University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ___2___

Date: ___2024/07/21___

Group Number: ____33____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Kevin Chu (Chao-Wu Chu)	85406312	b0u5x	kevinchu6601@gmail.com
Kaicheng Lu	14723720	o1u1i	kaichenglu2020@gmail.com
Jason Liang	75499566	w3s8d	cajasonliang@gmail.com

University of British Columbia, Vancouver
Department of Computer Science

Project Summary

Our project is a food delivering service that manages the process of a customer ordering food from a restaurant and having the food delivered to them. Customers are able to write reviews for both the restaurant and the driver responsible for delivering the food.

ER Diagram

We removed the MemberCustomer ISA Customer because the relationship was not meaningful, and thus did not contribute anything when added to our project. We added the attributes that were previously on MemberCustomer to Customer along with this change since we removed MemberCustomer.

We changed the primary key of Customer from having both email and phone#, and changed it to be only phone#. This is because a customer can be sufficiently uniquely identified with only phone#, and thus, it is unnecessary for email to also be a primary key.

For Food, we changed the primary key to only be foodName instead of both foodName and foodPrice. This is because foodName is sufficient enough in uniquely identifying the food. FoodPrice does not contribute to it being uniquely identified and thus we removed it from being a primary key.

For Restaurants, we changed the primary key from being both name and address to only address. Address is sufficient enough in uniquely identifying a restaurant, and thus we do not need the name attribute to also be a primary key. Therefore, we changed it to make only address be the primary key.

We added a reviewID attribute and used it as our primary key for Review rather than timestamp. This is because multiple comments can be posted during the same time, so it makes no sense to use it as a primary key because it does not uniquely identify a review. We thought reviewID was more fitting in identifying each one.

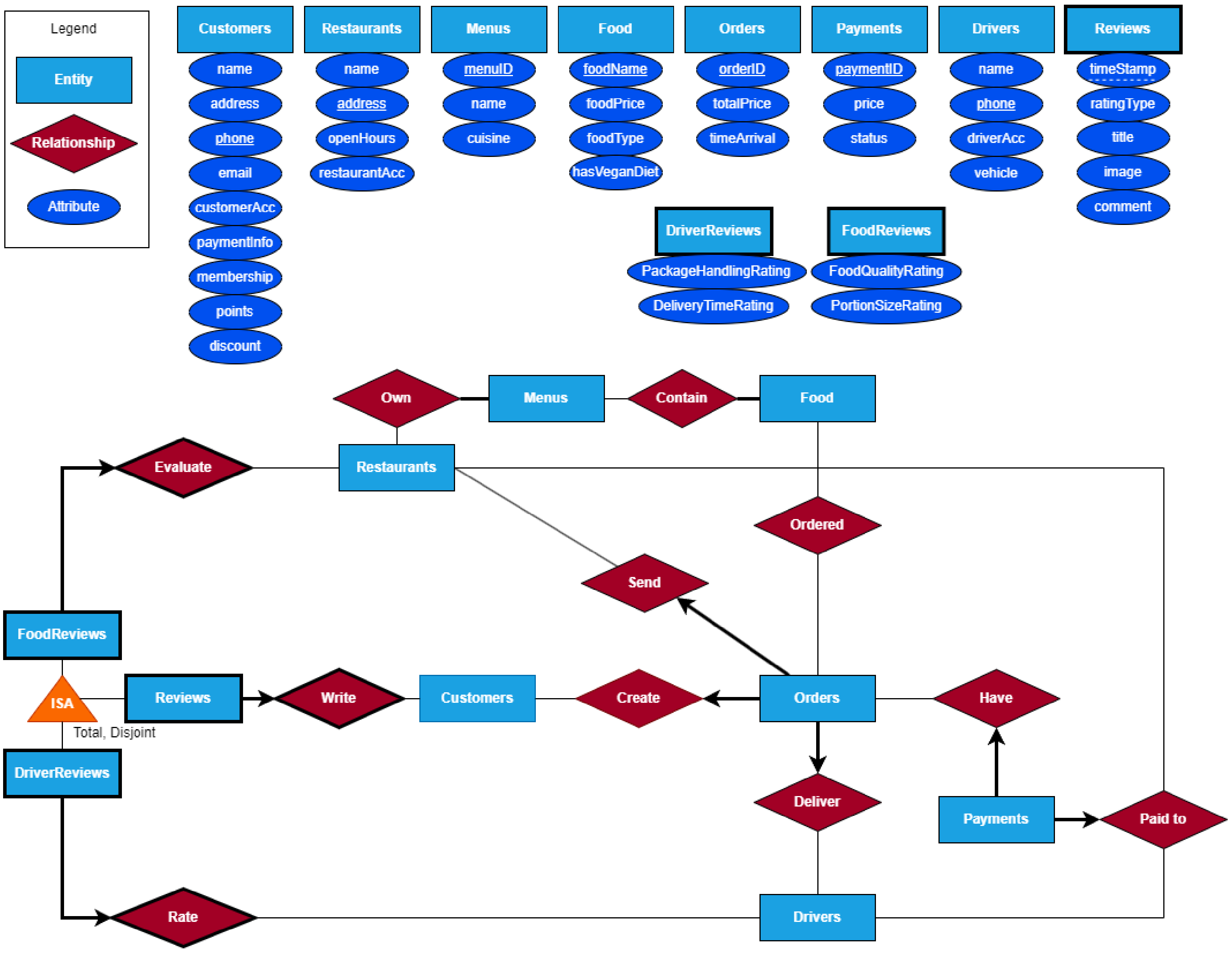
We added the attribute hasVeganDiet to the Food entity. This is because it would allow customers that are vegetarian or vegan to be able to easily find food that is suitable for them. At the same time, it gave another non-PK and non-CK FD.

