CPSC 304 Project

Milestone #: 2

Date: March 1, 2023

Group Number: 35

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Manek Gujral	27255066	r7g3b	manekgujral11@gmail.com
Kaiti Mok Rong	16915143	a1o2b	kmr.source@gmail.com
Kush Arora	77714640	l8i3b	kusharora339@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.)

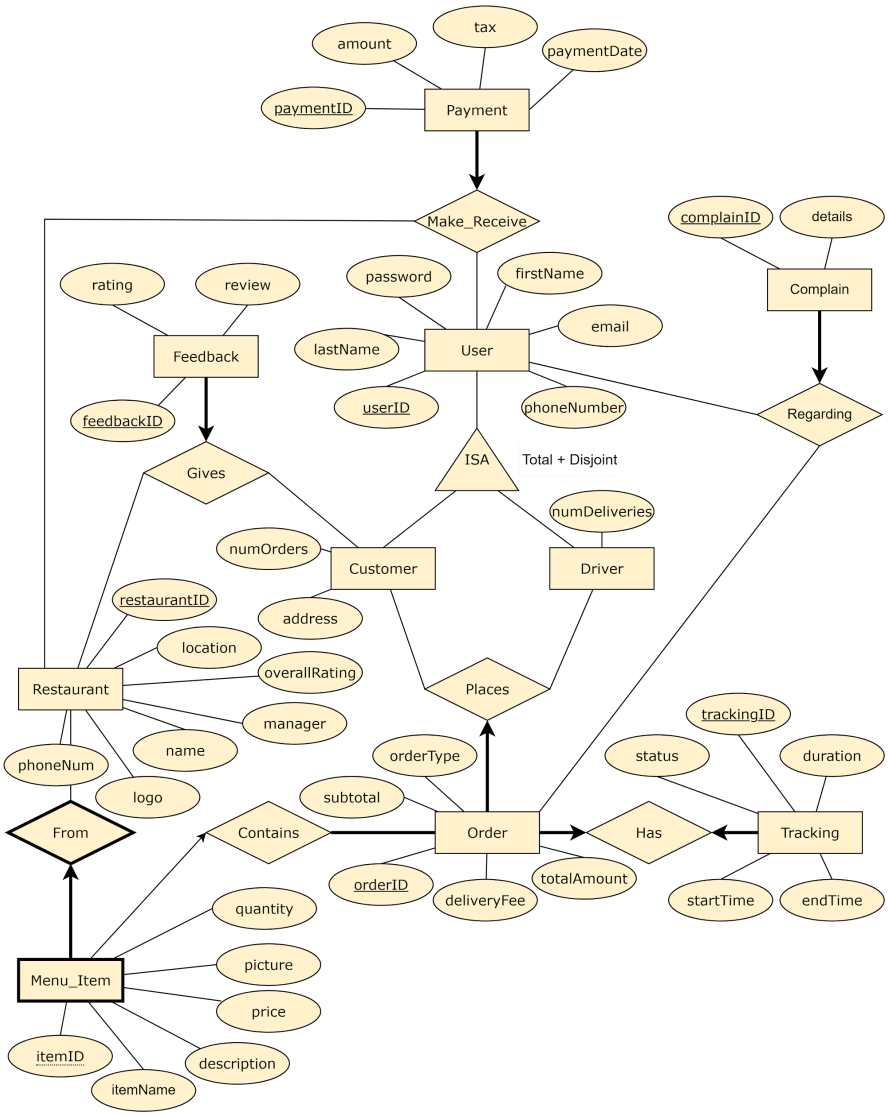
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

University of British Columbia, Vancouver
Department of Computer Science

Project Summary

Food Delivery Application will be the domain for our project. The domain will focus on customer and delivery management logistics, primarily on the data exchanged between the users and restaurants. The goal of our project is to identify how long it takes on average for a food order to be placed, prepared, and delivered from a restaurant to its customers, as well as how payments are made by customers and received by the restaurants and drivers. Customers can also provide feedback to restaurants. Orders can also have complaints made by the user.

ER diagram



Changes made for our current ER diagram:

1. Everything that was a x_id was changed to the format xID to make it more legible throughout the ER diagram entities
2. User Entity & Order Entity
    a. ISA relationship for users, the sub classes should not have additional attributes that act as keys. All user types should be identified by the same key attribute(s) **(TA suggestion Fixed)**
    b. A customer should be able to create a profile without having to place an order, thus the participation constraint here is not appropriate. **(TA suggestion Fixed)**
    c. Missing constraints, hierarchy isn't very meaningful. We made more meaningful so that it is total + disjoint for User ISA
    d. An order is placed by a customer, if you know the order you should know who placed it, this should be a many-to-one relationship.
3. User Entity:
    a. Removed admin subclass from our ISA to simplify the database, since the admin subclass does not really provide additional capabilities listed in the database specs **(TA suggestion Fixed)**
    b. Added attributes: phone_number, email, firstName, lastName
    c. Removed attributes: fullName
    d. we split full_name to two attributes first_name and last_name , as it will allow for better filters, search and sort capabilities for our queries
    e. user subclass Driver:
        i. Removed attributes: date_joined , email
        ii. deleted date_joined as it was not a useful attribute we needed for our database, and email was moved as User entity attribute
    f. For the user subclass Customer:
        i. Removed attributes: phone_number, email, date_joined
        ii. deleted date_joined as it was not a useful attribute we needed for our database, and email was moved as User entity attribute
4. Payment Entity
    a. The participation constraint on make/receive payment means all users must have a payment record. This does not seem appropriate as not all customers will have purchased/made a payment, or not all drivers has made a delivery and thus gotten paid. (**TA suggestion Fixed**)
    b. We removed the Tip entity and didn't make Tip as a weak entity because it didn't add much to our database. Instead we made the Menu Item a weak entity

