University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct 15,2023

Group Number: 123

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Kentaro Lim	44267326	klim10	kentarolim10@gmail.com
Riley Baines	98686033	rbaine01	rileybaines@outlook.com
Tony Liu	42641019	tl0226	tl0226yn@gmail.com

Summary

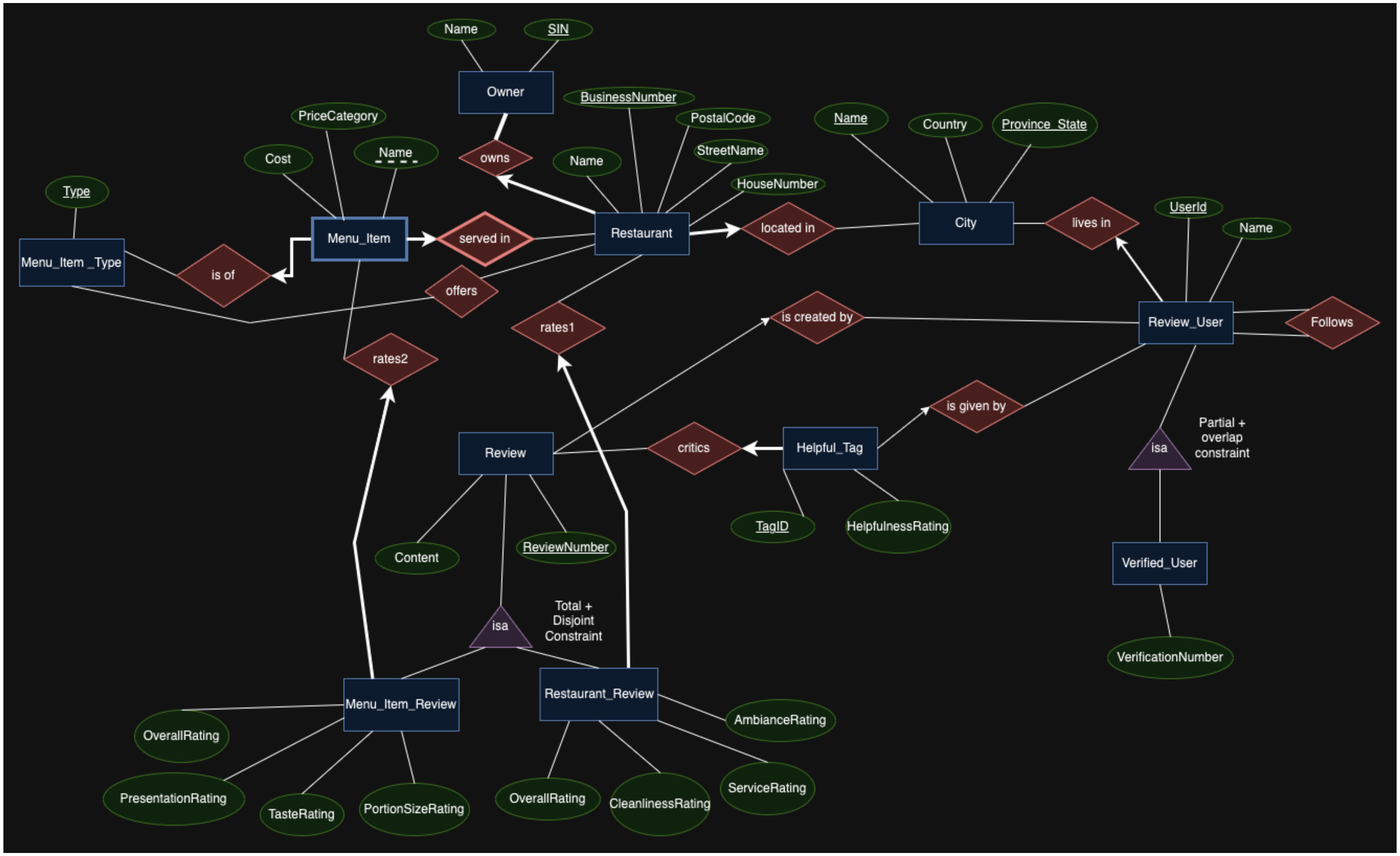
This project is an application that allows the user to read and write reviews for restaurants. However, the user is also allowed to read and write reviews for the food items themselves, giving a better understanding on recommendations and foods to avoid.

