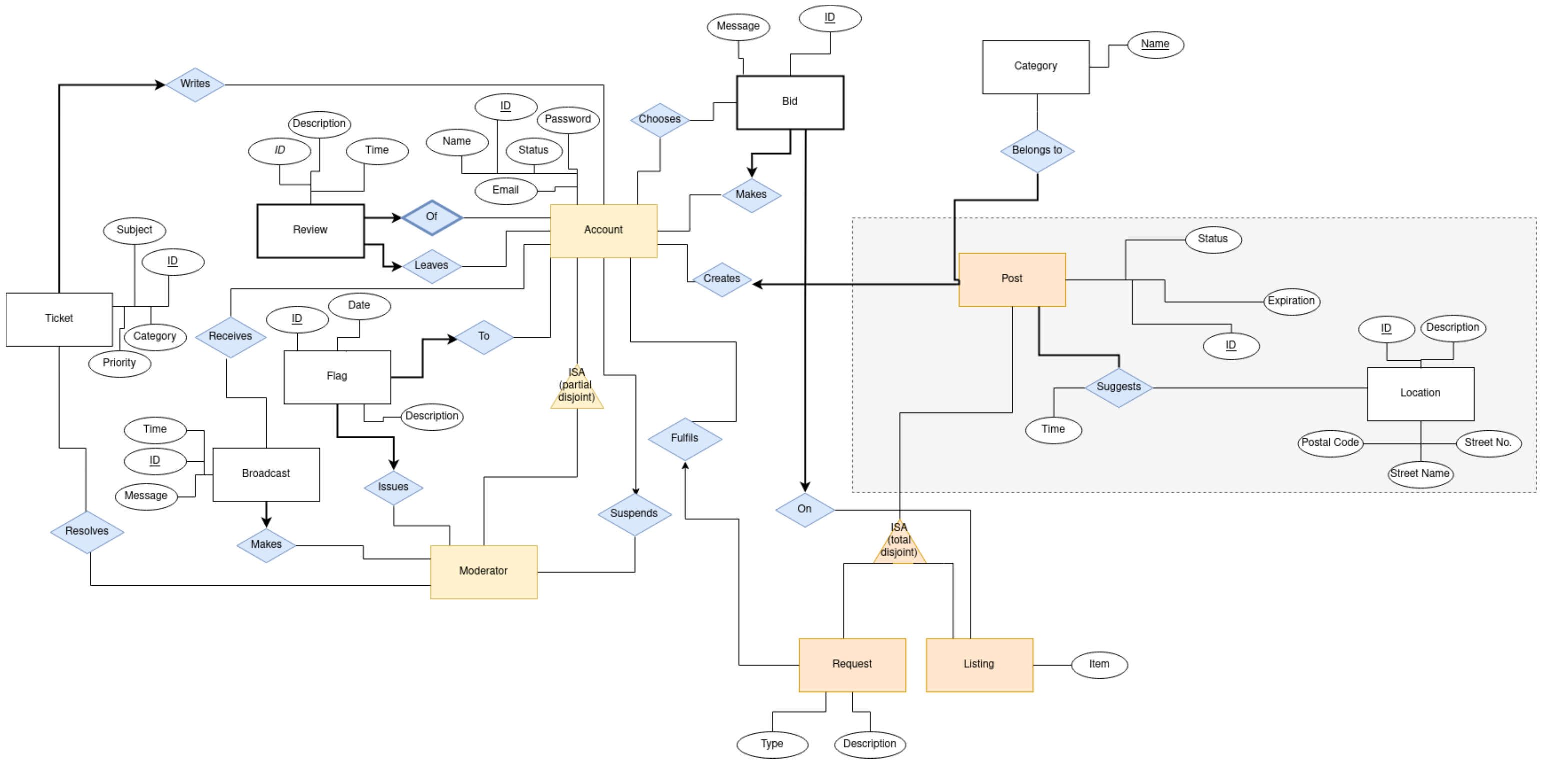
CPSC 304 Project Cover Page

Milestone #: 2

Date: October 25, 2021

Group Number: 50

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Amy Liu	12757563	j9v2b	aimyul@student.ubc.ca
Louise Cooke	42721332	x8x2b	louisecc@student.ubc.ca
Krish Thukral	69843365	g1c3b	krish.thukral@gmail.com



Changes from Milestone 1:

- Bid is no longer a weak entity
- Ticket has 2 new attributes: Category and Priority
- Location has new attributes: Id, StreetNo., StreetName, Postal Code
- Post no longer has the attribute of Item
- Request has new attributes: Type and Description
- Listing has a new attribute: Item

Note: Draw.io does not enable dash underlining of partial keys in weak entity sets; therefore, for the weak entity sets Review and Bid, we italicized the partial keys *ID*. We apologize for the inability to exactly use the conventions from lectures and the textbook.