For Driver, we changed the primary key to only be phone# instead of name and phone#. This is because phone# is enough to uniquely identify drivers. This makes it unnecessary to also have the name attribute as a primary key. Therefore, we made only phone# to be the primary key of Driver.

University of British Columbia, Vancouver
Department of Computer Science

We changed the attributes phone#, cusPhone#, and driverPhone# to be phone, cusPhone and driverPhone. This is because Oracle does not allow # in the name of attributes, so we got rid of the # in those respective attributes.

We also changed the Order relationship to be named Ordered because Oracle does not allow Tables to be named Order.



Relational Schema

Primary Keys are underlined, Foreign Keys are **bolded**.

- Customers (phone: varchar[15], name: varchar[255], address: varchar[255], email: varchar[255], customerAcc: varchar[20], paymentInfo: varchar[255], membership: varchar[3], points: int, discount: varchar[4]), email, customerAcc UNIQUE; name, address, email, customerAcc, paymentInfo, points NOT NULL; membership DEFAULT 'No'
  CK: phone, email, customerAcc

- Restaurants (address: varchar[255], name: varchar[255], openHours: varchar[255], restaurantAcc: varchar[20]), restaurantAcc UNIQUE; name, openHours, restaurantAcc NOT NULL
  CK: address, restaurantAcc

  - Relationship: Own (**restAddress**: varchar[255], **MID**: int)
    CK: {restAddress, MID}

- Menus (menuID: int, name: varchar[255], cuisine: varchar[255]), name NOT NULL
  CK: menuID

  - Relationship: Contain (**MID**: int, **fName**: varchar[255])
    CK: {MID, fName}

- Food (foodName: varchar[255], foodPrice: decimal[5, 2], foodType: varchar[255], hasVeganDiet: varchar[3]), foodPrice, foodType, hasVeganDiet NOT NULL
  CK: foodName
  - Relationship: Ordered (**fName**: varchar[255], **orderID**: int, orderedQuantity: int), orderedQuantity NOT NULL
    CK: {fName, orderID}

- Orders (orderID: int, totalPrice: decimal[5, 2] timeArrival: datetime, **cusPhone**: varchar[15], **restAddress**: varchar[255], **driverPhone**: varchar[15]), totalPrice, timeArrival, cusPhone, restAddress, driverPhone NOT NULL
  CK: orderID

- Drivers (phone: varchar[15], name: varchar[255], driverAcc: varchar[20], vehicle: varchar[255]), driverAcc UNIQUE; name, driverAcc, vehicle NOT NULL
  CK: phone, driverAcc