from the Restaurant. (**TA suggestion, fixed and modified our ER diagram, removed Tip entity)**

c. Simplify Payment entity relationship between Restaurant and User. We made it a ternary relationship where Many Payments can be made by a user (many-to-one) and many Payments can be make_recive by restaurants (many-to-one).
d. Attribute Removed: toID and fromID
e. Attribute Changed: total_amount→ amount
f. Attributes Added: paymentDate , tax

5. Tracking Entity & Order Entity & User Entity
    a. The participation constraint on monitor for users also does not seem appropriate. For example, a customer should not be forced to always have something delivered. So we removed monitor relation from Tracking and Users entity, instead we replaced it so it's a one-to-one relationship between Order and Tracking tracking. This means that every order has one trackingID associated with it and vice versa. **(TA suggestion, made modifications and fixed our ER diagram)**
6. Restaurant Entity
    a. If you know the order you should know which restaurant it is from (many to one). **(TA suggestion Fixed,** additionally not all restaurants need to have an order)
    b. Attributes Added: overallRating, phoneNum,logo, name, location, restaurantID
    c. Attributes Removed: address
7. Tip Entity (REMOVED)
    a. We removed the Tip entity and didn't make Tip as a weak entity because it didn't add much to our database.
8. Menu Item Entity (Changed to Weak entity) & Restaurant Entity & Order
    a. We made the Menu Item a weak entity of the Restaurant Entity.
    b. We changed it so the Weak entity Menu Item now has a many-to-one relationship between Order and Menu item. This means one order can contain many menu items from the Restaurant.
    c. We removed the relation "from" between Restaurant and Order
    d. instead made it so that there is a many-to-one relationship "from" Restaurant and Menu Item, therefore removing the original "has" relationship.
9. Menu Item entity
    a. Attributes Added: itemName
    b. Attributes changed: id → itemID, qty → quantity
    c. Attributes Removed: cusine_type

d. Removed cusines_type as having itemName was enough , we wanted to simplify our database

10. To make everything more uniform, we made all attributes with id to match their entity, for example, Entity: “Menu item” attribute for id will be menuID instead of id or menu_id
11. Feedback Entity
    a. We simplified the Feedback entity. We made it into a ternary relationship between Customer, Feedback and Restaurant. **(TA suggestion Fixed)**
    b. Attributes Added: review
    c. Attributes Removed: date_given
12. Complain Entity (ADDED)
    a. Attributes added: complainID, details
        i. One user can make multiple complaints regarding the many orders. This is separate from our Feedback entity as it is specific to the orders, so when users have problems with the order placed they can make a complaint and get it resolved.
13. Tracking Entity
    a. Attributes Added: status, trackingID, duration, startTime and endTime. endTime can be a candidate key which can give us the Status ie. completed or null
    b. Attributes Removed: time_to_arrive, as startTime and endTime will give us this information and duration.
14. Order Entity
    a. Attributes Added: orderType, subtotal, orderID, deliveryFee, totalAmount. Added these attributes gave us more normalization
    b. Attributes Removed: orderNum, total, time, date
15. Changed the color of the lines from red to black for the ER diagram easier to reasd

Relational Schema derived from our ER diagram

*****For attributes in the tables below Primary keys are underlined, Foreign keys are bolded*****

Payment_Make_Receive(paymentID: integer, **restaurantID**: integer, amount: real, tax: real, paymentDate: timestamp, **userID**: integer)
userID, restaurantID is not null
tax, amount, paymentDate is not null since valid payment needs non null attribute values.

Restaurant(restaurantID: integer, location: char[80], manager: char[30], name: char[30], phoneNum: char[10], logo: blob, overallRating: real)
phoneNum, location and name is not null since valid entries require non-null attribute values, all restaurants must have valid phoneNum, location and name

Feedback_Gives(feedbackID: integer, rating: integer, review: char[100], **userId**: integer, **restaurantID**: integer)
userID, restaurantID and rating are not null

Complain_Regarding(complainID: integer, details: char[250], **orderID**: integer, **userID:** integer)
orderID, userID and details are not null. Valid complain, requires details

Users(userID: integer, lastName: char[30], password: char[64], firstName: char[30], email: char[320], phoneNumber: char[10])
phoneNumber, email are unique
lastName, password, firstName, email, phoneNumber are not null since a valid user needs all attributes
CK: phoneNumber, email

Customer(**userID**: integer, numOrders: integer, address: char[80]])
Address is not null for valid customers.

