CPSC 304 Project Cover Page

Milestone #: 2

Date: 30 September 2024

Group Number: 26 .

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Kelvin Hang How Mah	57129686	r5i5o	kelvinmah02@gmail.com
Rioto Oka	54645734	q8b5g	okarioto@gmail.com
Aadesh Mehra	39288733	a5f9m	aadeshm03@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Each group must provide the following as a single PDF file:

1. **A completed cover page (template on Canvas)**
2. **A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.**

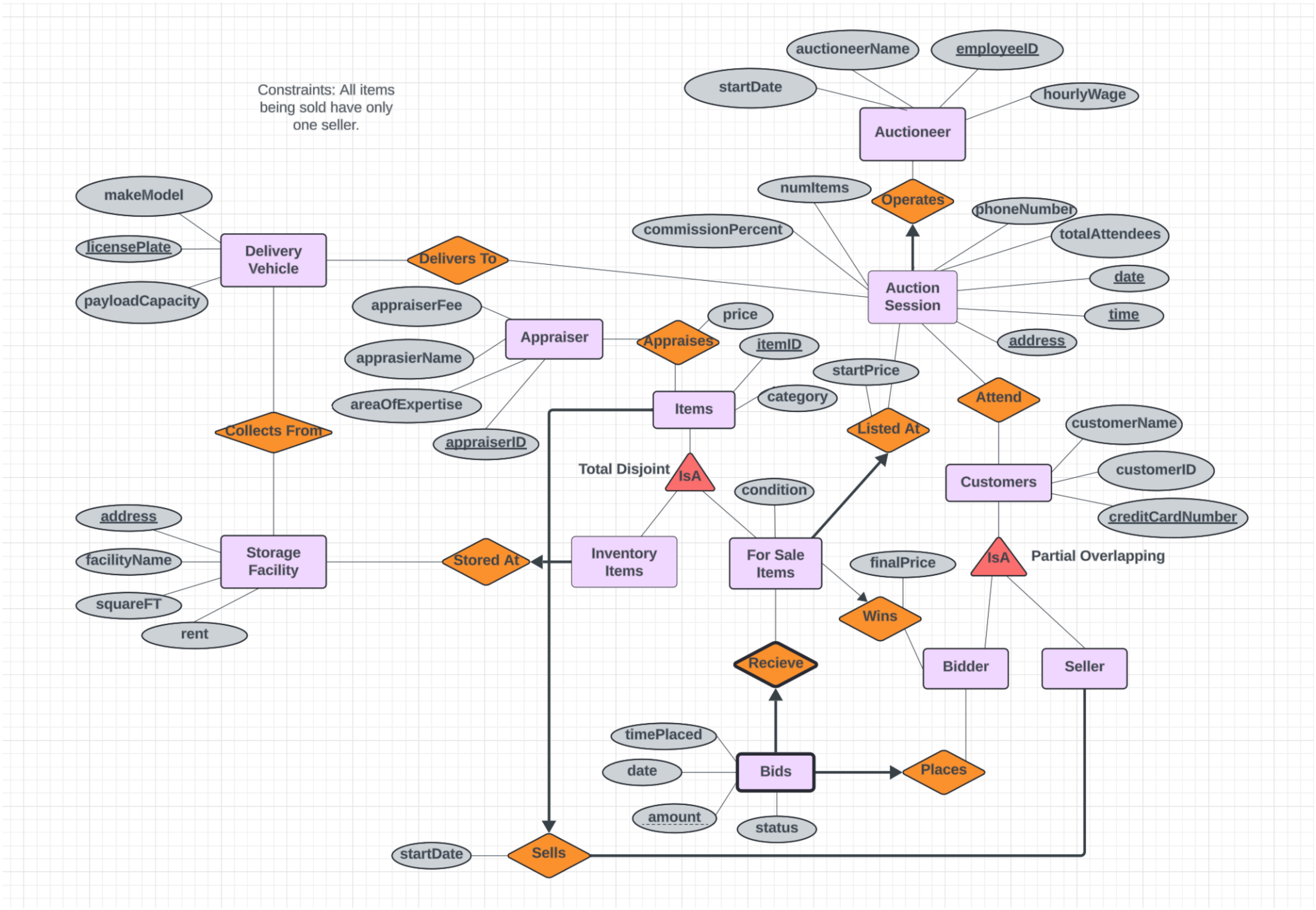
Our project will model the entities that include auction sessions, auctioneers, items, delivery vehicles, storage facilities, appraisers, and customers. In the auction domain, auctioneers facilitate the bidding process, items are auctioned, and customers participate as either sellers or bidders.

3. **The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why.If you have decided not to implement the suggestions given by your project mentor please be sure to leave a note stating why. This is not to say that you must do everything that your project mentor says. In many instances, there are trade-offs between design choices and your decision may be influenced by different factors. Your TAs will often leave suggestions that are meant to help massage your project into a form that will fit with the requirements in future project milestones. If you choose not to take their advice, it would be helpful for them to know why to better assist the group moving forward.**

Changes TO ER DIAGRAM:

- Added startDate to Auctioneer
- Added appraiserFee to Appraiser
- Moved condition from items to forSaleitem
- Added rent to storageFacility
- Added commission percent to auction session

All changes above were made to add functional dependencies


Constraints: All items being sold have only one seller.

