Final Updated Documentation

CPS510 Section 5

Karen Loac 501099937

Justin Lee 501109682

Zoravar Multani 501110311

Jefferey Wong 501103254

TA: Reza Barati

November 30th 2023

[**Final Application Description 2**](#_7tkckjut58jg)

[Application Details 2](#_1bhq8ashrwyk)

[Information 2](#_g5odrpc9t18w)

[Functions 2](#_1yv1ub4v86ir)

[Table Designs 3](#_1w52gf37da6g)

[**Final ER Model 6**](#_4m378q10c671)

[**Final Installation and Instructions 6**](#_jfwdon5f6lxc)

[**A01: Application Description 6**](#_oz8walrd2uzq)

[Application Details 6](#_8bjqlj5s1tjy)

[Information 7](#_ncw8k4k5rip1)

[Functions 7](#_wb3ttoy3yu0x)

[Tables 7](#_ma5vdix63tu7)

[**A02: ER Model 9**](#_zh5dvvx1uhgi)

[**A03: Schema Design 9**](#_4viosoimozoc)

[**A04P1: Designing Simple Queries 13**](#_uopnmrxyb9jw)

[**A04P2: Designing Advanced Queries, Joins & Views 16**](#_r61uu94r6n82)

[Queries 16](#_adoouh9s8wrf)

[Views 17](#_x4or8e25dxjc)

[**A05: Advanced Queries by Unix Shell Implementation 18**](#_9lorb358ech2)

[**A06: Functional Dependencies 21**](#_g419pmr0s5j)

[**A07: Normalization to 3NF 24**](#_2xrh0xwagntr)

[**A08: Normalization 3NF/BCNF by Algorithm 29**](#_u4uv9paou991)

[**A09: Demo by Unix Shell of 3NF/BCNF Tables 32**](#_i3ugkfh2ebhv)

[**A10: Relational Algebra Statements 43**](#_1jdykiezie3)

### **Final Application Description**

The application chosen to create for our project is the Auction Database Management System(ADBMS). The ADBMS allows users to sign up as auctioneers or bidders. Auctioneers will be able to manage their auctions, and bidders can easily track their current bids. The ADBMS will run like a simple auction website. Auctioneers will create an item listing, the highest bid wins the item once the timer has run out.

#### Application Details

Each user will have a unique username and password that will grant them access to what their account concerns, auctioneers will be able to see their auctions and bidders are able to see the items that they have bid for while the auction is live. The bidding system will be functioning as most auctions do with the top bid taking the item, and a time limit and a base bid the auctioneer chooses. The next big beyond the current top bid on an item will be a certain percentage of the previous top bid. The program will also have admin users to look over the quality assurance of the auctions taking place, keeping all auctions functioning.

#### Information

For a user to either sell or buy an item on the site, they would have to create an account. The account will contain various attributes such as their userID, name, username, and password. The primary key of a user is their userID, as it will be their unique identifier different from others. The admin account moderates the application, and requires the standard username and password, but also a special adminID that differentiates them. Items will be managed under an auction which contains the userID of the seller and the unique itemID, the composite primary key, as well as an auctionID and the latest date it was bid on. The items themselves require its primary key, itemID, item name, item description, minimum bid, start date, and buyout price. Item bids are also tracked, and use the itemID, userID of a buyer, and the top bid made by said user, where its primary key is a bidID. When a new top bid is made, the previous top bid and it’s date is stored with other past bids, a pastBidID is created to be the primary key. After winning a bid a user must ship the item, where the itemID, shipping address, and tracking number are used to keep track of the shipment, its primary key is the itemID. A buyer must pay using their financial details such as their card number, expiry date, billing address, and CVV, all of it identified by their userID, which is the primary key.

#### Functions

**Shipments** - Auctioneers with an item listing can ship their item to the bidder who won. Information needed is the item number and buyer’s shipping address. A tracking number is created.

**Payments** - Bidder pays auctioneer with credit information

**Bidding** - The auctioneer sets a base bid, during the bidding process other users will be able to bid up for the item based on the current top bid. The current top bid is stored separately after new top bid is made.

**Create Item Listing** - An auctioneer will be able to list an item in their possession for auction at a starting bid of their choice.

**Auction Modification** - Only admins are able to modify auctions in case of inappropriate listings. Admins will have their own special ID for access.

#### Table Designs

| **USER (bidder and auctioneer)** | | | |
| --- | --- | --- | --- |
| User\_ID | Name | Username | Password |
|  |  |  |  |
|  |  |  |  |

| **ADMIN** | | |
| --- | --- | --- |
| Admin ID | Username | Password |
|  |  |  |
|  |  |  |

| **ITEMS** | | |
| --- | --- | --- |
| Item\_ID | Item\_Name | Item\_Description |
|  |  |  |
|  |  |  |

| **ITEM PRICING** | | |
| --- | --- | --- |
| Item\_ID | Item\_minBid | Item\_buyPrice |
|  |  |  |
|  |  |  |

| **ITEM DATES** | | |
| --- | --- | --- |
| Item\_ID | Item\_startDate | Item\_endDate |
|  |  |  |
|  |  |  |

| **AUCTIONS** | | |
| --- | --- | --- |
| auction\_ID | user\_ID | item\_ID |
|  |  |  |
|  |  |  |

| **AUCTION STATUS** | |
| --- | --- |
| auction\_ID | auction\_latest |
|  |  |
|  |  |

| **FINANCE USER** | |
| --- | --- |
| user\_ID | finance\_number |
|  |  |
|  |  |

| **FINANCE DETAILS** | | | |
| --- | --- | --- | --- |
| finance\_number | finance\_address | finance\_expiry | finance\_cvv |
|  |  |  |  |
|  |  |  |  |

| **SHIPPING USER** | |
| --- | --- |
| user\_ID | shipping\_tracking |
|  |  |
|  |  |

| **SHIPPING DETAILS** | |
| --- | --- |
| shipping\_tracking | shipping\_reciever |
|  |  |
|  |  |

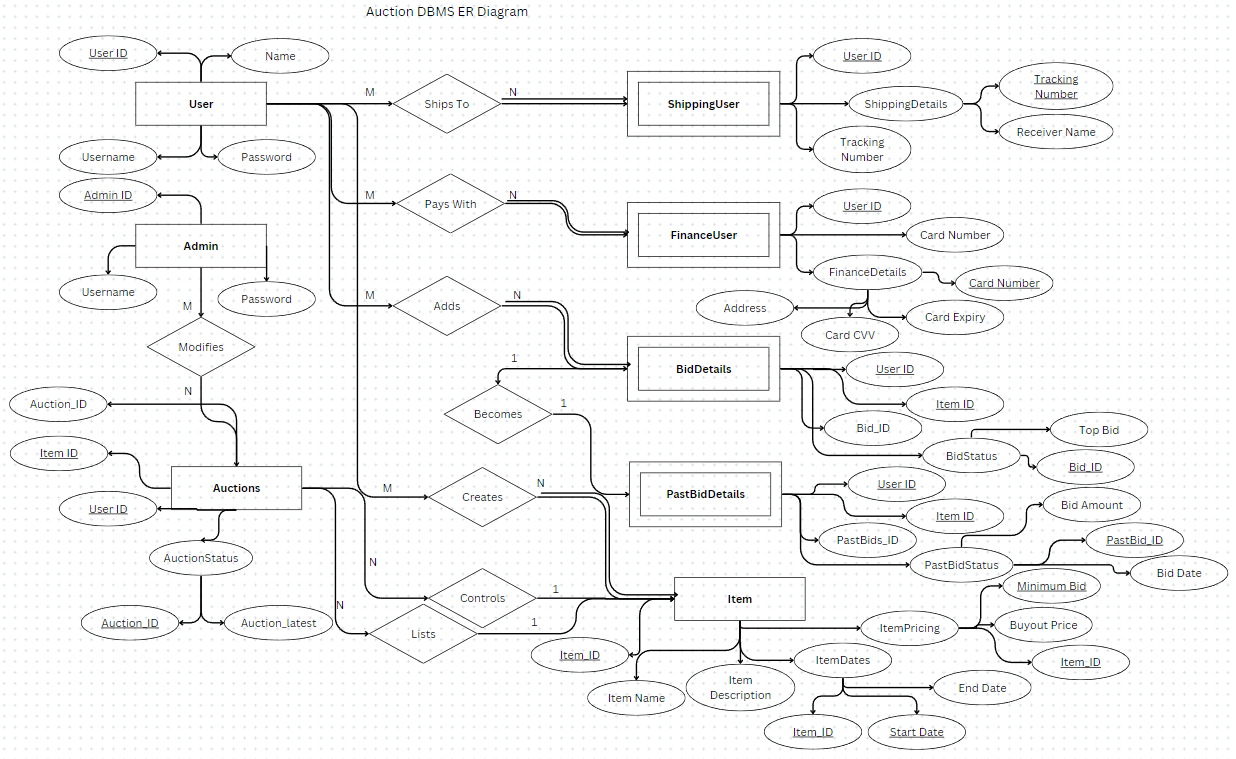
| **BID DETAILS** | | |
| --- | --- | --- |
| item\_ID | user\_ID | bids\_ID |
|  |  |  |
|  |  |  |

| **BID STATUS** | |
| --- | --- |
| bids\_ID | bids\_top |
|  |  |
|  |  |

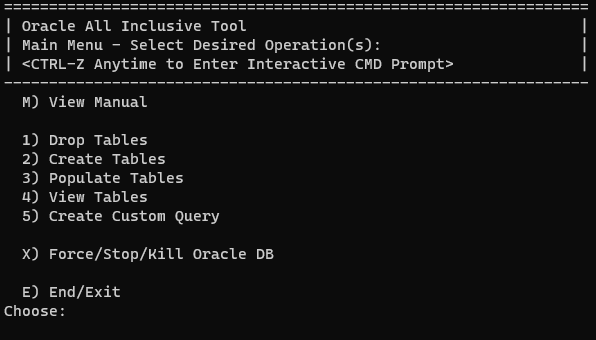
| **PAST BID DETAILS** | | |
| --- | --- | --- |
| user\_ID | item\_ID | pastBids\_ID |
|  |  |  |
|  |  |  |

| **PAST BID STATUS** | | |
| --- | --- | --- |
| pastBids\_ID | pastBid\_bidDate | pastBid\_bid |
|  |  |  |
|  |  |  |

### **Final ER Model**



### **Final Operational Instructions of UI**



Operations:

1. Drop all the tables in the Oracle DB
2. Create the tables for the auction DBMS
3. Put dummy data into the tables already created for the auction DBMS
4. View all the created tables with its dummy data
5. Create a custom query and view its output

### **A01: Application Description**

The application chosen to create for our project is the Auction Database Management System(ADBMS). The ADBMS allows users to sign up as auctioneers or bidders. Auctioneers will be able to manage their auctions, and bidders can easily track their current bids. The ADBMS will run like a simple auction website. Auctioneers will create an item listing, the highest bid wins the item once the timer has run out.

#### Application Details

Each user will have a unique username and password that will grant them access to what their account concerns, auctioneers will be able to see their auctions and bidders are able to see the items that they have bid for while the auction is live. The bidding system will be functioning as most auctions do with the top bid taking the item, and a time limit and a base bid the auctioneer chooses. The next big beyond the current top bid on an item will be a certain percentage of the previous top bid. The program will also have admin users to look over the quality assurance of the auctions taking place, keeping all auctions functioning.

#### Information

For a user to either sell or buy an item on the site, they would have to create an account. The account will contain various attributes such as their userID, name, username, and password. The primary key of a user is their userID, as it will be their unique identifier different from others. The admin account moderates the application, and requires the standard username and password, but also a special adminID that differentiates them. Items will be managed under an auction which contains the userID of the seller, the primary key, and a unique itemID. The items themselves require its primary key, itemID, item name, item description, minimum bid, start date, and buyout price. Item bids are also tracked, and use the itemID, userID of a buyer, and the top bid made by said user, where its primary key is the itemID. After winning a bid a user must ship the item, where the itemID, shipping address, and tracking number are used to keep track of the shipment, its primary key is the itemID. A buyer must pay using their financial details such as their card number, expiry date, billing address, and CVV, all of it identified by their userID, which is the primary key.