Driver(**userID**: integer, numDeliveries: integer)

Menu_Item(itemID: integer, **restaurantID**: integer, itemName: char[30], description: char[40], price: real, picture: blob, quantity: integer)
Quantity, price, itemName not null

Contains(**itemID**: integer, **restaurantID**: integer, **orderID**: integer)

Order_Places_Has(orderID: integer, totalAmount: real, **Customer_userID**: integer, **Driver_userID**: integer, **trackingID**: integer, deliveryFee: real, orderType: char[10], subtotal: real)
trackingID is unique,
trackingID, Customer_userID, orderType, subtotal and Driver_ID are not null,
totalAmount, deliveryFee, orderType, subtotal are not null since valid order needs non null attribute values.

Tracking(trackingID: integer, status: char[10], startTime: time, endTime: time, duration: time)
Status, startTime are not NULL

Functional Dependencies (FDs)

Payment_Make_Receive

Functional Dependencies:

1) paymentID -> amount, paymentDate, userID, restaurantID, tax
2) amount -> tax

Primary key

1) paymentID

Foreign Key

1) restaurantID

Users

Functional Dependencies:

1) userID -> firstName, lastName, email, phoneNumber, numOrders, address, numDeliveries
2) phoneNumber -> firstName, lastName, email, numOrders, address, numDeliveries, userID
3) email-> firstName, lastName, phoneNumber, numOrders, address, numDeliveries, userID

Primary key

1) userID

Candidate Key

1) phoneNumber
2) email

Complain

complainID -> details, userID, oderID

Primary key

1) complainID

Foreign Key

1) orderID
2) userID

Feedback

feedbackID -> rating, review, userID, restaurantID

Primary key

1) feedbackID

Foreign Key

1) userID
2) restaurantID

Restaurant

Functional dependencies

1) restaurantID -> location, manager, name, phoneNum, logo, overallRating
2) location -> name, overallRating
3) phoneNum -> location
4) Name -> logo

Primary Key

1) restaurantID

Menu_Item

Functional dependencies

1) itemID, restaurantID -> itemName, description, price, picture, location, manager, name

Primary Key

1) itemID
2) restaurantID

Foreign Key

1) restaurantID

Order

Functional Dependencies

1) orderID -> Customer_userID, Driver_userID, trackingID, totalAmount, orderType, deliveryFee, subtotal
2) orderType -> deliveryFee
3) deliveryFee, subtotal -> totalAmount

Primary Key

1) orderID

Foreign Key

1) trackingID
2) Customer_userID

3) Driver_ID

Tracking

Functional Dependencies

1) trackingID -> status, startTime, endTime, orderID, duration
2) endTime -> status
3) startTime, endTime → duration

Primary Key

2) trackingID

Normalization

*****For attributes in the tables and normalization Primary keys are underlined, Foreign keys are bolded*****

For Order:

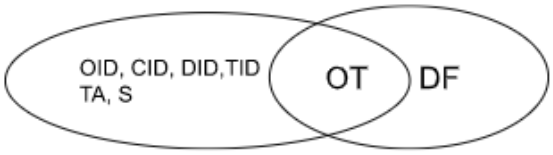
orderID = OID, Customer_userID = CID, Driver_userID = DID, trackingID = TID, totalAmount = TA, orderType = OT, deliveryFee = DF, subtotal = S

Order(OID, CID, DID, TA, OT, DF, S, TID)

OID -> CID, DID, TA, OT, DF, S, TID (OID is key)
OT -> DF
DF, S -> TA

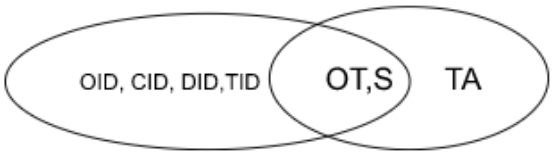
OT^+ = {OT, DF, S, TA}
DFS^+ = {DF, S, TA}
OTS^+ = {OT, DF, S, TA} so, implicitly, OT, S -> TA

Decompose on OT -> DF



OrderTable_1(DF, OT) (in BCNF)
OrderTable_2(OID, CID, DID, TA, S, TID, OT) (not in BCNF)

Decompose on OT,S -> TA



OrderTable_3(**OT**, S, TA) (in BCNF)
OrderTable_4(OID, CID, DID, **OT, S**, TID) (in BCNF)

OrderTable_1(deliveryFee: real, orderType: char[10]),
OrderTable_3(**orderType**: char[10], subtotal: real, totalAmount: real),
OrderTable_4(orderID: integer, **Customer_userID**: integer, **Driver_userID**: integer, **trackingID**: integer, **orderType**: char[10], **subtotal**: real)
constraints on OrderTables:
trackingID is unique,

trackingID, Customer_userID, orderType and Driver_ID are not null
totalAmount, deliveryFee, orderType, subtotal are not null since valid order needs non null attribute values.

