University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 15 - 10 - 2024

Group Number: 60

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Jennifer Tanojo	40026460	b7a5y	jennifer.olive.tan@gmail.com
Christopher Lew	87799201	a8t6b	christopherlew4@gmail.com
Jennifer Wang	72344476	f9p0l	jennifer26wang@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

Milestone 2 - Relational Schema, Normalization, SQL DDL, Tentative Queries

A brief project description:

a) Domain of the application

The domain of the application is food journaling and restaurant interaction, combining elements of personal food documentation, restaurant review and waitlist platforms, and online food services.

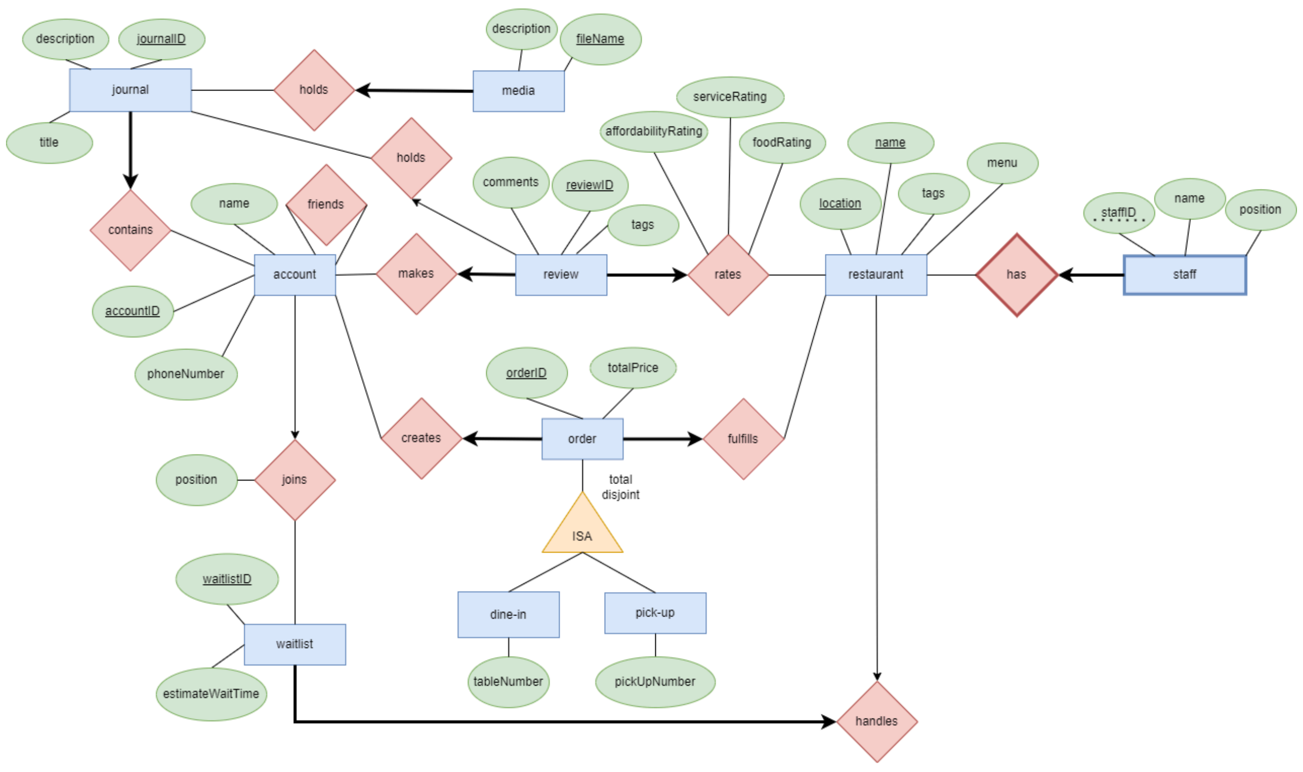
b) Aspects of the domain modeled by the database

The application's goal is to serve as a comprehensive food experience platform, where users can not only document their food experiences but also interact with other users and restaurants in various ways. This project is best understood as a personal food journaling and interaction platform, where users can not only document and share their dining experiences but also have restaurant-related activities in ways that enhance their journaling: leaving reviews, creating a food bucket list, ordering food online, and joining a waitlist for dine-in.

ER Diagram

We have made a few minor changes to the ER diagram since Milestone 1 to better accommodate our vision for this project, which we will list down below:

- We have established a participation constraint for Media in Journal, which states a Media must have a Journal.
- We also established a participation constraint for the Waitlist, such that one has to correspond to a Restaurant.



The Schema Derived from the ER diagram (above)

NOTE: Going forward, underlined attributes (___) represent Primary Keys (PK).