#### Functions

**Shipments** - Auctioneers with an item listing can ship their item to the bidder who won. Information needed is the item number and buyer’s shipping address. A tracking number is created.

**Payments** - Bidder pays auctioneer with credit information

**Bidding** - The auctioneer sets a base bid, during the bidding process other users will be able to bid up for the item based on the current top bid.

**Create Item Listing** - An auctioneer will be able to list an item in their possession for auction at a starting bid of their choice.

**Auction Modification** - Only admins are able to modify auctions in case of inappropriate listings. Admins will have their own special ID for access.

#### Tables

| **USER (bidder and auctioneer)** | | | |
| --- | --- | --- | --- |
| User\_ID | Name | Username | Password |
|  |  |  |  |
|  |  |  |  |

| **ADMIN** | | |
| --- | --- | --- |
| Admin ID | Username | Password |
|  |  |  |
|  |  |  |

| **AUCTIONS** | |
| --- | --- |
| User ID | Item ID |
|  |  |
|  |  |

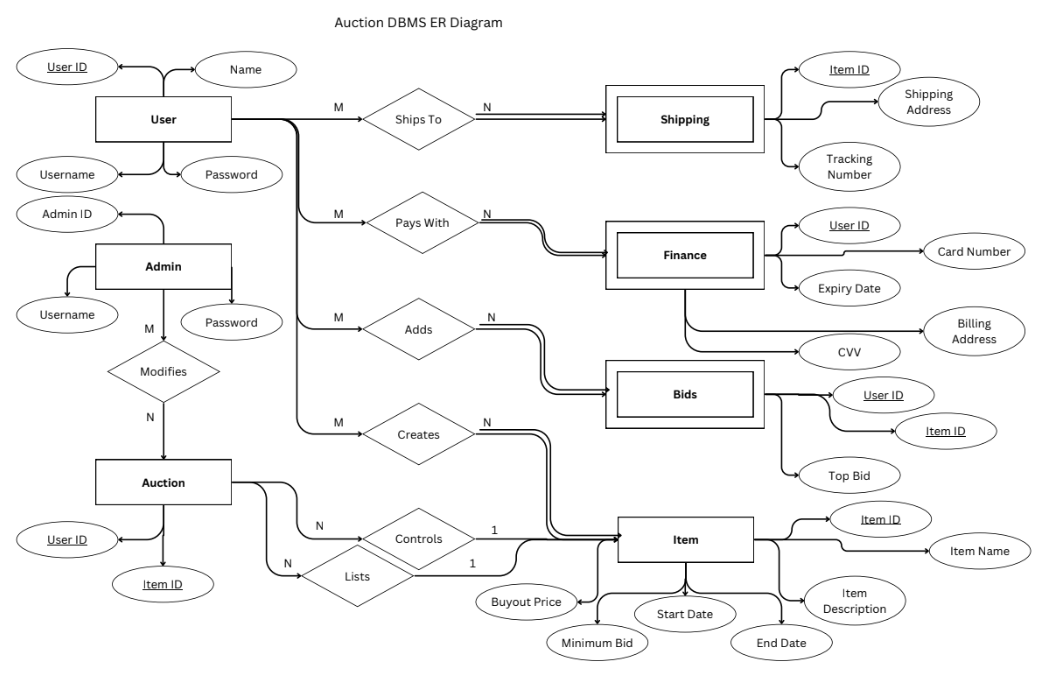
| **ITEMS** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Item ID | Item Name | Item Description | Minimum Bid | Start Date | End Date | Buyout Price |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

| **BIDS** | | |
| --- | --- | --- |
| Item ID | User ID | Top Bid |
|  |  |  |
|  |  |  |

| **SHIPPING** | | |
| --- | --- | --- |
| Item ID | Shipping Address | Tracking Number |
|  |  |  |
|  |  |  |

| **FINANCE** | | | | |
| --- | --- | --- | --- | --- |
| User ID | Card Number | Expiry Date | Billing Address | CVV |
|  |  |  |  |  |
|  |  |  |  |  |

### **A02: ER Model**



### **A03: Schema Design**

DROP TABLE auctions CASCADE CONSTRAINTS;

DROP TABLE admins CASCADE CONSTRAINTS;

DROP TABLE items CASCADE CONSTRAINTS;

DROP TABLE users CASCADE CONSTRAINTS;

DROP TABLE finance CASCADE CONSTRAINTS;

DROP TABLE shipping CASCADE CONSTRAINTS;

DROP TABLE bids CASCADE CONSTRAINTS;

-- items rely on auction

-- auctions rely on users and items

CREATE TABLE users (

user\_id INTEGER NOT NULL,

user\_name VARCHAR2(30) NOT NULL,

user\_username VARCHAR2(30) NOT NULL UNIQUE,

user\_password VARCHAR2(30) NOT NULL,

PRIMARY KEY(user\_id)

);

CREATE TABLE admins (

admin\_id INTEGER NOT NULL,

admin\_username VARCHAR2(30) NOT NULL UNIQUE,

admin\_password VARCHAR2(30) NOT NULL,

PRIMARY KEY(admin\_id)

);

CREATE TABLE items (

item\_id INTEGER NOT NULL,

item\_name VARCHAR2(30),

item\_description VARCHAR2(100),

item\_minBid DECIMAL(10, 2) DEFAULT 0.00,

-- DATE Format: YYYY-MM-DD

item\_startDate DATE NOT NULL,

item\_endDate DATE NOT NULL,

item\_buyPrice DECIMAL(10, 2),

PRIMARY KEY(item\_id)

);

CREATE TABLE auctions (

user\_id INTEGER REFERENCES users(user\_id),

item\_id INTEGER REFERENCES items(item\_id),

PRIMARY KEY(user\_id, item\_id)

);

CREATE TABLE finance (

user\_id INTEGER REFERENCES users(user\_id),

finance\_number INTEGER NOT NULL,

finance\_expiry DATE NOT NULL,

finance\_address VARCHAR2(30) NOT NULL,

finance\_cvv INTEGER NOT NULL,

PRIMARY KEY(user\_id)

);

CREATE TABLE shipping (

user\_id INTEGER REFERENCES users(user\_id),

shipping\_receiver VARCHAR2(200) NOT NULL,

shipping\_tracking VARCHAR2(50) NOT NULL,

PRIMARY KEY(user\_id)

);

CREATE TABLE bids (

item\_id INTEGER REFERENCES items(item\_id),

user\_id INTEGER REFERENCES users(user\_id),

bids\_top DECIMAL(10, 2) NOT NULL,

PRIMARY KEY(item\_id, user\_id)

);

-- Insert data into admins table

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (01, 'bossBaby', 'yummyyippy');

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (02, 'workerBaby1', 'yummyyippy1');

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (03, 'workerBaby2', 'yummyyippy2');

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (04, 'workerBaby3', 'yummyyippy3');

-- Insert data into users table

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (001, 'fraudkuna', 'megumi', 'ilygojo');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (002, 'slicedjo', 'ihatejjk', 'jjkcsmb');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (003, 'mrg5l', 'onepieceot', 'ilyluffykun');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (004, 'NotIntoJujutsu', 'OnePieceSuperfan', 'ilyluffykun');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (005, 'JJKIndifference', 'PirateKingLover', 'ilyluffykun');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (006, 'UninterestedInJJK', 'StrawHatForever', 'ilyluffykun');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (007, 'JJKPass', 'LuffyFanatic', 'ilyluffykun');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (008, 'JJKOUT', 'OPAdventureSeeker', 'ilyluffykun');

-- Insert data into items table

INSERT INTO items(item\_id, item\_name, item\_description, item\_minBid, item\_startDate, item\_endDate, item\_buyPrice)

VALUES (001, 'toiletPaper', 'soft', 2.00, '2023-09-27', '2023-10-11', 2000.00);

INSERT INTO items(item\_id, item\_name, item\_description, item\_minBid, item\_startDate, item\_endDate, item\_buyPrice)

VALUES (002, 'DIO', 'Hard', 69.00, '2023-09-27', '2023-09-29', 20.00);

INSERT INTO items(item\_id, item\_name, item\_description, item\_minBid, item\_startDate, item\_endDate, item\_buyPrice)

VALUES (003, 'Bar', 'spicy', 69.00, '2023-09-27', '2023-09-30', 20.00);

INSERT INTO items(item\_id, item\_name, item\_description, item\_minBid, item\_startDate, item\_endDate, item\_buyPrice)

VALUES (004, 'PURPLe', 'scam', 69.00, '2023-09-27', '2023-10-11', 5000.00);

-- Insert data into the bids table

INSERT INTO bids(item\_id, user\_id, bids\_top)

VALUES (001, 0001, 2000.00);

INSERT INTO bids(item\_id, user\_id, bids\_top)

VALUES (001, 0002, 2500.00);

INSERT INTO bids(item\_id, user\_id, bids\_top)

VALUES (001, 0003, 3500.00);

INSERT INTO bids(item\_id, user\_id, bids\_top)

VALUES (001, 0004, 5500.00);

-- Insert data into the shipping table

INSERT INTO shipping(user\_id, shipping\_receiver, shipping\_tracking)

VALUES (1, '6969-UniversityAve', '92453');

INSERT INTO shipping(user\_id, shipping\_receiver, shipping\_tracking)

VALUES (2, '937-UniversityAve', '95642');

INSERT INTO shipping(user\_id, shipping\_receiver, shipping\_tracking)

VALUES (3, '4201-UniversityAve', '91593');

INSERT INTO shipping(user\_id, shipping\_receiver, shipping\_tracking)

VALUES (4, '420-UniversityAve', '95192');

-- Insert data into the finance table with correct date format and quotes around address

INSERT INTO finance(user\_id, finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (1, 201321511231, '2025-09-02', '6969-UniversityAve', 554);

INSERT INTO finance(user\_id, finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (2, 201920034589, '2025-09-02', '937-UniversityAve', 657);

INSERT INTO finance(user\_id, finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (3, 201912498124, '2025-09-02', '937-UniversityAve', 212);

INSERT INTO finance(user\_id, finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (4, 201509572359, '2025-09-02', '937-UniversityAve', 537);

-- Insert data into the auctions

INSERT INTO auctions(user\_id, item\_id)

VALUES (0004, 001);

INSERT INTO auctions(user\_id, item\_id)

VALUES (0005, 002);

INSERT INTO auctions(user\_id, item\_id)

VALUES (0006, 003);

INSERT INTO auctions(user\_id, item\_id)

VALUES (0007, 004);

### **A04P1: Designing Simple Queries**

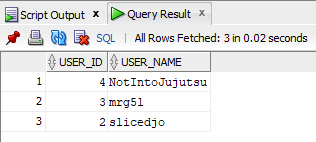
User

User Query: *List the usernames in ascending order with user IDs between 2 and 5, lowest to greatest.*

SELECT user\_id, user\_name FROM users

WHERE user\_id > 0002 AND user\_id < 0005

ORDER BY user\_name ASC;

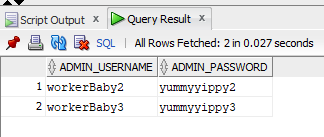


Admin

Admin Query: *List the admin usernames and their passwords that have an id number greater than two, lowest to greatest.*

SELECT admin\_username, admin\_password FROM admins

WHERE admin\_id > 2  
 ORDER BY admin\_id ASC;



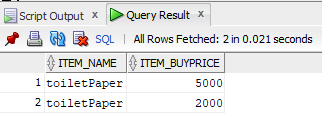
Item

Item Query: *List all the items with the same item name, sorted by greatest to least price*

SELECT item\_name, item\_buyPrice FROM items

WHERE item\_name = ‘toiletPaper’

ORDER BY item\_buyPrice DESC;

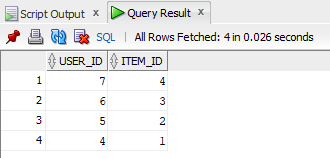


Auction

Auction Query: *List all unique items on auction by users between 0 and 9, by the item id from greatest to least.*