4. **The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:**
    a. **List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.**
    b. **Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.**

DeliveryVehicle(licensePlate: CHAR[6], makeModel: VARCHAR(255), payloadCapacity: INT)
PK: (licensePlate)

StorageFacility(address: VARCHAR(255), facilityName: VARCHAR(255), squareFT: INT, rent: INT)
PK: (address)

Appraiser(appraiserID: INT, appraiserName: VARCHAR(255), areaOfExpertise: VARCHAR(255))
PK: (appraiserID)

Auctioneer(employeeID: INT, auctioneerName: VARCHAR(255), hourlyWage: INT, startDate: DATE)
PK: (employeeID)

AuctionSession_Operates(date: DATE, time: VARCHAR(255), address: VARCHAR(255), employeeID: INT, phoneNumber: INT, totalAttendees: INT, numItems: INT, commissionPercent: INT)
PK: (date, time, address)
FK: (employeeID)
NOTNULL: (employeeID)

Attend(date: DATE, time: VARCHAR(255), address: VARCHAR(255), creditCardNumber: INT)
PK: (date, time, address, creditCardNumber)
FK: (date, time, address), (creditCardNumber)

DeliversTo(licensePlate: CHAR[6], date: DATE, time: VARCHAR(255), address: VARCHAR(255))
PK(licensePlate, date, time, address)
FK(licensePlate), (date, time, address)

CollectsFrom(address: VARCHAR(255), licensePlate: CHAR[6])
PK(address, licensePlate)
FK(address), (licensePlate)

Customers(creditCardNumber: INT, customerName: VARCHAR(255), customerID: INT)
PK(creditCardNumber)

Bidder(creditCardNumber: INT)

PK(creditCardNumber)
FK(creditCardNumber)

Seller(creditCardNumber: INT)
PK(creditCardNumber)
FK(creditCardNumber)

Appraises(appraiseerID: INT, itemID: INT, price: INT)
PK: (appraiserID, itemID)
FK: (appraiserID), (itemID)

InventoryItems_StoredAt(itemID: INT, address: VARCHAR(255))
PK: (itemID)
FK: (itemID), (address)
NOT NULL: (address)

ForSaleItems_ListedAt_Wins(itemID: INT, date: Date, time: VARCHAR(255), address: VARCHAR(255), creditCardNumber: INT, startPrice: INT, finalPrice: INT, condition: VARCHAR(255))
PK: (itemID)
FK: (date, time, address), (creditCardNumber)
NOT NULL: (date, time, address)

Items_Sells(itemId: INT, category: VARCHAR(255), startDate: DATE, creditCardNumber: INT)
PK: (itemID)
FK: (creditCardNumber)
NOT NULL: (creditCardNumber)

Assertions are needed to represent total participation from both sides (items to sells and customers to sells)

Bids_Recieve_Places(itemID: INT, timePlaced: VARCHAR(255), date: DATE, amount: INT, status: VARCHAR(255), creditCardNumber: INT)
PK: (itemID, amount)
FK: (itemID), (creditCardNumber)
NOTNULL: (creditCardNumber)

5. Functional Dependencies (FDs)

a. **Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key). PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A → A.**

Note: In your list of FDs, there must be some kind of valid FD other those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful)attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process. You do not need to have a non-PK/CK FD for each relation but be reasonable. If your TA feels that some non-PK/CK FDs have been omitted, your grade will be adjusted accordingly.

DeliveryVehicle
licensePlate -> makeModel, payloadCapacity
makeModel-> payloadCapacity

StorageFacility
address -> facilityName, squareFT, rent
squareFT -> rent

Appraiser
appraiserID -> appraiserName, areaOfExpertise, appraiserFee
areaOfExpertise -> appraiserFee

Auctioneer
employeeID -> auctioneerName, hourlyWage, startDate
startDate -> hourlyWage

Customers
creditCardNumber -> customerName, customerID
customerName -> customerID

AuctionSession_Operates
date, time, address -> employeeID ,phoneNumber, totalAttendees, numItems, commisionPercent
Address -> phoneNumber
numItems -> commisionPercent

Appraises
appraiserID, itemID->price

Items_Sells
itemID ->creditCardNumber, category, startDate,

InventoryItems_StoredAt
itemID->address

ForSaleItems_ListedAt_Wins
itemID, -> date, time, address, creditCardNumber, startPrice, finalPrice, condition
condition -> startPrice

Bids_Recieve_Places
itemID, amount->timePlaced, date, status, creditCardNumber
Date -> Status

Attend
date, time, address, creditCardNumber -> date, time, address, creditCardNumber

DeliversTo
licensePlate, date, time, address->licensePlate, date, time, address

CollectsFrom
address, licensePlate->address, licensePlate

Bidder
creditCardNumber->creditCardNumber

Seller
creditCardNumber->creditCardNumber

6. **Normalization**
    a. **Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition in a manner similar to that done in class. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown.**

    The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, …). ALL Tables must be listed, not only the ones post normalization.

DeliveryVehicle

licensePlate -> makeModel, payloadCapacity

makeModel-> payloadCapacity

1. Find Minimal Cover

   a. licensePlate -> makeModel

   licensePlate-> payloadCapacity

   makeModel-> payloadCapacity

   b. LHS already minimized

   c. Remove redundant FD: licensePlate -> payLoadCapacity

   d. Minimal Cover:

   licensePlate -> makeModel

   makeModel-> payloadCapacity

2. Decompose using synthesis:

   a. DV1(licensePlate, makeModel), DV2(makeModel, payloadCapacity)

   b. All attributes in key are in DV1

3. Decomposed tables:

   DV1(licensePlate: CHAR[6], makeModel: VARCHAR(255))

   PK: (licensePlate)

   FK: (makeModel)

   NOTNULL: (makeModel)

   DV2(makeModel: VARCHAR(255), payloadCapacity: INT)

   PK: (makeModel)