ER Diagram

Changes:

- For attributes, got rid of spaces and just use camel case
- For entities, remove spaces and use underscores
- Replace '#' in attributes for 'Number'
- Replace 'Province/State' with 'Province_State'
- Dashed underline for Name in MenuItem
- Switch Menu_Item_Review and Restaurant_Review_Positions
- Change Review and User "is given by" relation to "is created by"
- Change Restaurant and Restaurant_Review "rates" relation to "rates1"
- Change Menu_Item and Menu_Item_Review "rates" relation to "rates2"
- We decided to keep Menu_Item_Type with only one attribute being the primary key. The reason why we decided to use an entity to describe a menu item's type instead of an attribute is because many food items may have the same type (Ex. California Roll and Salmon Nigiri are of type Sushi). As a result, we choose an entity to reduce repetitiveness.
- Bolded the line from Restaurant to Owner in the owns relationship
- We decided to keep Review as an entity rather than a weak entity and remove the constraint saying that a Review has to have a User. This is because if we have a user that creates a review, and then we decide to delete the user, we still want to be able to keep the review.
- Similar to the Review, we removed the constraint saying that a Helpful_Tag has to have a User. If we delete a user, we still want to keep a helpful tag.
- Remove Rating from Review. We decided to move it into the two IsAs.
- Add OverallRating, CleanlinessRating, Service Rating, and AmbianceRating into Restaurant_Review
- Add OverallRating, TasteRating, PortionSizeRating, and PresentationRating into Menu_Item_Review
- Remove MenuItemName from Menu_Item_Review. This is already implicitly stated with the rates2 relation
- Remove RestaurantName from Restaurant_Review.
- In Restaurant, replace Address with HouseNumber, StreetName, and PostalCode
- In Menu_Item, add PriceCategory attribute

University of British Columbia, Vancouver
Department of Computer Science



University of British Columbia, Vancouver
Department of Computer Science

Schema

- City(Name: char(30) (PK), Province_State: char(30) (PK), Country: char(30))
- Review_User(UserId: char(30) (PK), Name: char(30), CityName: char(30) (FK - references Name in City) (NOT NULL), Province_State: char(30) (FK - references Province_State in City) (NOT NULL))
- Follows(UserId1: char(30) (PK) (FK - references UserId in User) (NOT NULL), UserId2: char(30) (PK) (FK - references UserId in User) (NOT NULL))
- Verified_User(UserId: char(30) (FK - references UserId in User) (NOT NULL), VerificationNumber: int (UNIQUE))
- Restaurant(BusinessNumber: int (PK), StreetName: char(50), HouseNumber: int, PostalCode: char(7), Name: char(50), CityName: char(30) (FK - references Name in City) (NOT NULL), Province_State: char(30) (FK - references Province_State in City) (NOT NULL), OwnerId: int (FK - references SIN in owner) (NOT NULL))
- Owner(SIN: int (PK), Name: char(30))
- Menu_Item(BusinessNumber: int (PK and FK referencing BusinessNumber in Restaurant), Name: char(50) (PK), Cost: float(to 2 decimal places), PriceCategory: int, Type: char(30) (FK - references Menu_Item_Type) (NOT NULL))
- Menu_Item_Type(Type: char(30) (PK))
- offers(Business Number: int (PK) (FK - references BusinessNumber in Restaurant), Type: char(30) (PK) (FK - references Menu_Item_Type))
- Review(ReviewNumber: int (PK), Content: char(4000), UserId: char(30) (FK - references UserId in User))
- Restaurant_Review(ReviewNumber: int (FK - references ReviewNumber in Review) (NOT NULL), BusinessNumber: int (FK - references BusinessNumber in Restaurant) (NOT NULL), AmbienceRating: int, CleanlinessRating: int, ServiceRating: int, OverallRating: int)
- Menu_Item_Review(ReviewNumber: int (FK - references ReviewNumber in Review) (NOT NULL), Business Number: int (FK - references BusinessNumber in Restaurant) (NOT NULL), PresentationRating: int, TasteRating: int, PortionSizeRating: int, OverallRating: int)
- Helpful_Tag(TagID: int(PK), HelpfulnessRating: int (NOT NULL), ReviewNumber: int (FK - references ReviewNumber in Review), UserId: char(30) (FK - references UserId in User))

