1. 애플리케이션에 대한 간단한 소개

이 애플리케이션은 온라인 경매 시스템으로, 사용자들이 물품을 판매하거나 구매할 수 있는 기능을 제공합니다. 기본 기능으로는 사용자 데이터 관리, 물품 데이터 관리, 입찰 데이터 관리, 그리고 거래 데이터 관리가 포함되어 있습니다. 사용자는 물품을 등록하고 입찰을 받을 수 있으며, 구매자들은 특정 조건에 맞는 물품을 검색하고 입찰을 하거나 즉시 구매할 수 있습니다.

주요 기능

- 사용자 데이터: 사용자 ID, 비밀번호, 관리자 여부 등의 정보로 사용자를 관리합니다.
- 물품 데이터: 물품의 카테고리, 설명, 상태, 판매자 ID, 즉시 구매가, 게시 날짜, 상태 등을 포함하여 각 물품의 상세 정보를 관리합니다.
- 입찰 데이터: 물품 ID, 입찰가, 입찰자, 입찰 날짜, 입찰 마감 날짜 등의 정보로 입찰 내역을 관리합니다.
- 거래 데이터: 판매된 물품 ID, 구매 날짜, 판매자 ID, 구매자 ID 등을 포함한 거래 정보를 관리하여 결제와 관련된 기록을 유지합니다.