StorageFacility

address -> facilityName, squareFT, rent

squareFT -> rent

1. Find Minimal Cover:

   a. address -> facilityName

   address -> squareFT

   address -> rent

   squareFT ->rent

   b. LHS already minimized

   c. Remove redundant FD: address -> rent

   d. Minimal cover:

   address -> facilityName

   address -> squareFT

   squareFT ->rent

2. Decompose using synthesis:

   a. SF1(address, facilityName), SF2(address, squareFT), SF3(squareFT, rent)

   b. All attributes in key are in SF1

3. Decomposed tables:

SF1(address: VARCHAR(255), facilityName: VARCHAR(255))
PK: (address)
FK: (address)
SF2(address: VARCHAR(255), squareFT: INT)
PK: (address)
FK: (squareFT)
NOT NULL: (squareFT)
SF3(squareFT: INT, rent: INT)
PK: (squareFT)

Appraiser

<u>appraiserID -> appraiserName, areaOfExpertise, appraiserFee</u>
areaOfExpertise -> appraiserFee

1. Find Minimal Cover:
   a. appraiserID -> appraiserName
   appraiserID -> areaOfExpertise
   appraiserID -> appraiserFee
   areaOfExpertise -> appraiserFee
   b. LHS already minimized
   c. Remove redundant FD: appraiserID -> appraiserFee
   d. Minimal cover:
   appraiserID -> appraiserName
   appraiserID -> areaOfExpertise
   areaOfExpertise -> appraiserFee
2. Decompose using synthesis:
   a. A1(appraiserID, appraiserName), A2(appraiserID, areaOfExpertise), A3(areaOfExpertise, appraiserFee)
   b. All attributes in key are in A1
3. Decomposed tables:
   A1(appraiserID: INT, appraiserName: VARCHAR(255))
   PK: (appraiserID)
   FK: (appraiserID)
   A2(appraiserID: INT, areaOfExpertise: VARCHAR(255))
   PK: (appraiserID)
   FK: (areaofExpertise)
   NOT NULL: (areaofExpertise)
   A3(areaOfExpertise, appraiserFee)
   PK: (areaOfExpertise)

Auctioneer

<u>employeeID -> auctioneerName, hourlyWage, startDate</u>

startDate -> hourlyWage

1. Find Minimal Cover:
    a. employeeID -> auctioneerName
       employeeID -> hourlyWage
       employeeID -> startDate
       startDate -> hourlyWage
    b. LHS already minimized
    c. Remove redundant FD: employeeID -> hourlyWage
    d. Minimal cover:
       employeeID -> auctioneerName
       employeeID -> startDate
       startDate -> hourlyWage
2. Decompose using synthesis:
    a. A1(employeeID, auctioneerName), A2 (employeeID, startDate), A3(startDate, hourlyWage)
    b. All attributes in key are in A1
3. Decomposed tables:
       A1(employeeID: INT, auctioneerName: VARCHAR(255))
           PK: (employeeID)
           FK: (employeeID),
       A2 (employeeID: INT, startDate: DATE)
           PK: (employeeID)
           FK: (startDate)
           NOT NULL: (startDate)
       A3(startDate: DATE, hourlyWage: INT)
           PK: (startDate),

Customers

<u>creditCardNumber -> customerName, customerID</u>

customerName -> customerID

1. Find Minimal Cover:
    a. creditCardNumber-> customerName
       creditCardNumber -> customerID
       customerName -> customerID
    b. LHS already minimized
    c. Remove redundant FD: creditCardNumber -> customerID
    d. Minimal Cover:
       creditCardNumber-> customerName
       customerName -> customerID
2. Decompose using synthesis:
    a. C1(creditCardNumber, customerName), C2(customerName, customerID)
    b. All attributes in key are in C1

3. Decomposed tables:

        C1(creditCardNumber: INT, customerName: VARCHAR(255))
            PK: (creditCardNumber)
            FK: (customerName)
            NOT NULL: (customerName)
        C2(customerName: VARCHAR(255), customerID: INT)
            PK: (customerName)

AuctionSession_Operates

date, time, address -> employeeID ,phoneNumber, totalAttendees, numItems, commissionPercent
address -> phoneNumber
numItems -> commissionPercent

1. Find Minimal Cover:
    a. date, time, address -> employeeID
       date, time, address -> phoneNumber
       date, time, address -> totalAttendees
       date, time, address -> numItems
       date, time, address -> commissionPercent
       address -> phoneNumber
       numItems -> comissionPercent
    b. LHS is already minimized
    c. Remove redundant FD: date, time, address -> phoneNumber, date, time, address -> commissionPercent
    d. Minimal cover:
       date, time, address -> employeeID
       date, time, address -> totalAttendees
       date, time, address -> numItems
       address -> phoneNumber
       numItems -> comissionPercent
2. Decompose using synthesis:
    a. ASO1(date, time, address, employeeID), ASO2(date, time, address, totalAttendees), ASO3(date, time, address, numItems), ASO4(address, phoneNumber), ASO5(numItems, comissionPercent)
    b. All attributes in key are in ASO1
3. Decomposed tables:

        ASO1(date: DATE, time: VARCHAR(255), address: VARCHAR(255), employeeID: INT)
            PK: (date, time, address)
            FK: (date, time, address), (employeeID)
            NOTNULL: (employeeID)
        ASO2(date: DATE, time: VARCHAR(255), address: VARCHAR(255), totalAttendees: INT)
            PK: (date, time, address)
            FK: (date, time, address)
        ASO3(date: DATE, time: VARCHAR(255), address: VARCHAR(255), numItems: INT)

PK: (date, time, address)
FK: (address), (numItems)
NOT NULL: (numItems)

ASO4(address: VARCHAR(255), phoneNumber: INT)
PK: (address)