Functional Dependencies

- **City:** Name, Province_State -> Country
- **User:** UserId -> Name, CityName, Province_State
- **Verified_User:** VerificationNumber -> UserId
- **Restaurant:**
  - BusinessNumber -> HouseNumber, StreetName, PostalCode, Name, OwnerId
  - PostalCode -> Province_State
  - HouseNumber, StreetName, PostalCode -> CityName, Province_State
- **Owner:** SIN -> Name
- **Menu_Item:**
  - BusinessNumber, Name -> Cost, Type, PriceCategory
  - Cost -> PriceCategory
- **Review:** ReviewNumber -> Content, UserId
- **Restaurant_Review:**
  - ReviewNumber -> BusinessNumber, OverallRating, CleanlinessRating, ServiceRating, AmbianceRating
  - CleanlinessRating, ServiceRating, AmbianceRating -> OverallRating
- **Menu_Item_Review:**
  - ReviewNumber -> BusinessNumber, OverallRating, TasteRating, PresentationRating, PortionSizeRating
  - TasteRating, PresentationRating, PortionSizeRating -> OverallRating
- **Helpful_Tag:** TagID -> ReviewNumber, UserId, HelpfulnessRating

Normalization

- City(Name: char(30) (PK), Province_State: char(30) (PK), Country: char(30))
- User(UserId: char(30) (PK), Name: char(30), CityName: char(30) (FK - references Name in City) (NOT NULL), Province_State: char(30) (FK - references Province_State in City) (NOT NULL))
- Follows(UserId1: char(30) (PK) (FK - references UserId in User) (NOT NULL), UserId2: char(30) (PK) (FK - references UserId in User) (NOT NULL))
- Verified_User(UserId: char(30) (FK - references UserId in User) (NOT NULL), VerificationNumber: int (UNIQUE))
- Owner(SIN: int (PK), Name: char(30))
- Menu_Item_Type(Type: char(30) (PK))
- Offers(BusinessNumber: int (PK) (FK - references BusinessNumber in Restaurant), Type: char(30) (PK) (FK - references Menu_Item_Type))
- Review(ReviewNumber: int (PK), Content: char(4000), UserId: char(30) (FK - references UserId in User))
- Helpful_Tag(TagID: int(PK), HelpfulnessRating: int (NOT NULL), ReviewNumber: int (FK - references ReviewNumber in Review), UserId: char(30) (FK - references UserId in User))
- Restaurant is currently not in 3NF
    - Make all functional dependencies into minimal cover
        - (BusinessNumber -> HouseNumber, StreetName, PostalCode, Name, OwnerId) has multiple attributes in RHS
            - BusinessNumber -> HouseNumber
            - BusinessNumber -> StreetName
            - BusinessNumber -> PostalCode
            - BusinessNumber -> Name
            - BusinessNumber -> OwnerId
        - (HouseNumber, StreetName, PostalCode -> CityName, Province_State) has multiple attributes in RHS
            - HouseNumber, StreetName, PostalCode -> CityName
            - HouseNumber, StreetName, PostalCode -> Province_State
        - (HouseNumber, StreetName, PostalCode -> Province_State) has unnecessary attributes on LHS
            - PostalCode -> Province_State