-

Schema

Primary keys are underlined and foreign keys are in bold.
Candidate keys and constraints are listed where applicable.
All id attributes must be unique!

Account (id: *integer*, name: *string*, password: *string*, email: *string*, status: *integer*)

- Constraints:
    - All attributes except for Status must be non-null.
    - Status is 0 when an account is suspended and 1 when an account is not suspended. Additional status values may be applied.

Moderator (**id:** *integer*)

- id references Account (id)
- Constraints:
    - All attributes must not be null

Bid (id: *integer*, message: *string*)

- Constraints:
    - All attributes must not be null

Makes (**aid**: *integer*, **bid**: *integer*)

- aid references Account (id)
- bid references Bid (id)
- Constraints:
    - All attributes must not be null

Resolves (**tid**: *integer*, **mid**: *integer*)

- tid references Ticket (id)
- md references Moderator (id)
- Constraints:
    - All attributes must not be null

Suspends (**aid**: *integer*, **mid**: *integer*)

- Constraints:
    - All attributes must not be null

Chooses (**aid**: *integer*, **bid**: *integer*)

- aid references Account (id)

- bid references Bid (id)
- Constraints:
    - All attributes must not be null

Ticket (<u>ID</u>: *integer*, **MID**: *integer*, **AID**: *integer*, Subject: *string*, Priority: *integer*, Category: *string*)
- mid references Moderator (id)
- aid references Account (id)
- Constraints:
    - Priority is an integer where higher number means more Priority
    - Priority is assigned based on Category
    - All attributes must not be null

Flag (<u>ID</u>: *integer*, **MID**: *integer*, **AID**: *integer*, date: *datetime*, description: *string*)
- mid references Moderator (id)
- aid references Account (id)
- Constraints;
    - All attributes must be non null

Broadcast (<u>ID</u>: *integer*, Time: *datetime*, Message: *string*)
- Constraints;
    - All attributes must be non null

Review (<u>ID</u>: *integer*, Time: *datetime*, Message: *string*)
- Constraints;
    - All attributes must be non null

Receives (<u>**b_id**</u>: *integer*, <u>**a_id**</u>: *integer*)
- b_id references Broadcast (id)
- a_id references Account (id)
- Constraints;
    - All attributes must be non null

Leaves (<u>**r_id**</u>: *integer*, <u>**a_id**</u>: *integer*)
- r_id references Review (id)
- a_id references Account (id)
- Constraints:

        - All attributes must be non null

Location (<u>locationid</u>: *integer*, description: *string*, streetname: *string*, streetno: *integer*, postalcode: *string*)
- Candidate key: description, streetname, streetno
- Constraints:
    - All attributes must not be null

Category (<u>name</u>: *string*)
- Constraints:
    - All attributes must be non null

Listing (**<u>postid</u>**: *integer*, status: *string*, item: *string*, expiration: *string*)
- postId references Post (id)
- Constraints:
    - status must be one of “open", "pending", "closed", or "expired”
    - if expiration date has passed, must have "expired" status
    - listingid, status, item, and expiration must not be null

Request (**<u>postid</u>**: *integer*, status: *string*, description: *string*, expiration: *string*, type: *string*)
- postId references Post (id)
- Constraints:
- status must be one of “open", "pending", "closed", or "expired”
    - if expiration date has passed, must have "expired" status
    - type must be one of “service", "item", or "information”
    - requestid, status, expiration, type must not be null

Creates (**<u>postid</u>**: *integer*, **accountid**: *integer*, createdon: *timestamp*, updatedon: *timestamp*)
- Constraints:
    - createdon and updatedon default to now
    - All attributes except updatedon must not be null