SELECT user\_id, item\_id FROM auctions

WHERE user\_id > 0 AND user\_id < 9  
 ORDER BY item\_id DESC;



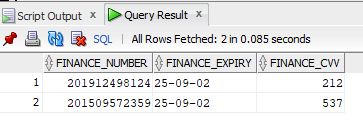
Finance

Finance Query: *List all finance numbers in descending order where user ID is greater than 2 with the corresponding expiration date and cvv*

SELECT finance\_number, finance\_expiry, finance\_cvv FROM finance

WHERE user\_id > 2

ORDER BY finance\_number DESC;

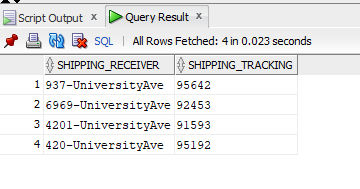


Shipping

Shipping Query: *List all auction recipients that have a tracking number between 0 and 90000 exclusive, from greatest to least*

SELECT shipping\_reciever, shipping\_tracking FROM shipping

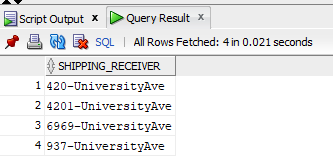
WHERE shipping\_tracking > 0 AND shipping\_tracking < 90000  
 ORDER BY shipping\_reciever DESC;



*List all unique buyers that have won an auction and are receiving an item, from least to greatest*

SELECT DISTINCT shipping\_receiver FROM shipping

ORDER BY shipping\_receiver ASC;



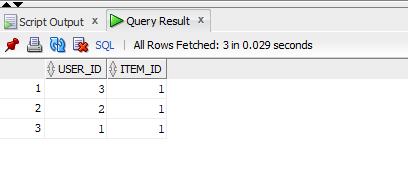
Bids

Bids Query: *List all top bids with the current highest bid with bids that are less than 5000$ and output them greatest to lowest by the value of the bid*

SELECT DISTINCT user\_id, item\_id FROM bids

WHERE bids\_top < 5000

ORDER BY bids\_top DESC;



### **A04P2: Designing Advanced Queries, Joins & Views**

#### Queries

1. *List all users who currently have an item being shipped from oldest to newest shipments(tracking numbers)*

SELECT u.user\_id, user\_name, s.shipping\_tracking

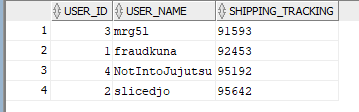
FROM shipping s, users u, sold i

WHERE s.shipping\_tracking > 0

AND s.user\_id = u.user\_id

AND s.shipping\_tracking= i.shipping\_tracking

ORDER BY shipping\_tracking ASC;



2. *List all of the auctions one user has put up*

SELECT a.user\_id, a.item\_id, i.item\_name

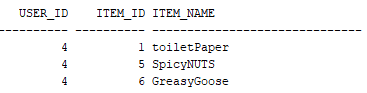
FROM auctions a, users u, items i

WHERE user\_id=4

AND a.user\_id = u.user\_id

AND a.item\_id = i.item\_id

ORDER BY item\_id ASC;



3. *List of items that have been bid on in the past 3 days, hence the most popular.*  
SELECT a.item\_id, a.auction\_latest, i.item\_endDate, b.bids\_top

FROM auctions a, bids b, items i

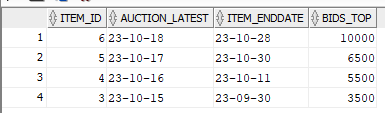
WHERE a.auction\_latest >= SYSDATE - 4

AND a.auction\_latest <= SYSDATE

AND a.item\_id = i.item\_id

AND b.item\_id = i.item\_id

ORDER BY a.auction\_latest DESC;



#### Views

1. *The total amount of bids a user has on an item*

CREATE VIEW BiddingActivity AS

(SELECT u.user\_id, COUNT(b.item\_id) AS total\_bids

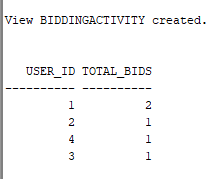
FROM pastbids b, users u

WHERE b.item\_id=1

AND u.user\_id = b.user\_id

GROUP BY u.user\_id

);



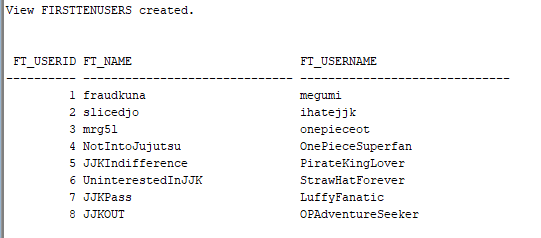
2. *Get the first ten user accounts created inclusive, and create a view with their id number, username, and name*  
CREATE VIEW FirstTenUsers (FT\_UserID, FT\_Name, FT\_Username) AS

(SELECT user\_id, user\_name, user\_username

FROM users

WHERE user\_id <= 10

);



3. *Get the users that have a financial card that expires within 10 days.*

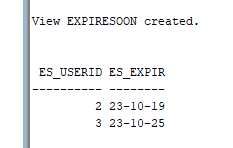
CREATE VIEW ExpireSoon (ES\_UserID, ES\_Expiry) AS

(SELECT user\_id, finance\_expiry

FROM finance

WHERE finance\_expiry > SYSDATE AND finance\_expiry <= SYSDATE + 10

);



### **A05: Advanced Queries by Unix Shell Implementation**

Query 1: List all usernames and passwords of users without names starting with “JJK”

(SELECT user\_username, user\_password

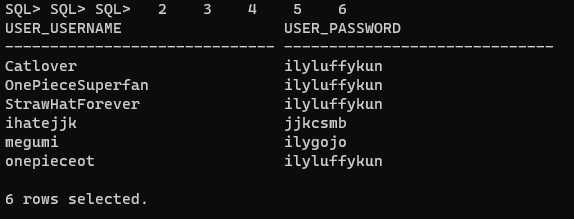
FROM users)

MINUS

(SELECT user\_username, user\_password

FROM users

WHERE user\_name LIKE 'JJK%');



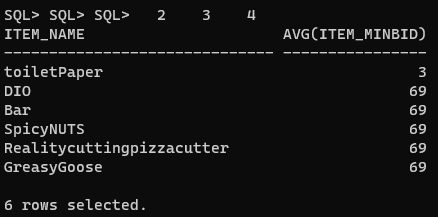
Query 2: List all item names and the average of their minimum bids, group by item names with averages less than 100

SELECT item\_name, AVG(item\_minBid)

FROM items

GROUP BY item\_name

HAVING AVG(item\_minBid) < 100;



Query 3: Check the existence of items with a bid over 5 dollars

SELECT i.item\_id, i.item\_name

FROM items i

WHERE EXISTS(

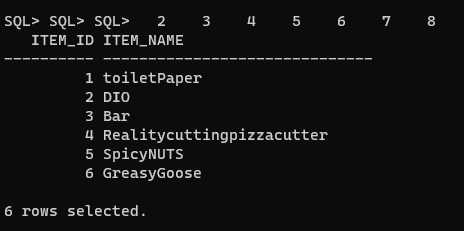
SELECT b.bids\_top

FROM bids b

WHERE b.bids\_top > 5

AND i.item\_id = b.item\_id

);



Query 4: List all item names and descriptions with an end date of '2023-10-11' and '2023-10-28'

SELECT item\_name, item\_description

FROM items

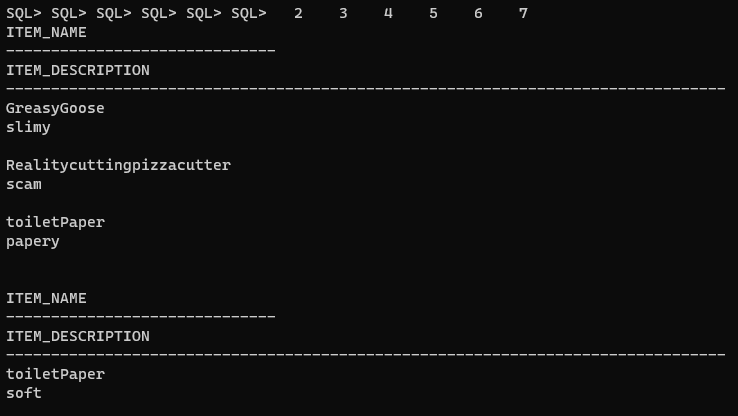
WHERE item\_endDate = '2023-10-11'

UNION

(SELECT item\_name, item\_description

FROM items

WHERE item\_endDate = '2023-10-28');



Query 5: The total number of bids a user has on an item

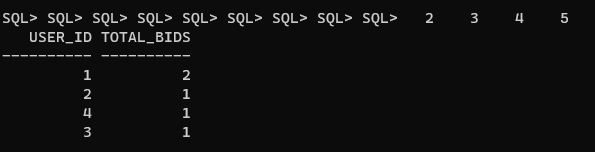
SELECT u.user\_id, COUNT(b.item\_id) AS total\_bids

FROM pastbids b, users u

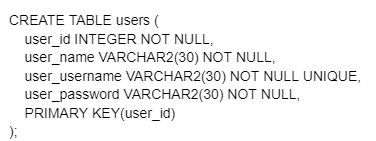
WHERE b.item\_id=1

AND u.user\_id = b.user\_id

GROUP BY u.user\_id;



### **A06: Functional Dependencies**



**user**(user\_id, user\_name, user\_username, user\_passsword)

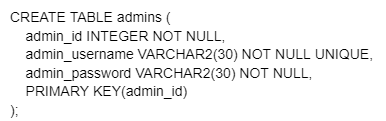
FD of user\_id = {

user\_id → user\_name,

user\_id → user\_username,

user\_id → user\_password

}



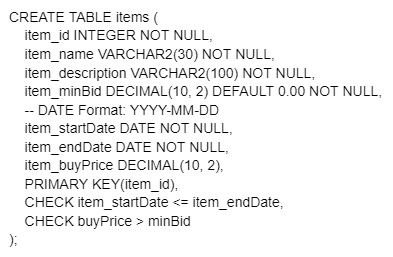
**admins**(admin\_id, admin\_username, admin\_password)