ASO5(numItems: INT, comissionPercent: INT)
PK: (numItems)

Appraises

appraiserID, itemID->price

1. Already in 3NF
2. Appraises(appraiseerID: INT, itemID: INT, price: INT)
   PK: (appraiserID, itemID)
   FK: (appraiserID), (itemID)

Items_Sells

itemID -> creditCardNumber, category, startDate

1. Already in 3NF
2. Items_Sells(itemId: INT, category: VARCHAR(255), startDate: DATE, creditCardNumber: INT)
   PK: (itemID)
   FK: (creditCardNumber)
   NOT NULL: (creditCardNumber)

InventoryItems_StoredAt

itemID->address

1. Already in 3NF
2. InventoryItems_StoredAt(itemID: INT, address: VARCHAR(255))
   PK: (itemID)
   FK: (itemID), (address)
   NOT NULL: (address)

ForSaleItems_ListedAt_Wins

itemID -> date, time, address, creditCardNumber, startPrice, finalPrice, condition

condition -> startPrice

1. Find Minimal Cover:
   a. itemID -> date
      itemID -> time

itemID -> address
itemID -> creditCardNumber
itemID -> startPrice
itemID -> finalPrice
itemID -> condition
condition -> startPrice

b. LHS already minimized

c. Remove redundant FDs: itemID -> startPrice

d. Minimal cover:
itemID -> date
itemID -> time
itemID -> address
itemID -> creditCardNumber
itemID -> finalPrice
itemID -> condition
condition -> startPrice

2. Decompose using synthesis:

a. FLW1(itemID, date), FLW2(itemID, time), FLW3(itemID, address), FLW4(itemID, creditCardNumber), FLW5(itemID, finalPrice), FLW6(itemID, condition), FLW7(condition, startPrice)

b. All attributes in key are in FLW1

3. Decomposed tables:

FLW1(itemID: INT, date: DATE)
PK: (itemID)
FK: (itemID)
NOTNULL: (date)

FLW2(itemID: INT, time: INT)
PK: (itemID)
FK: (itemID)
NOTNULL: (time)

FLW3(itemID: INT, address: VARCHAR(255))
PK: (itemID)
FK: (itemID), (address)
NOTNULL: (address)

FLW4(itemID: INT, creditCardNumber: INT)
PK: (itemID)
FK: (itemID), (creditCardNumber)

FLW5(itemID: INT, finalPrice: INT)
PK: (itemID)
FK: (itemID)

FLW6(itemID: INT, condition: VARCHAR(255))

PK: (itemID)

FK: (condition)

FLW7(condition: VARCHAR(255), startPrice: INT)

PK: (condition)

Bids_Recieve_Places

itemID, amount -> timePlaced, date, status, creditCardNumber

date -> status

1. Find Minimal Cover:
    a. itemID, amount -> timePlaced
    itemID, amount -> date
    itemID, amount -> status
    itemID, amount -> creditCardNumber
    date -> status
    b. LHS already minimized
    c. Remove redundant FDs: itemID, amount -> status
    d. Minimal cover:
    itemID, amount -> timePlaced
    itemID, amount -> date
    itemID, amount -> creditCardNumber
    date -> status
2. Decompose using synthesis:
    a. BRP1(itemID, amount, timePlaced), BRP2(itemID, amount, date), BRP3(itemID, amount, creditCardNumber), BRP4(date, status)
    b. All attributes in key are in BRP1
3. Decomposed tables:

BRP1(itemID: INT, amount: INT, timePlaced: VARCHAR(255))

PK: (itemID, amount)

FK: (itemID)

BRP2(itemID: INT, amount: INT, date: DATE)

PK: (itemID, amount)

FK: (itemID), (date)

NOT NULL: (date)

BRP3(itemID: INT, amount: INT, creditCardNumber: INT)

PK: (itemID, amount)

FK: (itemID), (creditCardNumber)

NOTNULL: (creditCardNumber)

BRP4(date: DATE, status: VARCHAR(255))

PK: (date)

Attend

date, time, address, creditCardNumber -> date, time, address, creditCardNumber

1. Already in 3NF
2. Attend(date: DATE, time: VARCHAR(255), address: VARCHAR(255), creditCardNumber: INT)
   PK: (date, time, address, creditCardNumber)
   FK: (date, time, address), (creditCardNumber)

DeliversTo

licensePlate, date, time, address->licensePlate, date, time, address

1. Already in 3NF
2. DeliversTo(licensePlate: CHAR[6], date: DATE, time: INT, address: VARCHAR(255))
   PK(licensePlate, date, time, address)
   FK(licensePlate), (date, time, address)

CollectsFrom

address, licensePlate->address, licensePlate

1. Already in 3NF
2. CollectsFrom(address: VARCHAR(255), licensePlate: CHAR[6])
   PK(address, licensePlate)
   FK(address), (licensePlate)

Bidder

creditCardNumber->creditCardNumber

1. Already in 3NF
2. Bidder(creditCardNumber: INT)
   PK(creditCardNumber)
   FK(creditCardNumber)

Seller

creditCardNumber->creditCardNumber

1. Already in 3NF
2. Seller(creditCardNumber: INT)
   PK(creditCardNumber)
   FK(creditCardNumber)

7. **The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc. Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR(255) data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which**

department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR(255) as the number of characters in a course name can vary greatly.