- (PostalCode -> Province_State) is written twice. Remove one.
- Get all coverages
  - BusinessNumber$^+$ = {BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId, CityName, Province_State}
  - PostalCode$^+$ = {PostalCode, Province_State}
  - (HouseNumber, StreetName, PostalCode)$^+$ = {HouseNumber, StreetName, PostalCode, CityName}
- Get Minimal Key
  - Chart with each attribute
    - LHS: BusinessNumber
    - Middle: PostalCode, HouseNumber, StreetName, PostalCode
    - RHS: OwnerId, Name
  - We've seen BusinessNumber has coverage of all attributes
  - Therefore, minimal key is BusinessNumber
- (PostalCode -> Province_State) violates 3NF for Restaurant
  - Split relation into Restaurant(BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId, CityName) and Postal_Area(PostalCode, Province_State)
- HouseNumber, StreetName, PostalCode -> CityName violates 3NF for Restaurant
  - Split relation into Restaurant(BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId) and Address(HouseNumber, StreetName, PostalCode, CityName)
- **These relations are now in 3NF so our new relations are:**
  - Restaurant(BusinessNumber: int (PK), StreetName: char(50) (FK - references StreetName in Address), House #: int (FK - references HouseNumber in Address), PostalCode: char(7) (FK - references PostalCode in Address, references PostalCode in Postal_Area), Name: char(50), OwnerId: int (FK - references SIN in Owner) (NOT NULL))
  - Postal_Area(PostalCode: char(7) (PK), Province_State: int)
  - Address(HouseNumber: int (PK), StreetName: char(50) (PK), PostalCode: char(7) (PK), CityName: char(30) (NOT NULL))

- Menu Item is currently not in 3NF
  - Make all functional dependencies into minimal cover
    - (BusinessNumber, Name -> Cost, Type, PriceCategory) has three attributes in RHS
      - BusinessNumber, Name -> Cost

            - BusinessNumber, Name -> Type
            - BusinessNumber, Name -> PriceCategory
    - Get all coverages
        - (BusinessNumber, Name)$^+$={BusinessNumber, Name, Cost, PriceCategory, Type}
        - (Cost)$^+$ = {Cost, PriceCategory}
    - Get Minimal Key
        - Chart with each attribute
            - LHS: BusinessNumber, Name
            - Middle: Cost
            - RHS: Name, PriceCategory, Type
        - BusinessNumber and Name has a coverage of all attributes
        - Therefore, minimal key is (BusinessNumber, Name)
    - (Cost -> PriceCategory) violates 3NF
        - Split relations into Menu Item(BusinessNumber, Name, Cost, Type) and Cost_Map(Cost, PriceCategory)
    - **These relations are in 3NF so our new relations are:**
        - Menu_Item(BusinessNumber: int (PK and FK referencing BusinessNumber in Restaurant), Name: char(50) (PK), Cost: float(to 2 decimal places) (FK - references Cost_Map), Type: char(30) (FK - references Menu_Item_Type) (NOT NULL))
        - Cost_Map(Cost: float(to 2 decimal places) (PK), PriceCategory: int)
- Restaurant_Review is not in 3NF
    - Make all functional dependencies into minimal cover
        - (ReviewNumber -> BusinessNumber, OverallRating, CleanlinessRating, ServiceRating, AmbianceRating) has multiple attributes on RHS
            - ReviewNumber -> BusinessNumber
            - ReviewNumber -> OverallRating
            - ReviewNumber -> CleanlinessRating
            - ReviewNumber -> ServiceRating
            - ReviewNumber -> AmbianceRating
    - Get all coverages
        - (ReviewNumber)$^+$ = {ReviewNumber, BusinessNumber, OverallRating, CleanlinessRating, ServiceRating, AmbianceRating}