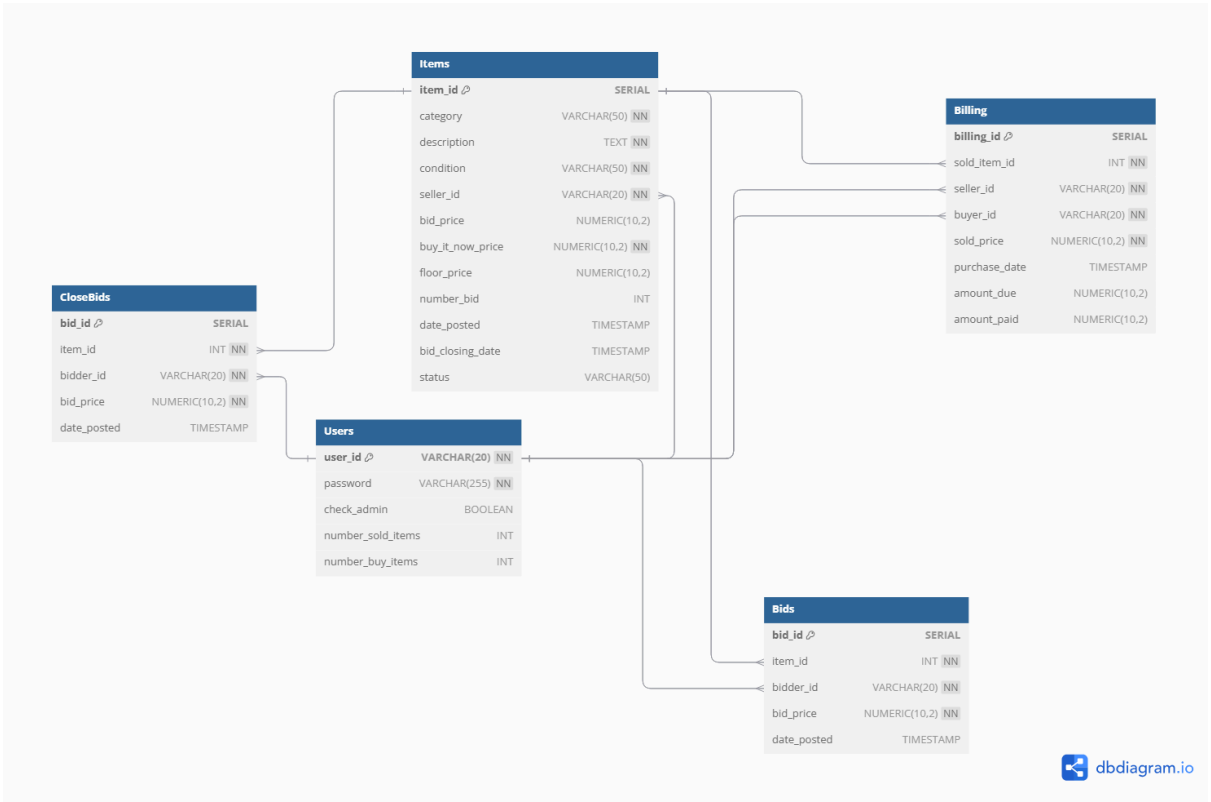
추가 기능

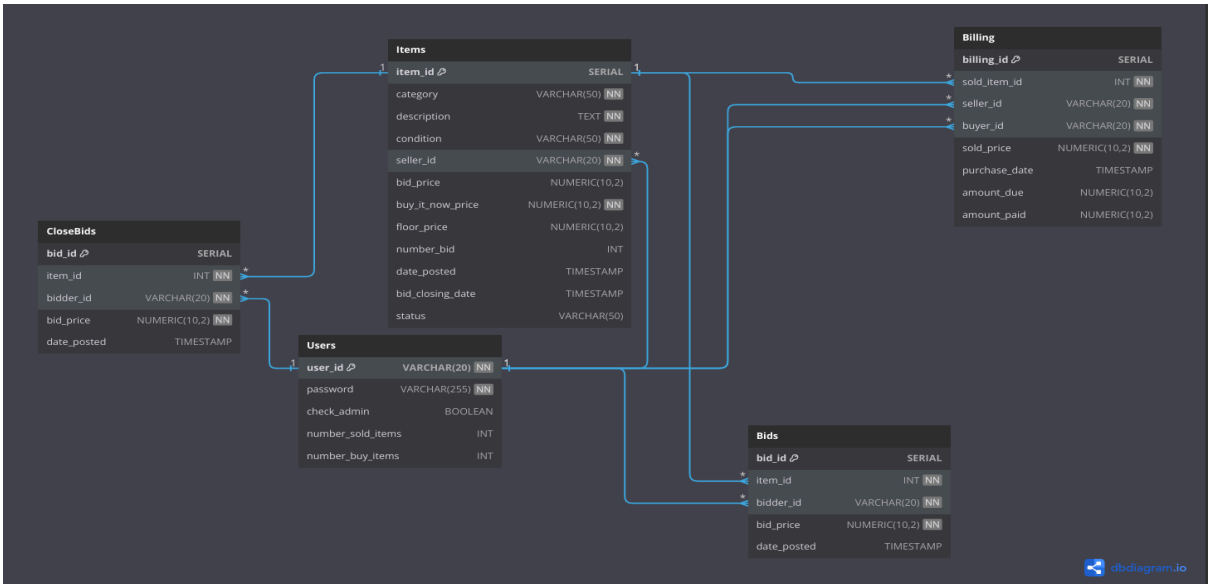
- 관리자 로그인 모드: 사용자가 기본 로그인 메뉴에서 관리자 계정으로 로그인하려고 할 경우, "관리자 로그인 모드를 선택하세요"라는 메시지를 표시하여, 관리자 전용 로그인 메뉴로 안내하는 기능을 추가했습니다
- 종료된 입찰 기록 관리: `CloseBids` 테이블을 추가로 구현하여 경매가 종료된 입찰 기록을 따로 저장함으로써 효율적인 데이터 관리가 가능하도록 했습니다. 이를 통해 종료된 입찰과 활성 입찰을 분리하여 관리할 수 있도록 하였습니다.