- **All Primary Keys are Candidate Keys**

Table definitions with specified keys:

- **Account**: Account(accountID : INTEGER, phoneNumber : INTEGER, name : VARCHAR)
- **Journal**: Journal(journalID : INTEGER, description : VARCHAR, title : VARCHAR, accountID (FK, NOT NULL) : INTEGER)
- **Friends**: Friends(accountID (FK) : INTEGER, friendID (FK) : INTEGER)
- **Review**: Review(reviewID : INTEGER, comments : VARCHAR, tags : VARCHAR, journalID (FK) : INTEGER, accountID (FK, NOT NULL) : INTEGER, restaurantName (FK, NOT NULL) : VARCHAR, restaurantLocation (FK, NOT NULL) : VARCHAR)
- **Rates**: Rates(foodRating : INTEGER, serviceRating : INTEGER, affordabilityRating : INTEGER, reviewID (FK, NOT NULL) : INTEGER, restaurantName (FK, NOT NULL) : VARCHAR, restaurantLocation (FK, NOT NULL) : VARCHAR)
- **Media**: Media(filename : VARCHAR, description : VARCHAR, journalID (FK, NOT NULL) : INTEGER)
- **DineInOrder**: DineInOrder(orderID : INTEGER, totalPrice : INTEGER, tableNumber : INTEGER, accountID (FK, NOT NULL): INTEGER, restaurantName (FK, NOT NULL) : VARCHAR, restaurantLocation (FK, NOT NULL) : VARCHAR)
- **PickupOrder**: PickupOrder(orderID : INTEGER, totalPrice : INTEGER, pickupNumber (UNIQUE): INTEGER, accountID (FK, NOT NULL) : INTEGER, restaurantName (FK, NOT NULL) : VARCHAR, restaurantLocation (FK, NOT NULL) : VARCHAR)
- **Restaurant**: Restaurant(name : VARCHAR, location : VARCHAR, cuisineTag : VARCHAR, menu : VARCHAR, waitlistID (FK, UNIQUE): INTEGER)
- **Restaurant_Staff**: Restaurant_Staff(restaurantName (FK) : VARCHAR, restaurantLocation (FK) : VARCHAR, staffID : INTEGER, name : VARCHAR, position : VARCHAR)
- **Waitlist**: Waitlist(waitlistID : INTEGER, estimateWaitTime : INTEGER, restaurantName (FK, UNIQUE, NOT NULL) : VARCHAR, restaurantLocation (FK, UNIQUE, NOT NULL) : VARCHAR)
- **Joins**: Joins(position : INTEGER, accountID (FK) : INTEGER, waitlistID (FK) : INTEGER)

Functional Dependencies (FDs)
For our design, we intuitively have the following FDs:

Account:
- accountID → phoneNumber, name
- phoneNumber → name

Journal:
- journalID → title, description, accountID
- title → description

Friends:
- The only FD in this relation is the trivial one(s):
  accountID, friendID → accountID, friendID
- *NOTE:* Friends does not have a non-PK/CK FD because it does not intuitively occur to us that we can add more meaningful attributes to this relation seeing as all we want to represent is that an account would have a friend and vice versa.

Review:
- reviewID → comments, tags, journalID, accountID, restaurantName, restaurantLocation
- journalID → tags, accountID

Rates:
- reviewID, restaurantName, restaurantLocation → foodRating, serviceRating, affordabilityRating
- *NOTE:* We wanted each part of our primary key to be unique as a trio, where this relation represents the restaurant's access to review the ratings under a given review.

Media:
- filename → description, journalID
- *NOTE*: Media does not have a non-PK/CK FD because it does not intuitively occur to us that we can add more meaningful attributes to this relation. As we solely wish to represent that a media object can be added to a journal and a description may be added.

DineInOrder:
- orderID → totalPrice, tableNumber, accountID, restaurantName, restaurantLocation
- *NOTE*: orderID and tableNumber are different as orderID is an ID for all food orders and tableNumber is about the staff delivering the food to the right table. Hence we cannot necessarily create another FD.

PickupOrder:

- orderID → totalPrice, pickupNumber, accountID, restaurantName, restaurantLocation
- *NOTE:* orderID and pickupNumber are different, and we cannot necessarily make another FD in this case, since the other attributes do not correlate with each other (e.g. orderID is for the app to ensure that we can refer to the order uniquely, and the pickupNumber is for reference when the user goes to the restaurant to pick-up their order).