FD of admin\_id = {

admin\_id → admin\_username,

admin\_id → admin\_password

}



**items**(item\_id, item\_name, item\_description, item\_minBid, item\_startDate, item\_endDate, item\_buyPrice)

FD of item\_id = {

item\_id → item\_name,

item\_id → item\_description,

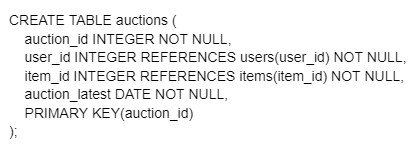
item\_id → item\_minBid,

item\_id → item\_buyPrice,

item\_id → item\_startDate,

item\_id → item\_endDate

}

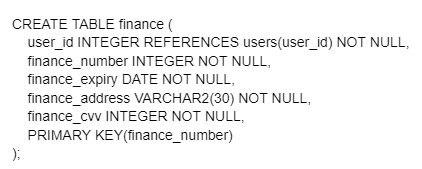


**auctions**(auction\_id, user\_id, item\_id, auction\_latest)

FD of auction\_id = {

auction\_id → auction\_latest

}



**finance**(finance\_number, user\_id finance\_expiry, finance\_address, finance\_cvv)

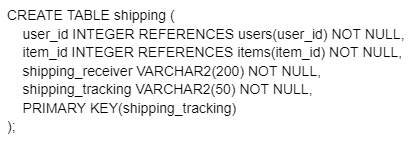
FD of finance\_number = {

finance\_number → finance\_expiry,

finance\_number → finance\_address,

finance\_number → finance\_cvv

}

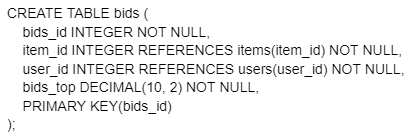


**shipping**(user\_id, shipping\_receiver, shipping\_tracking)

FD of shipping\_tracking = {

shipping\_tracking → shipping\_receiver

}

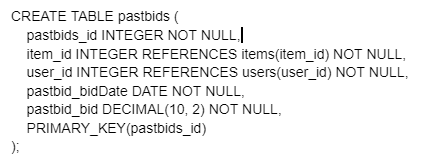


**bids**(bid\_id, item\_id, user\_id, bids\_top)

FD of bid\_id = {

bids\_id → bids\_top

}

****

**pastbids**(pastbids\_id, item\_id, user\_id, pastbid\_bidDate, pastbid\_bid)

FD of pastbids\_id = {

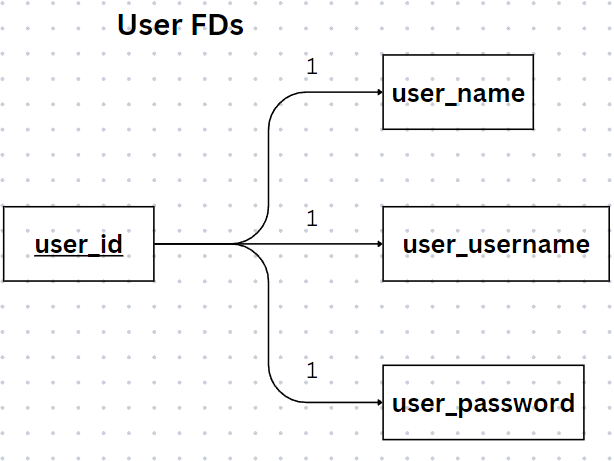
pastbids\_id → pastbid\_bidDate,

pastbids\_id → pastbid\_bid,

}

### **A07: Normalization to 3NF**

**User**(user\_id, user\_name, user\_username, user\_passsword)

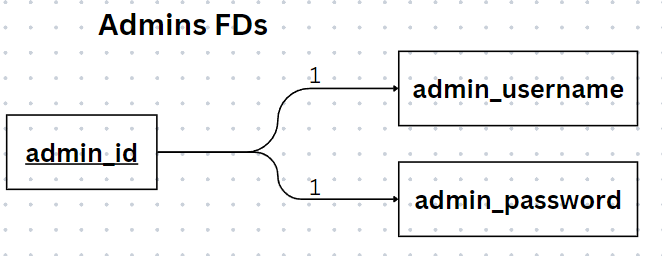


FD = {

1. user\_id → user\_name, user\_username, user\_password

}

**Admins**(admin\_id, admin\_username, admin\_password)

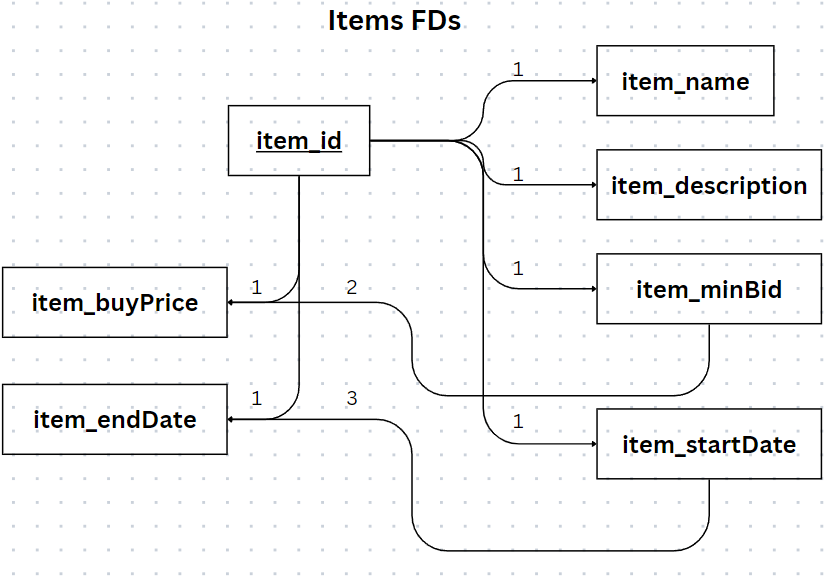


FD = {

1. admin\_id → admin\_username, admin\_password

}

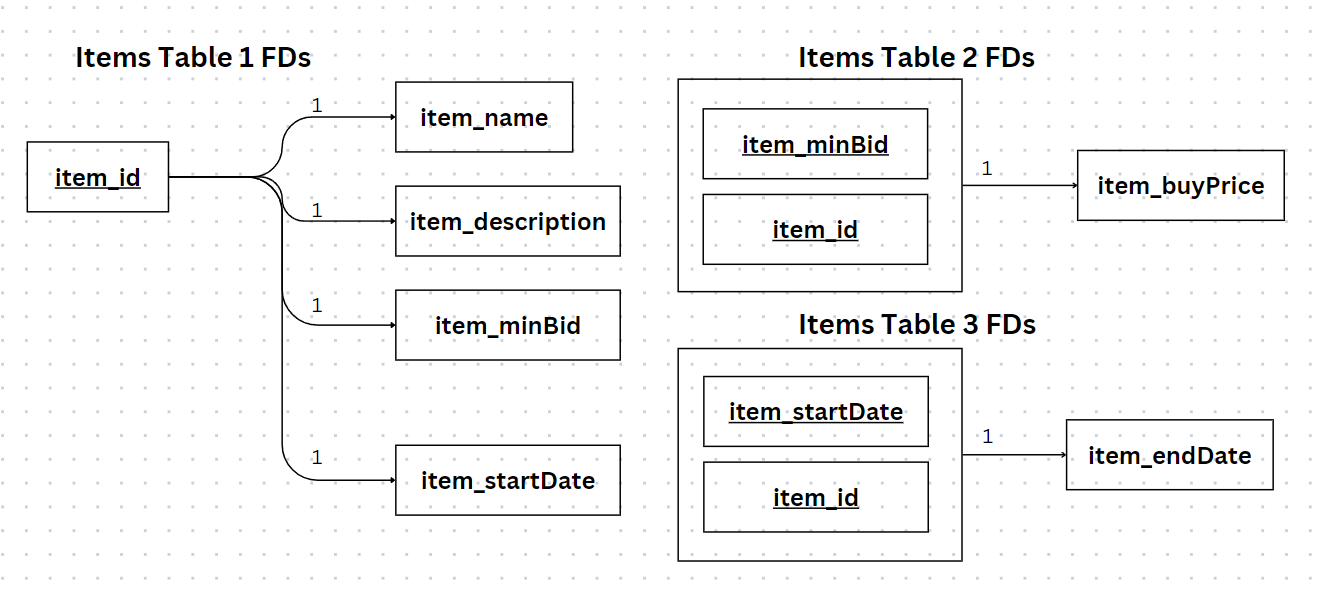
**Items**(item\_id, item\_name, item\_description, item\_minBid, item\_startDate, item\_endDate, item\_buyPrice)



FD = {

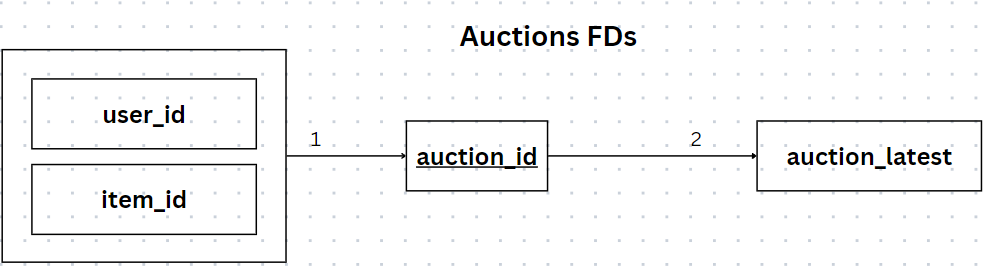
1. item\_id → item\_name, item\_description, item\_minBid, item\_buyPrice, item\_startDate, item\_endDate
2. item\_minBid → item\_buyPrice
3. item\_startDate → item\_endDate

}



**Auctions**(auction\_id, user\_id, item\_id, auction\_latest)

1NF and Transitive and Compound



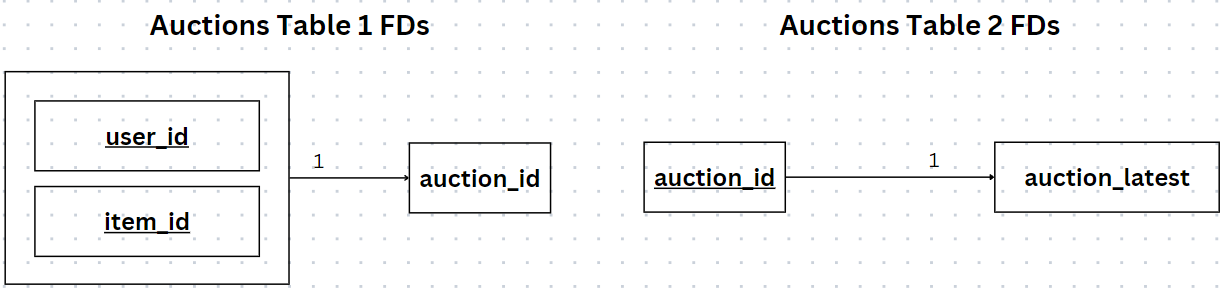
FD = {

1. user\_id, item\_id → auction\_id
2. auction\_id → auction\_latest

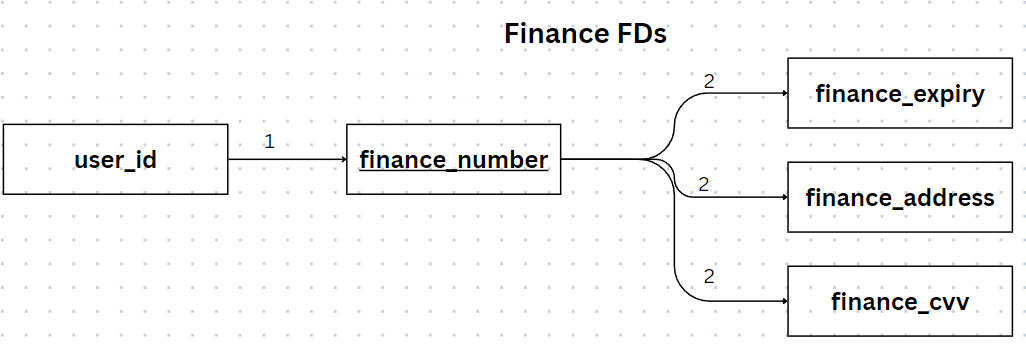
}

Decomposition from Compound FD -> 2NF

Decomposition from Transitive FD -> 3NF



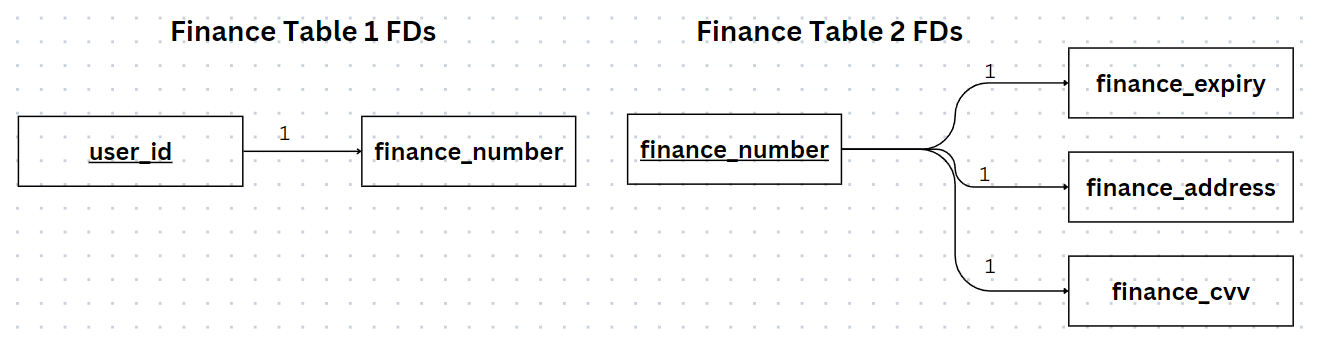
**Finance**(finance\_number, user\_id, finance\_expiry, finance\_address, finance\_cvv)



