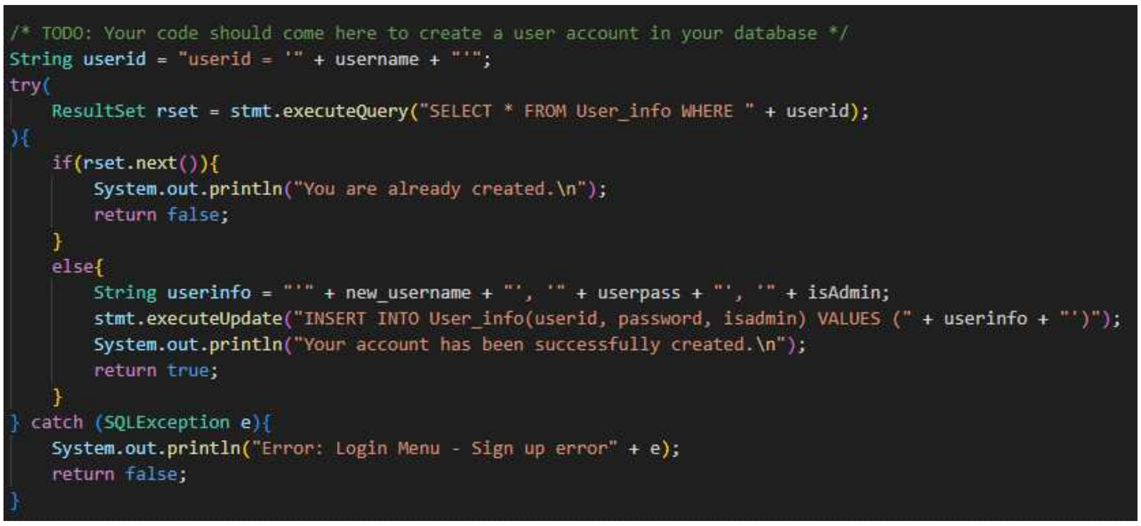
```
--
-- PostgreSQL database dump
--

-- Dumped from database version 14.17 (Ubuntu 14.17-0ubuntu0.22.04.1)
-- Dumped by pg_dump version 14.17 (Ubuntu 14.17-0ubuntu0.22.04.1)

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

--
-- Data for Name: user_info; Type: TABLE DATA; Schema: public; Owner: s21312738
--

INSERT INTO public.user_info (userid, password, isadmin) VALUES ('seo', '123', 'N');
INSERT INTO public.user_info (userid, password, isadmin) VALUES ('hwan', '123', 'Y');
INSERT INTO public.user_info (userid, password, isadmin) VALUES ('kim', '123', 'N');
INSERT INTO public.user_info (userid, password, isadmin) VALUES ('tom', '123', 'N');



--
-- Data for Name: item; Type: TABLE DATA; Schema: public; Owner: s21312738
--

INSERT INTO public.item (itemid, category, description, condition, sellerid) VALUES ('11', 'ELECTRONICS', 'ele', 'NEW', 'kim');
INSERT INTO public.item (itemid, category, description, condition, sellerid) VALUES ('2', 'BOOKS', 'bool', 'LIKE_NEW', 'kim');
INSERT INTO public.item (itemid, category, description, condition, sellerid) VALUES ('33', 'HOME', 'hom', 'NEW', 'seo');
INSERT INTO public.item (itemid, category, description, condition, sellerid) VALUES ('400', 'CLOTHING', 'cloths', 'ACCEPTABLE', 'kim');
INSERT INTO public.item (itemid, category, description, condition, sellerid) VALUES ('14', 'ELECTRONICS', 'tronic', 'ACCEPTABLE', 'seo');
INSERT INTO public.item (itemid, category, description, condition, sellerid) VALUES ('54', 'SPORTINGGOODS', 'golf', 'ACCEPTABLE', 'seo');
INSERT INTO public.item (itemid, category, description, condition, sellerid) VALUES ('133', 'ELECTRONICS', 'use', 'GOOD', 'kim');
INSERT INTO public.item (itemid, category, description, condition, sellerid) VALUES ('1111', 'ELECTRONICS', 'newele', 'NEW', 'kim');


--
-- Data for Name: auction; Type: TABLE DATA; Schema: public; Owner: s21312738
--

INSERT INTO public.auction (auctionid, itemid, startingprice, currentprice, highestbidder, buyitnowprice, bidstarttime, bidendtime, status
) VALUES (6, '54', 0, 0, NULL, 20000, '2025-05-12 20:31:52', '2025-05-12 20:33:00', 'EXPIRED');
INSERT INTO public.auction (auctionid, itemid, startingprice, currentprice, highestbidder, buyitnowprice, bidstarttime, bidendtime, status
) VALUES (1, '11', 0, 8000, 'seo', 10000, '2025-05-12 20:10:52', '2025-05-12 20:20:00', 'SOLD');
INSERT INTO public.auction (auctionid, itemid, startingprice, currentprice, highestbidder, buyitnowprice, bidstarttime, bidendtime, status
) VALUES (2, '2', 1000, 20000, 'seo', 20000, '2025-05-12 20:11:32', '2025-05-12 20:15:00', 'SOLD');
INSERT INTO public.auction (auctionid, itemid, startingprice, currentprice, highestbidder, buyitnowprice, bidstarttime, bidendtime, status
) VALUES (5, '14', 0, 0, NULL, 20000, '2025-05-12 20:26:34', '2025-05-12 20:28:00', 'EXPIRED');
INSERT INTO public.auction (auctionid, itemid, startingprice, currentprice, highestbidder, buyitnowprice, bidstarttime, bidendtime, status
) VALUES (3, '33', 10000, 30000, 'kim', 30000, '2025-05-12 20:15:51', '2025-05-12 20:20:00', 'SOLD');
INSERT INTO public.auction (auctionid, itemid, startingprice, currentprice, highestbidder, buyitnowprice, bidstarttime, bidendtime, status
) VALUES (4, '400', 0, 29000, 'seo', 30000, '2025-05-12 20:22:31', '2025-05-12 20:25:00', 'SOLD');
INSERT INTO public.auction (auctionid, itemid, startingprice, currentprice, highestbidder, buyitnowprice, bidstarttime, bidendtime, status
) VALUES (7, '133', 0, 25000, 'seo', 30000, '2025-05-12 20:35:17', '2025-05-12 20:37:00', 'SOLD');
INSERT INTO public.auction (auctionid, itemid, startingprice, currentprice, highestbidder, buyitnowprice, bidstarttime, bidendtime, status
) VALUES (8, '1111', 0, 27000, 'tom', 30000, '2025-05-12 20:40:52', '2025-05-12 20:42:00', 'SOLD');

--
-- Data for Name: bid; Type: TABLE DATA; Schema: public; Owner: s21312738
--

INSERT INTO public.bid (bidid, bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES (2, 'seo', 2, 20000, '2025-05-12 20:14:25', 'WON'
);
INSERT INTO public.bid (bidid, bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES (3, 'kim', 3, 30000, '2025-05-12 20:18:48', 'WON'
);
INSERT INTO public.bid (bidid, bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES (1, 'seo', 1, 8000, '2025-05-12 20:13:35', 'WON')
;
INSERT INTO public.bid (bidid, bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES (4, 'seo', 4, 29000, '2025-05-12 20:24:09', 'WON'
);
INSERT INTO public.bid (bidid, bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES (5, 'seo', 7, 25000, '2025-05-12 20:36:00', 'WON'
);
INSERT INTO public.bid (bidid, bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES (6, 'seo', 8, 25000, '2025-05-12 20:41:13', 'OUTB
ID');
INSERT INTO public.bid (bidid, bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES (7, 'tom', 8, 27000, '2025-05-12 20:41:44', 'WON'
);


--
-- Data for Name: billing; Type: TABLE DATA; Schema: public; Owner: s21312738
--

INSERT INTO public.billing (billingid, itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) VALUES (1, '2', 'seo', 'kim'
, 20000, '2025-05-12 20:14:25', 'COMPLETED');
INSERT INTO public.billing (billingid, itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) VALUES (2, '33', 'kim', 'seo
', 30000, '2025-05-12 20:18:48', 'COMPLETED');
INSERT INTO public.billing (billingid, itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) VALUES (3, '11', 'seo', 'kim
', 8000, '2025-05-12 20:13:35', 'COMPLETED');
INSERT INTO public.billing (billingid, itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) VALUES (4, '400', 'seo', 'ki
m', 29000, '2025-05-12 20:24:09', 'COMPLETED');
INSERT INTO public.billing (billingid, itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) VALUES (5, '133', 'seo', 'ki
m', 25000, '2025-05-12 20:36:00', 'COMPLETED');
INSERT INTO public.billing (billingid, itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) VALUES (6, '1111', 'tom', 'k
im', 27000, '2025-05-12 20:41:44', 'COMPLETED');


--
-- Name: auction_auctionid_seq; Type: SEQUENCE SET; Schema: public; Owner: s21312738
--

SELECT pg_catalog.setval('public.auction_auctionid_seq', 8, true);


--
-- Name: bid_bidid_seq; Type: SEQUENCE SET; Schema: public; Owner: s21312738
--

SELECT pg_catalog.setval('public.bid_bidid_seq', 7, true);


--
-- Name: billing_billingid_seq; Type: SEQUENCE SET; Schema: public; Owner: s21312738
--

SELECT pg_catalog.setval('public.billing_billingid_seq', 6, true);


--
-- PostgreSQL database dump complete
--

                                                                                                              106,0-1       Bot
```

4) Auction.java Code

For visibility, I will open the code Aduction.java in VScode and explain the implementation of the code focusing on the query.