        - (CleanlinessRating, ServiceRating, AmbianceRating)$^+$ = {CleanlinessRating, ServiceRating, AmbianceRating, OverallRating}
    - Get Minimal key
        - Chart with each attribute
            - LHS: ReviewNumber
            - Middle: CleanlinessRating, ServiceRating, AmbianceRating
            - RHS: BusinessNumber, OverallRating
        - ReviewNumber has coverage of all attributes
        - Therefore, minimal key is (ReviewNumber)
    - (CleanlinessRating, ServiceRating, AmbianceRating -> OverallRating) violates 3NF
        - Split relation into Restaurant_Review(ReviewNumber, BusinessNumber, CleanlinessRating, ServiceRating, AmbianceRating) and Restaurant_Rating(CleanlinessRating, ServiceRating, AmbianceRating, OverallRating)
    - **All relations are now in 3NF so our new relations are:**
        - Restaurant_Review(ReviewNumber: int (FK - references ReviewNumber in Review) (NOT NULL), BusinessNumber: int (FK - references BusinessNumber in Restaurant) (NOT NULL), AmbienceRating: int (FK - references AmbienceRating in Restaurant_Rating), CleanlinessRating: int (FK references CleanlinessRating in Restaurant_Rating), ServiceRating: int (FK references ServiceRating in Restaurant_Rating))
        - Restaurant_Rating(AmbienceRating: int (PK), CleanlinessRating: int (PK), ServiceRating: int (PK), OverallRating: int)
- Menu_Item_Review is not in 3NF
    - Make all functional dependencies into minimal cover
        - (ReviewNumber -> BusinessNumber, OverallRating, TasteRating, PresentationRating, PortionSizeRating) has multiple attributes on RHS
            - ReviewNumber -> BusinessNumber
            - ReviewNumber -> OverallRating
            - ReviewNumber -> TasteRating
            - ReviewNumber -> PresentationRating
            - ReviewNumber -> PortionSizeRating
    - Get all coverages
        - (ReviewNumber)$^+$ = {ReviewNumber, BusinessNumber, OverallRating, TasteRating, PresentationRating, PortionSizeRating}

- (TasteRating, PresentationRating, PortionSizeRating)$^+$ -> {TasteRating, PresentationRating, PortionSizeRating, OverallRating}

- Get minimal key
    - Chart with each attribute
        - LHS: ReviewNumber
        - Middle: TasteRating, PresentationRating, PortionSizeRating
        - RHS: BusinessNumber, OverallRating
    - ReviewNumber has coverage of all attributes
    - Therefore, minimal key is (ReviewNumber)
- (TasteRating, PresentationRating, PortionSizeRating -> OverallRating) violates in 3NF
    - Split relation into Menu_Item_Review(ReviewNumber, BusinessNumber, TasteRating, PresentationRating, PortionSizeRating) and MenuItemRating(TasteRating, PresentationRating, PortionSizeRating, OverallRating)
- **All relations are now in 3NF so our new relations are:**
    - Menu_Item_Review(ReviewNumber: int (FK - references ReviewNumber in Review) (NOT NULL), BusinessNumber: int (FK - references BusinessNumber in Restaurant) (NOT NULL), PresentationRating: int (FK - references PresentationRating in Menu_Item_Rating), TasteRating: int (FK - references TasteRating in Menu_Item_Rating), PortionSizeRating: int (FK - references PortionSizeRating in Menu_Item_Rating))
    - Menu_Item_Rating(PresentationRating: int (PK), TasteRating: int (PK), PortionSizeRating: int (PK), OverallRating: int)