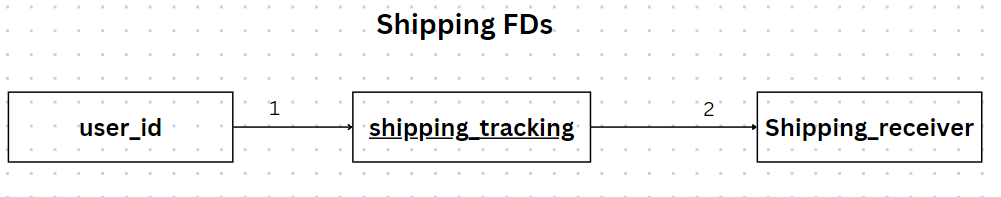
FD = {

1. user\_id → finance\_number
2. finance\_number → finance\_expiry, finance\_address, finance\_cvv

}



**Shipping**(user\_id, shipping\_receiver, shipping\_tracking)



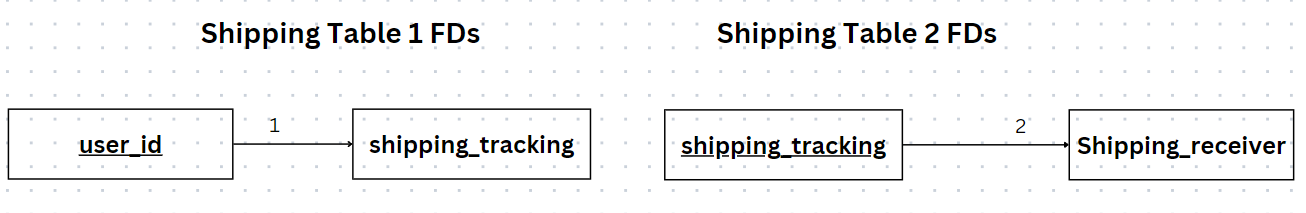
FD = {

1. user\_id → shipping\_tracking
2. shipping\_tracking → shipping\_receiver

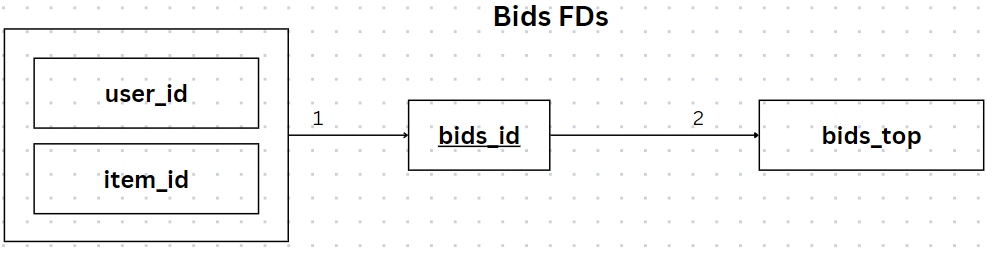
}

Decomposition from Compound FD -> 2NF

Decomposition from Transitive FD -> 3NF



**Bids**(bid\_id, item\_id, user\_id, bids\_top)



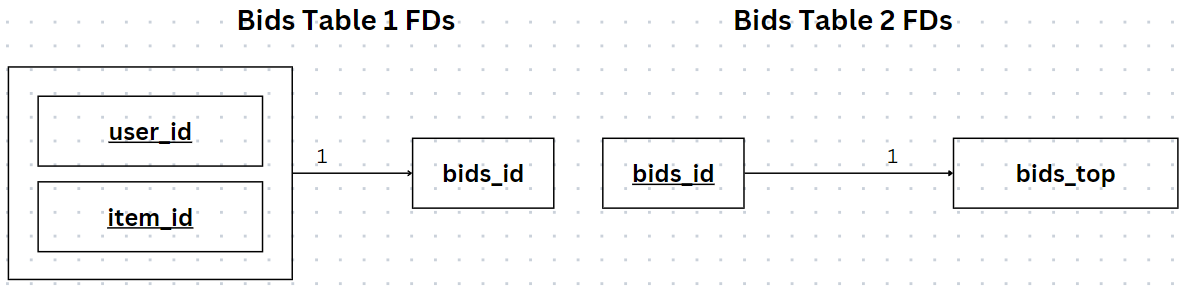
FD = {

1. user\_id, item\_id → bids\_id
2. bids\_id → bids\_top

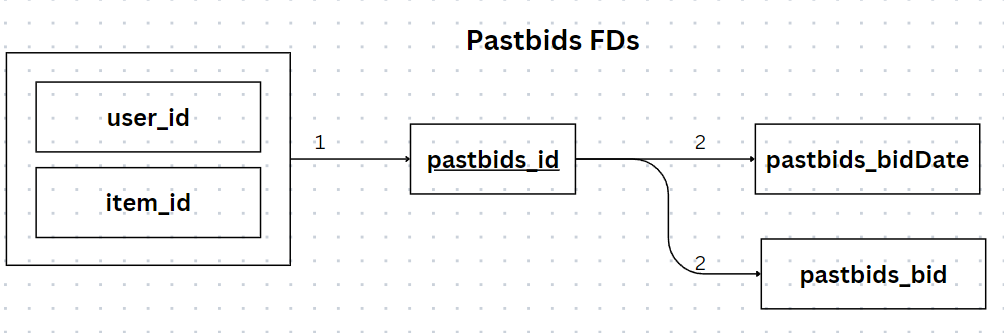
}

Decomposition from Compound FD -> 2NF

Decomposition from Transitive FD -> 3NF



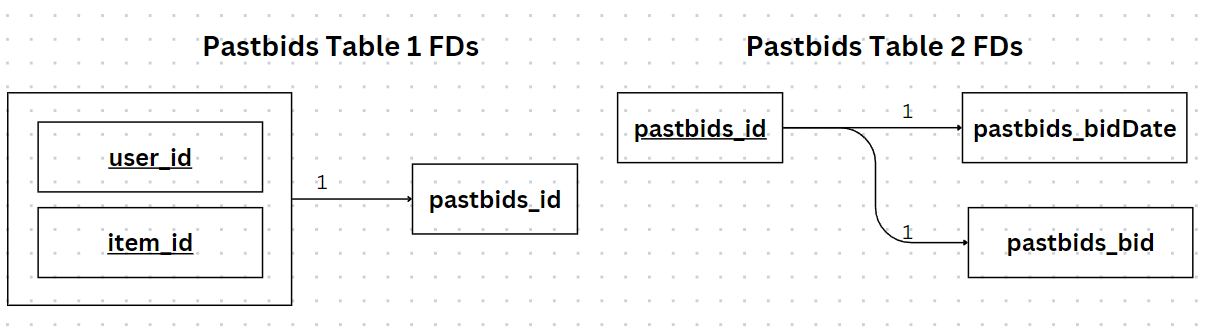
**pastbids**(pastbids\_id, item\_id, user\_id, pastbid\_bidDate, pastbid\_bid)

****

FD = {

1. user\_id, item\_id → pastbids\_id
2. pastbids\_id → pastbids\_bidDate, pastbids\_bid

}



### **A08: Normalization 3NF/BCNF by Algorithm**

**User Table:**

FDs = {

user\_id → user\_name, user\_username, user\_password

}

*Analysis:* No transitive or partial dependencies. This table is already in 3NF and BCNF.

**Admins Table:**

FDs = {

admin\_id → admin\_username, admin\_password

}

*Analysis:* No transitive or partial dependencies. This table is in 3NF and BCNF.

**Items Table:**

FDs = {

item\_id → item\_name, item\_description, item\_minBid, item\_buyPrice, item\_startDate, item\_endDate

item\_minBid → item\_buyPrice

item\_startDate → item\_endDate

}

*Analysis:* Contains transitive dependencies (item\_minBid → item\_buyPrice and item\_startDate → item\_endDate).

*Normalization:*

3NF Decomposition:

Create a new table ItemPricing(item\_minBid, item\_buyPrice) to remove the transitive dependency.

Create a new table ItemDates(item\_startDate, item\_endDate) to remove the other transitive dependency.

The remaining Items table has item\_id as the primary key.

*Resulting Tables:*

Items(item\_id, item\_name, item\_description, item\_minBid, item\_startDate)

ItemPricing(item\_id, item\_minBid, item\_buyPrice)

ItemDates(item\_id, item\_startDate, item\_endDate)

*BCNF Analysis:* Each table has all its non-key attributes fully functionally dependent on the primary key.

**Auctions Table:**

FDs = {

user\_id, item\_id → auction\_id

auction\_id → auction\_latest

}

*Analysis:* Contains a compound FD and a transitive dependency.

*Normalization:*

3NF Decomposition:

Create a new table AuctionStatus(auction\_id, auction\_latest) to remove the transitive dependency.

The remaining Auctions table has user\_id and item\_id as the compound primary key.

*Resulting Tables:*

Auctions(user\_id, item\_id, auction\_id)

AuctionStatus(auction\_id, auction\_latest)

*BCNF Analysis:* Each table has all its non-key attributes fully functionally dependent on the primary key.

**Finance Table:**

FDs = {

user\_id → finance\_number

finance\_number → finance\_expiry, finance\_address, finance\_cvv

}

*Analysis:* Transitive dependency present (finance\_number → finance\_expiry, finance\_address, finance\_cvv).

*Normalization:*

3NF Decomposition:

Create a new table FinanceDetails(finance\_number, finance\_expiry, finance\_address, finance\_cvv) to remove the transitive dependency.

The remaining Finance table has user\_id as the primary key.

*Resulting Tables:*

FinanceUser(user\_id, finance\_number)

FinanceDetails(finance\_number, finance\_expiry, finance\_address, finance\_cvv)

*BCNF Analysis:* Each table has all its non-key attributes fully functionally dependent on the primary key.

**Shipping Table:**

FDs = {

user\_id → shipping\_tracking

shipping\_tracking → shipping\_receiver

Analysis: Transitive dependency present.

}

*Normalization:*

3NF Decomposition:

Create a new table ShippingDetails(shipping\_tracking, shipping\_receiver) to remove the transitive dependency.

The remaining Shipping table has user\_id as the primary key.

*Resulting Tables:*

ShippingUser(user\_id, shipping\_tracking)

ShippingDetails(shipping\_tracking, shipping\_receiver)

*BCNF Analysis:* Each table has all its non-key attributes fully functionally dependent on the primary key.

**Bids Table:**

FDs = {

user\_id, item\_id → bids\_id

bids\_id → bids\_top

}

*Analysis:* Contains a compound FD and a transitive dependency.

*Normalization:*

3NF Decomposition:

Create a new table BidStatus(bids\_id, bids\_top) to remove the transitive dependency.

The remaining Bids table has user\_id and item\_id as the compound primary key.

*Resulting Tables:*

BidDetails(user\_id, item\_id, bids\_id)

BidStatus(bids\_id, bids\_top)

*BCNF Analysis:* Each table has all its non-key attributes fully functionally dependent on the primary key.

**Past Bids Table:**

FDs = {

user\_id, item\_id → pastbids\_id

pastbids\_id → pastbids\_bidDate, pastbids\_bid

}

*Analysis:* Contains a compound FD and a transitive dependency.

*Normalization:*

3NF Decomposition:

Create a new table PastBidStatus(pastbids\_id, pastbids\_bidDate, pastbids\_bid) to remove the transitive dependency.

The remaining Past Bids table has user\_id and item\_id as the compound primary key.

*Resulting Tables:*

PastBidDetails(user\_id, item\_id, pastbids\_id)

PastBidStatus(pastbids\_id, pastbids\_bidDate, pastbids\_bid)

*BCNF Analysis:* Each table has all its non-key attributes fully functionally dependent on the primary key.

**Summary:**

In summary, each table has been normalized to 3NF by removing transitive dependencies through decomposition. The resulting tables are also in BCNF, as every determinant is a candidate key, and there are no non-key attributes determining other non-key attributes.