Restaurant:

- name, location → cuisineTag, menu, waitlistID
- menu → cuisineTag

Restaurant_Staff:

- restaurantName, restaurantLocation, staffID → name, position
- staffID → name

Waitlist:

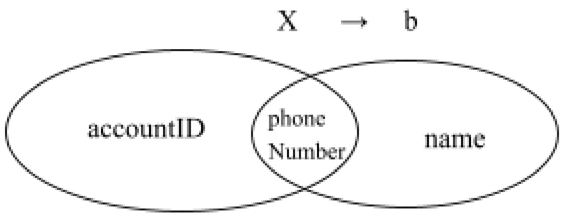
- waitlistID → estimateWaitTime, restaurantName, restaurantLocation
- restaurantName, restaurantLocation → estimateWaitTime

Joins:

- accountID, waitlistID → position
- *NOTE*: Joins does not have a non-PK/CK FD because it does not intuitively occur to us that we can add more meaningful attributes to this relation. As we only wish to represent an account can join a waitlist and have a position in the waitlist.

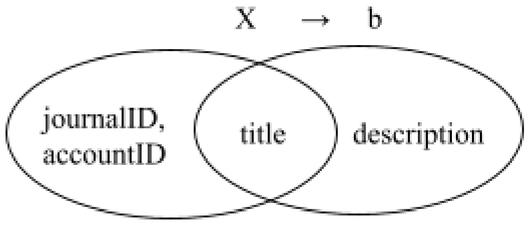
Normalization
From the listed Functional Dependencies (FDs) above, we see that Account, Journal, Review, Restaurant, Restaurant_Staff, and Waitlist are not in BCNF. We then do the necessary normalization below:

Account: Account(accountID, phoneNumber, name)
phoneNumber → name violates BCNF since phoneNumber is not a superkey.
Decomposing on phoneNumber → name:



Account1(phoneNumber, name)
Account2(accountID, phoneNumber)
Since both relationships have 2 attributes, they do not violate BCNF.

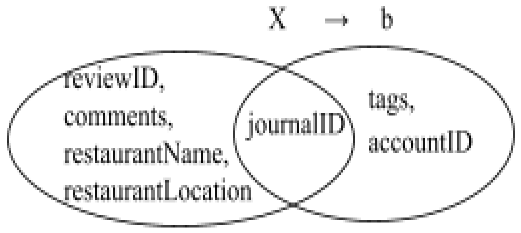
Journal: Journal(journalID, title, description, accountID)
We notice that the FD title → description violates BCNF since title is not a superkey. We decompose on title → description:



We have: Journal1(journalID, accountID, title)
Journal2(title, description)
They no longer violate the BCNF.

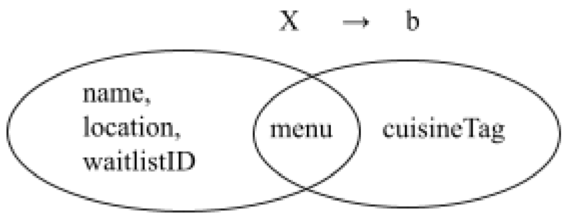
Review: Review(<u>reviewID</u>, comments, tags, journalID, accountID, restaurantName, restaurantLocation)

Notice that journalID → tags, accountID violates BCNF, since journalID not is a superkey. We decompose on this FD:



Now we have: Review1(<u>reviewID</u>, comments, journalID, restaurantName, restaurantLocation)
Review2(<u>journalID</u>, tags, accountID)
They no longer violate the BCNF.

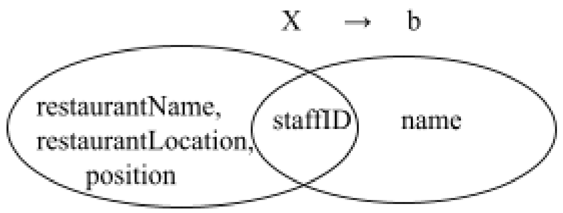
Restaurant: Restaurant(<u>name</u>, <u>location</u>, cuisineTag, menu, waitlistID)
menu → cuisineTag violates BCNF since menu is not a superkey.
Decomposing on menu → cuisineTag:



Restaurant1(<u>menu</u>, cuisineTag)
Restaurant2(<u>name</u>, <u>location</u>, waitlistID)
These relationships no longer violate BCNF.

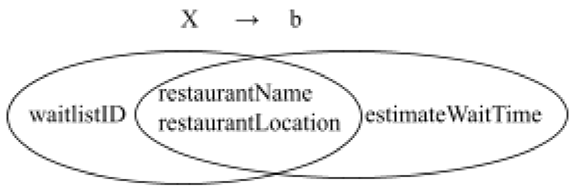
Restaurant_Staff: Restaurant_Staff (restaurantName, restaurantLocation, staffID, name, position)

We notice that the FD staffID → name violates BCNF since staffID is not a superkey. We decompose on staffID → name:



We have: Restaurant_Staff1(restaurantName, restaurantLocation, staffID, position)
Restaurant_Staff2(staffID, name)
They no longer violate the BCNF.