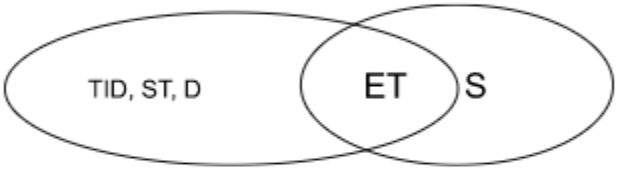
For Tracking:

trackingID = TID, status = S, startTime = ST, endTime = ET, duration = D
T(TID, S, ET, ST, D)

TID -> S, ET, ST, D (TID is key)
ET -> S
ST, ET -> D

$ET^+ = \{ET, S\}$
$STET^+ = \{D, S, ET, ST\}$

Decompose on ET -> S



Tracking_1(S, ET) (in BCNF)
Tracking_2(D, ET, ST, TID) (not in BCNF)

Decompose on ST, ET-> D

Tracking_3(D, **ET**, ST) (in BCNF)
Tracking_4(TID, **ET, ST**) (in BCNF)

TrackingTable_1(status: char[10], endTime: time)
TrackingTable_3(startTime: time, **endTime**: time, duration: time) (endTime refers to primary key(endTime) of TrackingTable_1)
TrackingTable_4(trackingID: integer, **startTime**: time, **endTime**: time) (startTime and endTime refer to primary key(startTime + endTime) of trackingTable_3)

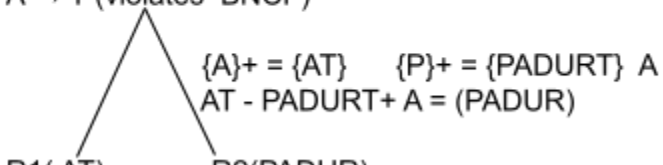
Constraints on TrackingTables:
Status, startTime are not NULL

For payment:

Let paymentID = P ,amount = A , tax = T , paymentDate = D , userID= U , restaurantID = R

1) {X}+
2) R - {X}+ + X

The candidate Key : P
P → A, D, U, R , T (BNCF)
A → T (violates BNCF)



R1(AT)
CK: AT
FD
AT → AT (BNCF)

R2(PADUR)
CK: P
FD
P-> A,D , U , R (BNCF)

PaymentTable_1(amount: real, tax: real)
PaymentTable_2(paymentID: integer, **amount**: real, paymentDate: timestamp, **userID:** integer, **restaurantID**: integer)

- amount refers to primary key (amount) of PaymentTable_1
- userID refers to the primary key (userID) in the user table
- restaurantID refers to the primary key (restaurantID) of RestaurantTable_6

Constraints on PaymentTables:
userID, restaurantID is not null
tax, amount, paymentDate is not null since valid payment needs non null attribute values.

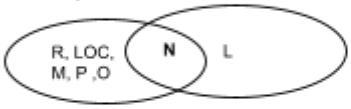
For Restaurant:

Let Location = LOC, name = N, Logo= L, overallRating = O , Manager = M, restaurantID = R, phoneNum = P

Functional Dependencies
R→ LOC, M, N, P, L, O (BNCF, CK is R)
N→L (violates BNCF)
LOC—> N,O
P→ LOC

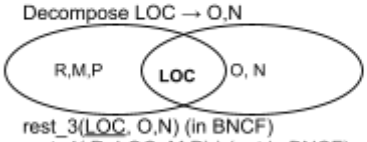
{N}+ = {N, L}
{LOC}+ = { LOC, N, O, L} simplify LOC →O,N
{P}+ = {P, LOC, N,O, L} simplify to P→ LOC

Decompose N→ L



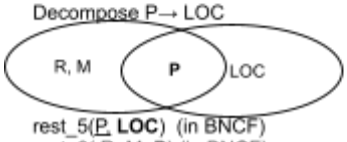
rest_1(N, L) (in BNCF)
rest_2(R, LOC, M, N, P, O) (not in BNCF)

Decompose LOC → O,N



rest_3(LOC, O,N) (in BNCF)
rest_4(R, LOC, M,P)) (not in BNCF)

Decompose P→ LOC



rest_5(P, **LOC**) (in BNCF)
rest_6(R, M, **P**) (in BNCF)

ResturantTable_1(name: char[30], logo: blob) PK: name
ResturantTable_3(Location: char[80], overallRating: real, **name**: char[30])

- name refers to the primary key (name) in ResturantTable_1

ResturantTable_5(phoneNum: char[10], **Location**: char[80])

- location refers to the primary key (location) in ResturantTable_3

ResturantTable_6(restaurantID: integer, manager: char[30], **phoneNum**: char[10])

- phoneNum refers to the primary key (phoneNum) in ResturantTable_5

Constraints on RestaurantTables:
phoneNum, location and name is not null since valid entries require non-null attribute values, all restaurants must have valid phoneNum, location and name

All tables after normalization:

Feedback_Gives(feedbackID: integer, rating: integer, review: char[100], **userId**: integer, **restaurantID**: integer)
userID, restaurantID and rating are not null

Complain_Regarding(complainID: integer, details: char[40], **orderID**: integer, **userID:** integer)
orderID, userID and details are not null.
Valid complain, requires details

Users(userID: integer, lastName: char[30], password: char[64], firstName: char[30], email: char[320], phoneNumber: char[10])
CK: phoneNumber, email
phoneNumber, email are unique
lastName, password, firstName, email, phoneNumber are not null since a valid user needs all attributes

Customer(**userID**: integer, numOrders: integer, address: char[80]])
Address is not null for valid customers.