Fulfills (<u>postid</u>: *integer*, **accountid**: *integer*, updatedon: *timestamp*)
- Constraints:
    - All attributes must not be null

BelongsTo (**<u>postid</u>**: *integer*, <u>category</u>: *string*)

- Constraints:
    - All attributes must not be null

Suggests (**postid**: *integer*, address: *string*, suggestedtime: *datetime*)
- postId references Post (id)
- Constraints:
    - All attributes must not be null

Post (postid: *integer*, type: *string*)
- Constraints:
    - postId must not be null

Non-trivial Functional Dependencies

Location:	StreetName, StreetNo -> PostalCode
Ticket:	Subject -> Priority
Account:	Id -> Name, Password, Email, Status
Bid:	Id -> Message

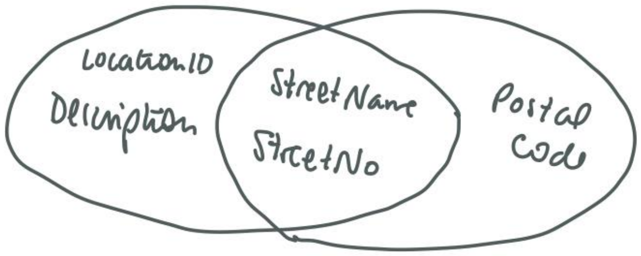
Normalization

Location(<u>LocationID</u>, Description, StreetName, StreetNo, PostalCode)

StreetName, StreetNo $\longrightarrow$ PostalCode

$\therefore$ not in BCNF or 3NF

<u>decomposition</u>



Pickup(<u>LocationID</u>, Description, StreetName, StreetNo)

Address(<u>StreetName</u>, <u>StreetNo</u>, PostalCode)

Tables derived:

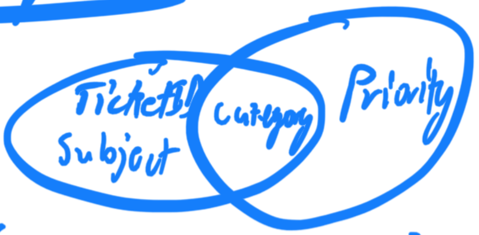
- Pickup (<u>pickupid</u>: *integer*, description: *string*, **streetname**: *string*, **streetno**: *integer*)
- Address (<u>streetname</u>: string, <u>streetno</u>: integer, postalcode: string)

Ticket(TicketID, Subject, Priority, Category)

Category → Priority

∴ not in BCNF or 3NF

decomposition



UrgencyLevel(Category, Priority)

Ticket Details (TicketID, Category, Subject)

Tables derived:

- UrgencyLevel (Category, Priority)
- TicketDetails (Ticket ID, Category, Subject)