Waitlist: Waitlist(waitlistID, estimateWaitTime, restaurantName, restaurantLocation)
Observe that the FD restaurantName, restaurantLocation → estimateWaitTime violates BCNF since restaurantName and restaurantLocation together is not a superkey. We decompose on the FD:



We have: Waitlist1(restaurantName, restaurantLocation, waitlistID)
Waitlist2(estimateWaitTime, restaurantName, restaurantLocation)
They no longer violate the BCNF.

All tables listed properly, with domains, post normalization:

- **Account1**: Account1(phoneNumber : INTEGER, name : VARCHAR)
- **Account2**: Account2(accountID : INTEGER, phoneNumber : INTEGER)
- **Journal1**: Journal1(journalID : INTEGER, accountID (FK, NOT NULL) : INTEGER, title : VARCHAR)
- **Journal2**: Journal2(title : VARCHAR, description : VARCHAR)
- **Friends**: Friends(accountID (FK) : INTEGER, friendID (FK) : INTEGER)
  Review2(journalID, tags, accountID)
- **Review1**: Review1(reviewID : INTEGER, comments : VARCHAR, journalID (FK) : INTEGER, restaurantName (FK, NOT NULL) : VARCHAR, restaurantLocation (FK, NOT NULL) : VARCHAR)
- **Review2**: Review2(journalID (FK) : INTEGER, tags : VARCHAR, accountID (FK, NOT NULL) : INTEGER)
- **Rates**: Rates(foodRating : INTEGER, serviceRating : INTEGER, affordabilityRating : INTEGER, reviewID (FK, NOT NULL) : INTEGER, restaurantName (FK, NOT NULL) : VARCHAR, restaurantLocation (FK, NOT NULL) : VARCHAR)
- **Media**: Media(filename : VARCHAR, description : VARCHAR, journalID (FK, NOT NULL) : INTEGER)
- **DineInOrder**: DineInOrder(orderID : INTEGER, totalPrice : INTEGER, tableNumber : INTEGER, accountID (FK, NOT NULL): INTEGER, restaurantName (FK, NOT NULL) : VARCHAR, restaurantLocation (FK, NOT NULL) : VARCHAR)
- **PickupOrder**: PickupOrder(orderID : INTEGER, totalPrice : INTEGER, pickupNumber (UNIQUE): INTEGER, accountID (FK, NOT NULL) : INTEGER, restaurantName (FK, NOT NULL) : VARCHAR, restaurantLocation (FK, NOT NULL) : VARCHAR)
- **Restaurant1**: Restaurant1(menu : VARCHAR, cuisineTag : VARCHAR)
- **Restaurant2**: Restaurant2(name : VARCHAR, location, waitlistID (FK, UNIQUE) : INTEGER)
- **Restaurant_Staff1**: Restaurant_Staff1(restaurantName (FK) : VARCHAR, restaurantLocation (FK) : VARCHAR, staffID : INTEGER, position : VARCHAR)
- **Restaurant_Staff2**: Restaurant_Staff2(staffID : INTEGER, name : VARCHAR)
- **Waitlist1**: Waitlist1(waitlistID : INTEGER, restaurantName (FK, UNIQUE, NOT NULL) : VARCHAR, restaurantLocation (FK, UNIQUE, NOT NULL) : VARCHAR)
- **Waitlist2**: Waitlist2(restaurantName (FK, UNIQUE, NOT NULL) : VARCHAR, restaurantLocation (FK, UNIQUE, NOT NULL) : VARCHAR, estimateWaitTime : INTEGER)
- **Joins**: Joins(position : INTEGER, accountID (FK) : INTEGER, waitlistID (FK) : INTEGER)