1. SignupMenu()

The SignupMenu allows you to register a new user or administrator by entering a new userid, password, and specifying whether to grant admin privileges.



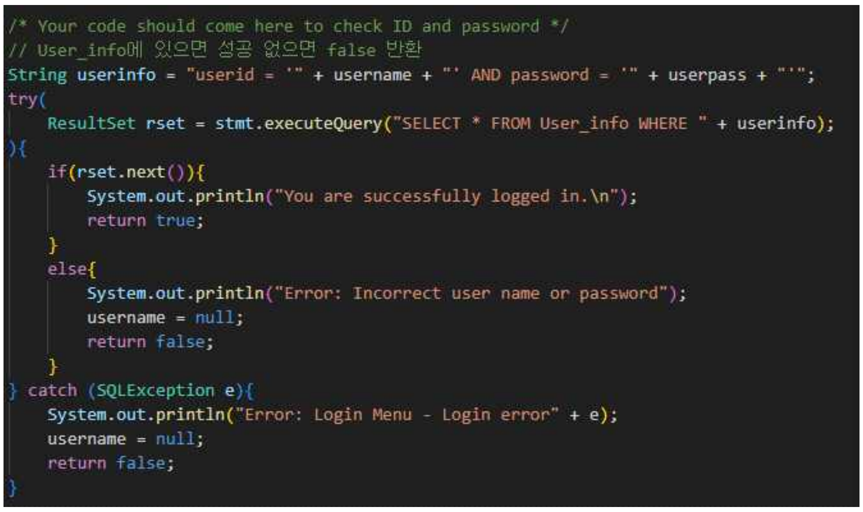
```java
/* TODO: Your code should come here to create a user account in your database */
String userid = "userid = '" + username + "'";
try(
    ResultSet rset = stmt.executeQuery("SELECT * FROM User_info WHERE " + userid);
){
    if(rset.next()){
        System.out.println("You are already created.\n");
        return false;
    }
    else{
        String userinfo = "'" + new_username + "', '" + userpass + "', '" + isAdmin;
        stmt.executeUpdate("INSERT INTO User_info(userid, password, isadmin) VALUES (" + userinfo + "')");
        System.out.println("Your account has been successfully created.\n");
        return true;
    }
} catch (SQLException e){
    System.out.println("Error: Login Menu - Sign up error" + e);
    return false;
}
```

However, a SELECT query is used to check if the given userid already exists.

If the ID already exists, the system outputs "You are already created." and returns False, redirecting the user back to the menu.

2. LoginMenu()

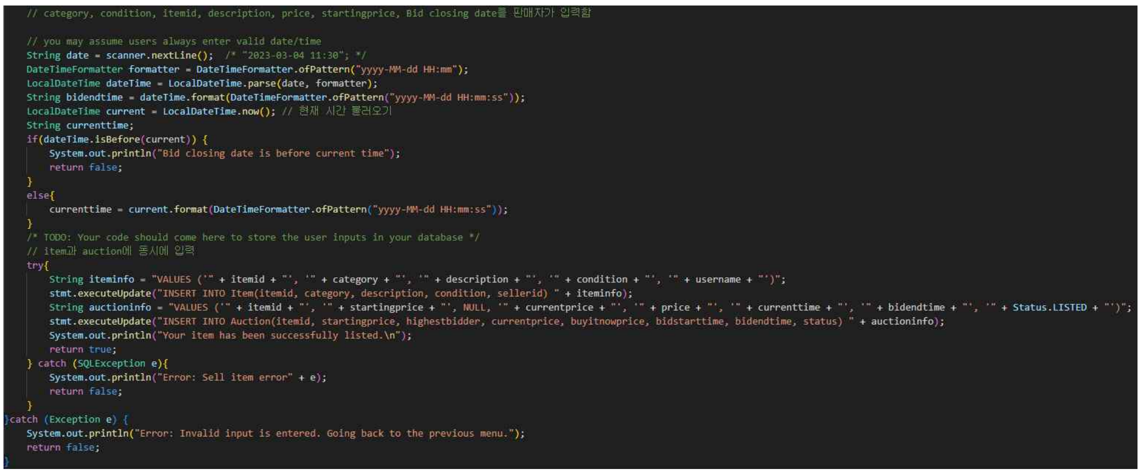
The login menu takes a userid and password as input. If the given userid exists in the user_info table, the login is successful. If not, the system returns false and redirects the user back to the menu.



```java
/* Your code should come here to check ID and password */
// User_info에 있으면 성공 없으면 false 반환
String userinfo = "userid = '" + username + "' AND password = '" + userpass + "'";
try(
    ResultSet rset = stmt.executeQuery("SELECT * FROM User_info WHERE " + userinfo);
){
    if(rset.next()){
        System.out.println("You are successfully logged in.\n");
        return true;
    }
    else{
        System.out.println("Error: Incorrect user name or password");
        username = null;
        return false;
    }
} catch (SQLException e){
    System.out.println("Error: Login Menu - Login error" + e);
    username = null;
    return false;
}
```

3. SellMenu()

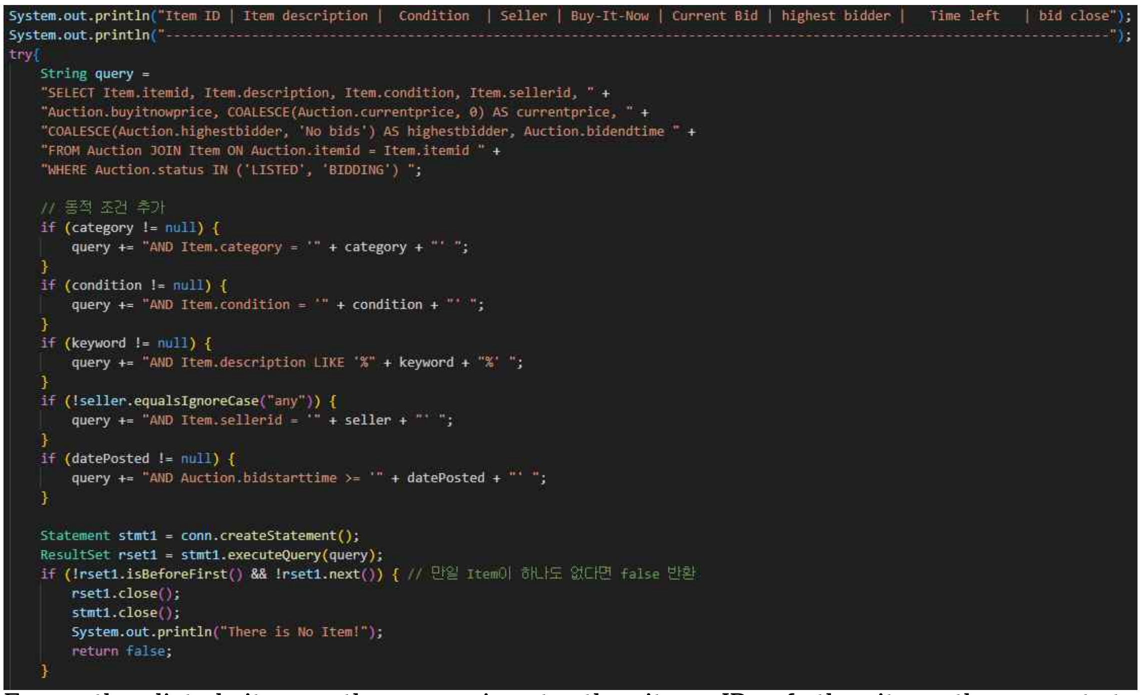
The user registers an item for auction by entering the category, condition, itemid, description, buy-it-now price, starting price, and bid closing date/time. The bid closing time cannot be set to a time earlier than the current time. Through this process, data is inserted into both the Item table and Auction table.



```java
    // category, condition, itemid, description, price, startingprice, Bid closing date를 판매자가 입력함

    // you may assume users always enter valid date/time
    String date = scanner.nextLine();  /* "2023-03-04 11:30"; */
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
    LocalDateTime dateTime = LocalDateTime.parse(date, formatter);
    String bidendtime = dateTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
    LocalDateTime current = LocalDateTime.now(); // 현재 시간 불러오기
    String currenttime;
    if(dateTime.isBefore(current)) {
        System.out.println("Bid closing date is before current time");
        return false;
    }
    else{
        currenttime = current.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
    }
    /* TODO: Your code should come here to store the user inputs in your database */
    // item과 auction에 동시에 입력
    try{
        String iteminfo = "VALUES ('" + itemid + "', '" + category + "', '" + description + "', '" + condition + "', '" + username + "')";
        stmt.executeUpdate("INSERT INTO Item(itemid, category, description, condition, sellerid) " + iteminfo);
        String auctioninfo = "VALUES ('" + itemid + "', '" + startingprice + "', NULL, '" + currentprice + "', '" + price + "', '" + currenttime + "', '" + bidendtime + "', '" + Status.LISTED + "')";
        stmt.executeUpdate("INSERT INTO Auction(itemid, startingprice, highestbidder, currentprice, buyitnowprice, bidstarttime, bidendtime, status) " + auctioninfo);
        System.out.println("Your item has been successfully listed.\n");
        return true;
    } catch (SQLException e){
        System.out.println("Error: Sell item error" + e);
        return false;
    }
}catch (Exception e) {
    System.out.println("Error: Invalid input is entered. Going back to the previous menu.");
    return false;
}
```