SQL to create tables:

```
CREATE TABLE Account(id INTEGER,
          name VARCHAR(30) NOT NULL,
          password VARCHAR(30) NOT NULL,
          email VARCHAR(30) NOT NULL,
          status INTEGER DEFAULT 1,
          PRIMARY KEY(id)
);

INSERT INTO Account VALUES (0, 'aimyul', 'aaaaaaaAAAAAAAAhelpme21321',
'aimyul@student.ubc.ca', 1);
INSERT INTO Account VALUES (1, 'louisecc', 'NoFreeLunchTheorem, cpsc340',
'louisecc@student.ubc.ca', 1);
INSERT INTO Account VALUES (2, 'krish.thrukal', 'iscream,youscream2',
'krish.thrukal@gmail.com', 1);
INSERT INTO Account VALUES (3, 'rapsody', 'P0ttingr?', 'bohem@test.com', 1);
INSERT INTO Account VALUES (4, 'Gerald', 'geoffr3y', 'a.crow@notreal.org', 0);
INSERT INTO Account VALUES (5, 'Penny', 'intherefrigerator', 'baby@baby.com', 1);
INSERT INTO Account VALUES (6, 'Fred', 'caw!', 'fredthescarecrow@cool.com', 1);
INSERT INTO Account VALUES (7, 'yxl', 'OnMars.', 'cool@cool.ca', 0);
INSERT INTO Account VALUES (8, 'dj south', ' Zzz123', 'nutty@elephant.food', 0);

CREATE TABLE Moderator (id INTEGER,
         PRIMARY KEY(id),
         FOREIGN KEY(id) REFERENCES Account
);

INSERT INTO Moderator(id) VALUES (0);
INSERT INTO Moderator(id) VALUES (1);
INSERT INTO Moderator(id) VALUES (2);
INSERT INTO Moderator(id) VALUES (5);
INSERT INTO Moderator(id) VALUES (6);

CREATE TABLE Bid (id INTEGER,
          message VARCHAR(120),
          PRIMARY KEY(id)
);

INSERT INTO Bid VALUES (0, 'would love to pick up');
INSERT INTO Bid VALUES (1, 'my daughter could use a chair like this! she''s 3 years old');
INSERT INTO Bid VALUES (2, 'nice');
INSERT INTO Bid VALUES (3, 'very interesting');
INSERT INTO Bid VALUES (4, 'that''s just lovely');
```

```
CREATE TABLE Makes(aid INTEGER,
        bid INTEGER,
        PRIMARY KEY(aid, bid),
        FOREIGN KEY(aid) REFERENCES Account,
        FOREIGN KEY(bid) REFERENCES Bid
);

INSERT INTO Makes VALUES (3, 0);
INSERT INTO Makes VALUES (4, 2);
INSERT INTO Makes VALUES (6, 3);
INSERT INTO Makes VALUES (3, 1);
INSERT INTO Makes VALUES (6, 4);

CREATE TABLE Resolves(tid Integer,
        mid Integer,
        PRIMARY KEY(tid, mid),
        FOREIGN KEY(tid) REFERENCES Ticket,
        FOREIGN KEY(mid) REFERENCES Moderator
);

INSERT INTO Resolves(tid, mid) VALUES (0, 0);
INSERT INTO Resolves(tid, mid) VALUES (1, 0);
INSERT INTO Resolves(tid, mid) VALUES (2, 1);
INSERT INTO Resolves(tid, mid) VALUES (3, 5);
INSERT INTO Resolves(tid, mid) VALUES (4, 1);

CREATE TABLE Suspends(aid Integer,
        mid Integer NOT NULL,
        PRIMARY KEY(aid),
        FOREIGN KEY(aid) REFERENCES Account
);

INSERT INTO Suspends VALUES (3, 5);
INSERT INTO Suspends VALUES (4, 5);
INSERT INTO Suspends VALUES (6, 5);
INSERT INTO Suspends VALUES (7, 5);
INSERT INTO Suspends VALUES (8, 0);

CREATE TABLE Chooses(aid Integer,
        bid Integer,
        PRIMARY KEY(aid, bid),
        FOREIGN KEY(aid) REFERENCES Account,
        FOREIGN KEY(bid) REFERENCES Bid
);

INSERT INTO Chooses VALUES (3, 4);
INSERT INTO Chooses VALUES (4, 4);
INSERT INTO Chooses VALUES (6, 0);
INSERT INTO Chooses VALUES (7, 2);
INSERT INTO Chooses VALUES (8, 1);

CREATE TABLE Ticket (
        t_id Integer not null,
        t_subject char(50),
        t_category char(11),
        t_priority boolean,
        PRIMARY KEY(t_id)
);

INSERT INTO Ticket VALUES (3, “bad customer service”,”customer service”,0);
INSERT INTO Ticket VALUES (4, “almost got stabbed”,”item quality”,1);
INSERT INTO Ticket VALUES (5, “almost ran over cat”,”good item quality”,1);
INSERT INTO Ticket VALUES (6, “ate a bagel”,”false information”,0);
INSERT INTO Ticket VALUES (7, “horrible ui”,”bad item quality”,1);

create table Flag (
        f_id Integer  not null,
        f_date char(20)
        f_description char(200)
        PRIMARY KEY(f_id)
);

INSERT INTO Flag VALUES (3, 1985-06-12, “bad behavior”);
INSERT INTO Flag VALUES (4, 1985-06-13, “selling drugs”);