### **A09: Demo by Unix Shell of 3NF/BCNF Tables**

-- Drop all tables

DROP TABLE users CASCADE CONSTRAINTS;

DROP TABLE admins CASCADE CONSTRAINTS;

DROP TABLE items CASCADE CONSTRAINTS;

DROP TABLE itemPricing CASCADE CONSTRAINTS;

DROP TABLE itemDates CASCADE CONSTRAINTS;

DROP TABLE auctions CASCADE CONSTRAINTS;

DROP TABLE auctionStatus CASCADE CONSTRAINTS;

DROP TABLE financeUser CASCADE CONSTRAINTS;

DROP TABLE financeDetails CASCADE CONSTRAINTS;

DROP TABLE shippingUser CASCADE CONSTRAINTS;

DROP TABLE shippingDetails CASCADE CONSTRAINTS;

DROP TABLE bidDetails CASCADE CONSTRAINTS;

DROP TABLE bidStatus CASCADE CONSTRAINTS;

DROP TABLE pastBidDetails CASCADE CONSTRAINTS;

DROP TABLE pastBidStatus CASCADE CONSTRAINTS;

-- Create users table

CREATE TABLE users (

user\_id INTEGER NOT NULL,

user\_name VARCHAR2(30) NOT NULL,

user\_username VARCHAR2(30) NOT NULL UNIQUE,

user\_password VARCHAR2(30) NOT NULL,

PRIMARY KEY(user\_id)

);

-- Create admins table

CREATE TABLE admins (

admin\_id INTEGER NOT NULL,

admin\_username VARCHAR2(30) NOT NULL UNIQUE,

admin\_password VARCHAR2(30) NOT NULL,

PRIMARY KEY(admin\_id)

);

-- Create items table

CREATE TABLE items (

item\_id INTEGER NOT NULL,

item\_name VARCHAR2(30) NOT NULL,

item\_description VARCHAR2(100) NOT NULL,

PRIMARY KEY(item\_id)

);

-- Create itemPricing table

CREATE TABLE itemPricing (

item\_id INTEGER REFERENCES items(item\_id),

item\_minBid DECIMAL (10, 2) DEFAULT 0.00 NOT NULL,

item\_buyPrice DECIMAL(10, 2),

PRIMARY KEY(item\_id, item\_minBid),

CHECK (item\_buyPrice > item\_minBid)

);

-- Create itemDates table

CREATE TABLE itemDates (

-- DATE format: YYYY-MM-DD

item\_id INTEGER REFERENCES items(item\_id),

item\_startDate DATE NOT NULL,

item\_endDate DATE NOT NULL,

PRIMARY KEY(item\_id, item\_startDate),

CHECK (item\_startDate <= item\_endDate)

);

-- Create auctions table

CREATE TABLE auctions (

auction\_id INTEGER NOT NULL UNIQUE,

user\_id INTEGER REFERENCES users(user\_id) NOT NULL,

item\_id INTEGER REFERENCES items(item\_id) NOT NULL,

PRIMARY KEY(user\_id, item\_id)

);

-- Create auctionStatus table

CREATE TABLE auctionsStatus (

auction\_id INTEGER REFERENCES auctions(auction\_id),

auction\_latest DATE NOT NULL,

PRIMARY KEY(auction\_id)

);

-- Create financeUser table

CREATE TABLE financeUser (

user\_id INTEGER REFERENCES users(user\_id),

finance\_number NUMBER(16) NOT NULL UNIQUE,

PRIMARY KEY(user\_id)

);

-- Create financeDetails table

CREATE TABLE financeDetails (

finance\_number NUMBER(16) REFERENCES financeUser(finance\_number),

finance\_expiry DATE NOT NULL,

finance\_address VARCHAR2(30) NOT NULL,

finance\_cvv INTEGER NOT NULL,

PRIMARY KEY(finance\_number)

);

-- Create shippingUser table

CREATE TABLE shippingUser (

user\_id INTEGER REFERENCES users(user\_id),

shipping\_tracking VARCHAR2(50) NOT NULL UNIQUE,

PRIMARY KEY(user\_id)

);

-- Create shippingDetails table

CREATE TABLE shippingDetails (

shipping\_tracking VARCHAR2(50) REFERENCES shippingUser(shipping\_tracking),

shipping\_receiver VARCHAR2(200) NOT NULL,

PRIMARY KEY(shipping\_tracking)

);

-- Create bidDetails table

CREATE TABLE bidDetails (

item\_id INTEGER REFERENCES items(item\_id),

user\_id INTEGER REFERENCES users(user\_id),

bids\_id INTEGER NOT NULL UNIQUE,

PRIMARY KEY(item\_id, user\_id)

);

-- Create bidStatus table

CREATE TABLE bidStatus (

bids\_id INTEGER REFERENCES bidDetails(bids\_id),

bids\_top DECIMAL(10, 2) NOT NULL,

PRIMARY KEY(bids\_id)

);

-- Create pastBidDetails table

CREATE TABLE pastBidDetails (

item\_id INTEGER REFERENCES items(item\_id),

user\_id INTEGER REFERENCES users(user\_id),

pastbids\_id INTEGER NOT NULL UNIQUE,

PRIMARY KEY(item\_id, user\_id)

);

-- Create pastBidStatus table

CREATE TABLE pastBidStatus (

pastbids\_id INTEGER REFERENCES pastBidDetails(pastbids\_id),

pastbid\_bidDate DATE NOT NULL,

pastbid\_bid DECIMAL(10, 2) NOT NULL,

PRIMARY KEY(pastbids\_id)

);

**-- INSERT STATEMENTS**

-- Insert data into users table

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (1, 'Zoravar Multani', 'gogurtlover15', 'ilovegogurt');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (2, 'Justin Lee', 'zesuckermol3r4t', 'iglazezora');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (3, 'Karen Loac', 'hater146', 'ihatejeffery');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (4, 'Jefferey Wong', 'pumpkineater', 'pantsonfire');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (5, 'Joaquin Kataoka', 'jhunglover', 'igrewmybeard');

INSERT INTO users(user\_id, user\_name, user\_username, user\_password)

VALUES (6, 'Justin Hung', 'keijilover', 'ipermmyhair');

-- Insert data into admins table

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (1, 'admin1', 'adminpassword1');

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (2, 'admin2', 'adminpassword2');

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (3, 'admin3', 'adminpassword2');

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (4, 'admin4', 'adminpassword2');

INSERT INTO admins(admin\_id, admin\_username, admin\_password)

VALUES (5, 'admin5', 'adminpassword2');

-- Insert data into items table

INSERT INTO items(item\_id, item\_name, item\_description)

VALUES (1, 'Item 1','Description for Item 1');

INSERT INTO items(item\_id, item\_name, item\_description)

VALUES (2, 'Item 2', 'Description for Item 2');

INSERT INTO items(item\_id, item\_name, item\_description)

VALUES (3, 'Item 3', 'Description for Item 3');

INSERT INTO items(item\_id, item\_name, item\_description)

VALUES (4, 'Item 4', 'Description for Item 4');

INSERT INTO items(item\_id, item\_name, item\_description)

VALUES (5, 'Item 5', 'Description for Item 5');

-- Insert data into itemPricing table

INSERT INTO itemPricing(item\_id, item\_minBid, item\_buyPrice)

VALUES (1, 10.00, 20.00);

INSERT INTO itemPricing(item\_id, item\_minBid, item\_buyPrice)

VALUES (2, 15.00, 30.00);

INSERT INTO itemPricing(item\_id, item\_minBid, item\_buyPrice)

VALUES (3, 10.00, 30.00);

INSERT INTO itemPricing(item\_id, item\_minBid, item\_buyPrice)

VALUES (4, 15.00, 35.00);

INSERT INTO itemPricing(item\_id, item\_minBid, item\_buyPrice)

VALUES (5, 20.00, 40.00);

-- Insert data into itemDates table

INSERT INTO itemDates(item\_id, item\_startDate, item\_endDate)

VALUES (1, '2023-11-01', '2023-11-10');

INSERT INTO itemDates(item\_id, item\_startDate, item\_endDate)

VALUES (2, '2023-11-05', '2023-11-15');

INSERT INTO itemDates(item\_id, item\_startDate, item\_endDate)

VALUES (3, '2023-12-01', '2023-12-10');

INSERT INTO itemDates(item\_id, item\_startDate, item\_endDate)

VALUES (4, '2023-12-05', '2023-12-15');

INSERT INTO itemDates(item\_id, item\_startDate, item\_endDate)

VALUES (5, '2023-12-10', '2023-12-20');

-- Insert data into auctions table

INSERT INTO auctions(auction\_id, user\_id, item\_id)

VALUES (1, 1, 1);

INSERT INTO auctions(auction\_id, user\_id, item\_id)

VALUES (2, 2, 2);

INSERT INTO auctions(auction\_id, user\_id, item\_id)

VALUES (3, 3, 3);

INSERT INTO auctions(auction\_id, user\_id, item\_id)

VALUES (4, 4, 4);

INSERT INTO auctions(auction\_id, user\_id, item\_id)

VALUES (5, 5, 5);

-- Insert data into auctionStatus table

INSERT INTO auctionStatus(auction\_id, auction\_latest)

VALUES (1, '2023-11-08');

INSERT INTO auctionStatus(auction\_id, auction\_latest)

VALUES (2, '2023-11-12');

INSERT INTO auctionStatus(auction\_id, auction\_latest)

VALUES (3, '2023-12-08');

INSERT INTO auctionStatus(auction\_id, auction\_latest)

VALUES (4, '2023-12-12');

INSERT INTO auctionStatus(auction\_id, auction\_latest)

VALUES (5, '2023-12-16');

-- Insert data into financeUser table

INSERT INTO financeUser(user\_id, finance\_number)

VALUES (1, 1234567890123456);

INSERT INTO financeUser(user\_id, finance\_number)

VALUES (2, 9876543210987654);

INSERT INTO financeUser(user\_id, finance\_number)

VALUES (3, 1111222233334444);

INSERT INTO financeUser(user\_id, finance\_number)

VALUES (4, 5555666677778888);

INSERT INTO financeUser(user\_id, finance\_number)

VALUES (5, 9999888877775555);

-- Insert data into financeDetails table

INSERT INTO financeDetails(finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (1234567890123456, '2024-12-31', '1735 Danforth Ave', 123);

INSERT INTO financeDetails(finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (9876543210987654, '2025-06-30', '123 Yonge St', 456);

INSERT INTO financeDetails(finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (1111222233334444, '2024-12-31', '710 King St W', 456);

INSERT INTO financeDetails(finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (5555666677778888, '2025-06-30', '344 Bathurst St', 789);

INSERT INTO financeDetails(finance\_number, finance\_expiry, finance\_address, finance\_cvv)

VALUES (9999888877775555, '2026-01-15', '552 Yonge St', 123);

-- Insert data into shippingUser table

INSERT INTO shippingUser(user\_id, shipping\_tracking)

VALUES (1, 'SHIPPING123');

INSERT INTO shippingUser(user\_id, shipping\_tracking)

VALUES (2, 'SHIPPING456');

INSERT INTO shippingUser(user\_id, shipping\_tracking)

VALUES (3, 'SHIPPING789');

INSERT INTO shippingUser(user\_id, shipping\_tracking)

VALUES (4, 'SHIPPING101');

INSERT INTO shippingUser(user\_id, shipping\_tracking)

VALUES (5, 'SHIPPING1234');

-- Insert data into shippingDetails table

INSERT INTO shippingDetails(shipping\_tracking, shipping\_receiver)

VALUES ('SHIPPING123', 'John Doe');

INSERT INTO shippingDetails(shipping\_tracking, shipping\_receiver)

VALUES ('SHIPPING456', 'Jane Smith');

INSERT INTO shippingDetails(shipping\_tracking, shipping\_receiver)

VALUES ('SHIPPING789', 'Alice Johnson');

INSERT INTO shippingDetails(shipping\_tracking, shipping\_receiver)

VALUES ('SHIPPING101', 'Bob Smith');

INSERT INTO shippingDetails(shipping\_tracking, shipping\_receiver)

VALUES ('SHIPPING1234', 'Eve Adams');

-- Insert data into bidDetails table

INSERT INTO bidDetails(item\_id, user\_id, bids\_id)

VALUES (1, 1, 1);

INSERT INTO bidDetails(item\_id, user\_id, bids\_id)

VALUES (2, 2, 2);

INSERT INTO bidDetails(item\_id, user\_id, bids\_id)