DeliveryVehicle

```
CREATE TABLE DeliveryVehicle1 (
   licensePlate CHAR(6) PRIMARY KEY,
   makeModel VARCHAR(255) NOT NULL,
   FOREIGN KEY (makeModel) REFERENCES DeliveryVehicle2(makeModel) ON DELETE CASCADE
);

CREATE TABLE DeliveryVehicle2 (
   makeModel VARCHAR(255) PRIMARY KEY,
   payloadCapacity INT
);
```

StorageFacility

```
CREATE TABLE StorageFacility1 (
   address VARCHAR(255),
   facilityName VARCHAR(255),
   PRIMARY KEY (address),
   FOREIGN KEY (address) REFERENCES StorageFacility2(address) ON DELETE CASCADE
);


CREATE TABLE StorageFacility2 (
   address VARCHAR(255) PRIMARY KEY,
   squareFT INT NOT NULL,
   FOREIGN KEY (squareFT) REFERENCES StorageFacility3(squareFT) ON DELETE CASCADE
);

CREATE TABLE StorageFacility3 (
   squareFT INT PRIMARY KEY,
   rent INT
);
```

Appraiser

```
CREATE TABLE Appraiser1 (
   appraiserID INT PRIMARY KEY,
   appraiserName VARCHAR(255),
   FOREIGN KEY (appraiserID) REFERENCES Appraiser2(appraiserID) ON DELETE CASCADE
);
```

```
CREATE TABLE Appraiser2 (
   appraiserID INT PRIMARY KEY,
   areaOfExpertise VARCHAR(255)NOT NULL,
   FOREIGN KEY (areaOfExpertise) REFERENCES Appraiser3(areaOfExpertise) ON DELETE CASCADE
);

CREATE TABLE Appraiser3 (
   areaOfExpertise VARCHAR(255) PRIMARY KEY,
   appraiserFee INT
);
```

Auctioneer

```
CREATE TABLE Auctioneer1 (
   employeeID INT PRIMARY KEY,
   auctioneerName VARCHAR(255),
   FOREIGN KEY (employeeID) REFERENCES Auctioneer2(employeeID) ON DELETE CASCADE
);

CREATE TABLE Auctioneer2 (
   employeeID INT PRIMARY KEY,
   startDate DATE NOT NULL,
   FOREIGN KEY (startDate) REFERENCES Auctioneer3(startDate) ON DELETE CASCADE
);

CREATE TABLE Auctioneer3 (
   startDate DATE PRIMARY KEY,
   hourlyWage INT
);
```

Customers

```
CREATE TABLE Customers1(
   creditCardNumber INT PRIMARY KEY,
   customerName VARCHAR(255) NOT NULL,
   FOREIGN KEY (customerName) REFERENCES Customers2(customerName) ON DELETE CASCADE
);

CREATE TABLE Customers2 (
   customerName VARCHAR(255) PRIMARY KEY,
   customerID INT
);
```

AuctionSession_Operates

```
CREATE TABLE AuctionSession_Operates1 (
   date DATE,
   time INT,
   address VARCHAR(255),
   employeeID INT NOT NULL,
   PRIMARY KEY (date, time, address),
   FOREIGN KEY (date, time, address) REFERENCES AuctionSession_Operates2(date, time, address) ON DELETE CASCADE
   FOREIGN KEY (employeeID) REFERENCES Auctioneer1 ON DELETE NO ACTION
);

CREATE TABLE AuctionSession_Operates2 (
   date DATE,
   time INT,
   address VARCHAR(255),
   totalAttendees INT,
   PRIMARY KEY (date, time, address),
   FOREIGN KEY (date, time, address) REFERENCES AuctionSession_Operates3(date, time, address) ON DELETE CASCADE,
);

CREATE TABLE AuctionSession_Operates3 (
   date DATE,
   time INT,
   address VARCHAR(255),
   numItems INT NOT NULL,
   PRIMARY KEY (date, time, address),
   FOREIGN KEY (numItems) REFERENCES AuctionSession_Operates5(numItems) ON DELETE CASCADE,
   FOREIGN KEY (address) REFERENCES AuctionSession_Operates4(address) ON DELETE CASCADE
);

CREATE TABLE AuctionSession_Operates4 (
   address VARCHAR(255) PRIMARY KEY,
   phoneNumber INT
);

CREATE TABLE AuctionSession_Operates5 (
   numItems INT PRIMARY KEY,
   commissionPercent INT
);
```

Appraises

```
CREATE TABLE Appraises1 (
   appraiserID INT,
   itemID INT,
   price INT,
   PRIMARY KEY (appraiserID, itemID),
   FOREIGN KEY (appraiserID) REFERENCES Appraiser1(appraiserID) ON DELETE CASCADE,
   FOREIGN KEY(itemID) REFERENCES Items_Sells(itemID) ON DELETE NO ACTION
);
```

Items_Sells

```
CREATE TABLE Items_Sells (
   itemID INT PRIMARY KEY,
   category VARCHAR(255),
   startDate DATE,
   creditCardNumber INT NOT NULL,
   FOREIGN KEY (creditCardNumber) REFERENCES Customers1(creditCardNumber) ON DELETE
CASCADE
);
```

InventoryItems_StoredAt

```
CREATE TABLE InventoryItems_StoredAt (
   itemID INT PRIMARY KEY,
   address VARCHAR(255) DEFAULT ‘HQ’ NOT NULL ,
   FOREIGN KEY (address) REFERENCES StorageFacility1(address) ON DELETE SET DEFAULT
   FOREIGN KEY (itemID) REFERENCES Item_Sells(itemID) ON DELETE CASCADE
);
```