SQL DDL Statements to Create the Tables Above

```
CREATE TABLE Account1 (
        phoneNumber INTEGER  PRIMARY KEY,
        name VARCHAR
)

CREATE TABLE Account2 (
        accountID INTEGER  PRIMARY KEY,
        phoneNumber INTEGER
)

CREATE TABLE Journal1 (
        journalID INTEGER  PRIMARY KEY,
        title VARCHAR,
        accountID INTEGER NOT NULL,
        FOREIGN KEY (accountID) REFERENCES Account2(accountID)
)

CREATE TABLE Journal2 (
        title VARCHAR  PRIMARY KEY,
        description VARCHAR
)

CREATE TABLE Friends (
        accountID INTEGER,
        friendID INTEGER,
        PRIMARY KEY (accountID, friendID),
        FOREIGN KEY (accountID) REFERENCES Account2(accountID),
        FOREIGN KEY (friendID) REFERENCES Account2(accountID)
)

CREATE TABLE Review1 (
        comments VARCHAR,
        reviewID INTEGER  PRIMARY KEY,
        journalID INTEGER,
        restaurantName VARCHAR  NOT NULL,
        restaurantLocation VARCHAR NOT NULL,
        FOREIGN KEY (journalID) REFERENCES Journal1(journalID),
	FOREIGN KEY (restaurantName, restaurantLocation) REFERENCES
	Restaurant2(name, location)
)

CREATE TABLE Review2 (
	tags VARCHAR,
	journalID INTEGER PRIMARY KEY,
	accountID INTEGER  NOT NULL,
	FOREIGN KEY (journalID) REFERENCES Journal1(journalID),
	FOREIGN KEY (accountID) REFERENCES Account2(accountID)
)

CREATE TABLE Rates (
	foodRating INTEGER,
	serviceRating INTEGER,
	affordabilityRating INTEGER,
	reviewID INTEGER NOT NULL,
	restaurantName VARCHAR  NOT NULL,
	restaurantLocation VARCHAR NOT NULL,
	PRIMARY KEY (reviewID, restaurantName, restaurantLocation),
	FOREIGN KEY (reviewID) REFERENCES Review2(reviewID),
	FOREIGN KEY (restaurantName, restaurantLocation) REFERENCES
	Restaurant2(name, location)
)

CREATE TABLE Media (
	filename VARCHAR  PRIMARY KEY,
	description VARCHAR,
	journalID INTEGER  NOT NULL,
	FOREIGN KEY (journalID) REFERENCES Journal1(journalID)
)

CREATE TABLE DineInOrder (
	orderID INTEGER  PRIMARY KEY,
	totalPrice INTEGER,
	tableNumber INTEGER,
	accountID INTEGER NOT NULL,
	restaurantName VARCHAR  NOT NULL,
	restaurantLocation VARCHAR NOT NULL,
	FOREIGN KEY (accountID) REFERENCES Account2(accountID),
        FOREIGN KEY (restaurantName, restaurantLocation) REFERENCES
        Restaurant2(name, location)
)

CREATE TABLE PickupOrder (
        orderID INTEGER  PRIMARY KEY,
        totalPrice INTEGER,
        pickupNumber INTEGER,
        accountID INTEGER  NOT NULL,
        restaurantName VARCHAR  NOT NULL,
        restaurantLocation VARCHAR NOT NULL,
        FOREIGN KEY (accountID) REFERENCES Account2(accountID),
        FOREIGN KEY (restaurantName, restaurantLocation) REFERENCES
        Restaurant2(name, location)
)

CREATE TABLE Restaurant1 (
        cuisineTag VARCHAR,
        menu VARCHAR PRIMARY KEY
)

CREATE TABLE Restaurant2 (
        name VARCHAR,
        location VARCHAR,
        waitlistID INTEGER UNIQUE,
        PRIMARY KEY (name, location),
        FOREIGN KEY (waitlistID) REFERENCES Waitlist1(waitlistID)
)

CREATE TABLE Restaurant_Staff1 (
        restaurantName VARCHAR,
        restaurantLocation VARCHAR,
        staffID INTEGER,
        position VARCHAR,
        PRIMARY KEY (restaurantName, restaurantLocation, staffID)
        FOREIGN KEY (restaurantName, restaurantLocation) REFERENCES
        Restaurant2(name, location), ON DELETE CASCADE
)
```

```
CREATE TABLE Restaurant_Staff2 (
        staffID INTEGER PRIMARY KEY,
        name VARCHAR
)

CREATE TABLE Waitlist1 (
        waitlistID INTEGER  PRIMARY KEY,
        restaurantName VARCHAR NOT NULL,
        restaurantLocation VARCHAR NOT NULL,
        UNIQUE restaurantName,
        UNIQUE restaurantLocation,
        FOREIGN KEY (restaurantName, restaurantLocation) REFERENCES
        Restaurant2(name, location)
)

CREATE TABLE Waitlist2 (
        estimateWaitTime INTEGER,
        restaurantName VARCHAR NOT NULL,
        restaurantLocation VARCHAR NOT NULL,
        PRIMARY KEY (restaurantName, restaurantLocation),
        UNIQUE restaurantName,
        UNIQUE restaurantLocation,
        FOREIGN KEY (restaurantName, restaurantLocation) REFERENCES
        Restaurant2(name, location)
)

CREATE TABLE Joins (
        position INTEGER,
        accountID INTEGER,
        waitlistID INTEGER,
        PRIMARY KEY (accountID, waitlistID),
        FOREIGN KEY (accountID) REFERENCES Account2(accountID),
        FOREIGN KEY (waitlistID) REFERENCES Waitlist1(waitlistID)
)
```

INSERT Statements to Populate Each Table with At Least 5 Tuples

Account1:

```
INSERT
INTO Account1(phoneNumber, name)
VALUES (‘1234567890’, ‘A’)
INSERT
INTO Account1(phoneNumber, name)
VALUES (‘2345678901’, ‘B’)
INSERT
INTO Account1(phoneNumber, name)
VALUES (‘3456789012’, ‘C’)
INSERT
INTO Account1(phoneNumber, name)
VALUES (‘4567890123’, ‘D’)
INSERT
INTO Account1(phoneNumber, name)
VALUES (‘5678901234’, ‘E’)
```