VALUES (3, 3, 6);

INSERT INTO bidDetails(item\_id, user\_id, bids\_id)

VALUES (4, 4, 7);

INSERT INTO bidDetails(item\_id, user\_id, bids\_id)

VALUES (5, 5, 8);

-- Insert data into bidStatus table

INSERT INTO bidStatus(bids\_id, bids\_top)

VALUES (1, 30.00);

INSERT INTO bidStatus(bids\_id, bids\_top)

VALUES (2, 40.00);

INSERT INTO bidStatus(bids\_id, bids\_top)

VALUES (6, 50.00);

INSERT INTO bidStatus(bids\_id, bids\_top)

VALUES (7, 60.00);

INSERT INTO bidStatus(bids\_id, bids\_top)

VALUES (8, 70.00);

-- Insert data into pastBidDetails table

INSERT INTO pastBidDetails(item\_id, user\_id, pastbids\_id)

VALUES (1, 1, 1);

INSERT INTO pastBidDetails(item\_id, user\_id, pastbids\_id)

VALUES (2, 2, 2);

INSERT INTO pastBidDetails(item\_id, user\_id, pastbids\_id)

VALUES (3, 3, 6);

INSERT INTO pastBidDetails(item\_id, user\_id, pastbids\_id)

VALUES (4, 4, 7);

INSERT INTO pastBidDetails(item\_id, user\_id, pastbids\_id)

VALUES (5, 5, 8);

-- Insert data into pastBidStatus table

INSERT INTO pastBidStatus(pastbids\_id, pastbid\_bidDate, pastbid\_bid)

VALUES (1, '2023-11-08', 25.00);

INSERT INTO pastBidStatus(pastbids\_id, pastbid\_bidDate, pastbid\_bid)

VALUES (2, '2023-11-10', 35.00);

INSERT INTO pastBidStatus(pastbids\_id, pastbid\_bidDate, pastbid\_bid)

VALUES (6, '2023-12-05', 45.00);

INSERT INTO pastBidStatus(pastbids\_id, pastbid\_bidDate,en pastbid\_bid)

VALUES (7, '2023-12-10', 55.00);

INSERT INTO pastBidStatus(pastbids\_id, pastbid\_bidDate, pastbid\_bid)

VALUES (8, '2023-12-15', 65.00);

SELECT \* FROM users;

SELECT \* FROM admins;

SELECT \* FROM items;

SELECT \* FROM itemPricing;

SELECT \* FROM itemDates;

SELECT \* FROM auctions;

SELECT \* FROM auctionStatus;

SELECT \* FROM financeUser;

SELECT \* FROM financeDetails;

SELECT \* FROM shippingUser;

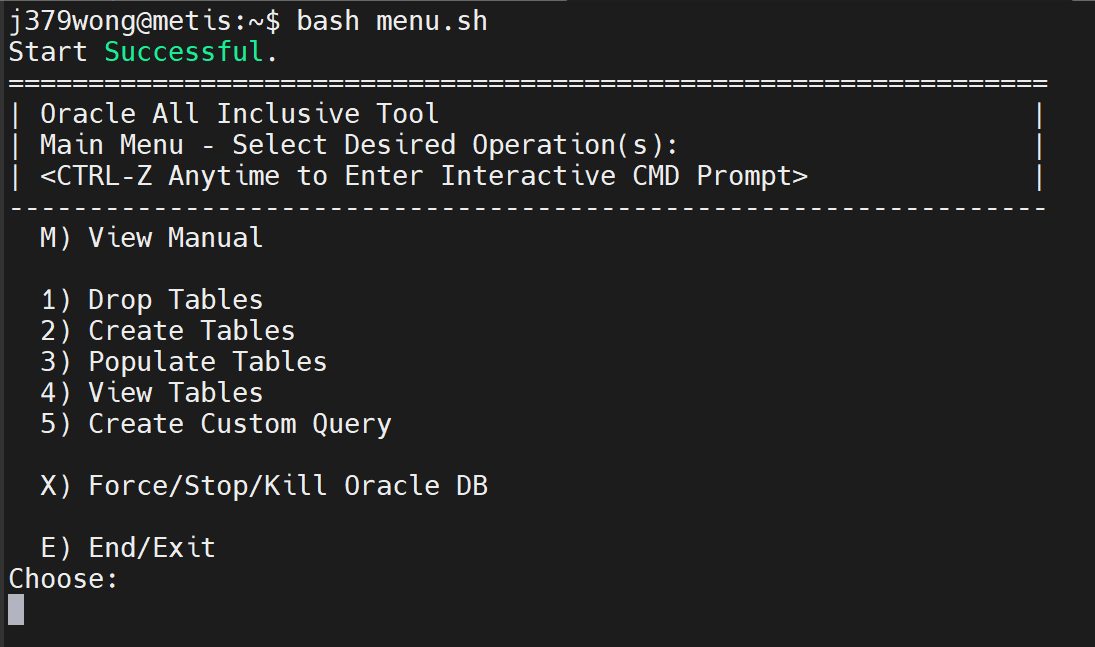
SELECT \* FROM shippingDetails;

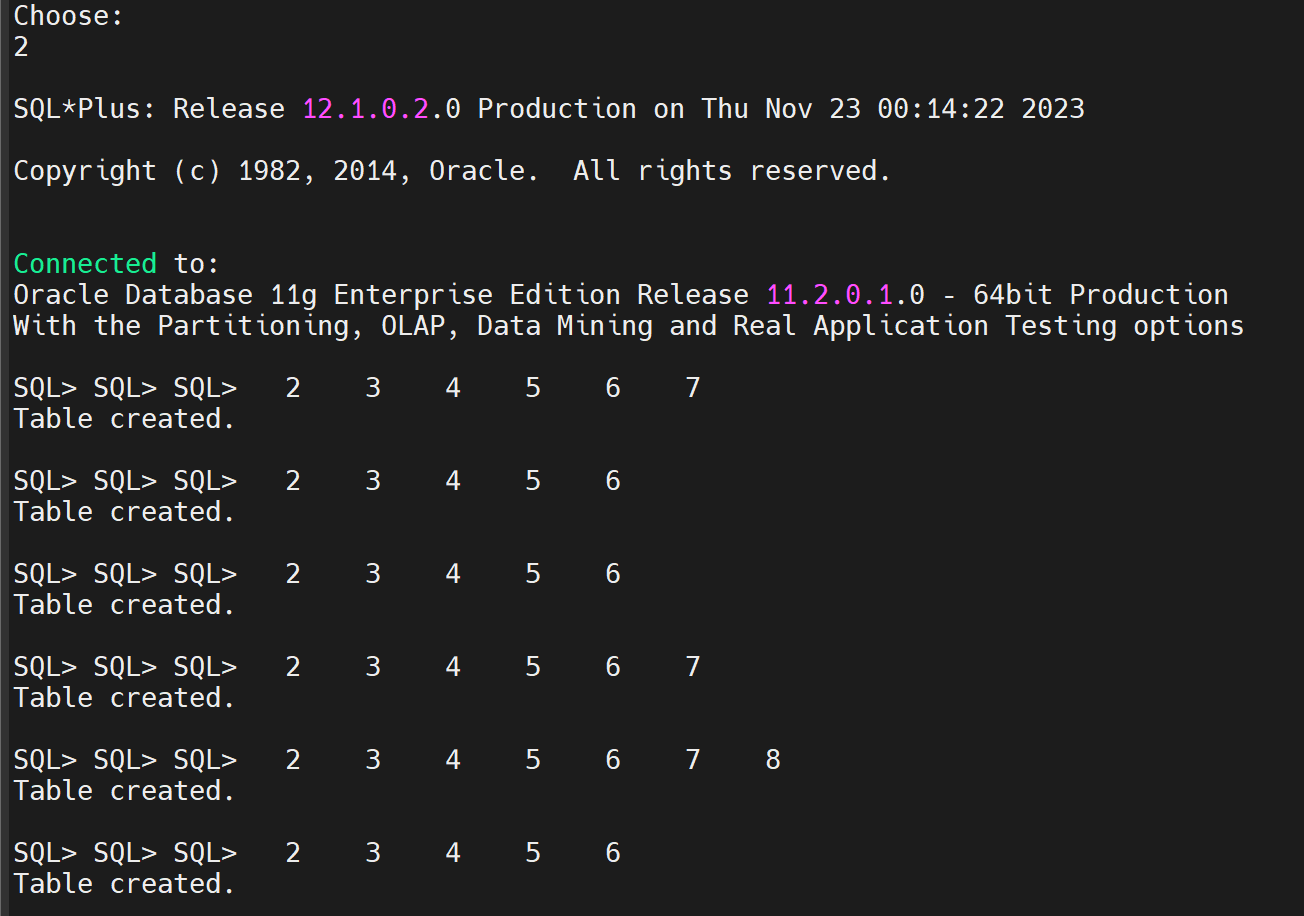
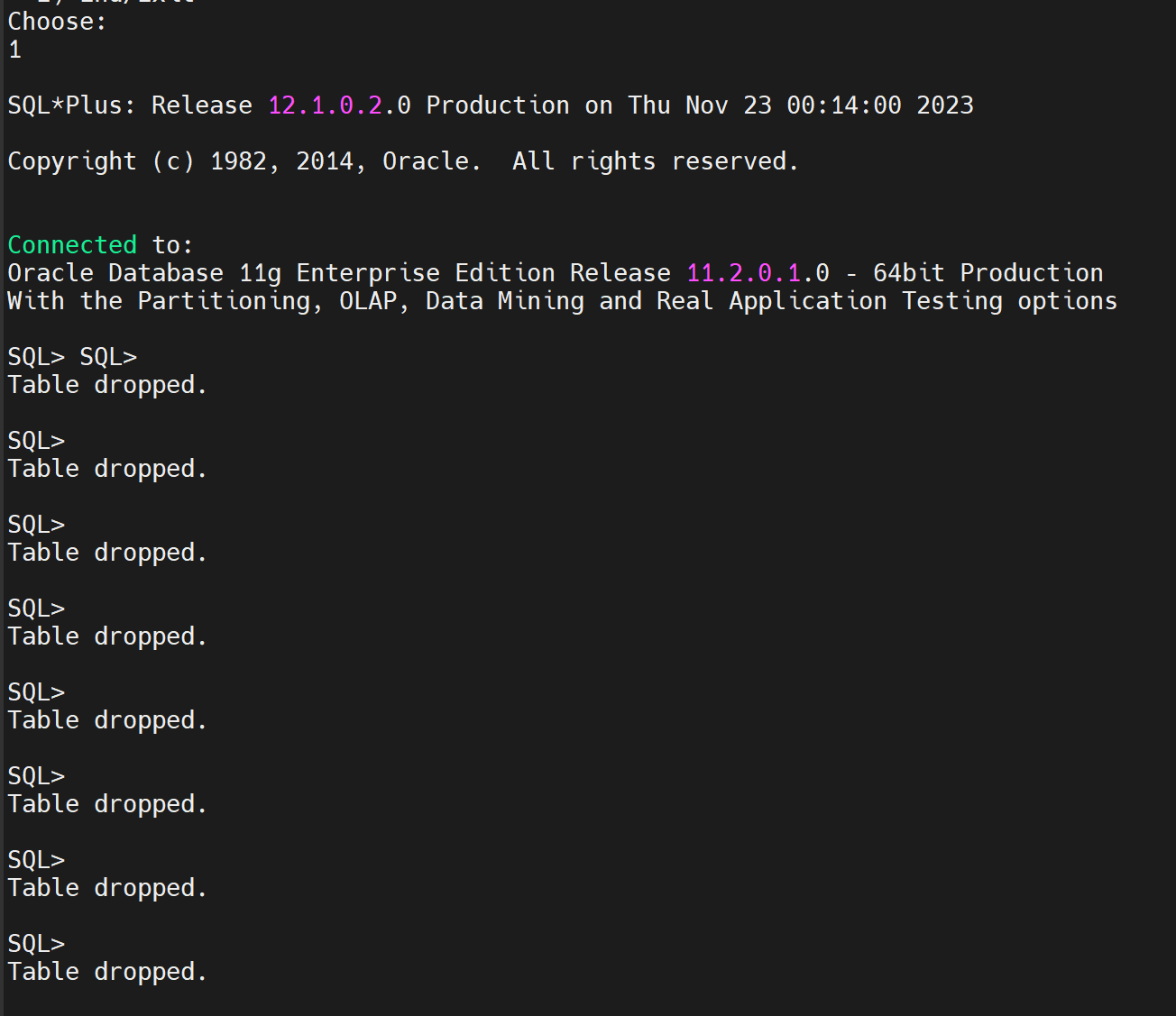
SELECT \* FROM bidDetails;