2. E-R 모델을 사용한 스키마 다이어그램
"이 프로젝트의 스키마 다이어그램은 https://dbdiagram.io의 도구를 참고하여 작성하였습니다."

Users와 Items: 한 사용자는 여러 아이템을 판매할 수 있음 (1:N)
Users와 Bids: 한 사용자는 여러 입찰을 할 수 있음 (1:N)

Items와 Bids: 한 아이템은 여러 입찰을 받을 수 있음(1:N)
Items와 CloseBids: 한 아이템은 여러 종료된 입찰을 가질 수 있음 (1:N)
Items와 Billing: 한 아이템은 하나의 거래 내역을 가짐 (1:1)
Users와 Billing: 사용자는 판매자 또는 구매자로서 여러 거래에 참여할 수 있음음 (1:N)





3. 애플리케이션에서 지원하는 기능 목록 및 각 기능에 대한 **SQL** 쿼리

1. LoginMenu 함수 내의 **SQL**쿼리

기능 설명:
사용자가 애플리케이션에 로그인할 수 있도록 합니다. 입력한 사용자 ID와 비밀번호가 정확할 경우에만 로그인할 수 있습니다.

SQL 쿼리:

```
SELECT * FROM Users WHERE (user_id = ?) AND (password = ?);
```

쿼리 설명:
입력된 `user_id`와 `password`가 `Users` 테이블의 `user_id` 및 `password`와 일치하는지 확인합니다. 해당 조건을 만족하는 레코드가 있으면 로그인에 성공하며, 없을 경우 로그인에 실패합니다.

2. SellMenu 함수 내의 **SQL** 쿼리

기능 설명:

판매자가 새로운 아이템을 경매에 등록할 수 있도록 합니다. 카테고리, 설명, 상태, 판매자 ID, 즉시 구매 가격, 마감 날짜, 상태 등의 정보를 입력하여 아이템을 등록합니다.

SQL 쿼리:

```
INSERT INTO Items (category, description, condition, seller_id, bid_price, buy_it_now_price, floor_price, number_bid, date_posted, bid_closing_date, status) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

쿼리 설명:

`Items` 테이블에 새로운 레코드를 추가하여 아이템을 등록합니다. `category`, `description`, `condition`, `seller_id`, `buy_it_now_price`, `bid_closing_date`, `status` 등의필드에 각각의 값을 입력합니다. 이 쿼리를 통해 입력된 정보를 기반으로 새로운 아이템이 경매 목록에 추가됩니다.

3. SignupMenu 함수 내의 **SQL** 쿼리

기능 설명:
새로운 사용자가 애플리케이션에 회원가입할 수 있도록 합니다. 사용자 ID, 비밀번호, 관리자 여부를 입력하여 계정을 생성합니다.

SQL 쿼리:

```
INSERT INTO Users (user_id, password, check_admin) VALUES (?, ?, ?);
```

쿼리 설명:
`Users` 테이블에 새로운 사용자를 추가하는 쿼리입니다. `user_id`, `password`, `check_admin` 필드에 각각의 값을 입력하여 사용자 계정을 생성합니다. `user_id`는 고유해야 하며, 이 쿼리를 통해 입력된 정보가 데이터베이스에 저장되어 사용자가 애플리케이션에 접근할 수 있게 됩니다.

4. AdminMenu 함수 내의 **SQL** 쿼리

기능 설명:
관리자가 애플리케이션에 로그인할 수 있도록 합니다. 관리자의 ID와 비밀번호를 확인하여 관리자 권한으로 접근을 허용합니다.

SQL 쿼리:

```
SELECT * FROM Users WHERE user_id = ? AND password = ? AND check_admin = true;
```

쿼리 설명:
`Users` 테이블에서 `user_id`와 `password`가 일치하고, `check_admin`이 `true`인 사용자를 검색하는 쿼리입니다. 해당 조건을 만족하는 레코드가 있으면 관리자 로그인이 성공하며, 그렇지 않으면 접근이 거부됩니다. 이 쿼리는 사용자가 관리자 권한을 가지고 있는지를 확인하는 역할을 합니다.

AdminMenu 1. Print Sold Items per Category

기능 설명:
특정 카테고리에 속하는 판매 완료된 아이템 목록을 조회합니다. 각 아이템의 판매 정보와 수수료 정보를 출력합니다.

SQL 쿼리:

```
SELECT b.sold_item_id AS item_id, b.purchase_date AS sold_date, b.seller_id, b.buyer_id, b.sold_price AS price, FLOOR(b.sold_price * 0.1) AS commission