Account2:

```
INSERT
INTO Account2(accountID, phoneNumber)
VALUES (‘0001’, ‘1234567890’)
INSERT
INTO Account2(accountID, phoneNumber)
VALUES (‘0002’, ‘2345678901’)
INSERT
INTO Account2(accountID, phoneNumber)
VALUES (‘0003’, ‘3456789012’)
INSERT
INTO Account2(accountID, phoneNumber)
VALUES (‘0004’, ‘4567890123’)
INSERT
INTO Account2(accountID, phoneNumber)
VALUES (‘0005’, ‘5678901234’)
```

Journal1:

```
INSERT
INTO Journal1(journalID, title, accountID)
VALUES (‘1’, ‘test1’, ‘0001’)
INSERT
INTO Journal1(journalID, title, accountID)
VALUES (‘2’, ‘test2’, ‘0001’)
INSERT
INTO Journal1(journalID, title, accountID)
VALUES (‘3’, ‘test3’, ‘0001’)
INSERT
INTO Journal1(journalID, title, accountID)
VALUES (‘4’, ‘test4’, ‘0002’)
INSERT
INTO Journal1(journalID, title, accountID)
VALUES (‘5’, ‘test5’, ‘0003’)
```

Journal2:

```
INSERT
INTO Journal2(title, description)
VALUES (‘test1’, ‘this is test1’s description’)
INSERT
INTO Journal2(title, description)
VALUES (‘test2’, ‘this is test2’s description’)
INSERT
INTO Journal2(title, description)
VALUES (‘test3’, ‘this is test3’s description’)
INSERT
INTO Journal2(title, description)
VALUES (‘test4’, ‘this is test4’s description’)
INSERT
INTO Journal2(title, description)
VALUES (‘test5’, ‘this is test5’s description’)
```

Friends:

```
INSERT
INTO Friends(accountID, friendID)
VALUES(‘0001’, ‘0002’)
INSERT
INTO Friends(accountID, friendID)
VALUES(‘0003’, ‘0004’)
INSERT
INTO Friends(accountID, friendID)
VALUES(‘0005’, ‘0006’)
INSERT
INTO Friends(accountID, friendID)
VALUES(‘0006’, ‘0001’)
INSERT
INTO Friends(accountID, friendID)
VALUES(‘0003’, ‘0005’)
```

Review1:

```
INSERT
INTO Review1(comments, reviewID, journalID, restaurantName, restaurantLocation)
VALUES (‘test1’s comment’, ‘123’, ‘1’, ‘test1’s Food’, ‘test1’s location’)
INSERT
INTO Review1(comments, reviewID, journalID, restaurantName, restaurantLocation)
VALUES (‘test2’s comment’, ‘234’, ‘2’, ‘test2’s Food’, ‘test2’s location’)
INSERT
INTO Review1(comments, reviewID, journalID, restaurantName, restaurantLocation)
VALUES (‘test3’s comment’, ‘345’, ‘3’, ‘test3’s Food’, ‘test3’s location’)
INSERT
INTO Review1(comments, reviewID, journalID, restaurantName, restaurantLocation)
VALUES (‘test4’s comment’, ‘456’, ‘4’, ‘test4’s Food’, ‘test4’s location’)
INSERT
INTO Review1(comments, reviewID, journalID, restaurantName, restaurantLocation)
VALUES (‘test5’s comment’, ‘567’, ‘5’, ‘test5’s Food’, ‘test5’s location’)
```

Review2:

```
INSERT
INTO Review2(tags, journalID, accountID)
VALUES (‘American’, ‘1’, ‘0001’)
INSERT
INTO Review2(tags, journalID, accountID)
VALUES (‘Seafood’, ‘2’, ‘0004’)
INSERT
INTO Review2(tags, journalID, accountID)
VALUES (‘Italian’, ‘3’, ‘0003’)
INSERT
INTO Review2(tags, journalID, accountID)
VALUES (‘Japanese’, ‘4’, ‘0001’)
INSERT
INTO Review2(tags, journalID, accountID)
VALUES (‘American’, ‘5’, ‘0002’)
```