Driver(**userID**: integer, numDeliveries: integer)

Menu_Item(itemID: integer, **restaurantID**: integer, itemName: char[30], description: char[40], price: real, picture: blob, quantity: integer)
Quantity, price, itemName not null

Contains(**itemID**: integer, **restaurantID**: integer, **orderID**: integer)

OrderTable_1(deliveryFee: real, orderType: char[10]),
OrderTable_2(**orderType**: char[10], subtotal: real, totalAmount: real) (orderType refers primary key (orderType) of orderTable_1)
OrderTable_3(orderID: integer, **Customer_userID**: integer, **Driver_userID**: integer, **trackingID**: integer, **orderType**: char[10], **subtotal**: real) (orderType and subtotal refer primary key (orderType + subtotal) of orderTable_2)
Constraints on OrderTables:
trackingID is unique,
trackingID, Customer_userID, orderType, subtotal and Driver_ID are not null

totalAmount, deliveryFee, orderType, subtotal are not null since valid order needs non null attribute values.

TrackingTable_1(status: char[10], endTime: timestamp),
TrackingTable_2(startTime: timestamp, **endTime**: timestamp, duration: interval) (endTime refers to primary key(endTime) of TrackingTable_1),
TrackingTable_3(trackingID: integer, **startTime**: timestamp, **endTime**: timestamp) (startTime and endTime refer to primary key(startTime + endTime) of trackingTable_2)
Constraints on TrackingTables:
Status, startTime are not NULL

PaymentTable_1(amount: real, tax: real)
PaymentTable_2(paymentID: integer, **amount**: real, paymentDate: timestamp, **userID:** integer, **restaurantID**: integer)

- amount refers to primary key (amount) of PaymentTable_1
- userID refers to the primary key (userID) in the user table
- restaurantID refers to the primary key (restaurantID) of RestaurantTable_4

Constraints on PaymentTables:
userID, restaurantID is not null
tax, amount, paymentDate is not null since valid payment needs non null attribute values.

ResturantTable_1(name: char[30], logo: blob)
ResturantTable_2(Location: char[80], overallRating: real, **name**: char[30])

- name refers to the primary key (name) in ResturantTable_1

ResturantTable_3(phoneNum: char[10], **Location**: char[80])

- location refers to the primary key (location) in ResturantTable_2

ResturantTable_4(restaurantID: integer, manager: char[30], **phoneNum**: char[10])

- phoneNum refers the primary key (phoneNum) in ResturantTable_3

Constraints on RestaurantTables:
phoneNum, location and name is not null since valid entries require non-null attribute values, all restaurants must have valid phoneNum, location and name