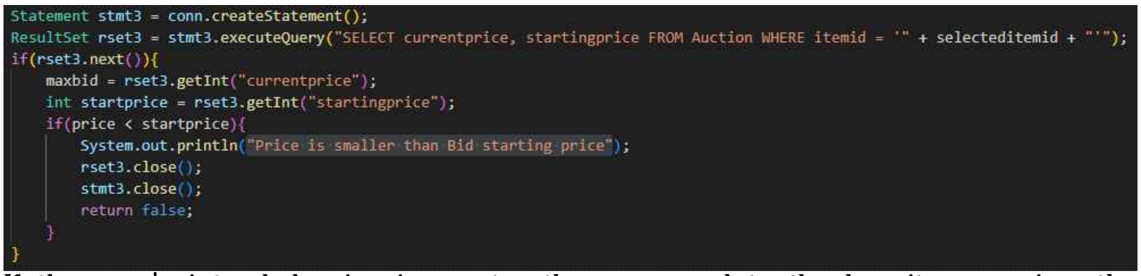
4. BuyItem()

The user searches for items they wish to purchase by entering the category, condition, description, seller ID (enter 'any' if the buyer wants to see items from any seller), and datePosted. A SELECT query is used to list and display items that match the given criteria. If no items match the conditions, the system displays "There is No Item!", returns false, and redirects the user back to the menu.



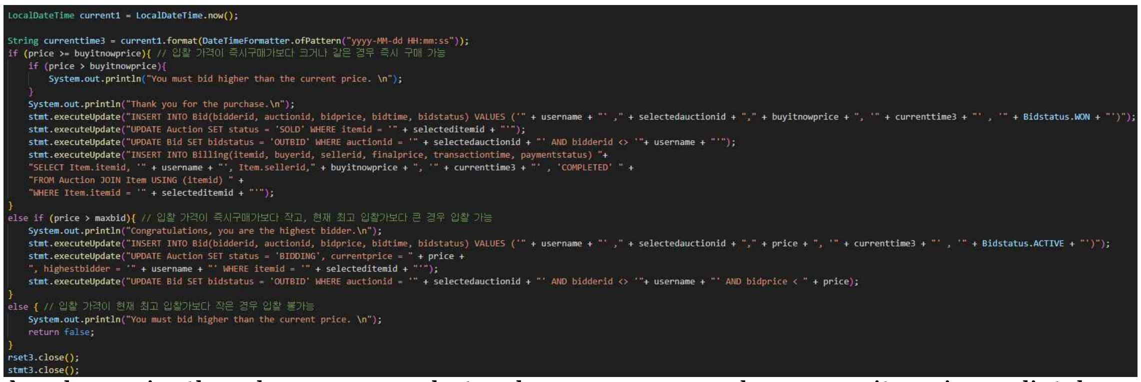
```java
System.out.println("Item ID | Item description |  Condition  | Seller | Buy-It-Now | Current Bid | highest bidder |   Time left   | bid close");
System.out.println("-------------------------------------------------------------------------------------------------------------------");
try{
    String query =
    "SELECT Item.itemid, Item.description, Item.condition, Item.sellerid, " +
    "Auction.buyitnowprice, COALESCE(Auction.currentprice, 0) AS currentprice, " +
    "COALESCE(Auction.highestbidder, 'No bids') AS highestbidder, Auction.bidendtime " +
    "FROM Auction JOIN Item ON Auction.itemid = Item.itemid " +
    "WHERE Auction.status IN ('LISTED', 'BIDDING') ";

    // 동적 조건 추가
    if (category != null) {
        query += "AND Item.category = '" + category + "' ";
    }
    if (condition != null) {
        query += "AND Item.condition = '" + condition + "' ";
    }
    if (keyword != null) {
        query += "AND Item.description LIKE '%" + keyword + "%' ";
    }
    if (!seller.equalsIgnoreCase("any")) {
        query += "AND Item.sellerid = '" + seller + "' ";
    }
    if (datePosted != null) {
        query += "AND Auction.bidstarttime >= '" + datePosted + "' ";
    }

    Statement stmt1 = conn.createStatement();
    ResultSet rset1 = stmt1.executeQuery(query);
    if (!rset1.isBeforeFirst() && !rset1.next()) { // 만일 Item이 하나도 없다면 false 반환
        rset1.close();
        stmt1.close();
        System.out.println("There is No Item!");
        return false;
    }
```

From the listed items, the user inputs the item ID of the item they want to purchase and enters their intended purchase price. If the intended price is lower than the item's starting price, the system displays "Price is smaller than Bid starting price", returns false, and goes back to the menu.



```java
Statement stmt3 = conn.createStatement();
ResultSet rset3 = stmt3.executeQuery("SELECT currentprice, startingprice FROM Auction WHERE itemid = '" + selecteditemid + "'");
if(rset3.next()){
    maxbid = rset3.getInt("currentprice");
    int startprice = rset3.getInt("startingprice");
    if(price < startprice){
        System.out.println("Price is smaller than Bid starting price");
        rset3.close();
        stmt3.close();
        return false;
    }
}
```

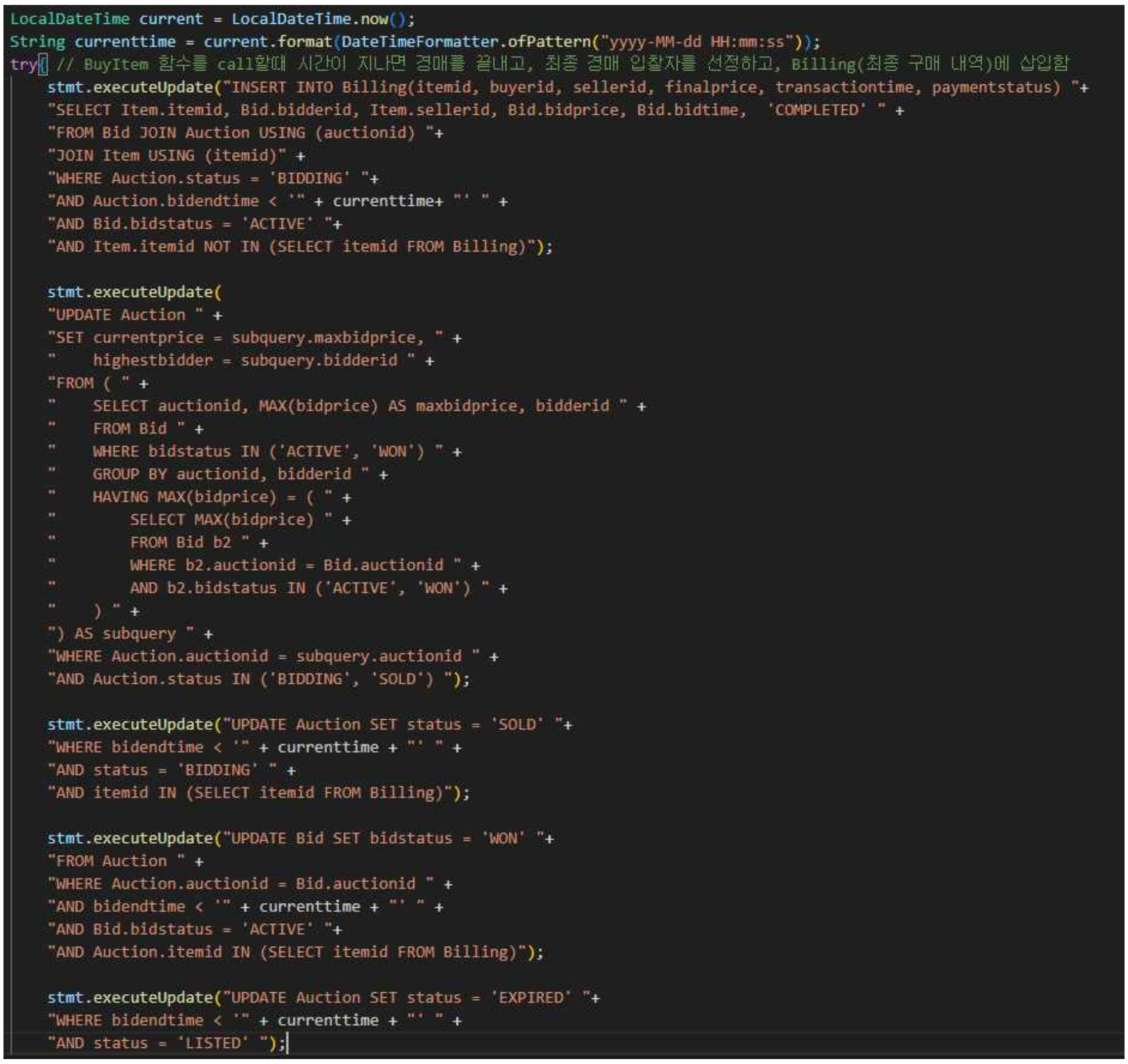
If the user's intended price is greater than or equal to the buy-it-now price, the item is purchased immediately. (If the price is higher, the system still displays "You must bid higher than the current price.", but the item is bought at the buy-it-now price.) If the intended price is less than the buy-it-now price but greater than the current highest bid, the bid is accepted at the entered price. However, if the intended price is less than or equal to the current highest bid, the bid is rejected, and the system displays "You must bid higher than the current price.", returns false, and goes back to the menu.



```java
LocalDateTime current1 = LocalDateTime.now();

String currenttime3 = current1.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
if (price >= buyitnowprice){ // 입찰 가격이 즉시구매가보다 크거나 같은 경우 즉시 구매 가능
    if (price > buyitnowprice){
        System.out.println("You must bid higher than the current price. \n");
    }
    System.out.println("Thank you for the purchase.\n");
    stmt.executeUpdate("INSERT INTO Bid(bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES ('" + username + "' ," + selectedauctionid + "," + buyitnowprice + ", '" + currenttime3 + "' , '" + Bidstatus.WON + "')");
    stmt.executeUpdate("UPDATE Auction SET status = 'SOLD' WHERE itemid = '" + selecteditemid + "'");
    stmt.executeUpdate("UPDATE Bid SET bidstatus = 'OUTBID' WHERE auctionid = '" + selectedauctionid + "' AND bidderid <> '"+ username + "'");
    stmt.executeUpdate("INSERT INTO Billing(itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) "+
    "SELECT Item.itemid, '" + username + "', Item.sellerid," + buyitnowprice + ", '" + currenttime3 + "' , 'COMPLETED' " +
    "FROM Auction JOIN Item USING (itemid) " +
    "WHERE Item.itemid = '" + selecteditemid + "'");
}
else if (price > maxbid){ // 입찰 가격이 즉시구매가보다 작고, 현재 최고 입찰가보다 큰 경우 입찰 가능
    System.out.println("Congratulations, you are the highest bidder.\n");
    stmt.executeUpdate("INSERT INTO Bid(bidderid, auctionid, bidprice, bidtime, bidstatus) VALUES ('" + username + "' ," + selectedauctionid + "," + price + ", '" + currenttime3 + "' , '" + Bidstatus.ACTIVE + "')");
    stmt.executeUpdate("UPDATE Auction SET status = 'BIDDING', currentprice = " + price +
    ", highestbidder = '" + username + "' WHERE itemid = '" + selecteditemid + "'");
    stmt.executeUpdate("UPDATE Bid SET bidstatus = 'OUTBID' WHERE auctionid = '" + selectedauctionid + "' AND bidderid <> '"+ username + "' AND bidprice < " + price);
}
else { // 입찰 가격이 현재 최고 입찰가보다 작은 경우 입찰 불가능
    System.out.println("You must bid higher than the current price. \n");
    return false;
}
rset3.close();
stmt3.close();
```