Rates:

```
INSERT
INTO Rates(foodRating, serviceRating, affordabilityRating, reviewID, restaurantName,
restaurantLocation)
VALUES(‘3’, ‘4’, ‘5’, ‘123’, ‘test1’s Food’, ‘test1’s location’)
INSERT
INTO Rates(foodRating, serviceRating, affordabilityRating, reviewID, restaurantName,
restaurantLocation)
VALUES(‘4’, ‘4’, ‘4’, ‘234’, ‘test2’s Food’, ‘test2’s location’)
INSERT
INTO Rates(foodRating, serviceRating, affordabilityRating, reviewID, restaurantName,
restaurantLocation)
VALUES(‘5’, ‘4’, ‘4’, ‘345’, ‘test3’s Food’, ‘test3’s location’)
INSERT
INTO Rates(foodRating, serviceRating, affordabilityRating, reviewID, restaurantName,
restaurantLocation)
VALUES(‘4’, ‘4’, ‘3’, ‘456’, ‘test4’s Food’, ‘test4’s location’)
INSERT
INTO Rates(foodRating, serviceRating, affordabilityRating, reviewID, restaurantName,
restaurantLocation)
VALUES(‘5’, ‘5’, ‘5’, ‘567’, ‘test5’s Food’, ‘test5’s location’)
```

Media:

```
INSERT
INTO Media(filename, description, journalID)
VALUES(‘file1.jpg’, ‘file1 desc’, ‘1’)
INSERT
INTO Media(filename, description, journalID)
VALUES(‘file2.jpg’, ‘file2 desc’, ‘1’)
INSERT
INTO Media(filename, description, journalID)
VALUES(‘file3.jpg’, ‘file3 desc’, ‘1’)
INSERT
INTO Media(filename, description, journalID)
VALUES(‘file4.png’, ‘file4 desc’, ‘2’)
INSERT
INTO Media(filename, description, journalID)
VALUES(‘file5.png’, ‘file5 desc’, ‘3’)
```

DineInOrder:

```
INSERT
INTO DineInOrder(orderID, totalPrice, tableNumber, accountID, restaurantName,
restaurantLocation)
VALUES (‘90001’, ‘100’, ‘7001’, ‘0001’, ‘test1’s Food’, ‘test1’s location’)
INSERT
INTO DineInOrder(orderID, totalPrice, tableNumber, accountID, restaurantName,
restaurantLocation)
VALUES (‘90002’, ‘50’, ‘7005’, ‘0002’, ‘test2’s Food’, ‘test2’s location’)
INSERT
INTO DineInOrder(orderID, totalPrice, tableNumber, accountID, restaurantName,
restaurantLocation)
VALUES (‘90003’, ‘200’, ‘7003’, ‘0003’, ‘test3’s Food’, ‘test3’s location’)
INSERT
INTO DineInOrder(orderID, totalPrice, tableNumber, accountID, restaurantName,
restaurantLocation)
VALUES (‘90004’, ‘15’, ‘7002’, ‘0004’, ‘test4’s Food’, ‘test4’s location’)
INSERT
INTO DineInOrder(orderID, totalPrice, tableNumber, accountID, restaurantName,
restaurantLocation)
VALUES (‘90005’, ‘75’, ‘7006’, ‘0005’, ‘test5’s Food’, ‘test5’s location’)
```

PickupOrder:
INSERT
INTO DineInOrder(orderID, totalPrice, pickupNumber, accountID, restaurantName, restaurantLocation)
VALUES (‘90006’, ‘80’, ‘7010’, ‘0001’, ‘test1’s Food’, ‘test1’s location’)
INSERT
INTO DineInOrder(orderID, totalPrice, pickupNumber, accountID, restaurantName, restaurantLocation)
VALUES (‘90007’, ‘16’, ‘7011’, ‘0003’, ‘test2’s Food’, ‘test2’s location’)
INSERT
INTO DineInOrder(orderID, totalPrice, pickupNumber, accountID, restaurantName, restaurantLocation)
VALUES (‘90008’, ‘38’, ‘7012’, ‘0002’, ‘test3’s Food’, ‘test3’s location’)
INSERT
INTO DineInOrder(orderID, totalPrice, pickupNumber, accountID, restaurantName, restaurantLocation)
VALUES (‘90009’, ‘80’, ‘7013’, ‘0004’, ‘test4’s Food’, ‘test4’s location’)
INSERT
INTO DineInOrder(orderID, totalPrice, pickupNumber, accountID, restaurantName, restaurantLocation)
VALUES (‘90010’, ‘110’, ‘7014’, ‘0005’, ‘test5’s Food’, ‘test5’s location’)

Restaurant1:
INSERT
INTO Restaurant1(cuisineTag, menu)
VALUES (‘American’, ‘Burgers, Fries, Milkshakes’)
INSERT
INTO Restaurant1(cuisineTag, menu)
VALUES (‘Seafood’, ‘Fish, Lobster, Crab’)
INSERT
INTO Restaurant1(cuisineTag, menu)
VALUES (‘Italian’, ‘Pasta, Pizza’)
INSERT
INTO Restaurant1(cuisineTag, menu)
VALUES (‘Japanese’, ‘Sushi, Udon, Ramen’)
INSERT
INTO Restaurant1(cuisineTag, menu)
VALUES (‘Chinese’, ‘Hot pot, Milk tea’)