SQL DDL statements

```
CREATE TABLE Users(
  userID INT,
  password VARCHAR(64) NOT NULL,
  firstName VARCHAR(30) NOT NULL,
  lastName VARCHAR(30) NOT NULL,
  email VARCHAR(320) NOT NULL,
  phoneNumber VARCHAR(10) NOT NULL,
  PRIMARY KEY(userID),
  UNIQUE(email, phoneNumber)
);

CREATE TABLE Customer(
  userID INT,
  numOrders INT,
  address VARCHAR(80) NOT NULL,
  PRIMARY KEY(userID),
  FOREIGN KEY(userID)
    REFERENCES Users(userID)
);

CREATE TABLE Driver(
  userID INT,
  numDeliveries INT,
  PRIMARY KEY(userID),
  FOREIGN KEY(userID)
    REFERENCES Users(userID)
);

CREATE TABLE TrackingTable_1(
  status VARCHAR(10) NOT NULL,
  endTime TIMESTAMP,
  PRIMARY KEY(endTime)
);
```

```
CREATE TABLE TrackingTable_2(
  startTime TIMESTAMP,
  endTime TIMESTAMP,
  duration INTERVAL DAY(0) TO SECOND(0),
  PRIMARY KEY(startTime, endTime),
  FOREIGN KEY (endTime)
    REFERENCES TrackingTable_1(endTime)
);

CREATE TABLE TrackingTable_3(
  trackingID INT,
  startTime TIMESTAMP NOT NULL,
  endTime TIMESTAMP,
  PRIMARY KEY(trackingID),
  FOREIGN KEY (startTime, endTime)
    REFERENCES TrackingTable_2(startTime, endTime)
);

CREATE TABLE OrderTable_1(
  deliveryFee REAL NOT NULL,
  orderType VARCHAR(10),
  PRIMARY KEY(orderType)
);

CREATE TABLE OrderTable_2(
  orderType VARCHAR(10),
  subtotal REAL NOT NULL,
  totalAmount REAL NOT NULL,
  PRIMARY KEY(orderType, subtotal),
  FOREIGN KEY(orderType)
    REFERENCES OrderTable_1(orderType)
);
```

```
CREATE TABLE OrderTable_3(
  orderID INT,
  Customer_userID INT NOT NULL,
  Driver_userID INT NOT NULL,
  trackingID INT NOT NULL,
  orderType VARCHAR(10) NOT NULL,
  subtotal REAL NOT NULL,
  PRIMARY KEY(orderID),
  UNIQUE(trackingID),
  FOREIGN KEY(Customer_userID)
    REFERENCES Customer(userID),
  FOREIGN KEY(Driver_userID)
    REFERENCES Driver(userID),
  FOREIGN KEY(trackingID)
    REFERENCES TrackingTable_3(trackingID) ,
  FOREIGN KEY(orderType, subtotal)
    REFERENCES OrderTable_2(orderType, subtotal)
);

CREATE TABLE Complain_Regarding(
  complainID INT,
  details VARCHAR(250) NOT NULL,
  orderID INT NOT NULL,
  userID INT NOT NULL,
  PRIMARY KEY(complainID),
  FOREIGN KEY(orderID)
    REFERENCES OrderTable_3(orderID),
  FOREIGN KEY(userID)
    REFERENCES Users(userID)
);

CREATE TABLE RestaurantTable_1(
  name VARCHAR(30),
  logo BLOB,
  PRIMARY KEY(name)
);
```

```
CREATE TABLE RestaurantTable_2(
  location VARCHAR(80),
  name VARCHAR(30) NOT NULL,
  overallRating REAL,
  PRIMARY KEY(location),
  FOREIGN KEY(name)
    REFERENCES RestaurantTable_1(name)
);

CREATE TABLE RestaurantTable_3(
  phoneNum VARCHAR(10),
  location VARCHAR(80) NOT NULL,
  PRIMARY KEY(phoneNum),
  FOREIGN KEY(location)
    REFERENCES RestaurantTable_2(location)
);

CREATE TABLE RestaurantTable_4(
  restaurantID INT,
  manager VARCHAR(30),
  phoneNum VARCHAR(10) NOT NULL,
  PRIMARY KEY(restaurantID),
  FOREIGN KEY(phoneNum)
    REFERENCES RestaurantTable_3(phoneNum)
);
```

```
CREATE TABLE Menu_Item(
  itemID INT,
  restaurantID INT,
  itemName VARCHAR(30) NOT NULL,
  description VARCHAR(40),
  price REAL NOT NULL,
  picture BLOB,
  quantity INT NOT NULL,
  PRIMARY KEY(itemID, restaurantID),
  FOREIGN KEY(restaurantID)
    REFERENCES RestaurantTable_4(restaurantID)
);

CREATE TABLE Feedback_Gives(
  feedbackID INT,
  rating INT NOT NULL,
  review VARCHAR(100),
  userID INT NOT NULL,
  restaurantID INT NOT NULL,
  PRIMARY KEY(feedbackID),
  FOREIGN KEY(userID)
    REFERENCES Users(userID) ,
  FOREIGN KEY(restaurantID)
    REFERENCES RestaurantTable_4(restaurantID)
);
```

```
CREATE TABLE Contains(
  itemID INT,
  restaurantID INT,
  orderID INT,
  PRIMARY KEY(itemID, restaurantID, orderID),
  FOREIGN KEY(itemID, restaurantID)
    REFERENCES Menu_Item(itemID, restaurantID),
  FOREIGN KEY(restaurantID)
    REFERENCES RestaurantTable_4(restaurantID) ,
  FOREIGN KEY(orderID)
    REFERENCES OrderTable_3(orderID)
);

CREATE TABLE PaymentTable_1(
  amount REAL,
  tax REAL NOT NULL,
  PRIMARY KEY(amount)
);

CREATE TABLE PaymentTable_2(
  paymentID INT,
  amount REAL NOT NULL,
  paymentDate TIMESTAMP NOT NULL,
  userID INT NOT NULL,
  restaurantID INT NOT NULL,
  PRIMARY KEY(paymentID),
  FOREIGN KEY(amount)
    REFERENCES PaymentTable_1(amount),
  FOREIGN KEY(userID)
    REFERENCES Users(userID) ,
  FOREIGN KEY(restaurantID)
    REFERENCES RestaurantTable_4(restaurantID)
);
```