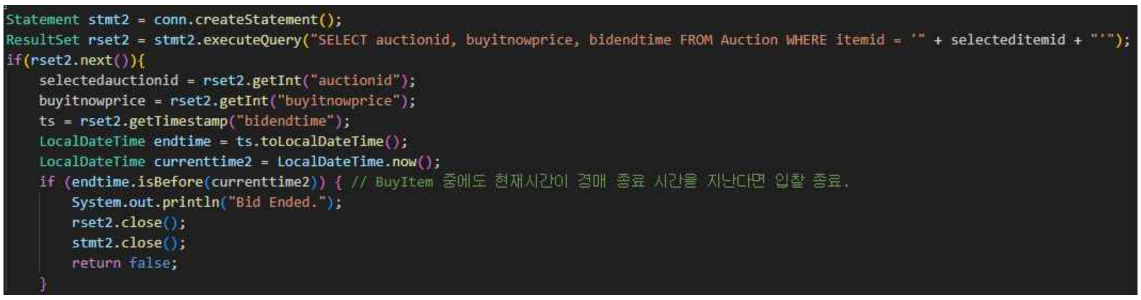
As shown in the above screenshot, when a user purchases an item immediately, a record is inserted into the Bid table (where ‘WON’ indicates that the user has won the auction), and the item's Auction status is updated to ‘SOLD’. Additionally, the statuses of other bidders who had placed bids on the item are updated to ‘OUTBID’. Since the purchase has been completed, a transaction record is also inserted into the Billing table.Similarly, when a user places a bid, a record is inserted into the Bid table (where ‘ACTIVE’ indicates that the user is currently the highest bidder), and the item's Auction status is updated to BIDDING. The statuses of other bidders who had placed bids on the same item are updated to ‘OUTBID’.

(Because of the size of the picture, I will explain it on the next page.)



```java
LocalDateTime current = LocalDateTime.now();
String currenttime = current.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
try{ // BuyItem 함수를 call할때 시간이 지나면 경매를 끝내고, 최종 경매 입찰자를 선정하고, Billing(최종 구매 내역)에 삽입함
    stmt.executeUpdate("INSERT INTO Billing(itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) "+
    "SELECT Item.itemid, Bid.bidderid, Item.sellerid, Bid.bidprice, Bid.bidtime,  'COMPLETED' " +
    "FROM Bid JOIN Auction USING (auctionid) "+
    "JOIN Item USING (itemid)" +
    "WHERE Auction.status = 'BIDDING' "+
    "AND Auction.bidendtime < '" + currenttime+ "' " +
    "AND Bid.bidstatus = 'ACTIVE' "+
    "AND Item.itemid NOT IN (SELECT itemid FROM Billing)");

    stmt.executeUpdate(
    "UPDATE Auction " +
    "SET currentprice = subquery.maxbidprice, " +
    "    highestbidder = subquery.bidderid " +
    "FROM ( " +
    "    SELECT auctionid, MAX(bidprice) AS maxbidprice, bidderid " +
    "    FROM Bid " +
    "    WHERE bidstatus IN ('ACTIVE', 'WON') " +
    "    GROUP BY auctionid, bidderid " +
    "    HAVING MAX(bidprice) = ( " +
    "        SELECT MAX(bidprice) " +
    "        FROM Bid b2 " +
    "        WHERE b2.auctionid = Bid.auctionid " +
    "        AND b2.bidstatus IN ('ACTIVE', 'WON') " +
    "    ) " +
    ") AS subquery " +
    "WHERE Auction.auctionid = subquery.auctionid " +
    "AND Auction.status IN ('BIDDING', 'SOLD') ");

    stmt.executeUpdate("UPDATE Auction SET status = 'SOLD' "+
    "WHERE bidendtime < '" + currenttime + "' " +
    "AND status = 'BIDDING' " +
    "AND itemid IN (SELECT itemid FROM Billing)");

    stmt.executeUpdate("UPDATE Bid SET bidstatus = 'WON' "+
    "FROM Auction " +
    "WHERE Auction.auctionid = Bid.auctionid " +
    "AND bidendtime < '" + currenttime + "' " +
    "AND Bid.bidstatus = 'ACTIVE' "+
    "AND Auction.itemid IN (SELECT itemid FROM Billing)");

    stmt.executeUpdate("UPDATE Auction SET status = 'EXPIRED' "+
    "WHERE bidendtime < '" + currenttime + "' " +
    "AND status = 'LISTED' ");
```