ForSaleItems_ListedAt_Wins

```
CREATE TABLE ForSaleItems_ListedAt_Wins1 (
   itemID INT PRIMARY KEY,
   date DATE NOT NULL,
   FOREIGN KEY (itemID) REFERENCES ForSaleItems_ListedAt_Wins2(itemID) ON DELETE CASCADE
   FOREIGN KEY (date, time, address -implicitly from itemID-) REFERENCES
AuctionSession_Operates(date, time, address) ON DELETE CASCADE
    FOREIGN KEY (itemID) REFERENCES Item_Sells(itemID) ON DELETE CASCADE

);
CREATE TABLE ForSaleItems_ListedAt_Wins2 (
   itemID INT PRIMARY KEY,
   time INT NOT NULL,
   FOREIGN KEY (itemID) REFERENCES ForSaleItems_ListedAt_Wins3(itemID) ON DELETE CASCADE
);
CREATE TABLE ForSaleItems_ListedAt_Wins3 (
   itemID INT PRIMARY KEY,
   address VARCHAR(255) NOT NULL,
   FOREIGN KEY (itemID) REFERENCES ForSaleItems_ListedAt_Wins4(itemID) ON DELETE CASCADE,
   FOREIGN KEY (address) REFERENCES AuctionSession_Operates4(address) ON DELETE CASCADE
);
CREATE TABLE ForSaleItems_ListedAt_Wins4 (
   itemID INT PRIMARY KEY,
   creditCardNumber INT,
   FOREIGN KEY (itemID) REFERENCES ForSaleItems_ListedAt_Wins5(itemID) ON DELETE CASCADE,
   FOREIGN KEY (creditCardNumber) REFERENCES Customers1(creditCardNumber) ON DELETE SET NULL
);
CREATE TABLE ForSaleItems_ListedAt_Wins5 (
   itemID INT PRIMARY KEY,
   finalPrice INT,
   FOREIGN KEY (itemID) REFERENCES ForSaleItems_ListedAt_Wins6(itemID) ON DELETE CASCADE
);
CREATE TABLE ForSaleItems_ListedAt_Wins6 (
   itemID INT PRIMARY KEY,
   condition VARCHAR(255) NOT NULL,
   FOREIGN KEY (condition) REFERENCES ForSaleItems_ListedAt_Wins7(condition) ON DELETE CASCADE
);
CREATE TABLE ForSaleItems_ListedAt_Wins7 (
   condition VARCHAR(255) PRIMARY KEY,
   startPrice INT
);
```

Bids_Recieve_Places

```
CREATE TABLE Bids_Recieve_Places1(
   itemID INT,
   amount INT,
   timePlaced VARCHAR(255)
   PRIMARY KEY (itemID, amount),
   FOREIGN KEY (itemID) REFERENCES Bids_Recieve_Places2(itemID) ON DELETE CASCADE
);
CREATE TABLE Bids_Recieve_Places2(
   itemID INT,
   amount INT,
   date DATE NOT NULL,
   PRIMARY KEY (itemID, amount),
   FOREIGN KEY (itemID) REFERENCES Bids_Recieve_Places3(itemID) ON DELETE CASCADE,
   FOREIGN KEY (date) REFERENCES Bids_Recieve_Places4(date) ON DELETE CASCADE
);
CREATE TABLE Bids_Recieve_Places3(
   itemID INT,
   amount INT,
   creditCardNumber INT NOT NULL,
   PRIMARY KEY (itemID, amount),
   FOREIGN KEY (itemID) REFERENCES Bids_Recieve_Places1 (itemID) ON DELETE CASCADE,
   FOREIGN KEY (creditCardNumber) REFERENCES Customer1 (creditCardNumber) ON DELETE CASCADE
);
CREATE TABLE Bids_Recieve_Places4(
   date DATE PRIMARY KEY,
   status VARCHAR(255),
);
```

Attend

```
    CREATE TABLE Attend (
      date DATE,
      time INT,
      address VARCHAR(255),
      creditCardNumber INT,
      PRIMARY KEY (date, time, address, creditCardNumber),
      FOREIGN KEY (creditCardNumber) REFERENCES Customers1 (creditCardNumber))ON DELETE
CASCADE
     FOREIGN KEY (date, time, address) REFERENCES AuctionSession_Operates1(date, time, address) ON
DELETE CASCADE
    );
```

DeliversTo

```
    CREATE TABLE DeliversTo(
      licensePlate CHAR(6),
      date DATE,
      time INT,
      address VARCHAR(255),
      PRIMARY KEY (licensePlate, date, time, address),
      FOREIGN KEY (licensePlate) REFERENCES DeliveryVehicle1(licensePlate) ON DELETE ON DELETE
CASCADE,
      FOREIGN KEY (date, time, address) REFERENCES AuctionSession_Operates1(date, time, address) ON
DELETE CASCADE
    );
```

CollectsFrom

```
CREATE TABLE CollectsFrom (
    address VARCHAR(255),
    licensePlate CHAR(6),
    PRIMARY KEY (address, licensePlate),
    FOREIGN KEY (address) REFERENCES StorageFacility1(address) ON DELETE CASCADE,
    FOREIGN KEY (licensePlate) REFERENCES DeliveryVehicle1(licensePlate) ON DELETE CASCADE
);
```

Bidder

```
 CREATE TABLE Bidder (
    creditCardNumber INT,
    PRIMARY KEY (creditCardNumber),
    FOREIGN KEY (creditCardNumber) REFERENCES Customer1 ON DELETE CASCADE
);
```

Seller

```
CREATE TABLE Seller (
    creditCardNumber INT,
    PRIMARY KEY (creditCardNumber)
    FOREIGN KEY (creditCardNumber) REFERENCES Customer1 ON DELETE CASCADE
);
```

Since oracle doesn't support ON UPDATE, we refrain from updating values frequently. If an update is required, we will do it manually