INSERT INTO Flag VALUES (6, 1985-06-14, “against community guidelines”);
INSERT INTO Flag VALUES (7, 1985-06-15, “unusual post”);
INSERT INTO Flag VALUES (8, 1985-06-16, “hate speech”);

create table Broadcast (
        b_id Integer  not NULL,
        b_message char(200),
        B_date char(20),
        b_time char (11),
        PRIMARY KEY(b_id)
);
```

```
INSERT INTO Broadcast(1, “hello partner”, 2021-10-30 12:00:00);
INSERT INTO Broadcast(2, “today is a great day”, 2021-11-05 15:00:00);
INSERT INTO Broadcast(4, “stop posting stuff”, 2021-10-30 15:30:00');
INSERT INTO Broadcast(5, “start posting stuff”, '2021-10-30 15:45:00');
INSERT INTO Broadcast(8, “the weather is good isn't it”, '2021-10-30 15:00:00');
INSERT INTO Broadcast(9, “welcome to our interface”, '2021-10-30 15:00:00');

create table Review
        (r_id Integer not NULL,
        r_description char(200)
        r_time datetime(11)
        PRIMARY KEY(r_id));

INSERT INTO Review VALUES (6, “best product”, 2021-10-30 12:00:00);
INSERT INTO Review VALUES (10,, “worst product”, 2021-11-05 15:00:00);
INSERT INTO Review VALUES (6,  “all right product”, 2021-10-30 15:30:00);
INSERT INTO Review VALUES (8, “too far”, 2021-10-30 15:45:00);
INSERT INTO Review VALUES (9, “nice and close”, 2021-10-30 15:00:00);

create table Receives (
        b_id char(11) not NULL,
        a_id char(11) not NULL,
        PRIMARY KEY(b_id,a_id),
        FOREIGN KEY(b_id) references Broadcast,
        FOREIGN KEY(a_id) references Account
);

INSERT INTO Receives VALUES (3, 4);
INSERT INTO Receives VALUES (4, 4);
INSERT INTO Receives VALUES (6, 0);
INSERT INTO Receives VALUES (7, 2);
INSERT INTO Receives VALUES (8, 1);

create table Leaves (
        r_id Integer not NULL,
        a_id Integer not NULL,
        PRIMARY KEY(r_id,a_id),
        FOREIGN KEY(r_id) references Review,
	FOREIGN KEY(a_id) references Account
);

INSERT INTO Leaves VALUES (3, 4);
INSERT INTO Leaves VALUES (4, 4);
INSERT INTO Leaves VALUES (6, 0);
INSERT INTO Leaves VALUES (7, 2);
INSERT INTO Leaves VALUES (8, 1);

CREATE TABLE Category (
	name VARCHAR(20) PRIMARY KEY NOT NULL
);

CREATE TABLE Post (
	postid INTEGER PRIMARY KEY NOT NULL,
	type VARCHAR(10) NOT NULL
;)

CREATE TABLE Listing (
	postid INTEGER FOREIGN KEY References Post(postid),
	status VARCHAR(10) NOT NULL,
	item VARCHAR(50) NOT NULL,
	expiration DATETIME NOT NULL,
	PRIMARY KEY postid
);

CREATE TABLE Request (
	postid INTEGER FOREIGN KEY References Post(postid) NOT NULL,
	status VARCHAR(10) NOT NULL,
	description VARCHAR(50) NOT NULL,
	expiration DATETIME NOT NULL,
	type VARCHAR(10) NOT NULL,
	PRIMARY KEY postid
);

CREATE TABLE Creates (
	postid INTEGER FOREIGN KEY References Post(postid) NOT NULL,
	accountid INTEGER FOREIGN KEY References Account(accountid) NOT NULL,
	createdon TIMESTAMP NOT NULL DEFAULT NOW,
	updatedon TIMESTAMP NOT NULL DEFAULT NOW,
	PRIMARY KEY (postid, accountid)
);
```

```
CREATE TABLE Fulfills (
        postid INTEGER FOREIGN KEY References Post(postid) NOT NULL,
        accountid INTEGER FOREIGN KEY References Account(accountid) NOT NULL,
        updatedon TIMESTAMP NOT NULL DEFAULT NOW,
        PRIMARY KEY (postid, accountid)
);

CREATE TABLE BelongsTo (
        postid FOREIGN KEY References Post(postid) NOT NULL,
        category VARCHAR(20) NOT NULL,
        PRIMARY KEY (postid, category)
);

CREATE TABLE Suggests (
        postid FOREIGN KEY References Post(postid) NOT NULL,
        pickupid FOREIGN KEY References Pickup(pickupid) NOT NULL,
        suggestedtime DATETIME NOT NULL,
        PRIMARY KEY (postid, pickupid, suggestedtime)
);

CREATE TABLE Address (
        streetname VARCHAR(30) NOT NULL,
        streetno INTEGER,
        postalcode VARCHAR(6) NOT NULL,
        PRIMARY KEY (streetname, streetno)
);

CREATE TABLE Pickup (
        pickupid INTEGER PRIMARY KEY NOT NULL,
        description VARCHAR(100) NOT NULL,