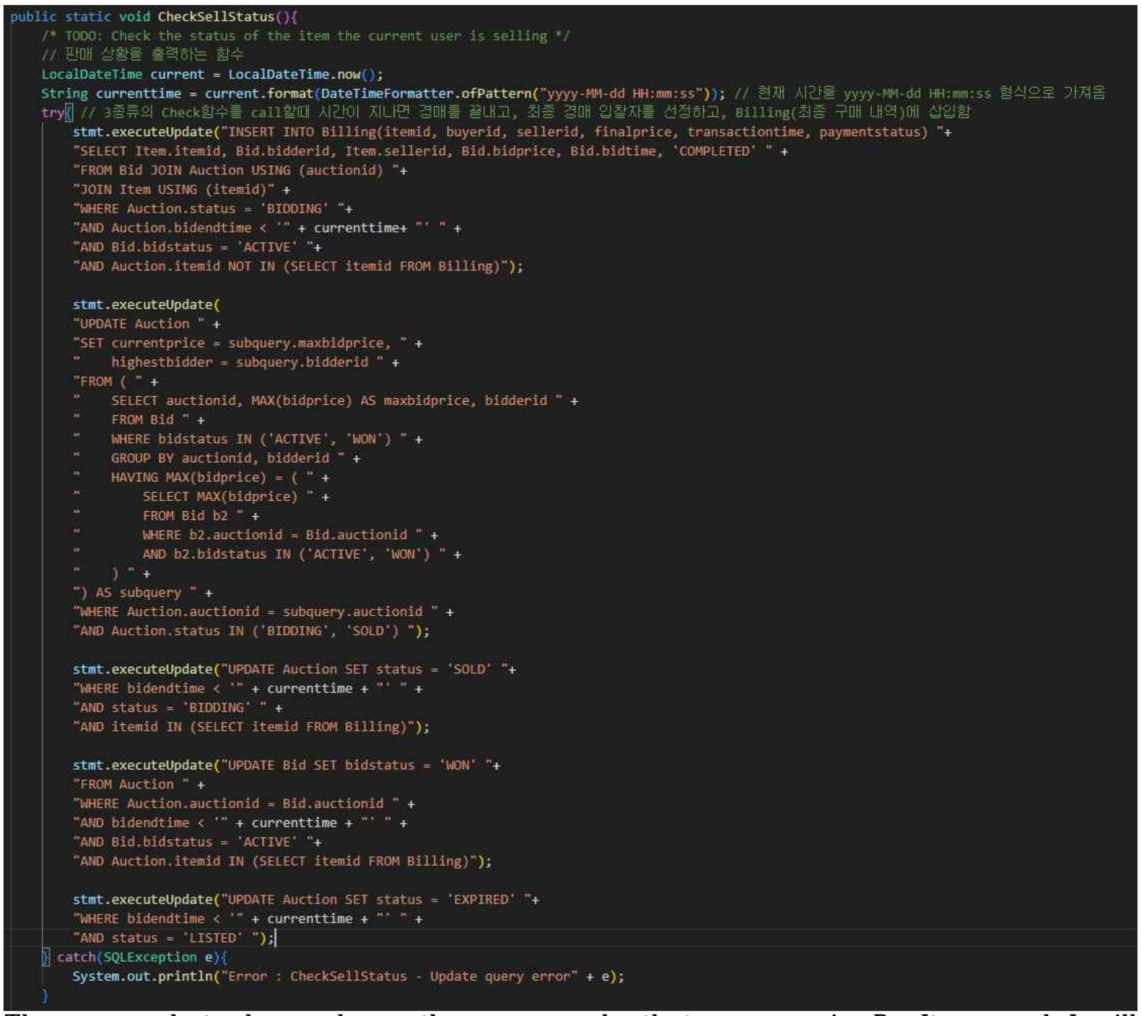
<u>(The code in that above screenshot runs before the Above Select query to list the items.)</u> There is an important point to note in the BuyItem process. If, during the time the user is searching for items to purchase, the current time passes the item's bid closing time(bidendtime), the bid must be ended, the highest bidder must be selected as the final winner, and the transaction details must be inserted into the Billing table at the highest bid price. In this case, even if an item originally matched the user's search condition, it's not displayed in the item list if the current time has already passed the bid closing time. <u>(Bid Closing Time Check – 1st)</u>



```java
Statement stmt2 = conn.createStatement();
ResultSet rset2 = stmt2.executeQuery("SELECT auctionid, buyitnowprice, bidendtime FROM Auction WHERE itemid = '" + selecteditemid + "'");
if(rset2.next()){
    selectedauctionid = rset2.getInt("auctionid");
    buyitnowprice = rset2.getInt("buyitnowprice");
    ts = rset2.getTimestamp("bidendtime");
    LocalDateTime endtime = ts.toLocalDateTime();
    LocalDateTime currenttime2 = LocalDateTime.now();
    if (endtime.isBefore(currenttime2)) { // BuyItem 중에도 현재시간이 경매 종료 시간을 지난다면 입찰 종료.
        System.out.println("Bid Ended.");
        rset2.close();
        stmt2.close();
        return false;
    }
```

Also, during the process of selecting an item from the listed results and entering the intended purchase price, if the current time has passed the bid closing time, the system should display "Bid Ended." (Bid Closing Time Check - 2nd)
Through this process, when the bidding period ends, the final purchase price is determined as the highest bid, even if there are multiple bidders for the item. Only the bidder who placed the highest bid is awarded the item at the final purchase price. (Handling Bidding and Payments)

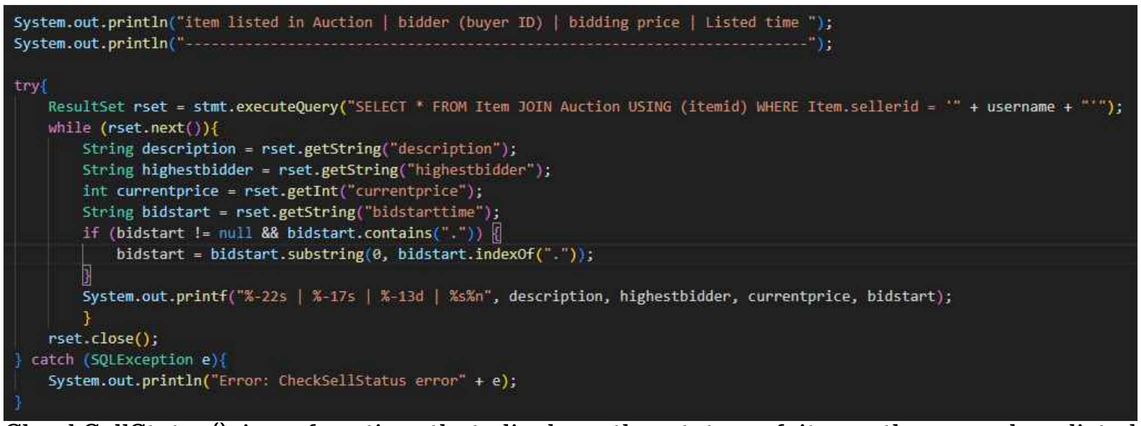
5. CheckSellStatus()



```java
public static void CheckSellStatus(){
    /* TODO: Check the status of the item the current user is selling */
    // 판매 상황을 출력하는 함수
    LocalDateTime current = LocalDateTime.now();
    String currenttime = current.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")); // 현재 시간을 yyyy-MM-dd HH:mm:ss 형식으로 가져옴
    try{ // 3종류의 Check함수를 call할때 시간이 지나면 경매를 끝내고, 최종 경매 입찰자를 선정하고, Billing(최종 구매 내역)에 삽입함
        stmt.executeUpdate("INSERT INTO Billing(itemid, buyerid, sellerid, finalprice, transactiontime, paymentstatus) "+
        "SELECT Item.itemid, Bid.bidderid, Item.sellerid, Bid.bidprice, Bid.bidtime, 'COMPLETED' " +
        "FROM Bid JOIN Auction USING (auctionid) "+
        "JOIN Item USING (itemid)" +
        "WHERE Auction.status = 'BIDDING' "+
        "AND Auction.bidendtime < '" + currenttime+ "' " +
        "AND Bid.bidstatus = 'ACTIVE' "+
        "AND Auction.itemid NOT IN (SELECT itemid FROM Billing)");

        stmt.executeUpdate(
        "UPDATE Auction " +
        "SET currentprice = subquery.maxbidprice, " +
        "    highestbidder = subquery.bidderid " +
        "FROM ( " +
        "    SELECT auctionid, MAX(bidprice) AS maxbidprice, bidderid " +
        "    FROM Bid " +
        "    WHERE bidstatus IN ('ACTIVE', 'WON') " +
        "    GROUP BY auctionid, bidderid " +
        "    HAVING MAX(bidprice) = ( " +
        "        SELECT MAX(bidprice) " +
        "        FROM Bid b2 " +
        "        WHERE b2.auctionid = Bid.auctionid " +
        "        AND b2.bidstatus IN ('ACTIVE', 'WON') " +
        "    ) " +
        ") AS subquery " +
        "WHERE Auction.auctionid = subquery.auctionid " +
        "AND Auction.status IN ('BIDDING', 'SOLD') ");

        stmt.executeUpdate("UPDATE Auction SET status = 'SOLD' "+
        "WHERE bidendtime < '" + currenttime + "' " +
        "AND status = 'BIDDING' " +
        "AND itemid IN (SELECT itemid FROM Billing)");

        stmt.executeUpdate("UPDATE Bid SET bidstatus = 'WON' "+
        "FROM Auction " +
        "WHERE Auction.auctionid = Bid.auctionid " +
        "AND bidendtime < '" + currenttime + "' " +
        "AND Bid.bidstatus = 'ACTIVE' "+
        "AND Auction.itemid IN (SELECT itemid FROM Billing)");

        stmt.executeUpdate("UPDATE Auction SET status = 'EXPIRED' "+
        "WHERE bidendtime < '" + currenttime + "' " +
        "AND status = 'LISTED' ");
    } catch(SQLException e){
        System.out.println("Error : CheckSellStatus - Update query error" + e);
    }
```

The screenshot above shows the same code that appears in BuyItem, and I will explain it in detail.
The first INSERT query inserts data into the Billing table by retrieving the bidder ID, bid price, and bid time from the Bid table for items in the Auction table where the current time has passed the bid closing time, the auction status is 'BIDDING', and the bidder's bid status is 'ACTIVE'.
The second UPDATE query updates information that has not yet been updated for the highest bid or completed purchase. It updates the relevant information for bids