INSERT statements to populate each table with at least 5 tuples

```
INSERT INTO Users VALUES ('1', 'pass1', 'bob', 'marley', 'bob_m@gmail.com', '2508634536');
INSERT INTO Users VALUES ('2', 'pass2', 'harry', 'lewis', 'harry_l@gmail.com', '2505634536');
INSERT INTO Users VALUES ('3', 'pass3', 'josh', 'bradley', 'josh_b@gmail.com', '2508623656');
INSERT INTO Users VALUES ('4', 'pass4', 'tobi', 'brown', 'tobi@gmail.com', '1508634536');
INSERT INTO Users VALUES ('5', 'pass5', 'simon', 'minter', 'simon@gmail.com', '2236634536');
INSERT INTO Users VALUES ('6', 'pass6', 'vikram', 'singh', 'vik_s@gmail.com', '1158634536');
INSERT INTO Users VALUES ('7', 'pass7', 'ethan', 'payne', 'ethan@gmail.com', '9505634536');
INSERT INTO Users VALUES ('8', 'pass8', 'jj', 'wyatt', 'j_w@gmail.com', '2501113656');
INSERT INTO Users VALUES ('9', 'pass9', 'carol', 'baskin', 'carolb@gmail.com', '9998634536');
INSERT INTO Users VALUES ('10', 'pass10', 'jalen', 'turner', 'jt@gmail.com', '8556634536');

INSERT INTO Customer VALUES ('1', '5', '5621 Dunbar St');
INSERT INTO Customer VALUES ('2', '4', '1131 Dunbar St');
INSERT INTO Customer VALUES ('3', '6', '1364 W 11th Ave');
INSERT INTO Customer VALUES ('4', '11', '1033 Second St');
INSERT INTO Customer VALUES ('5', '100', '5621 Dunbar St');

INSERT INTO Driver VALUES ('6', '5');
INSERT INTO Driver VALUES ('7', '15');
INSERT INTO Driver VALUES ('8', '55');
INSERT INTO Driver VALUES ('9', '54');
INSERT INTO Driver VALUES ('10', '4');

INSERT INTO TrackingTable_1 VALUES ('COMPLETED', TIMESTAMP '2023-02-01 12:15:00');
INSERT INTO TrackingTable_1 VALUES ('COMPLETED', TIMESTAMP '2023-02-02 16:00:00');
INSERT INTO TrackingTable_1 VALUES ('COMPLETED', TIMESTAMP '2023-02-03 09:00:00');
INSERT INTO TrackingTable_1 VALUES ('COMPLETED', TIMESTAMP '2023-02-04 18:00:00');
INSERT INTO TrackingTable_1 VALUES ('COMPLETED', TIMESTAMP '2023-02-05 11:45:00');
INSERT INTO TrackingTable_1 VALUES ('PROCESSING', TIMESTAMP '9999-01-01 00:00:00');

INSERT INTO TrackingTable_2 VALUES (TIMESTAMP '2023-02-01 12:00:00', TIMESTAMP
'2023-02-01 12:15:00', INTERVAL '15' MINUTE);
INSERT INTO TrackingTable_2 VALUES (TIMESTAMP '2023-02-02 15:30:00', TIMESTAMP
'2023-02-02 16:00:00', INTERVAL '30' MINUTE);
```

```
INSERT INTO TrackingTable_2 VALUES (TIMESTAMP '2023-02-03 08:45:00', TIMESTAMP
'2023-02-03 09:00:00', INTERVAL '15' MINUTE);
INSERT INTO TrackingTable_2 VALUES (TIMESTAMP '2023-02-04 17:00:00', TIMESTAMP
'2023-02-04 18:00:00', INTERVAL '1' HOUR);
INSERT INTO TrackingTable_2 VALUES (TIMESTAMP '2023-02-05 11:20:00', TIMESTAMP
'2023-02-05 11:45:00', INTERVAL '25' MINUTE);
INSERT INTO TrackingTable_2 VALUES (CURRENT_TIMESTAMP, TIMESTAMP '9999-01-01
00:00:00', NULL);

INSERT INTO TrackingTable_3 VALUES ('1', TIMESTAMP '2023-02-01 12:00:00', TIMESTAMP
'2023-02-01 12:15:00');
INSERT INTO TrackingTable_3 VALUES ('2', TIMESTAMP '2023-02-02 15:30:00', TIMESTAMP
'2023-02-02 16:00:00');
INSERT INTO TrackingTable_3 VALUES ('3', TIMESTAMP '2023-02-03 08:45:00', TIMESTAMP
'2023-02-03 09:00:00');
INSERT INTO TrackingTable_3 VALUES ('4', TIMESTAMP '2023-02-04 17:00:00', TIMESTAMP
'2023-02-04 18:00:00');
INSERT INTO TrackingTable_3 VALUES ('5', TIMESTAMP '2023-02-05 11:20:00', TIMESTAMP
'2023-02-05 11:45:00');

INSERT INTO OrderTable_1 VALUES ('5.99', 'Platinum');
INSERT INTO OrderTable_1 VALUES ('4.99', 'Gold');
INSERT INTO OrderTable_1 VALUES ('3.99', 'Silver');
INSERT INTO OrderTable_1 VALUES ('2.99', 'Bronze');
INSERT INTO OrderTable_1 VALUES ('1.99', 'Standard');

INSERT INTO OrderTable_2 VALUES ('Platinum', '100', '105.99');
INSERT INTO OrderTable_2 VALUES ('Gold', '100', '104.99');
INSERT INTO OrderTable_2 VALUES ('Silver', '90', '93.99');
INSERT INTO OrderTable_2 VALUES ('Bronze', '56', '58.99');
INSERT INTO OrderTable_2 VALUES ('Standard', '100', '101.99');

INSERT INTO OrderTable_3 VALUES ('1', '1', '6', '1', 'Standard', '100');
INSERT INTO OrderTable_3 VALUES ('2', '2', '7', '2', 'Gold', '100');