- Reviews (**cusPhone**: varchar[15], timeStamp: datetime, title: varchar[50], image: varchar[255], comment: varchar[255], ratingType: varchar[8]), title, comment, ratingType NOT NULL
  CK: {cusPhone, timeStamp}

  - ISA: FoodReviews (**cusPhone**: varchar[15], **timeStamp**: datetime, **restAddress**: varchar[255], FoodQualityRating: int, PortionSizeRating: int), restAddress NOT NULL
    CK: {cusPhone, timeStamp}

  - ISA: DriverReviews (**cusPhone**: varchar[15], **timeStamp**: datetime, **driverPhone**: varchar[15], PackageHandlingRating: int, DeliveryTimeRating: int), driverPhone NOT NULL
    CK: {cusPhone, timeStamp}

- Payment (paymentID: int, price: decimal[5, 2], status: varchar[7], **orderID**: int, **restAddress**: varchar[255], **driverPhone**: varchar[15]), price, status, orderID, restAddress, driverPhone NOT NULL
  CK: paymentID

Functional Dependencies

Entity	FD
Customers	phone → name, address, email, paymentInfo, customerAcc, membership, points, discounts email → name, address, phone, paymentInfo, customerAcc, membership, points, discounts customerAcc → name, address, email, phone, paymentInfo, membership, points, discounts
Restaurants	restaurantAcc → name, address, openHours address → name, openHours, restaurantAcc
Menus	menuID → name, cuisine name → cuisine
Food	foodName → foodPrice, foodType, hasVeganDiet foodType → hasVeganDiet
Orders	orderID → totalPrice, timeArrival, cusPhone, restAddress, driverPhone
Drivers	driverAcc → name, phone, vehicle phone → driverAcc, vehicle, name
Reviews	cusPhone, timeStamp → title, image, comment, ratingType comment → ratingType
FoodReviews	cusPhone, timeStamp → restAddress, FoodQualityRating, PortionSizeRating
DriverReviews	cusPhone, timeStamp → driverPhone, PackageHandlingRating, DeliveryTimeRating
Payment	paymentID → price, status, orderID, restAddress, driverPhone orderID → price, restAddress, driverPhone

Normalization

- Customer (phone, name, address, email, customerAcc, membership, points, discount)

  CK: {phone, email}

  Closures:

  Phone$^+$ = {name, address, email, paymentInfo, customerAcc, membership, points discounts}

  email$^+$ = {name, address, phone, paymentInfo, customerAcc, membership, points discounts}

  customerAcc = {name, address, phone, email, paymentInfo, customerAcc, membership, points, discounts}

  This relation is in BCNF so no decomposition is needed.

- Own (<u>**restAddress**</u>, <u>**MID**</u>)

  CK: {restAddress, MID}

  This relation is in BCNF as there are no functional dependencies.

- Restaurant (name, <u>address</u>, openHours, restaurantAcc)

  CK: address, restaurantAcc

  Closures:

  RestaurantAcc$^+$ = {name, address, openHours}

  Address$^+$ = {name, openHours, restaurantAcc}

  This relation is in BCNF so no decomposition is needed.

- Menus (menuID, name, cuisine)

  CK: menuID

  Closures:

  MenuID$^+$ = {name, cuisine}

  Name$^+$ = {cuisine}

  So this relation violates BCNF and 3NF.

  Decomposition:

  name $\rightarrow$ cuisine

  Menu1 (name, cuisine) Menu2(menuID, name)

  Hence, Menu1 and Menu2 are in BCNF

  Final relations:

  Menu1 (<u>name</u>: varchar[255], cuisine: varchar[255])

CK: name

Menu2 (menuID: int, **name**: varchar[255]), name NOT NULL

CK: MenuID

- Contain (**MID**, **fName**)

  CK: {MID, fName}

  This relation is in BCNF as there are no functional dependencies.

- Food (foodName, foodPrice, foodType, hasVeganDiet)

  CK: foodName

  Closures:

  foodName$^+$ = {foodName, foodPrice, foodType, hasVeganDiet}

  foodType$^+$ = {foodType, hasVeganDiet}

  So this relation violates BCNF and 3NF.