Restaurant2:

```
INSERT
INTO Restaurant2(name, location, waitlistID)
VALUES(‘test1’s name’, ‘test1’s location’, ‘1’)
INSERT
INTO Restaurant2(name, location, waitlistID)
VALUES(‘test2’s name’, ‘test2’s location’, ‘2’)
INSERT
INTO Restaurant2(name, location, waitlistID)
VALUES(‘test3’s name’, ‘test3’s location’, ‘3’)
INSERT
INTO Restaurant2(name, location, waitlistID)
VALUES(‘test4’s name’, ‘test4’s location’, ‘4’)
INSERT
INTO Restaurant2(name, location, waitlistID)
VALUES(‘test5’s name’, ‘test5’s location’, ‘5’)
```

Restaurant_Staff1:

```
INSERT
INTO Restaurant_Staff1(restaurantName, restaurantLocation, staffID, position)
VALUES(‘test1’s Food’, ‘test1’s location’, ‘01’, ‘Chef’)
INSERT
INTO Restaurant_Staff1(restaurantName, restaurantLocation, staffID, position)
VALUES(‘test2’s Food’, ‘test2’s location’, ‘02’, ‘Server’)
INSERT
INTO Restaurant_Staff1(restaurantName, restaurantLocation, staffID, position)
VALUES(‘test3’s Food’, ‘test3’s location’, ‘03’, ‘Barista’)
INSERT
INTO Restaurant_Staff1(restaurantName, restaurantLocation, staffID, position)
VALUES(‘test4’s Food’, ‘test4’s location’, ‘04’, ‘Chef’)
INSERT
INTO Restaurant_Staff1(restaurantName, restaurantLocation, staffID, position)
VALUES(‘test5’s Food’, ‘test5’s location’, ‘05’, ‘Server’)
```

Restaurant_Staff2:

```
INSERT
INTO Restaurant_Staff2(staffID, name)
VALUES(‘01’, ‘staff1’)
INSERT
INTO Restaurant_Staff2(staffID, name)
VALUES(‘02’, ‘staff2’)
INSERT
INTO Restaurant_Staff2(staffID, name)
VALUES(‘03’, ‘staff3’)
INSERT
INTO Restaurant_Staff2(staffID, name)
VALUES(‘04’, ‘staff4’)
INSERT
INTO Restaurant_Staff2(staffID, name)
VALUES(‘05’, ‘staff5’)
```

Waitlist1:

```
INSERT
INTO Waitlist1(waitlistID, restaurantName, restaurantLocation)
VALUES (‘1’, ‘test1’s name’, ‘test1’s location’)
INSERT
INTO Waitlist1(waitlistID, restaurantName, restaurantLocation)
VALUES (‘2’, ‘test2’s name’, ‘test2’s location’)
INSERT
INTO Waitlist1(waitlistID, restaurantName, restaurantLocation)
VALUES (‘3’, ‘test3’s name’, ‘test3’s location’)
INSERT
INTO Waitlist1(waitlistID, restaurantName, restaurantLocation)
VALUES (‘4’, ‘test4’s name’, ‘test4’s location’)
INSERT
INTO Waitlist1(waitlistID, restaurantName, restaurantLocation)
VALUES (‘5’, ‘test5’s name’, ‘test5’s location’)
```

Waitlist2:

```
INSERT
INTO Waitlist2(estimateWaitTime, restaurantName, restaurantLocation)
VALUES (‘20’, ‘test1’s name’, ‘test1’s location’)
INSERT
INTO Waitlist2(estimateWaitTime, restaurantName, restaurantLocation)
VALUES (‘70’, ‘test2’s name’, ‘test2’s location’)
INSERT
INTO Waitlist2(estimateWaitTime, restaurantName, restaurantLocation)
VALUES (‘10’, ‘test3’s name’, ‘test3’s location’)
INSERT
INTO Waitlist2(estimateWaitTime, restaurantName, restaurantLocation)
VALUES (‘0’, ‘test4’s name’, ‘test4’s location’)
INSERT
INTO Waitlist2(estimateWaitTime, restaurantName, restaurantLocation)
VALUES (‘15’, ‘test5’s name’, ‘test5’s location’)
```

Joins:

```
INSERT
INTO Joins(position, accountID, waitlistID)
VALUES (‘1’, ‘0001’, ‘1’)
INSERT
INTO Joins(position, accountID, waitlistID)
VALUES (‘2’, ‘0002’, ‘1’)
INSERT
INTO Joins(position, accountID, waitlistID)
VALUES (‘1’, ‘0003’, ‘2’)
INSERT
INTO Joins(position, accountID, waitlistID)
VALUES (‘1’, ‘0004’, ‘3’)
INSERT
INTO Joins(position, accountID, waitlistID)
VALUES (‘2’, ‘0005’, ‘3’)
```