8. **INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.**

DeliveryVehicle

```
INSERT INTO DV1 (licensePlate, makeModel)
VALUES ('ABC123', 'Ford Transit'), ('DEF456', 'Mercedes Sprinter'), ('GHI789', 'Ram ProMaster'),
('JKL012', 'Chevrolet Express'), ('MNO345', 'Nissan NV3500'), ('PQR678', 'Ford F-150'), ('STU901',
'Chevrolet Silverado 1500'), ('VWX234', 'Ram 1500'), ('YZA567', 'Toyota Tacoma'), ('BCD890', 'GMC
Sierra 2500HD');

INSERT INTO DV2 (makeModel, payloadCapacity)
VALUES ('Ford Transit', 3500), ('Mercedes Sprinter', 4000), ('Ram ProMaster', 4200), ('Chevrolet
Express', 3000), ('Nissan NV3500', 3700), ('Ford F-250', 4000), ('Chevrolet Silverado 2500', 4000), ('Ram
3500', 7680);
```

StorageFacility

```
INSERT INTO StorageFacility1 (address, facilityName)
```

```
VALUES ('123 Main St', 'Main Street Storage'), ('456 Oak Ave', 'Oak Avenue Self Storage'), ('789 Pine Dr',
'Pine Drive Secure Storage'), ('101 Maple Ln', 'Maple Lane Storage Solutions'), ('202 Birch Blvd', 'Birch
Boulevard Storage Center');

INSERT INTO StorageFacility2 (address, squareFT)
VALUES ('123 Main St', 10000), ('456 Oak Ave', 20000), ('789 Pine Dr', 40000), ('101 Maple Ln', 60000),
('202 Birch Blvd', 80000);


INSERT INTO StorageFacility3 (squareFT, rent)
VALUES (10000, 7000), (20000, 13500), (40000, 27000), (60000, 40000), (80000, 52000);
```

Appraiser

```
INSERT INTO Appraiser1 (appraiserID, appraiserName)
VALUES (101, 'Tom Brown'), (102, 'Jane Miller'), (103, 'David Wilson'), (104, 'Lisa Adams'), (105, 'John
Clark');

INSERT INTO Appraiser2 (appraiserID, areaOfExpertise)
VALUES (101, 'Toys'), (102, 'Sports Memorabilia' ), (103, 'Fine Art'), (104, 'Vintage Cars'), (105, 'Jewelry');

INSERT INTO Appraiser3 (areaOfExpertise, appraiserFee)
VALUES ('Toys', 1800), ('Sports Memorabilia', 2200), ('Fine Art', 5000), ('Vintage Cars', 6000), (''Jewelry',
2500);
```

Auctioneer

```
INSERT INTO Auctioneer1 (employeeID, auctioneerName)
VALUES (201, 'Sarah Johnson'), (202, 'Michael Smith'), (203, 'Emily Davis'), (204, 'James Wilson'), (205,
'Linda Taylor');

INSERT INTO Auctioneer2 (employeeID, startDate)
VALUES (201, '2020-01-15'), (202, '2021-05-22'), (203, '2019-03-10'), (204, '2022-08-30'), (205,
'2023-06-01');

INSERT INTO Auctioneer3 (startDate, hourlyWage)
VALUES ('2020-01-15', 30), ('2021-05-22', 35), ('2019-03-10', 28), ('2022-08-30', 40), ('2023-06-01', 32);
```

Customers

```
INSERT INTO Customers1 (creditCardNumber, customerName)
VALUES (1234567812345678, 'Alice Johnson'), (2345678923456789, 'Bob Smith'), (3456789034567890,
'Catherine Brown'), (4567890145678901, 'David Wilson'), (5678901256789012, 'Emily Davis');

INSERT INTO Customers2 (customerName, customerID)
VALUES ('Alice Johnson', 1), ('Bob Smith', 2), ('Catherine Brown', 3), ('David Wilson', 4), ('Emily Davis',
5);
```

AuctionSession_Operates

```
INSERT INTO AuctionSession_Operates1 (date, time, address, employeeID)
```

```
VALUES ('2024-10-15', 14:00, '123 Auction St', 201), ('2024-10-16', 15:00, '456 Bid Ave', 202),
('2024-10-17', 16:00, '789 Sale Rd', 203), ('2024-10-18', 17:00, '321 Sell Blvd', 204), ('2024-10-19',
18:00, '654 Offer Pl', 205);
```

```
INSERT INTO AuctionSession_Operates2 (date, time, address, totalAttendees)
VALUES ('2024-10-15', 14:00, '123 Auction St', 25), ('2024-10-16', 15:00, '456 Bid Ave', 40),
('2024-10-17', 16:00, '789 Sale Rd', 55), ('2024-10-18', 17:00, '321 Sell Blvd', 30), ('2024-10-19', 18:00,
'654 Offer Pl', 60);
```

```
INSERT INTO AuctionSession_Operates3 (date, time, address, numItems)
VALUES ('2024-10-15', '14:00', '123 Auction St', 50), ('2024-10-16', '15:00', '456 Bid Ave', 100),
('2024-10-17', '16:00', '789 Sale Rd', 150), ('2024-10-18', '17:00', '321 Sell Blvd', 200), ('2024-10-19',
'18:00', '654 Offer Pl', 250);
```

```
INSERT INTO AuctionSession_Operates4 (address, phoneNumber)
VALUES ('123 Auction St', 5551234567), ('456 Bid Ave', 5552345678), ('789 Sale Rd', 5553456789), ('321
Sell Blvd', 5554567890), ('654 Offer Pl', 5555678901);
```

```
INSERT INTO AuctionSession_Operates5 (numItems, commissionPercent)
VALUES (50, 10),(100, 15), (150, 12), (200, 8), (250, 5);
```