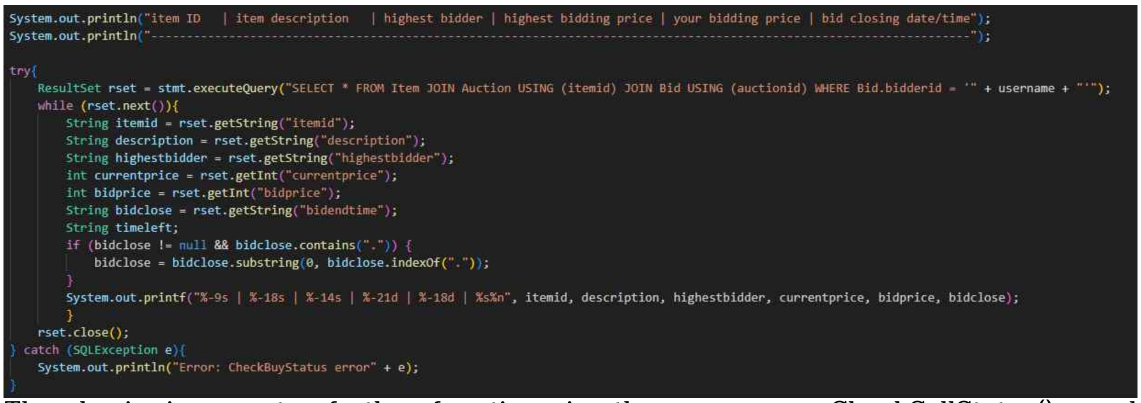
(or purchases) where the bid status is 'ACTIVE' or 'WON', for items whose status is 'BIDDING' or 'SOLD', based on the highest bid (or final purchase) price.
The third UPDATE query changes the status of the item to 'SOLD' if the current time has passed the bid closing time and the item is still marked as 'BIDDING'.
The fourth UPDATE query updates the bid status of the current highest bidder (with status 'ACTIVE') to 'WON', if the bid closing time has passed.
The fifth UPDATE query changes the status of items from 'LISTED' to 'EXPIRED' if the bid closing time has passed and no bids were placed.
Each time any of the three Check functions — CheckSellStatus(), CheckBuyStatus(), or CheckAccount() — is called, these five queries are executed. <u>(Bid Closing Time Check - 3rd)</u>



```java
System.out.println("item listed in Auction | bidder (buyer ID) | bidding price | Listed time ");
System.out.println("------------------------------------------------------------------------");

try{
    ResultSet rset = stmt.executeQuery("SELECT * FROM Item JOIN Auction USING (itemid) WHERE Item.sellerid = '" + username + "'");
    while (rset.next()){
        String description = rset.getString("description");
        String highestbidder = rset.getString("highestbidder");
        int currentprice = rset.getInt("currentprice");
        String bidstart = rset.getString("bidstarttime");
        if (bidstart != null && bidstart.contains(".")) {
            bidstart = bidstart.substring(0, bidstart.indexOf("."));
        }
        System.out.printf("%-22s | %-17s | %-13d | %s%n", description, highestbidder, currentprice, bidstart);
        }
    rset.close();
} catch (SQLException e){
    System.out.println("Error: CheckSellStatus error" + e);
}
```

CheckSellStatus() is a function that displays the status of items the user has listed for selling.

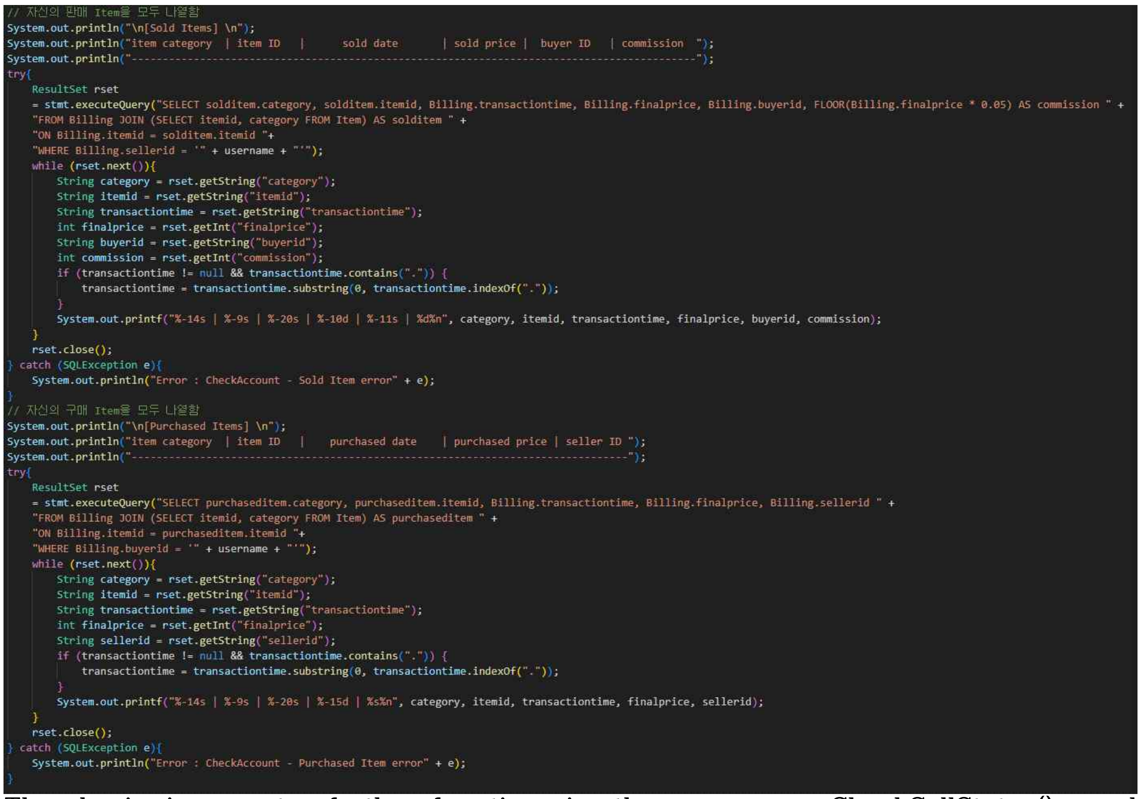
6. CheckBuyStatus()



```java
System.out.println("item ID   | item description   | highest bidder | highest bidding price | your bidding price | bid closing date/time");
System.out.println("-------------------------------------------------------------------------------------------------------------------");

try{
    ResultSet rset = stmt.executeQuery("SELECT * FROM Item JOIN Auction USING (itemid) JOIN Bid USING (auctionid) WHERE Bid.bidderid = '" + username + "'");
    while (rset.next()){
        String itemid = rset.getString("itemid");
        String description = rset.getString("description");
        String highestbidder = rset.getString("highestbidder");
        int currentprice = rset.getInt("currentprice");
        int bidprice = rset.getInt("bidprice");
        String bidclose = rset.getString("bidendtime");
        String timeleft;
        if (bidclose != null && bidclose.contains(".")) {
            bidclose = bidclose.substring(0, bidclose.indexOf("."));
        }
        System.out.printf("%-9s | %-18s | %-14s | %-21d | %-18d | %s%n", itemid, description, highestbidder, currentprice, bidprice, bidclose);
        }
    rset.close();
} catch (SQLException e){
    System.out.println("Error: CheckBuyStatus error" + e);
}
```

The beginning part of the function is the same as CheckSellStatus(), and CheckBuyStatus() is a function that displays the status of items the user has biddered.