University of British Columbia, Vancouver
Department of Computer Science

SQL DDL CREATE TABLE Statements

```
CREATE TABLE City(
  Name VARCHAR(30),
  Province_State VARCHAR(30),
  Country VARCHAR(30),
  PRIMARY KEY(Name, Province_State)
  );

CREATE TABLE Review_User(
  UserId VARCHAR(30) PRIMARY KEY,
  Name VARCHAR(30),
  CityName VARCHAR(30) NOT NULL,
  Province_State VARCHAR(30) NOT NULL,
  FOREIGN KEY (CityName, Province_State) REFERENCES City ON DELETE NO ACTION
  );

CREATE TABLE Follows(
  UserId1 VARCHAR(30) NOT NULL,
  UserId2 VARCHAR(30) NOT NULL,
  PRIMARY KEY(UserId1,UserId2),
  FOREIGN KEY (UserId1) REFERENCES Review_User ON DELETE CASCADE,
  FOREIGN KEY (UserId2) REFERENCES Review_User ON DELETE CASCADE
 );

CREATE TABLE Verified_User(
  UserId VARCHAR(30) NOT NULL,
  VerificationNumber INT UNIQUE,
  FOREIGN KEY (UserId) REFERENCES Review_User ON DELETE CASCADE
 );

CREATE TABLE Owner(
  SIN INT PRIMARY KEY,
  Name VARCHAR(30)
 );

CREATE TABLE Postal_Area(
  PostalCode VARCHAR(7) PRIMARY KEY,
  Province_State VARCHAR(30)
  );

CREATE TABLE Address(
  HouseNumber INT,
  StreetName VARCHAR(50),
  PostalCode VARCHAR(7),
  CityName VARCHAR(30) NOT NULL,
  PRIMARY KEY (HouseNumber,StreetName,PostalCode)
  );

CREATE TABLE Restaurant(
  BusinessNumber INT PRIMARY KEY,
  HouseNumber INT,
  StreetName VARCHAR(50),
  PostalCode VARCHAR(7),
  Name VARCHAR(50),
  OwnerId INT DEFAULT -1 NOT NULL,
  FOREIGN KEY (HouseNumber,StreetName,PostalCode) REFERENCES Address ON DELETE SET NULL,
  FOREIGN KEY (PostalCode) REFERENCES Postal_Area ON DELETE NO ACTION,
  FOREIGN KEY (OwnerId) REFERENCES Owner ON DELETE SET DEFAULT,
  );

CREATE TABLE Menu_Item_Type(
  Type VARCHAR(30) PRIMARY KEY
  );

CREATE TABLE offers(
  BusinessNumber INT,
  Type VARCHAR(30),
  PRIMARY KEY (BusinessNumber,Type),
  FOREIGN KEY (BusinessNumber) REFERENCES Restaurant ON DELETE CASCADE,
  FOREIGN KEY (Type) REFERENCES Menu_Item_Type ON DELETE NO ACTION
  );

CREATE TABLE Cost_Map(
  Cost FLOAT(7,2) PRIMARY KEY,
  PriceCategory INT
  );

CREATE TABLE Menu_Item(
  BusinessNumber INT,