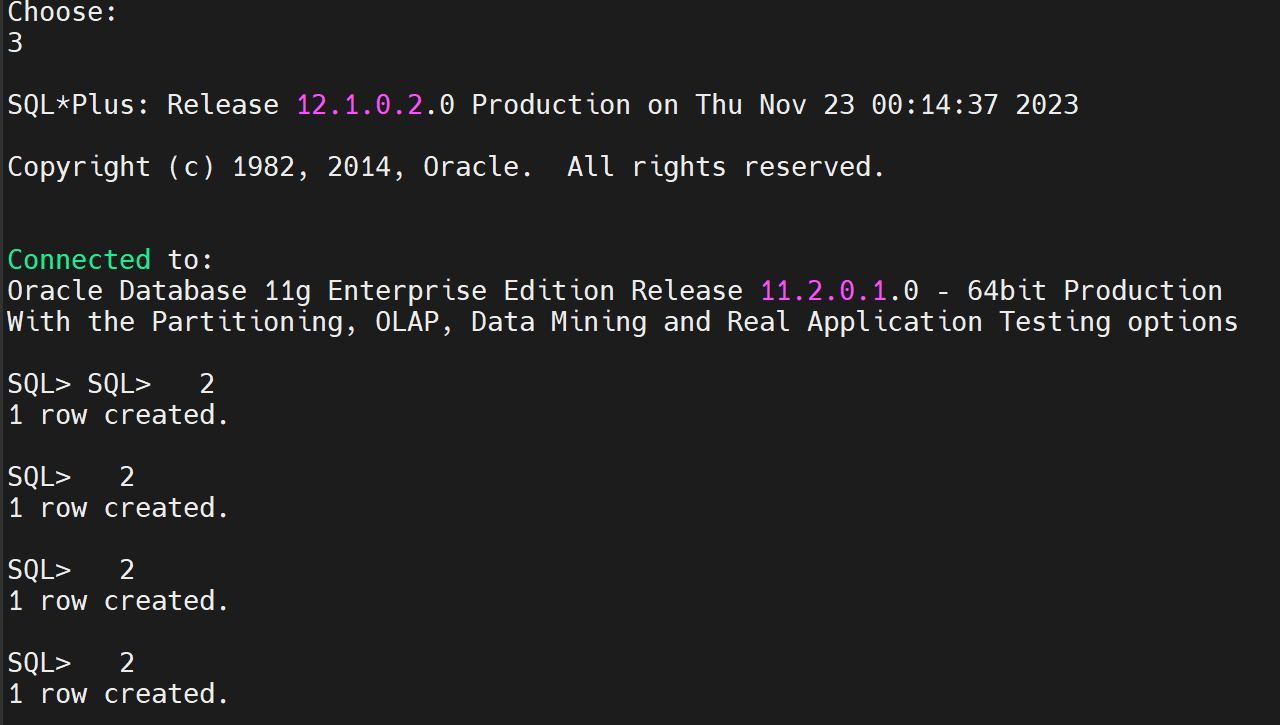
SELECT \* FROM bidStatus;

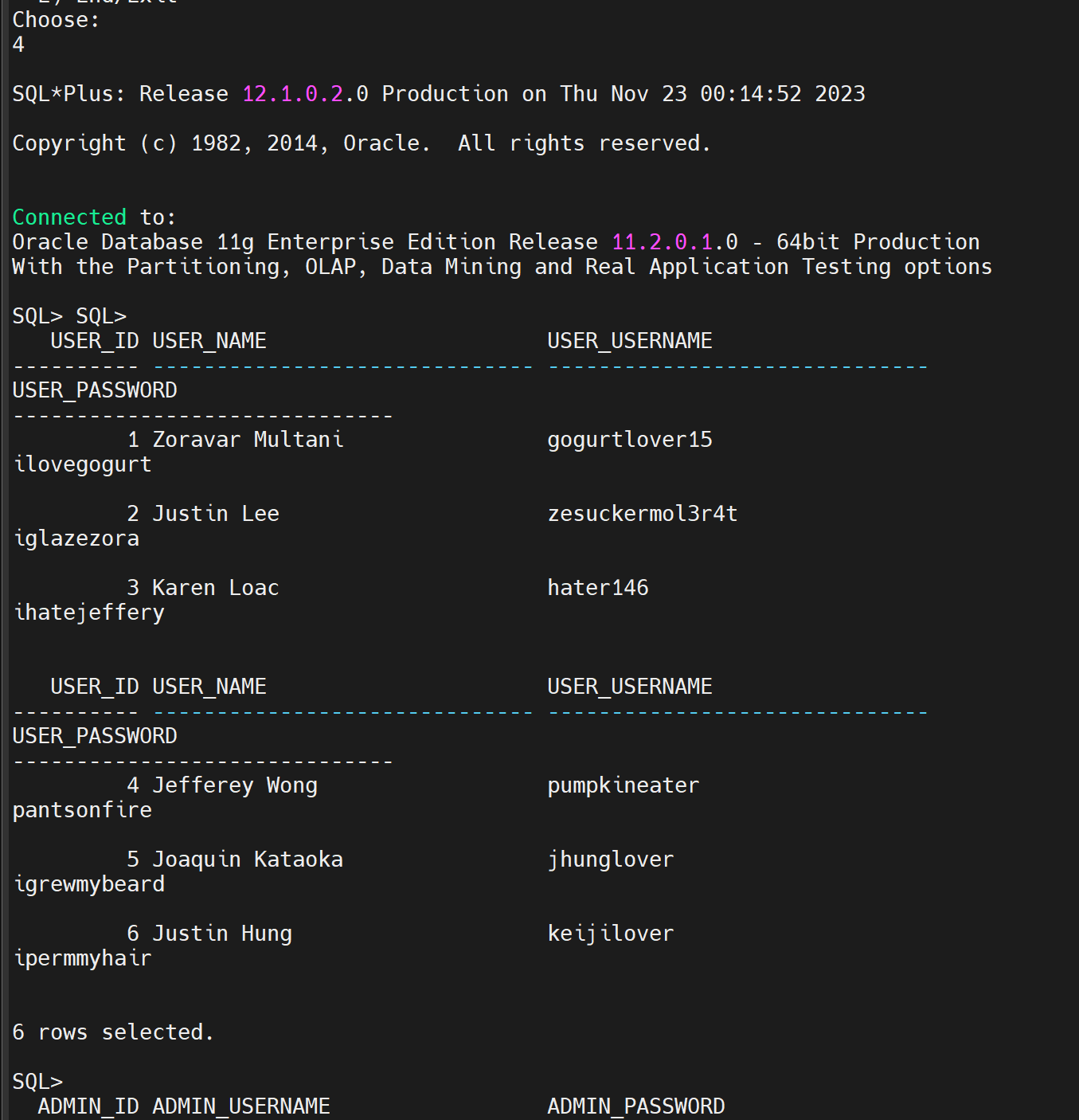
SELECT \* FROM pastBidDetails;

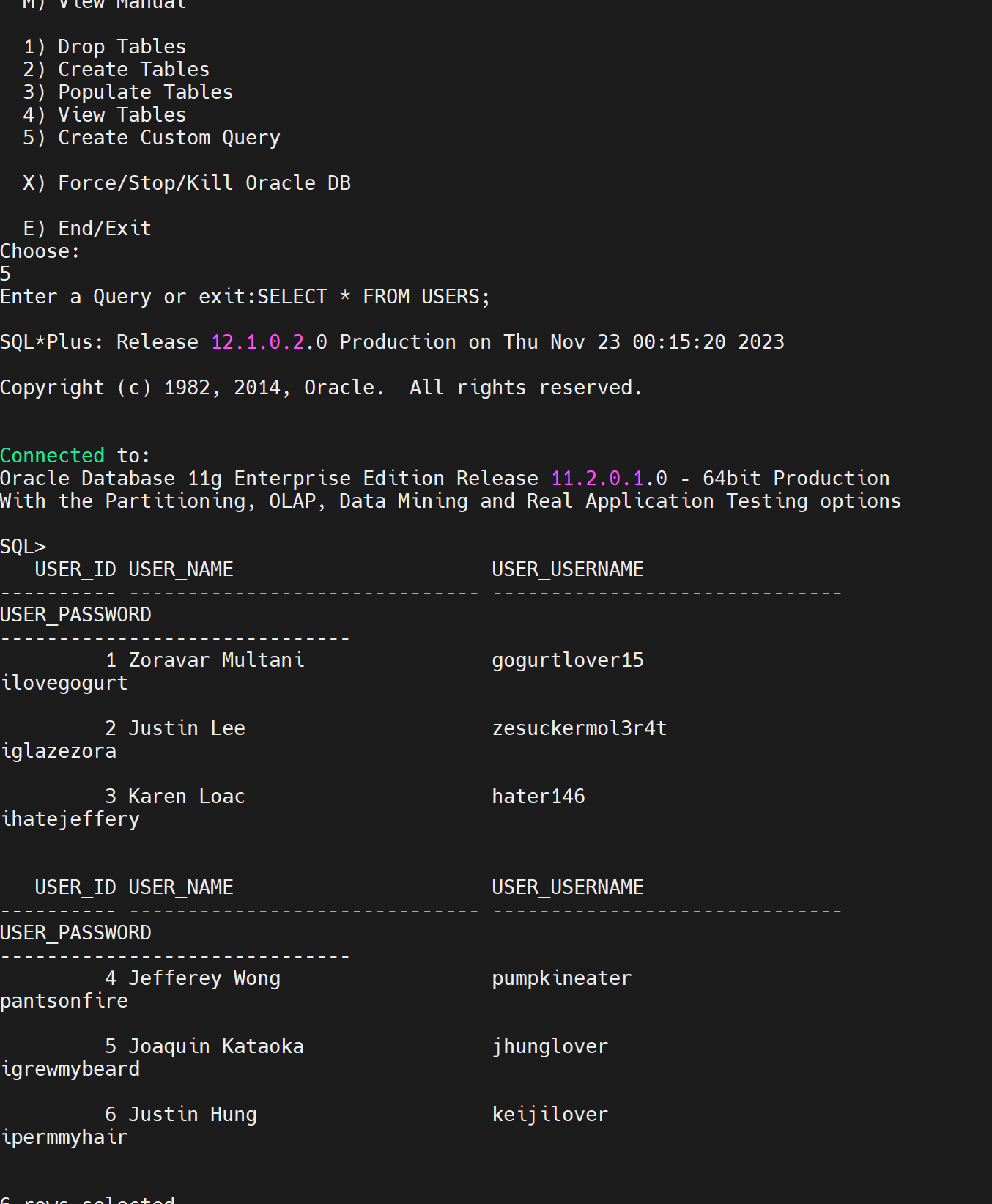
SELECT \* FROM pastBidStatus;











### **A10: Relational Algebra Statements**

QUERY 1:

SELECT \* from users →

It is the same for all select all statements for all tables in the DBMS.

QUERY 2:

SELECT user\_id from users →

QUERY 3:

(SELECT user\_username, user\_password

FROM users)

MINUS

(SELECT user\_username, user\_password

FROM users

WHERE user\_name LIKE 'JJK%'); →

QUERY 4:

SELECT item\_name, item\_description

FROM items

WHERE item\_endDate = '2023-10-11'

UNION

(SELECT item\_name, item\_description

FROM items

WHERE item\_endDate = '2023-10-28'); →

QUERY 5:

SELECT u.user\_id, COUNT(b.item\_id) AS total\_bids

FROM pastbids b, users u

WHERE b.item\_id=1

AND u.user\_id = b.user\_id

GROUP BY u.user\_id;