FROM Billing AS b JOIN Items AS i ON b.sold_item_id =i.item_id
```

```
WHERE i.category = ?;
```

쿼리 설명:
`Billing` 테이블과 `Items` 테이블을 조인하여 특정 카테고리에 속하는 판매 완료된 아이템의 정보를 검색하는 쿼리입니다. `category`가 지정된 값과 일치하는 항목을 조회하며, 결과에는 아이템 ID, 판매 날짜, 판매자 ID, 구매자 ID, 판매 가격, 그리고 10% 수수료가 포함됩니다. `FLOOR` 함수는 수수료를 정수로 내림 처리합니다.

AdminMenu 2. Print Account Balance for Seller

기능 설명:
특정 판매자가 판매한 아이템 목록을 조회합니다. 각 아이템의 판매 날짜, 구매자, 판매 가격 및 수수료 정보를 출력합니다.

SQL 쿼리:

```
SELECT sold_item_id AS item_id, purchase_date AS sold_date,
buyer_id, sold_price AS price,ROUND(sold_price * 0.1, 1) AS
commission

FROM Billing

WHERE seller_id = ?;
```

쿼리 설명:
`Billing` 테이블에서 특정 `seller_id`를 가진 판매자가 판매한 아이템의 정보를 검색하는 쿼리입니다. 결과에는 아이템 ID, 판매 날짜, 구매자 ID, 판매 가격, 그리고 10% 수수료가 포함됩니다. `ROUND` 함수는 수수료를 소수점 첫째 자리에서 반올림하여 표시합니다.

AdminMenu 3. Print Seller Ranking

기능 설명**:**
판매자별로 판매된 아이템 수와 총 수익을 조회하여, 수익 순으로 정렬하여 출력합니다. 각 판매자의 순위와 수익 정보를 확인할 수 있습니다.

SQL 쿼리**:**

```
SELECT seller_id, COUNT(sold_item_id) AS
items_sold,SUM(sold_price - sold_price * 0.1) AS total_profit
```

```
FROM Billing

GROUP BY seller_id

ORDER BY total_profit DESC;
```

쿼리 설명:
`Billing` 테이블에서 `seller_id`별로 판매된 아이템 수(`items_sold`)와 총 수익(`total_profit`)을 계산하는 쿼리입니다. `SUM` 함수는 각 판매자가 번 총 수익에서 10% 수수료를 뺀 값을 합산하여 계산합니다. `GROUP BY`를 사용해 판매자별로 집계하고, `ORDER BY total_profit DESC`로 수익이 높은 순서대로 정렬하여 결과를 표시합니다.

AdminMenu 4. Print Buyer Ranking

기능 설명:
구매자별로 구매한 아이템 수와 총 지출 금액을 조회하여, 지출 금액 순으로 정렬하여 출력합니다. 각 구매자의 순위와 지출 정보를 확인할 수 있습니다.

SQL 쿼리:

```
SELECT buyer_id, COUNT(sold_item_id) AS items_purchased,
SUM(sold_price) AS total_spent

FROM Billing

GROUP BY buyer_id

ORDER BY total_spent DESC;
```

쿼리 설명:
`Billing` 테이블에서 `buyer_id`별로 구매한 아이템 수(`items_purchased`)와 총 지출 금액(`total_spent`)을 계산하는 쿼리입니다. `SUM` 함수는 각 구매자가 지출한 총 금액을 합산하여 계산합니다. `GROUP BY`를 사용해 구매자별로 집계하고, `ORDER BY total_spent DESC`로 지출 금액이 높은 순서대로 정렬하여 결과를 표시합니다.

5. CheckSellStatus 함수 내의 **SQL** 쿼리

기능 설명:
판매자가 등록한 활성화된 아이템에 대해 입찰된 정보를 조회합니다. 이 기능을 통해 판매자는 자신이 올린 상품의 현재 입찰 상황을 확인할 수 있습니다.

SQL 쿼리:

```
SELECTi.item_id,i.status,b.bidder_id,b.bid_price,b.date_posted

FROM Items AS i LEFT JOIN Bids AS b ON i.item_id = b.item_id

WHERE i.seller_id = ? AND i.status = 'active';
```

쿼리 설명:
`Items` 테이블과 `Bids` 테이블을 `item_id`를 기준으로 조인하여, 특정 판매자(`seller_id`)가 등록한 활성화된(`status = 'active'`) 아이템의 입찰 정보를 가져오는 쿼리입니다. `LEFT JOIN`을 사용하여 입찰 기록이 없는 아이템도 포함하여 조회할 수 있습니다. 결과에는 아이템 ID, 상태, 입찰자 ID, 입찰 가격, 입찰 날짜가 포함됩니다.

6. BuyItem 함수 내의 **SQL** 쿼리

기능 설명:
사용자가 검색 조건에 맞는 아이템을 조회하고, 각 아이템의 현재 입찰가와 최고 입찰자 정보를 확인할 수 있도록 합니다. 사용자는 카테고리, 조건, 설명, 판매자 및 등록 날짜로 아이템을 필터링할 수 있습니다.

SQL 쿼리:

```
SELECT i.item_id, i.description, i.condition, i.seller_id,
i.buy_it_now_price, COALESCE(MAX(b.bid_price), 0) AS
current_bid,COALESCE(MAX(b.bidder_id), 'No Bids') AS
highest_bidder, i.bid_closing_date

FROM Items AS i

LEFT JOIN Bids AS b ON i.item_id = b.item_id

WHERE i.date_posted >= ?
```

(조건부로 추가되는 부분)

- 카테고리가 선택된 경우: `AND i.category = ?`
- 조건이 지정된 경우: `AND i.condition = ?`
- 설명에 특정 키워드가 포함된 경우: `AND i.description LIKE ?`
- 특정 판매자의 아이템을 검색하는 경우: `AND i.seller_id = ?`

마지막으로, 결과는 다음 필드별로 그룹화됩니다:

```
GROUP BY i.item_id, i.description, i.condition, i.seller_id,
i.buy_it_now_price, i.bid_closing_date
```

쿼리 설명:

Items 테이블과 Bids 테이블을 조인하여 조건에 맞는 아이템을 조회합니다. 필터링된 아이템에 대해 현재 최고 입찰가(current_bid)와 최고 입찰자(highest_bidder) 정보를 포함하여 표시합니다. COALESCE 함수를 통해 입찰이 없는 경우 기본값으로 0 또는 'No Bids'를 반환합니다.

기능 설명:
사용자가 아이템에 대해 구매하거나 입찰할 때, 유효한지 확인하기 위해 즉시 구매 가격과 입찰 마감 날짜를 조회합니다.

SQL 쿼리:

```
SELECT buy_it_now_price, bid_closing_date

FROM Items

WHERE item_id = ?
```

쿼리 설명:
Items 테이블에서 item_id가 일치하는 아이템의 buy_it_now_price와 bid_closing_date를 조회합니다. 이를 통해 사용자가 설정한 가격이 즉시 구매 가격 이상인지와 입찰이 아직 유효한지를 확인할 수 있습니다.

기능 설명:
사용자가 즉시 구매 가격 이상으로 구매를 완료할 때, 거래 내역을 기록하고 해당 아이템의 입찰 정보를 종료 처리합니다.

1. 거래 내역 추가

SQL 쿼리:

```
String insertBilling = "INSERT INTO Billing (sold_item_id,
seller_id, buyer_id, sold_price, amount_due, amount_paid,
purchase_date) " +

                       "VALUES (?, ?, ?, ?, ?, ?,
CAST(CURRENT_TIMESTAMP AS TIMESTAMP(0)))";
```

쿼리 설명:
아이템이 즉시 구매된 경우, Billing 테이블에 거래가 발생한 아이템의 ID, 판매자 ID, 구매자 ID, 그리고 최종 판매 가격을 기록합니다.

Items 테이블의 **status**를 **'sold'**로 업데이트

기능 설명:
거래가 완료된 후, 해당 아이템의 상태를 active에서 sold로 변경하여 더 이상 입찰이나 구매가 불가능하게 합니다.

SQL 쿼리:

```
UPDATE Items SET status = 'sold' WHERE item_id = ?
```

쿼리 설명:
지정된 item_id에 해당하는 레코드의 status 값을 'sold'로 업데이트합니다.

거래 완료 후 **Items** 테이블의 **number_bid** 증가

기능 설명:
입찰 또는 즉시 구매 시 해당 아이템의 입찰 횟수인 number_bid 값을 1 증가시킵니다.

SQL 쿼리:

```
UPDATE Items SET number_bid = number_bid + 1 WHERE item_id = ?
```

쿼리 설명:
지정된 item_id에 해당하는 레코드의 number_bid 값을 1 증가시켜 입찰 또는 구매가 이루어졌음을 기록합니다.

판매자의 **number_sold_items** 증가

기능 설명:
거래가 완료된 후, 해당 아이템을 판매한 판매자의 판매 횟수를 1 증가시킵니다.

SQL 쿼리:

```
UPDATE Users SET number_sold_items = number_sold_items + 1
WHERE user_id = ?
```

쿼리 설명:
지정된 user_id에 해당하는 사용자의 number_sold_items 값을 1 증가시켜 판매 기록을 업데이트합니다.

구매자의 **number_buy_items** 증가

기능 설명:
거래가 완료된 후, 해당 아이템을 구매한 구매자의 구매 횟수를 1 증가시킵니다.

SQL 쿼리:

```
UPDATE Users SET number_buy_items = number_buy_items + 1 WHERE user_id = ?
```

쿼리 설명:
지정된 user_id에 해당하는 사용자의 number_buy_items 값을 1 증가시켜 구매 기록을 업데이트합니다.

2. 입찰 기록 종료 처리

SQL 쿼리:

```
INSERT INTO CloseBids (item_id, bidder_id, bid_price, date_posted)

SELECT item_id, bidder_id, bid_price,  CAST(date_posted AS TIMESTAMP(0))

FROM Bids

WHERE item_id = ?
```

쿼리 설명:
Bids 테이블에서 해당 아이템의 모든 입찰 정보를 CloseBids 테이블로 이동하여, 종료된 입찰로 처리합니다.

3. 기존 입찰 기록 삭제

SQL 쿼리:

```
DELETE FROM Bids

WHERE item_id = ?
```

쿼리 설명:
Bids 테이블에서 해당 아이템의 입찰 내역을 삭제하여, 아이템이 더 이상 입찰 상태가 아님을 나타냅니다.

이러한 일련의 쿼리들이 실행되어 아이템이 즉시 구매되었을 때의 처리를 완성합니다.

기능 설명:
사용자가 지정한 가격으로 특정 아이템에 입찰을 할 때, Bids 테이블에 입찰 정보를 추가합니다.

SQL 쿼리:

```
INSERT INTO Bids (item_id, bidder_id, bid_price, date_posted)

VALUES (?, ?, ?, CURRENT_TIMESTAMP)
```

쿼리 설명:
Bids 테이블에 새로운 입찰 기록을 추가합니다. item_id는 입찰된 아이템의 ID, bidder_id는 입찰한 사용자의 ID, bid_price는 입찰 가격이며, date_posted는 현재 시간(CURRENT_TIMESTAMP)으로 설정되어 입찰 시간이 기록됩니다.

이 쿼리를 통해 특정 아이템에 대한 사용자의 입찰이 데이터베이스에 저장됩니다.