  Decomposition:

  foodType → hasVeganDiet

  Food1(foodType, hasVeganDiet), Food2(foodName, foodPrice, foodType)

  Hence, Food1 and Food2 are in BCNF.

  Final relations:

  Food1 (foodType: varchar[255], hasVeganDiet: varchar[3]) , hasVeganDiet NOT NULL

  CK: foodType

  Food2 (foodName: varchar[255], foodPrice: decimal[5, 2], **foodType**: varchar[255]), foodPrice, foodType NOT NULL

  CK: foodName

- Ordered (**fName**, **orderID**, orderedQuantity)

  CK: {fName, orderID}

  This relation is in BCNF as there are no functional dependencies.

- Orders (orderID, totalPrice, timeArrival, **cusPhone**, **restAddress**, **driverPhone**)

  CK: orderID

  Closures:

  orderID$^+$ = {orderID, totalPrice, timeArrival, cusPhone, restAddress, driverPhone}

  This relation is in BCNF so no decomposition is needed.

- Drivers (<u>phone</u>, name, driverAcc, vehicle)
  CK: phone, driverAcc

  Closures:
  driverAcc$^+$ = {name, phone, vehicle}
  phone$^+$ = {name, driverAcc, vehicle}
  This relation is in BCNF so no decomposition is needed.

- Reviews (**<u>cusPhone</u>**, <u>timeStamp</u>, title, image, comment, ratingType)
  CK: {cusPhone, timeStamp}

  Closures:
  {cusPhone, timeStamp}$^+$ = {cusPhone, timeStamp, title, image, comment, ratingType}
  comment$^+$ = {comment, ratingType}
  So this relation violates BCNF and 3NF.

  Decomposition:
  comment → ratingType
  Review1(<u>comment</u>, ratingType), Review2(**<u>cusPhone</u>**, <u>timeStamp</u>, title, image, **comment**)
  Hence, Review1 and Review2 are in BCNF.

  Final relations:
  Reviews1 (<u>comment</u>: varchar[255], ratingType: varchar[8]), ratingType NOT NULL
  CK: comment
  Reviews2 (**<u>cusPhone</u>**: varchar[15], <u>timeStamp</u>: datetime, title: varchar[50], image: varchar[255], **comment**: varchar[255]), title, comment NOT NULL
  CK: {cusPhone, timeStamp}

- FoodReviews (**<u>cusPhone</u>**, **<u>timeStamp</u>**, **restAddress**, FoodQualityRating, PortionSizeRating)
  CK: {cusPhone, timeStamp}

  Closures:
  {cusPhone, timeStamp}$^+$ = {cusPhone, timeStamp, restAddress, FoodQualityRating, PortionSizeRating}
  This relation is in BCNF so no decomposition is needed.

- **DriverReviews** (**cusPhone**, **timeStamp**, **driverPhone**, PackageHandlingRating, DeliveryTimeRating)

  CK: {cusPhone, timeStamp}

  Closures:

  {cusPhone, timeStamp}$^+$ = {cusPhone, timeStamp, driverPhone, PackageHandlingRating, DeliveryTimeRating}

  This relation is in BCNF so no decomposition is needed.

- Payment (paymentID, price, status, orderID, restAddress, driverPhone)

  Closures:

  paymentID$^+$ = {paymentID, price, status, orderID, restAddress, driverPhone}

  orderID$^+$ = {orderID, price, restAddress, driverPhone}

  So this relation violates BCNF and 3NF

  Decomposition:

  orderID → price, restAddress, driverPhone

  Payment1(**orderID**, price, **restAddress**, **driverPhone**), Payment2(**paymentID**, status, **orderID**)

  Hence, Payment1 and Payment2 are in BCNF

  Final relations:

  Payment1(**orderID**: int, price decimal[5, 2], **restAddress**: varchar[255], **driverPhone**: varchar[15]), price, restAddress, driverPhone NOT NULL

  CK: orderID

  Payment2(paymentID: int, status: varchar[7], **orderID**: int), status, orderID NOT NULL