  Name VARCHAR(50),
  Type VARCHAR(30) NOT NULL,
  Cost FLOAT(7,2),
  PRIMARY KEY (BusinessNumber, Name),
  FOREIGN KEY (BusinessNumber) REFERENCES Restaurant ON DELETE CASCADE,
  FOREIGN KEY (Type) REFERENCES Menu_Item_Type ON DELETE NO ACTION,
  FOREIGN KEY (Cost) REFERENCES Cost_Map ON DELETE NO ACTION,
  );

CREATE TABLE Review(
  ReviewNumber INT PRIMARY KEY,
  Content VARCHAR(4000),
  UserId VARCHAR(30),
  FOREIGN KEY (UserId) REFERENCES Review_User ON DELETE SET NULL,
  );

CREATE TABLE Restaurant_Rating(
  AmbienceRating INT,
  CleanlinessRating INT,
  ServiceRating INT,
  OverallRating INT,
  PRIMARY KEY (AmbienceRating,CleanlinessRating,ServiceRating)
  );

CREATE TABLE Restaurant_Review(
  ReviewNumber INT NOT NULL,
  BusinessNumber INT DEFAULT -1 NOT NULL,
  AmbienceRating INT,
  CleanlinessRating INT,
  ServiceRating INT,
  FOREIGN KEY (ReviewNumber) REFERENCES Review ON DELETE CASCADE,
  FOREIGN KEY (BusinessNumber) REFERENCES Restaurant ON DELETE SET DEFAULT,
  FOREIGN KEY (AmbienceRating,CleanlinessRating,ServiceRating) REFERENCES Restaurant_Rating ON DELETE CASCADE
  );

CREATE TABLE Menu_Item_Rating(
  PresentationRating INT,
  TasteRating INT,
  PortionSizeRating INT,
  OverallRating INT,
  PRIMARY KEY (PresentationRating,TasteRating,PortionSizeRating)
  );

CREATE TABLE Menu_Item_Review(
  ReviewNumber INT NOT NULL,
  BusinessNumber INT DEFAULT -1 NOT NULL,
  MenuItemName VARCHAR(50) DEFAULT "N/A" NOT NULL,
  PresentationRating INT,
  TasteRating INT,
  PortionSizeRating INT,
  FOREIGN KEY (MenuItemName) REFERENCES Menu_Item ON DELETE SET DEFAULT
  FOREIGN KEY (ReviewNumber) REFERENCES Review ON DELETE CASCADE,
  FOREIGN KEY (BusinessNumber) REFERENCES Restaurant ON DELETE SET DEFAULT,
  FOREIGN KEY (PresentationRating,TasteRating,PortionSizeRating) REFERENCES Menu_Item_Rating
  );

CREATE TABLE Helpful_Tag(
  TagID INT PRIMARY KEY,
  HelpfulnessRating INT NOT NULL,
  ReviewNumber INT,
  UserId VARCHAR(30),
  FOREIGN KEY (ReviewNumber) REFERENCES Review ON DELETE SET NULL,
  FOREIGN KEY (UserId) REFERENCES Review_User ON DELETE SET NULL
  );
```