        streetname FOREIGN KEY References Address(streetname) NOT NULL,
        streetno FOREIGN KEY References Address(streetno)
);

INSERT INTO Category VALUES('Furniture');
INSERT INTO Category VALUES('Clothing');
INSERT INTO Category VALUES('Electronics');
INSERT INTO Category VALUES('Food');
INSERT INTO Category VALUES('Books');
INSERT INTO Category VALUES('Musical equipment');
INSERT INTO Category VALUES('Sports equipment');
INSERT INTO Category VALUES('Coupons');

INSERT INTO Post VALUES(1, 'Listing');
INSERT INTO Post VALUES(2, 'Listing');
INSERT INTO Post VALUES(3, 'Request');
INSERT INTO Post VALUES(4, 'Listing');
INSERT INTO Post VALUES(5, 'Listing');
INSERT INTO Post VALUES(6, 'Request');
INSERT INTO Post VALUES(7, 'Request');
INSERT INTO Post VALUES(8, 'Listing');
INSERT INTO Post VALUES(9, 'Listing');

INSERT INTO Listing VALUES(1, 'Open', 'JBL headphones', 2021-11-15 12:00:00);
INSERT INTO Listing VALUES(2, 'Closed', 'Aritzia sweater', 2021-10-15 12:00:00);
INSERT INTO Listing VALUES(4, 'Pending', 'Desk', 2021-11-21 12:00:00);
INSERT INTO Listing VALUES(5, 'Pending', 'Dune', 2021-11-22 12:00:00);
INSERT INTO Listing VALUES(8, 'Open', '10% off at Pizza Garden', 2021-12-23 12:00:00);
INSERT INTO Listing VALUES(9, 'Open', 'Quest bars', 2021-12-24 12:00:00);

INSERT INTO Request VALUES(3, 'Open', 'Toaster oven', 2021-11-15 12:00:00, 'Item');
INSERT INTO Request VALUES(6, 'Closed', 'Truck for moving', 2021-10-15 12:00:00,
'Service');
INSERT INTO Request VALUES(7, 'Pending', 'Thinking Fast and Slow', 2021-11-21 12:00:00,
'Item');
INSERT INTO Request VALUES(10, 'Open', 'Where can I find a good mechanic?',
2021-11-22 12:00:00, 'Information');
INSERT INTO Request VALUES(11, 'Open', 'Free haircut on campus', 2021-12-23 12:00:00,
'Service');

INSERT INTO Creates VALUES(1, 3);
INSERT INTO Creates VALUES(2, 3);
INSERT INTO Creates VALUES(3, 3);
INSERT INTO Creates VALUES(5, 4);
INSERT INTO Creates VALUES(11, 4);

INSERT INTO Fulfills VALUES(1, 5);
INSERT INTO Fulfills VALUES(2, 5);
INSERT INTO Fulfills VALUES(3, 2);
INSERT INTO Fulfills VALUES(5, 1);
INSERT INTO Fulfills VALUES(11, 2);

INSERT INTO BelongsTo VALUES(1, 'Electronics');
INSERT INTO BelongsTo VALUES(2, 'Clothing');
INSERT INTO BelongsTo VALUES(3, 'Electronics');
INSERT INTO BelongsTo VALUES(4, 'Furniture');
INSERT INTO BelongsTo VALUES(5, 'Books');
INSERT INTO BelongsTo VALUES(6, 'Transportation');
INSERT INTO BelongsTo VALUES(7, 'Books');
INSERT INTO BelongsTo VALUES(8, 'Coupons');
INSERT INTO BelongsTo VALUES(9, 'Food');
INSERT INTO BelongsTo VALUES(10, 'Transportation');
INSERT INTO BelongsTo VALUES(11, 'Other');

INSERT INTO Suggests(1, 1, '2021-10-30 12:00:00');
INSERT INTO Suggests(2, 4, '2021-11-05 15:00:00');
INSERT INTO Suggests(4, 3, '2021-10-30 15:30:00');
INSERT INTO Suggests(5, 2, '2021-10-30 15:45:00');
INSERT INTO Suggests(8, 1, '2021-10-30 15:00:00');
INSERT INTO Suggests(9, 5, '2021-10-30 15:00:00');

INSERT INTO Address VALUES('University Boulevard', 6133, 'V6T1Z1');
INSERT INTO Address VALUES('University Boulevard', 6200, 'V6T1Z3');
INSERT INTO Address VALUES('Main Mall', 2366, 'V6T1Z4');
INSERT INTO Address VALUES('Crescent Road', 6301, 'V6T1Z2');
INSERT INTO Address VALUES('Walter Gage Road', 6011, 'V6T0B4');
INSERT INTO Address VALUES('University Boulevard', 6131, 'V6T2A1');

INSERT INTO Pickup VALUES(1, 'in front of Honour Roll', 'University Boulevard', 6133);
INSERT INTO Pickup VALUES(2, 'in fourth floor lounge', 'University Boulevard', 6133);
INSERT INTO Pickup VALUES(3, 'Starbucks', 'University Boulevard', 6200);
INSERT INTO Pickup VALUES(4, 'ICICS, west doors', 'Main Mall', 2366);
INSERT INTO Pickup VALUES(5, 'UBC Rose Garden', 'Crescent Road', 6301);
INSERT INTO Pickup VALUES(6, 'Great Dane Coffee', 'Walter Gage Road', 6011);
INSERT INTO Pickup VALUES(7, 'MacInnes Field', 'University Boulevard', 6131);
```