```
// Items 테이블의 number_bid 증가

                        String updateNumberBid = "UPDATE
Items SET number_bid = number_bid + 1 WHERE item_id = ?";

                        PreparedStatement updateBidStmt =
conn.prepareStatement(updateNumberBid);
```

```
                        updateBidStmt.setInt(1,
selectedItemID);

                        updateBidStmt.executeUpdate();
```

7. CheckBuyStatus 함수 내의 **SQL** 쿼리

기능 설명:
사용자가 입찰한 아이템들의 상태를 확인하고, 각 아이템에 대한 최고 입찰가와 사용자의 입찰가를 함께 표시합니다.

SQL 쿼리:

```
SELECT i.item_id, i.description, COALESCE(b1.bidder_id, 'No
Bids') AS highest_bidder,COALESCE(MAX(b1.bid_price), 0) AS
highest_bidding_price,b2.bid_price AS your_bidding_price,
i.bid_closing_date

FROM Items AS i LEFT JOIN Bids AS b1 ON i.item_id = b1.item_id

LEFT JOIN Bids AS b2 ON i.item_id = b2.item_id AND
b2.bidder_id = ?

WHERE b2.bidder_id = ?

GROUP BY i.item_id, i.description, b1.bidder_id, b2.bid_price,
i.bid_closing_date
```

쿼리 설명:

- `Items` 테이블을 `Bids` 테이블과 조인하여 사용자가 입찰한 아이템의 정보와 현재 최고 입찰가를 조회합니다.
- `b1`은 최고 입찰가와 최고 입찰자를 확인하기 위한 조인으로, `b1.bid_price`의 최대값을 `highest_bidding_price`로 가져옵니다.
- `b2`는 사용자의 입찰가(`your_bidding_price`)를 확인하기 위한 조인입니다.
- `COALESCE`를 사용해 최고 입찰자와 입찰가가 없을 경우 각각 `No Bids`와 `0`을 기본값으로 반환합니다.
- `GROUP BY`는 아이템별로 입찰 상태를 그룹화하여 각 아이템의 최고 입찰가와 사용자의 입찰가를 함께 보여줍니다.

8. CheckAccount 함수 내의 **SQL** 쿼리

기능 설명:
판매자가 자신이 판매한 아이템에 대한 상세 정보를 확인합니다. 이 정보에는 판매일, 판매 가격, 구매자 ID, 수수료, 판매자가 실제로 받을 금액이 포함됩니다.

SQL 쿼리:

```
SELECT i.category, b.sold_item_id, b.purchase_date,
b.sold_price, b.buyer_id,ROUND(b.sold_price * 0.1, 1) AS
commission,ROUND(b.sold_price - b.sold_price * 0.1, 0) AS
net_amount_due,i.bid_closing_date

FROM Billing AS b

JOIN Items AS i ON b.sold_item_id = i.item_id

WHERE b.seller_id = ?
```

쿼리 설명:

- `Billing` 테이블과 `Items` 테이블을 조인하여 판매된 아이템의 상세 정보를 가져옵니다.
- `ROUND(b.sold_price * 0.1, 1)`을 사용해 10% 수수료를 계산하여 `commission`으로 표시하며, 소수점 첫째 자리까지 표시합니다.
- `ROUND(b.sold_price - b.sold_price * 0.1, 0)`로 수수료를 제외한 실제 지급 금액을 `net_amount_due`로 계산하여 표시합니다.
- `WHERE b.seller_id = ?` 조건을 사용해 로그인한 판매자에 해당하는 기록만 조회합니다.

기능 설명:
구매자가 자신이 구매한 아이템에 대한 상세 정보를 확인합니다. 정보에는 카테고리, 아이템 ID, 구매 날짜, 구매 가격, 판매자 ID가 포함됩니다.

SQL 쿼리:

```
SELECT i.category, b.sold_item_id, b.purchase_date,
b.sold_price, b.seller_id

FROM Billing AS b

JOIN Items AS i ON b.sold_item_id = i.item_id

WHERE b.buyer_id = ?
```

쿼리 설명:

- `Billing` 테이블과 `Items` 테이블을 조인하여 구매한 아이템의 상세 정보를 가져옵니다.
- `i.category`, `b.sold_item_id`, `b.purchase_date`, `b.sold_price`, `b.seller_id`를 선택하여 구매자의 구매 내역을 표시합니다.
- `WHERE b.buyer_id = ?` 조건을 통해 로그인한 구매자에 해당하는 기록만 조회합니다.