INSERT INTO OrderTable_3 VALUES ('3', '3', '8', '3', 'Platinum', '100');
INSERT INTO OrderTable_3 VALUES ('4', '4', '9', '4', 'Gold', '100');
INSERT INTO OrderTable_3 VALUES ('5', '5', '10', '5', 'Platinum', '100');
```

```
INSERT INTO Complain_Regarding VALUES('1', 'Driver was rude.', '1', '1');
INSERT INTO Complain_Regarding VALUES('2', 'Driver left food outside of my house without calling.', '2', '2');
INSERT INTO Complain_Regarding VALUES('3', 'Customer was very angry and stareted shouting like crazy.', '3', '8');
INSERT INTO Complain_Regarding VALUES('4', 'Customer was rude.', '4', '9');
INSERT INTO Complain_Regarding VALUES('5', 'Driver was shouting.', '5', '5');

INSERT INTO RestaurantTable_1 VALUES('KFC', NULL);
INSERT INTO RestaurantTable_1 VALUES('Dominos', NULL);
INSERT INTO RestaurantTable_1 VALUES('Pizza Hut', NULL);
INSERT INTO RestaurantTable_1 VALUES('Cactus Club', NULL);
INSERT INTO RestaurantTable_1 VALUES('Social', NULL);

INSERT INTO RestaurantTable_2 VALUES('1050 Burard St.', 'KFC', '4.2');
INSERT INTO RestaurantTable_2 VALUES('1150 Dunbar St.', 'Dominos', '4.0');
INSERT INTO RestaurantTable_2 VALUES('2250 Burard St.', 'Pizza Hut', '3.2');
INSERT INTO RestaurantTable_2 VALUES('3456 West 11th Ave.', 'Cactus Club', '4.4');
INSERT INTO RestaurantTable_2 VALUES('5621 Dunbar St.', 'Social', '5.0');

INSERT INTO RestaurantTable_3 VALUES ('2504566532', '1050 Burard St.');
INSERT INTO RestaurantTable_3 VALUES ('2504566531', '1150 Dunbar St.');
INSERT INTO RestaurantTable_3 VALUES ('2504566533', '2250 Burard St.');
INSERT INTO RestaurantTable_3 VALUES ('2504566534', '3456 West 11th Ave.');
INSERT INTO RestaurantTable_3 VALUES ('2504566535', '5621 Dunbar St.');

INSERT INTO RestaurantTable_4 VALUES ('1', 'Kush Arora', '2504566532');
INSERT INTO RestaurantTable_4 VALUES ('2', 'Jason Tatum', '2504566531');
INSERT INTO RestaurantTable_4 VALUES ('3', 'Dwight Howard', '2504566533');
INSERT INTO RestaurantTable_4 VALUES ('4', 'Phil Foden', '2504566534');
INSERT INTO RestaurantTable_4 VALUES ('5', 'Leo Messi', '2504566535');

INSERT INTO Menu_Item VALUES ('1', '1', 'Zinger Burger', 'its finger lickin food', '10.00', NULL, '10');
INSERT INTO Menu_Item VALUES ('2', '1', 'Zinger Twister', 'its finger lickin food', '8.99', NULL, '1');
```

```
INSERT INTO Menu_Item VALUES ('1', '2', 'Philly Steak', 'Philly Steak', '14.99', NULL, '1');
INSERT INTO Menu_Item VALUES ('2', '2', 'Pepperoni', 'tastylicious', '12.99', NULL, '1');
INSERT INTO Menu_Item VALUES ('1', '3', 'Triple Crown', 'yummy pizza!!!', '20.99', NULL, '1');

INSERT INTO Feedback_Gives VALUES ('1', '3.0', 'It was amazing', '1', '1');
INSERT INTO Feedback_Gives VALUES ('2', '4.4', 'It was tasty', '2', '1');
INSERT INTO Feedback_Gives VALUES ('3', '5.0', 'phenomenal', '3', '2');
INSERT INTO Feedback_Gives VALUES ('4', '3.5', 'wowww It was amazing', '4', '3');
INSERT INTO Feedback_Gives VALUES ('5', '2.8', 'It was not that tasty', '5', '5');

INSERT INTO Contains VALUES ('1', '1', '1');
INSERT INTO Contains VALUES ('1', '1', '2');
INSERT INTO Contains VALUES ('1', '1', '3');
INSERT INTO Contains VALUES ('1', '1', '4');
INSERT INTO Contains VALUES ('1', '1', '5');

INSERT INTO PaymentTable_1 VALUES ('101.99', '5.0995');
INSERT INTO PaymentTable_1 VALUES ('10.00', '5.00');
INSERT INTO PaymentTable_1 VALUES ('50.00', '2.50');
INSERT INTO PaymentTable_1 VALUES ('74.00', '3.70');
INSERT INTO PaymentTable_1 VALUES ('83.00', '4.15');

INSERT INTO PaymentTable_2 VALUES ('1', '101.99', TIMESTAMP '2023-02-01 12:00:00', '1', '1');
INSERT INTO PaymentTable_2 VALUES ('2', '101.99', TIMESTAMP '2023-02-02 15:30:00', '2', '1');
INSERT INTO PaymentTable_2 VALUES ('3', '101.99', TIMESTAMP '2023-02-03 08:45:00', '3', '2');
INSERT INTO PaymentTable_2 VALUES ('4', '101.99', TIMESTAMP '2023-02-04 17:00:00', '4', '3');
INSERT INTO PaymentTable_2 VALUES ('5', '101.99', TIMESTAMP '2023-02-05 11:20:00', '5', '5');
```