Appraises

```
INSERT INTO Appraises1 (appraiserID, itemID, price)
VALUES (101, 1001, 485),  (102, 1002, 315), (103, 1003, 210),  (101, 1004, 170), (102, 1005, 430);
```

Items_Sells

```
INSERT INTO Items_Sells (itemID, category, startDate, creditCardNumber)
VALUES (1001, 'Furniture', '2024-10-01', 1234567812345678), (1002, 'Electronics', '2024-10-05',
2345678923456789), (1003, 'Antiques', '2024-10-10', 3456789034567890), (1004, 'Art', '2024-10-12',
4567890145678901), (1005, 'Collectibles', '2024-10-15', 5678901256789012);
```

InventoryItems_StoredAt

```
INSERT INTO InventoryItems_StoredAt (itemID, address)
VALUES (1001, '123 Auction St'), (1002, '456 Bid Ave'), (1003, '789 Sale Rd'), (1004, '321 Sell Blvd'),
(1005, '654 Offer Pl');
```

ForSaleItems_ListedAt_Wins

```
INSERT INTO ForSaleItems_ListedAt_Wins1 (itemID, date)
VALUES (1001, '2024-10-15'), (1002, '2024-10-16'), (1003, '2024-10-17'), (1004, '2024-10-18'),
(1005, '2024-10-19');
```

```
INSERT INTO ForSaleItems_ListedAt_Wins2 (itemID, time) VALUES (1001, '12:00'), (1002, '13:00'),
(1003, '14:00'), (1004, '15:00'), (1005, '16:00');
```

```
INSERT INTO ForSaleItems_ListedAt_Wins3 (itemID, address)
VALUES (1001, '123 Auction St'), (1002, '456 Bid Ave'), (1003, '789 Sale Rd'), (1004, '321 Sell Blvd'),
(1005, '654 Offer Pl');

INSERT INTO ForSaleItems_ListedAt_Wins4 (itemID, creditCardNumber)
VALUES (1001, 1234567812345678), (1002, 2345678923456789), (1003, NULL), (1004,
4567890145678901), (1005, 5678901256789012);

INSERT INTO ForSaleItems_ListedAt_Wins5 (itemID, finalPrice)
VALUES (1001, 550), (1002, 350), (1003, 220), (1004, 180), (1005, 450);

INSERT INTO ForSaleItems_ListedAt_Wins6 (itemID, condition)
VALUES (1001, 'New'), (1002, 'Used'), (1003, 'Refurbished'), (1004, 'Damaged'), (1005, 'Open Box');

INSERT INTO ForSaleItems_ListedAt_Wins7 (condition, startPrice)
VALUES ('New', 500), ('Used', 300), ('Refurbished', 200), ('Damaged', 150), ('Open Box', 400);
```

Bids_Recieve_Places

```
INSERT INTO Bids_Recieve_Places1 (itemID, amount, timePlaced)
VALUES (1001, 500, '10:30'), (1002, 300, '11:00'), (1003, 200, '11:30'), (1004, 150, '12:00'), (1005, 400,
'12:30');

INSERT INTO Bids_Recieve_Places2 (itemID, amount, date)
VALUES (1001, 500, '2024-10-15'), (1002, 300, '2024-10-16'), (1003, 200, '2024-10-17'), (1004, 150,
'2024-10-18'), (1005, 400, '2024-10-19');

INSERT INTO Bids_Recieve_Places3 (itemID, amount, creditCardNumber)
VALUES (1001, 500, 1234567812345678), (1002, 300, 2345678923456789), (1003, 200,
3456789034567890), (1004, 150, 4567890145678901), (1005, 400, 5678901256789012);

INSERT INTO Bids_Recieve_Places4 (date, status)
VALUES ('2024-10-15', 'Open'), ('2024-10-16', 'Closed'), ('2024-10-17', 'Pending'), ('2024-10-18', 'Open'),
('2024-10-19', 'Closed');
```

Attend

```
INSERT INTO Attend (date, time, address, creditCardNumber)
VALUES ('2024-10-15', '14:00', '123 Main St', 1234567812345678), ('2024-10-16', '15:00', '456 Elm St',
2345678923456789), ('2024-10-17', '16:00', '789 Oak St', 3456789034567890), ('2024-10-18', '17:00',
'321 Pine St', 4567890145678901), ('2024-10-19', '18:00', '654 Maple St', 5678901256789012);
```

DeliversTo

```
INSERT INTO DeliversTo (licensePlate, date, time, address)
VALUES ('ABC123', '2024-10-15', '14:00', '123 Main St'), ('DEF456', '2024-10-16', '15:00', '456 Oak Ave'),
('GHI789', '2024-10-17', '16:00', '789 Pine Dr'), ('JKL012', '2024-10-18', '17:00', '101 Maple Ln'),
('MNO345', '2024-10-19', '18:00', '202 Birch Blvd');
```

CollectsFrom

```
INSERT INTO CollectsFrom (address, licensePlate)
VALUES ('123 Main St', 'ABC123'), ('456 Oak Ave', 'DEF456'), ('789 Pine Dr', 'GHI789'), ('101 Maple Ln',
'JKL012'), ('202 Birch Blvd', 'MNO345');
```

Bidder

```
INSERT INTO Bidder (creditCardNumber)
VALUES (1234567890123456), (2345678901234567), (3456789012345678), (4567890123456789),
(5678901234567890);
```

Seller

```
INSERT INTO Seller (creditCardNumber)
VALUES (1111222233334444), (2222333344445555), (3333444455556666), (4444555566667777),
(5555666677778888);
```