  CK: paymentID

University of British Columbia, Vancouver
Department of Computer Science

SQL DDL (CREATE and INSERT)

Create Tables:

```
CREATE TABLE Customers (
        phone                   varchar(15),
        name                    varchar(255)            NOT NULL,
        address                 varchar(255)            NOT NULL,
        email                   varchar(255)            NOT NULL,
        customerAcc             varchar(20)             NOT NULL,
        paymentInfo             varchar(255)            NOT NULL,
        membership              varchar(3)              DEFAULT 'No',
        points                  int                     NOT NULL,
        discount                varchar(4),
        PRIMARY KEY (phone),
        UNIQUE (email, customerAcc)
);

CREATE TABLE Own (
        restAddress             varchar(255),
        MID                     int,
        PRIMARY KEY (restAddress, MID),
        FOREIGN KEY (MID) REFERENCES Menu2(menuID)
                ON DELETE CASCADE,
        FOREIGN KEY (restAddress) REFERENCES Restaurants(address)
                ON DELETE CASCADE
);

CREATE TABLE Restaurants (
        address                 varchar(255),
        name                    varchar(255)            NOT NULL,
        openHours               varchar(255)            NOT NULL,
        restaurantAcc           varchar(20)             NOT NULL,
        PRIMARY KEY (address),
        UNIQUE (restaurantAcc)
);
```

```
CREATE TABLE Menu1 (
        name                varchar(255),
        cuisine             varchar(255),
        PRIMARY KEY (name)
);


CREATE TABLE Menu2 (
        menuID              int,
        name                varchar(255)        NOT NULL,
        PRIMARY KEY (menuID),
        FOREIGN KEY (name) REFERENCES Menu1(name)
                ON DELETE CASCADE
);


CREATE TABLE Contain (
        MID                 int,
        fName               varchar(255),
        PRIMARY KEY (MID, fName),
        FOREIGN KEY (MID) REFERENCES Menu2(menuID)
                ON DELETE CASCADE,
        FOREIGN KEY (fName) REFERENCES Food2(foodName)
                ON DELETE CASCADE
);


CREATE TABLE Food1 (
        foodType            varchar(255),
        hasVeganDiet        varchar(3)          NOT NULL,
        PRIMARY KEY (foodType)
);
```

```
CREATE TABLE Food2 (
        foodName            varchar(255),
        foodPrice           decimal(5, 2)       NOT NULL,
        foodType            varchar(255)        NOT NULL,
        PRIMARY KEY (foodName),
        FOREIGN KEY (foodType) REFERENCES Food1(foodType)
                ON DELETE NO ACTION
);

CREATE TABLE Ordered (
        fName               varchar(255),
        orderID             int,
        orderedQuantity     int                 NOT NULL,
        PRIMARY KEY (fName, orderID),
        FOREIGN KEY (fName) REFERENCES Food2(foodName)
                ON DELETE NO ACTION,
        FOREIGN KEY (orderID) REFERENCES Orders(orderID)
                ON DELETE CASCADE
);

CREATE TABLE Orders (
        orderID             int,
        totalPrice          decimal(5, 2)       NOT NULL,
        timeArrival         datetime            NOT NULL,
        cusPhone            varchar(15)         NOT NULL,
        restAddress         varchar(255)        NOT NULL,
        driverPhone         varchar(15)         NOT NULL,
        PRIMARY KEY (orderID),