## 7. CheckAccount()

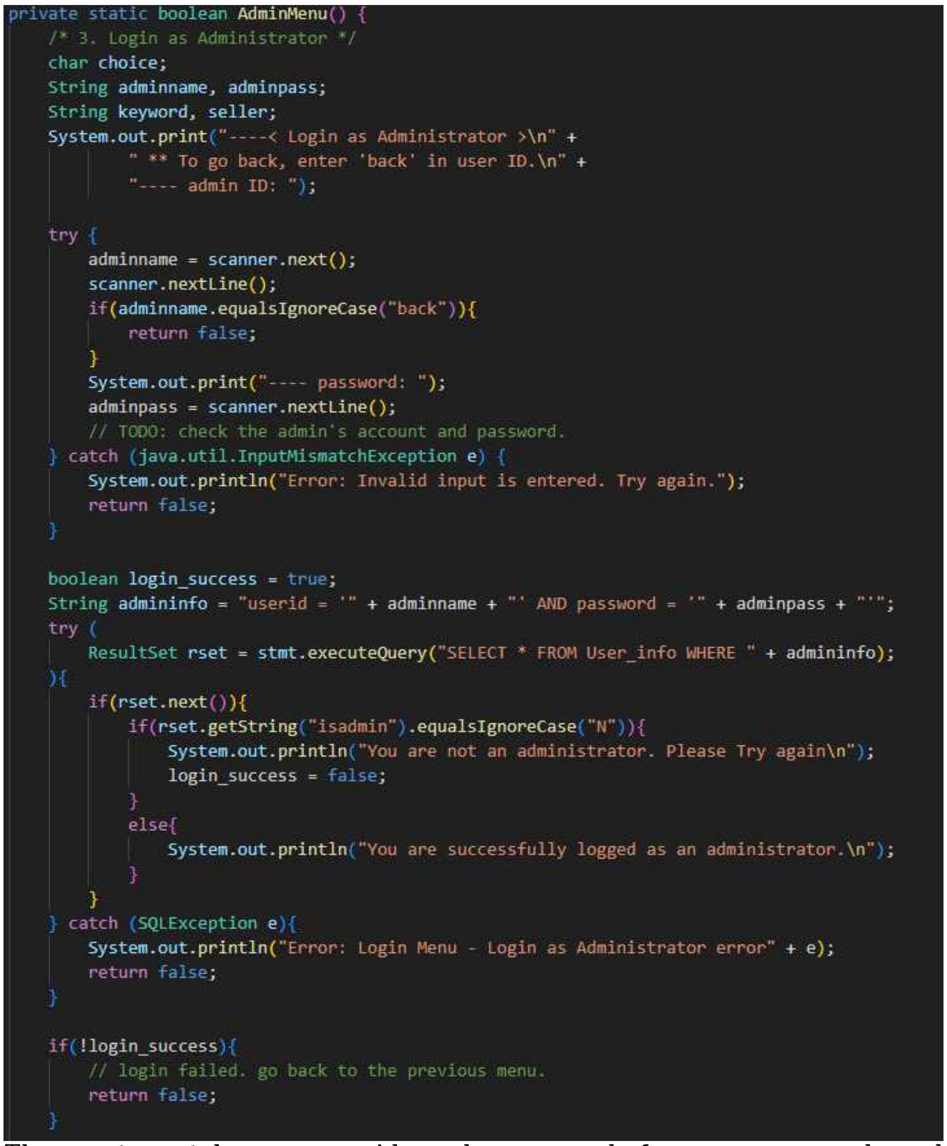


```java
// 자신의 판매 Item을 모두 나열함
System.out.println("\n[Sold Items] \n");
System.out.println("item category  | item ID   |      sold date       | sold price |  buyer ID   | commission  ");
System.out.println("------------------------------------------------------------------------------------------");
try{
    ResultSet rset
    = stmt.executeQuery("SELECT solditem.category, solditem.itemid, Billing.transactiontime, Billing.finalprice, Billing.buyerid, FLOOR(Billing.finalprice * 0.05) AS commission " +
    "FROM Billing JOIN (SELECT itemid, category FROM Item) AS solditem " +
    "ON Billing.itemid = solditem.itemid "+
    "WHERE Billing.sellerid = '" + username + "'");
    while (rset.next()){
        String category = rset.getString("category");
        String itemid = rset.getString("itemid");
        String transactiontime = rset.getString("transactiontime");
        int finalprice = rset.getInt("finalprice");
        String buyerid = rset.getString("buyerid");
        int commission = rset.getInt("commission");
        if (transactiontime != null && transactiontime.contains(".")) {
            transactiontime = transactiontime.substring(0, transactiontime.indexOf("."));
        }
        System.out.printf("%-14s | %-9s | %-20s | %-10d | %-11s | %d%n", category, itemid, transactiontime, finalprice, buyerid, commission);
    }
    rset.close();
} catch (SQLException e){
    System.out.println("Error : CheckAccount - Sold Item error" + e);
}
// 자신의 구매 Item을 모두 나열함
System.out.println("\n[Purchased Items] \n");
System.out.println("item category  | item ID   |    purchased date    | purchased price | seller ID ");
System.out.println("-------------------------------------------------------------------------------");
try{
    ResultSet rset
    = stmt.executeQuery("SELECT purchaseditem.category, purchaseditem.itemid, Billing.transactiontime, Billing.finalprice, Billing.sellerid " +
    "FROM Billing JOIN (SELECT itemid, category FROM Item) AS purchaseditem " +
    "ON Billing.itemid = purchaseditem.itemid "+
    "WHERE Billing.buyerid = '" + username + "'");
    while (rset.next()){
        String category = rset.getString("category");
        String itemid = rset.getString("itemid");
        String transactiontime = rset.getString("transactiontime");
        int finalprice = rset.getInt("finalprice");
        String sellerid = rset.getString("sellerid");
        if (transactiontime != null && transactiontime.contains(".")) {
            transactiontime = transactiontime.substring(0, transactiontime.indexOf("."));
        }
        System.out.printf("%-14s | %-9s | %-20s | %-15d | %s%n", category, itemid, transactiontime, finalprice, sellerid);
    }
    rset.close();
} catch (SQLException e){
    System.out.println("Error : CheckAccount - Purchased Item error" + e);
}
```

The beginning part of the function is the same as CheckSellStatus(), and CheckAccount() is a function that joins with the Billing table to display all items the user has sold and all items the user has purchased as recorded in the final transaction history.

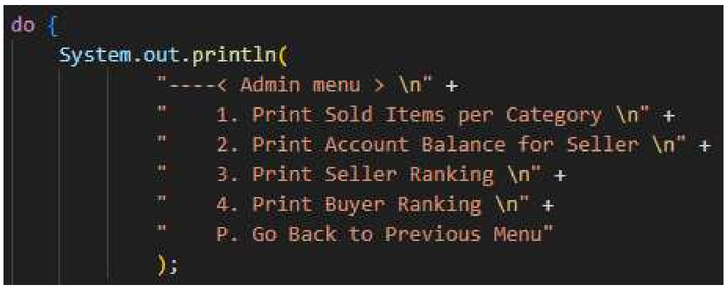
(Because of the size of the picture, I will explain it on the next page.)

8. AdminMenu()



```java
private static boolean AdminMenu() {
    /* 3. Login as Administrator */
    char choice;
    String adminname, adminpass;
    String keyword, seller;
    System.out.print("----< Login as Administrator >\n" +
            " ** To go back, enter 'back' in user ID.\n" +
            "---- admin ID: ");

    try {
        adminname = scanner.next();
        scanner.nextLine();
        if(adminname.equalsIgnoreCase("back")){
            return false;
        }
        System.out.print("---- password: ");
        adminpass = scanner.nextLine();
        // TODO: check the admin's account and password.
    } catch (java.util.InputMismatchException e) {
        System.out.println("Error: Invalid input is entered. Try again.");
        return false;
    }

    boolean login_success = true;
    String admininfo = "userid = '" + adminname + "' AND password = '" + adminpass + "'";
    try (
        ResultSet rset = stmt.executeQuery("SELECT * FROM User_info WHERE " + admininfo);
    ){
        if(rset.next()){
            if(rset.getString("isadmin").equalsIgnoreCase("N")){
                System.out.println("You are not an administrator. Please Try again\n");
                login_success = false;
            }
            else{
                System.out.println("You are successfully logged as an administrator.\n");
            }
        }
    } catch (SQLException e){
        System.out.println("Error: Login Menu - Login as Administrator error" + e);
        return false;
    }

    if(!login_success){
        // login failed. go back to the previous menu.
        return false;
    }
```

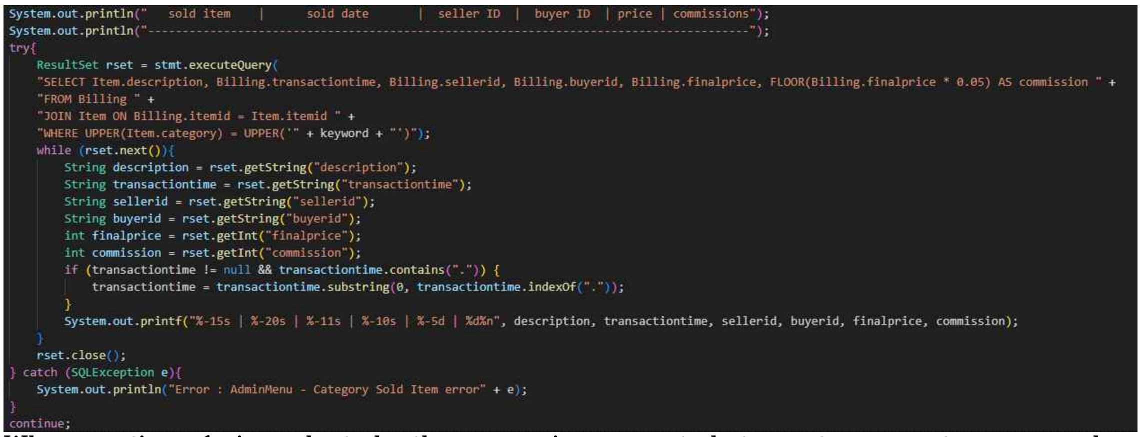
The system takes a userid and password from a user who claims to have administrator privileges (i.e., users who answered "Y" to the prompt "Is this user an administrator? (Y/N):"). If the user's isadmin value is 'Y', the login is successful. If it is 'N', the login fails, returns false, and the user is redirected to the menu.



```java
do {
    System.out.println(
            "----< Admin menu > \n" +
            "    1. Print Sold Items per Category \n" +
            "    2. Print Account Balance for Seller \n" +
            "    3. Print Seller Ranking \n" +
            "    4. Print Buyer Ranking \n" +
            "    P. Go Back to Previous Menu"
            );
```

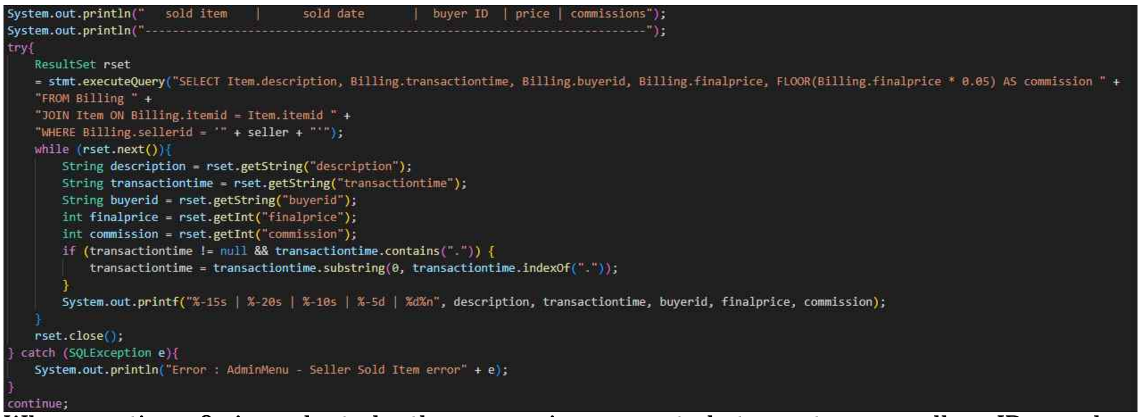
Upon successful login, the four options shown in the above screenshot are

presented to the user as the Admin menu, which is implemented using a do-while loop, allowing the user to repeatedly enter choices.
In addition, each time one of the four options is selected in the Admin menu, the time-based UPDATE queries that were present in BuyItem and the three Check functions are executed by default.