Necessary Queries:

1. INSERT Operation - Provide an interface for the user to specify some input for the insert operation.
    a. User inserts a listing that they make.

2. DELETE Operation - Implement a cascade-on-delete situation (or an alternative that was agreed to by the TA if the DB system doesn't provide this). Provide an interface for the user to specify some input for the deletion operation.
    a. User deletes a bid they've posted.

3. UPDATE Operation - Provide an interface for the user to specify some 1 input for the update operation.

    a. User updates status of one of their listings.

4. SELECTION - Create one query of this category and provide an interface for the user to specify the values of the selection conditions to be returned. Example:
   SELECT …
   FROM …
   WHERE Field1 = :Var1 AND Field2 > :Var2

- User searches for posts of a certain category
- SELECT * FROM POST WHERE cid = 2;

5. PROJECTION
- Create one query of this category and provide an interface for the user to specify the values of the selection conditions to be returned. Example:
  SELECT …
  FROM …
  WHERE Field1 = :Var1 AND Field2 > :Var2

- User can look up other users by name/email
- SELECT * FROM Account WHERE email LIKE '%@gmail.com';

6. JOIN Operation
- Create one query in this category, which joins at least 2 tables and performs a meaningful query, and provide an interface for the user to execute this query. The user must provide at least one value to qualify in the WHERE clause (e.g. join the Customer and the Transaction table to find the names and phone numbers of all customers who have purchased a specific item).

- Joining accounts and flags to see how many a user (with a certain name or email) has

7. Aggregation with Group By
- Create one query that requires the use of distinct aggregation (min, max, average, or count are all fine), and provide an interface (e.g., HTML button/dropdown, etc.) for the user to execute this query.

- Count number of requests a user has fulfilled

8. Aggregation with Having
- Create one meaningful query that requires the use of a HAVING clause, and provide an interface (e.g., HTML 1 button/dropdown, etc.) for the user to execute this query.

- Having a certain expiration date

9. Queries: Nested Aggregation with Group By
- Create one query that finds some aggregated value for each group (e.g., use a nested subquery, such as finding the average number of items purchased per customer, subject to some constraint). Some examples for the Sailors table are given in the project specs. Note the difference between this query and the above Aggregation Query. You must use separate distinct queries for this criterion and the Aggregation Query (i.e., do not double dip).

- Within users who have posted bids on their item, a poster can filter by review over 4 stars

10. Queries: Division
- Create one query of this category and provide an interface (i.e., HTML button, etc.) for the user to execute this query (e.g., find all the customers who bought all the items).

- Find all the users who made a bid on a listing