        FOREIGN KEY (cusPhone) REFERENCES Customers(phone)
                ON DELETE NO ACTION,
        FOREIGN KEY (restAddress) REFERENCES Restaurants(address)
                ON DELETE NO ACTION,
        FOREIGN KEY (driverPhone) REFERENCES Drivers(phone)
                ON DELETE NO ACTION
);
```

```
CREATE TABLE Drivers (
        phone               varchar(15),
        name                varchar(255)        NOT NULL,
        driverAcc           varchar(20)         NOT NULL,
        vehicle             varchar(255)        NOT NULL,
        PRIMARY KEY (phone),
        UNIQUE (driverAcc)
);


CREATE TABLE Review1 (
        comment             varchar(255),
        ratingType          varchar(8)          NOT NULL,
        PRIMARY KEY (comment)
);


CREATE TABLE Review2 (
        cusPhone            varchar(15),
        timeStamp           datetime,
        title               varchar(50)         NOT NULL,
        image               varchar(255),
        comment             varchar(255)        NOT NULL,
        PRIMARY KEY (cusPhone, timeStamp),
        FOREIGN KEY (cusPhone) REFERENCES Customers(phone)
                ON DELETE CASCADE,
        FOREIGN KEY (comment) REFERENCES Review1(comment)
                ON DELETE CASCADE
);
```

```
CREATE TABLE FoodReviews (
        cusPhone                varchar(15),
        timeStamp               datetime,
        restAddress             varchar(255)            NOT NULL,
        FoodQualityRating       int,
        PortionSizeRating       int,
        PRIMARY KEY (cusPhone, timeStamp),
        FOREIGN KEY (cusPhone) REFERENCES Customers(phone)
                ON DELETE CASCADE,
        FOREIGN KEY (timeStamp) REFERENCES Review2(timeStamp)
                ON DELETE CASCADE,
        FOREIGN KEY (restAddress) REFERENCES Restaurants(address)
                ON DELETE CASCADE
);


CREATE TABLE DriverReviews (
        cusPhone                        varchar(15),
        timeStamp                       datetime,
        driverPhone                     varchar(15)     NOT NULL,
        PackageHandlingRating           int,
        DeliveryTimeRating              int,
        PRIMARY KEY (cusPhone, timeStamp),
        FOREIGN KEY (cusPhone) REFERENCES Customers(phone)
                ON DELETE CASCADE,
        FOREIGN KEY (timeStamp) REFERENCES Review2(timeStamp)
                ON DELETE CASCADE,
        FOREIGN KEY (driverPhone) REFERENCES Drivers(phone)
                ON DELETE CASCADE
);
```

```
CREATE TABLE Payment1 (
        orderID             int,
        price               decimal(5, 2)       NOT NULL,
        restAddress         varchar(255)        NOT NULL,
        driverPhone         varchar(15)         NOT NULL,
        PRIMARY KEY (orderID),
        FOREIGN KEY (orderID) REFERENCES Orders(orderID)
                ON DELETE CASCADE,
        FOREIGN KEY (restAddress) REFERENCES Restaurants(address)
                ON DELETE NO ACTION,
        FOREIGN KEY (driverPhone) REFERENCES Drivers(phone)
                ON DELETE NO ACTION
);


CREATE TABLE Payment2 (
        paymentID           int,
        status              varchar(7)          NOT NULL,
        orderID             int                 NOT NULL,
        PRIMARY KEY (paymentID),
        FOREIGN KEY (orderID) REFERENCES Payment1(orderID)
                ON DELETE CASCADE
);
```