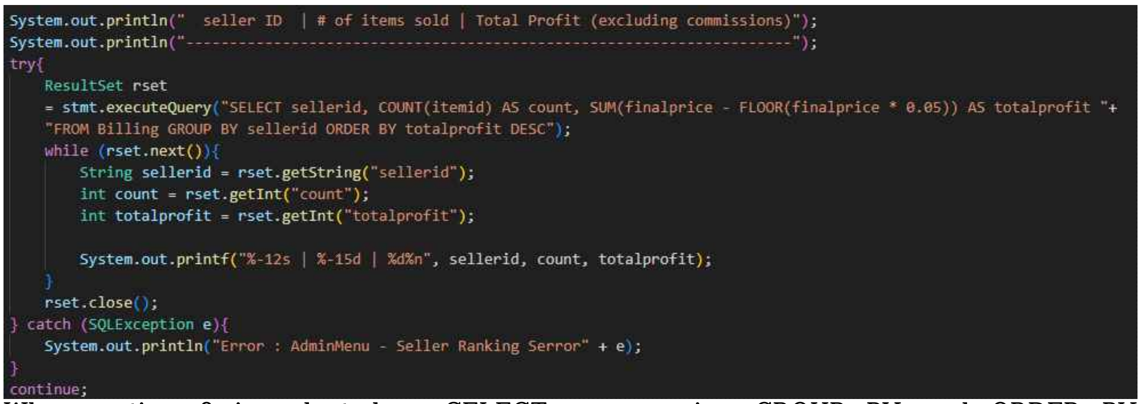
```java
System.out.println("   sold item    |      sold date       |  seller ID  |  buyer ID  | price | commissions");
System.out.println("-------------------------------------------------------------------------------------");
try{
    ResultSet rset = stmt.executeQuery(
    "SELECT Item.description, Billing.transactiontime, Billing.sellerid, Billing.buyerid, Billing.finalprice, FLOOR(Billing.finalprice * 0.05) AS commission " +
    "FROM Billing " +
    "JOIN Item ON Billing.itemid = Item.itemid " +
    "WHERE UPPER(Item.category) = UPPER('" + keyword + "')");
    while (rset.next()){
        String description = rset.getString("description");
        String transactiontime = rset.getString("transactiontime");
        String sellerid = rset.getString("sellerid");
        String buyerid = rset.getString("buyerid");
        int finalprice = rset.getInt("finalprice");
        int commission = rset.getInt("commission");
        if (transactiontime != null && transactiontime.contains(".")) {
            transactiontime = transactiontime.substring(0, transactiontime.indexOf("."));
        }
        System.out.printf("%-15s | %-20s | %-11s | %-10s | %-5d | %d%n", description, transactiontime, sellerid, buyerid, finalprice, commission);
    }
    rset.close();
} catch (SQLException e){
    System.out.println("Error : AdminMenu - Category Sold Item error" + e);
}
continue;
```

When option 1 is selected, the user is prompted to enter a category, and a SELECT query is used to display all items sold in that category.



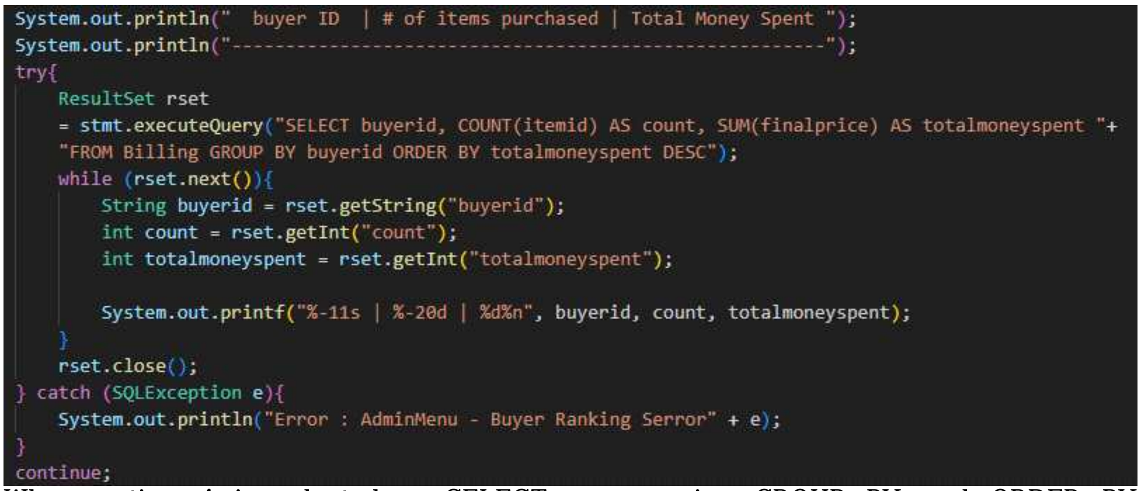
```java
System.out.println("   sold item    |      sold date       |  buyer ID  | price | commissions");
System.out.println("------------------------------------------------------------------------");
try{
    ResultSet rset
    = stmt.executeQuery("SELECT Item.description, Billing.transactiontime, Billing.buyerid, Billing.finalprice, FLOOR(Billing.finalprice * 0.05) AS commission " +
    "FROM Billing " +
    "JOIN Item ON Billing.itemid = Item.itemid " +
    "WHERE Billing.sellerid = '" + seller + "'");
    while (rset.next()){
        String description = rset.getString("description");
        String transactiontime = rset.getString("transactiontime");
        String buyerid = rset.getString("buyerid");
        int finalprice = rset.getInt("finalprice");
        int commission = rset.getInt("commission");
        if (transactiontime != null && transactiontime.contains(".")) {
            transactiontime = transactiontime.substring(0, transactiontime.indexOf("."));
        }
        System.out.printf("%-15s | %-20s | %-10s | %-5d | %d%n", description, transactiontime, buyerid, finalprice, commission);
    }
    rset.close();
} catch (SQLException e){
    System.out.println("Error : AdminMenu - Seller Sold Item error" + e);
}
continue;
```

When option 2 is selected, the user is prompted to enter a seller ID, and a SELECT query is used to display all items sold by that seller.



```java
System.out.println("  seller ID   | # of items sold | Total Profit (excluding commissions)");
System.out.println("----------------------------------------------------------------------");
try{
    ResultSet rset
    = stmt.executeQuery("SELECT sellerid, COUNT(itemid) AS count, SUM(finalprice - FLOOR(finalprice * 0.05)) AS totalprofit "+
    "FROM Billing GROUP BY sellerid ORDER BY totalprofit DESC");
    while (rset.next()){
        String sellerid = rset.getString("sellerid");
        int count = rset.getInt("count");
        int totalprofit = rset.getInt("totalprofit");

        System.out.printf("%-12s | %-15d | %d%n", sellerid, count, totalprofit);
    }
    rset.close();
} catch (SQLException e){
    System.out.println("Error : AdminMenu - Seller Ranking Serror" + e);
}
continue;
```

When option 3 is selected, a SELECT query using GROUP BY and ORDER BY (descending) is executed to display a ranking of sellers based on their total sell amount (after deducting the commission).


```
System.out.println("  buyer ID   | # of items purchased | Total Money Spent ");
System.out.println("--------------------------------------------------------");
try{
    ResultSet rset
    = stmt.executeQuery("SELECT buyerid, COUNT(itemid) AS count, SUM(finalprice) AS totalmoneyspent "+
    "FROM Billing GROUP BY buyerid ORDER BY totalmoneyspent DESC");
    while (rset.next()){
        String buyerid = rset.getString("buyerid");
        int count = rset.getInt("count");
        int totalmoneyspent = rset.getInt("totalmoneyspent");

        System.out.printf("%-11s | %-20d | %d%n", buyerid, count, totalmoneyspent);
    }
    rset.close();
} catch (SQLException e){
    System.out.println("Error : AdminMenu - Buyer Ranking Serror" + e);
}
continue;
```
When option 4 is selected, a SELECT query using GROUP BY and ORDER BY (descending) is executed to display a ranking of buyers based on their total purchase amount.

Additionally, pressing P returns the user to the previous menu.

The Auction system operates based on this mechanism.

In addition, the method for running the program is as follows
(as described in the README file) :
Compile → Use the command ‘make all’
Run → Use the command ‘make run’