University of British Columbia, Vancouver
Department of Computer Science

Insert Statements:

```
INSERT INTO   Customers (name, address, phone, email, customerAcc, paymentInfo,
              membership, points, discount)
VALUES        ('Alice Smith', '123 Broadway', ‘6041112222’, 'alice@example.com', 'alice001',
              'Credit Card', 'yes', 150, '2%'),
              ('Bob Johnson', '456 Broadway', ‘6043334444’, 'bob@example.com', 'bob2002',
              'PayPal', 'no', 80, NULL),
              ('John Ha', '789 Broadway', ‘6045556666’, 'john2002@example.com', 'john2002',
              'Debit Card', 'yes', 200, '5%'),

              ('John Brown', '101 Broadway', ‘6047778888’, 'john2001@example.com',
              'john2001', 'Credit Card', 'yes', 50, '1%'),
              ('John Green', '202 Broadway', ‘6049990000’, 'john1999@example.com',
              'john1999', 'Credit Card', 'no', 180, '4%');


INSERT INTO   Restaurants (name, address, openHours, restaurantAcc)
VALUES        ('Saladland', '100 Broadway', '10:00-22:00', 'saladland'),
              ('Burgerland', '200 Broadway', '11:00-23:00', 'burgerland'),
              ('Pastaland', '300 Broadway', '09:00-21:00', 'pastaland'),
              ('Noodleland', '400 Broadway', '08:00-20:00', 'noodleland'),
              ('Sushiland', '500 Broadway', '12:00-24:00', 'sushiland'),
              ('Pizzaland', '600 Broadway', '12:00-22:00', 'pizzaland');


INSERT INTO   Menu1 (name, cuisine)
VALUES        ('Classic Chicken Salad', 'Italian'),
              ('Burger Menu', 'American'),
              ('Pasta Menu', 'Italian'),
              ('Noodle Menu', 'Chinese'),
              ('Sushi Menu', 'Japanese'),
              ('Pizza Menu', 'Italian');
```

```
INSERT INTO   Menu2 (menuID, name)
VALUES        (1, 'Salad Menu'),
              (2, 'Burger Menu'),
              (3, 'Pasta Menu'),
              (4, 'Noodle Menu'),
              (5, 'Sushi Menu'),
              (6, 'Pizza Menu');

INSERT INTO   Own (restAddress, MID)
VALUES        ('100 Broadway', 1),
              ('200 Broadway', 2),
              ('300 Broadway', 3),
              ('400 Broadway', 4),
              ('500 Broadway', 5),
              ('600 Broadway', 6);


INSERT INTO   Contain (MID, fName)
VALUES        (1, 'classic chicken salad'),
              (2, 'double cheeseburger'),
              (3, 'creamed spinach pasta'),
              (4, 'beef noodle soup'),
              (5, 'California Roll'),
              (6, 'medium classic pepperoni pizza');


INSERT INTO   Food1 (foodType, hasVeganDiet)
VALUES        (‘salad’, ‘yes’),
              (‘burger’, ‘no’),
              (‘pasta’, ‘yes’),
              (‘Chinese noodle soup’, ‘no’),
              (‘sushi’, ‘yes’),
              (‘pizza’, ‘yes’);
```

```
INSERT INTO   Food2 (foodName, foodPrice, foodType)
VALUES        (‘classic chicken salad’, 6.00, ‘salad’),
              (‘double cheeseburger’, 11.00, ‘burger’),
              (‘creamed spinach pasta’, 15.00, ‘pasta’),
              (‘beef noodle soup’, 18.00, ‘Chinese noodle soup’),
              (‘California Roll’, 10.00, ‘sushi’),
              (‘medium classic pepperoni pizza’, 17.00, ‘pizza’);

INSERT INTO   Ordered (fName, orderID, orderedQuantity)
VALUES        (‘classic chicken salad’, 1, 1),
              (‘double cheeseburger’, 1, 1),
              (‘double cheeseburger’, 2, 2),
              (‘creamed spinach pasta’, 3, 2),
              (‘beef noodle soup’, 4, 1),
              (‘California Roll’, 5, 3),
              (‘medium classic pepperoni pizza’, 6, 2);


INSERT INTO   Orders (orderID, totalPrice, timeArrival, cusPhone, restAddress, driverPhone)
VALUES        (1, 17.00, ‘2024-07-25 18:30:00’, ‘6041112222’, ‘100 Broadway’, ‘6041111111’),
              (2, 22.00, ‘2024-08-10 15:00:00’, ‘6043334444’, ‘200 Broadway’, ‘6042222222’),
              (3, 30.00, ‘2024-09-01 12:05:00’, ‘6045556666’, ‘300 Broadway’, ‘6043333333’),
              (4, 18.00, ‘2024-12-01 09:25:00’, ‘6045556666’, ‘400 Broadway’, ‘6044444444’),
              (5, 30.00, ‘2025-02-28 11:10:00’, ‘6047778888’, ‘500 Broadway’, ‘6045555555’),
              (6, 34.00, ‘2025-07-26 13:45:00’, ‘6049990000’, ‘600 Broadway’, ‘6046666666’);


INSERT INTO   Drivers (name, phone, driverAcc, vehicle)
VALUES        (‘Robert Chambers’, ‘6041111111’, ‘rc1994’, ‘Tesla Model Y’),
              (‘Lucy Armstrong’, ‘6042222222’, ‘rabbit458’, ‘Toyota Corolla’),
              (‘Aaron Sharp’, ‘6043333333’, ‘yankeesforever88’, ‘Porsche 718 Cayman’),
              (‘Robert Chambers’, ‘6044444444’, ‘iamkeanureeves777’, ‘Tesla Model Y’),
              (‘Rafael Walls’, ‘6045555555’, ‘rflw999’, ‘Ford Escape’),
              (‘William Robinson’, ‘6046666666’, ‘robot55123’, ‘Tesla Model Y’);
```

```
INSERT INTO   Review1 (comment, ratingType)
VALUES        (‘very good’, ‘positive’),
              (‘good’, ‘positive’),
              (‘good’, ‘positive’),
              (‘not bad’, ‘positive’),
              (‘disgusting’, ‘negative’),
              (‘smells bad’, ‘negative’);


INSERT INTO   Review2 (cusPhone, timeStamp, title, image, comment)
VALUES        (‘6041112222’, ‘2024-07-25 19:30:00’, ‘Amazing’, ‘link_1’, ‘very good’),
              (‘6043334444’, ‘2024-08-10 16:00:00’, ‘Fine’, ‘link_2’, ‘good’),
              (‘6045556666’, ‘2024-09-01 13:05:00’, ‘I like this’, ‘link_3’, ‘good’),
              (‘6045556666’, ‘2024-12-01 10:25:00’, ‘Good’, ‘link_4’, ‘not bad’),
              (‘6047778888’, ‘2025-02-28 12:10:00’, ‘So bad’, ‘link_5’, ‘disgusting’),
              (‘6049990000’, ‘2025-07-26 14:45:00’, ‘Not recommended’, ‘link_6’, ‘smells
              bad’);

INSERT INTO   FoodReviews (cusPhone, timeStamp, restAddress, FoodQualityRating,
              PortionSizeRating)
VALUES        (‘6041112222’, ‘2024-07-25 19:30:00’, ‘100 Broadway’, 5, 4),
              (‘6043334444’, ‘2024-08-10 16:00:00’, ‘200 Broadway’, 4, 5),
              (‘6045556666’, ‘2024-09-01 13:05:00’, ‘300 Broadway’, 4, 4),
              (‘6045556666’, ‘2024-12-01 10:25:00’, ‘400 Broadway’, 3, 3),
              (‘6047778888’, ‘2025-02-28 12:10:00’, ‘500 Broadway’, 2, 1),
              (‘6049990000’, ‘2025-07-26 14:45:00’, ‘600 Broadway’, 1, 2);

INSERT INTO   DriverReviews (cusPhone, timeStamp, driverPhone, PackageHandlingRating,
              DeliveryTimeRating)
VALUES        (‘6041112222’, ‘2024-07-25 19:30:00’, ‘6041111111’, 5, 5),
              (‘6043334444’, ‘2024-08-10 16:00:00’, ‘6042222222’, 5, 4),
              (‘6045556666’, ‘2024-09-01 13:05:00’, ‘6043333333’, 4, 4),
              (‘6045556666’, ‘2024-12-01 10:25:00’, ‘6044444444’, 2, 2),
              (‘6047778888’, ‘2025-02-28 12:10:00’, ‘6045555555’, 4, 4),
              (‘6049990000’, ‘2025-07-26 14:45:00’, ‘6046666666’, 5, 5);
```

```
INSERT INTO   Payment1 (orderID, price, restAddress, driverPhone)
VALUES        (1, ‘17.00’, ‘100 Broadway’, ‘6041111111’),
              (2, ‘22.00’, ‘200 Broadway’, ‘6042222222’),
              (3, ‘30.00’, ‘300 Broadway’, ‘6043333333’),
              (4, ‘18.00’, ‘400 Broadway’, ‘6044444444’),
              (5, ‘30.00’, ‘500 Broadway’, ‘6045555555’),
              (6, ‘34.00’, ‘600 Broadway’, ‘6046666666’);


INSERT INTO   Payment2 (paymentID, status, orderID)
VALUES        (1, ‘success’, 1),
              (2, ‘success’, 2),
              (3, ‘failure’, 3),
              (4, ‘success’, 3),
              (5, ‘success’, 4),
              (6, ‘success’, 5),
              (7, ‘success’, 6);
```