University of British Columbia, Vancouver
Department of Computer Science

SQL DDL INSERT Statements

```
INSERT INTO City(Name, Province_State, Country) VALUES ('Vancouver', 'British Columbia', 'Canada');
INSERT INTO City(Name, Province_State, Country) VALUES ('Richmond', 'British Columbia', 'Canada');
INSERT INTO City(Name, Province_State, Country) VALUES ('Toronto', 'Ontario', 'Canada');
INSERT INTO City(Name, Province_State, Country) VALUES ('Ottawa', 'Ontario', 'Canada');
INSERT INTO City(Name, Province_State, Country) VALUES ('Los Angeles', 'California', 'United States of America');

INSERT INTO Review_User(UserId,Name,CityName,Province_State) VALUES ('user1','John','Toronto','Ontario');
INSERT INTO Review_User(UserId,Name,CityName,Province_State) VALUES ('user2','Jane','Los Angeles','California');
INSERT INTO Review_User(UserId,Name,CityName,Province_State) VALUES ('user3','Kentaro','Vancouver','British Columbia');
INSERT INTO Review_User(UserId,Name,CityName,Province_State) VALUES ('user4','Tony','Vancouver','British Columbia');
INSERT INTO Review_User(UserId,Name,CityName,Province_State) VALUES ('user5','Riley','Vancouver','British Columbia');
INSERT INTO Review_User(UserId,Name,CityName,Province_State) VALUES ('user6','ChatGPT','Vancouver','British Columbia');

INSERT INTO Follows(UserId1, UserId2) VALUES (‘user1’, ‘user2’);
INSERT INTO Follows(UserId1, UserId2) VALUES(‘user3’, ‘user4’);
INSERT INTO Follows(UserId1, UserId2) VALUES(‘user1’, ‘user3’);
INSERT INTO Follows(UserId1, UserId2) VALUES(‘user4’, ‘user2’);
INSERT INTO Follows(UserId1, UserId2) VALUES(‘user5’, ‘user6’);

INSERT INTO Verified_Users(UserId, VerificationNumber) VALUES (‘user1’, 1);
INSERT INTO Verified_Users(UserId, VerificationNumber) VALUES (‘user2’, 2);
INSERT INTO Verified_Users(UserId, VerificationNumber) VALUES (‘user3’, 3);
INSERT INTO Verified_Users(UserId, VerificationNumber) VALUES (‘user4’, 4);
INSERT INTO Verified_Users(UserId, VerificationNumber) VALUES (‘user5’, 5);
INSERT INTO Owner(SIN, Name) VALUES (123456789, ‘Bob’);
INSERT INTO Owner(SIN, Name) VALUES (456789123, ‘Jick’);
INSERT INTO Owner(SIN, Name) VALUES (789123456, ‘Jane’);
INSERT INTO Owner(SIN, Name) VALUES (987654321, ‘Bob’);
INSERT INTO Owner(SIN, Name) VALUES (654321987, ‘Jane’);

INSERT INTO Postal_Area(PostalCode, Province_State) VALUES (‘V6T 1Z4’, ‘British Columbia’);
INSERT INTO Postal_Area(PostalCode, Province_State) VALUES (‘T2X 2L9’, ‘Alberta’);
INSERT INTO Postal_Area(PostalCode, Province_State) VALUES (‘V6B 1M8’, ‘British Columbia’);
INSERT INTO Postal_Area(PostalCode, Province_State) VALUES (‘K1A 0A9’, ‘Ontario’);
INSERT INTO Postal_Area(PostalCode, Province_State) VALUES (‘M5V 3L9’, ‘Ontario’);

INSERT INTO Address(HouseNumber, StreetName, PostalCode, CityName) VALUES (601, ‘W Hastings Street’, ‘V6B 1M8’, ‘Vancouver’);
INSERT INTO Address(HouseNumber, StreetName, PostalCode, CityName) VALUES (2205, ‘Lower Mall’, ‘V6T 1Z4’, ‘Vancouver’);
INSERT INTO Address(HouseNumber, StreetName, PostalCode, CityName) VALUES (6363, ‘Agronomy Road’, ‘V6T 1Z4’, ‘Vancouver’);
INSERT INTO Address(HouseNumber, StreetName, PostalCode, CityName) VALUES (290, ‘Bremner Blvd’, ‘M5V 3L9’, ‘Toronto’);
INSERT INTO Address(HouseNumber, StreetName, PostalCode, CityName) VALUES (1, ‘Wellington Street’, ‘K1A 0A9’, ‘Ottawa’);

INSERT INTO Restaurant(BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId) VALUES (123, 601, ‘W Hastings
Street’, ‘V6B 1M8’, ‘Bob’s Generic Pizza Place’, 123456789);
INSERT INTO Restaurant(BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId) VALUES (234, 2205, ‘Lower
Mall’, ‘V6T 1Z4’, ‘The Point’, 789123456);
INSERT INTO Restaurant(BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId) VALUES (345, 6363, ‘Agronomy
Road’, ‘V6T 1Z4’, ‘Orchard Commons’, 456789123);
INSERT INTO Restaurant(BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId) VALUES (456, 6363, ‘Agronomy
Road’, ‘V6T 1Z4’, ‘McDonalds’, 654321987);
INSERT INTO Restaurant(BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId) VALUES (999, 1, ‘Wellington
Street’, ‘K1A 0A9’, ‘Bob’s Underground Food Court’, 987654321);
INSERT INTO Restaurant(BusinessNumber, HouseNumber, StreetName, PostalCode, Name, OwnerId) VALUES (909, 290, ‘Bremner
Blvd’, ‘M5V 3L9’, ‘Bob’s Sky High Food Court’, 987654321);
```

```
INSERT INTO Menu_Item_Type(Type) VALUES (‘Pizza’);
INSERT INTO Menu_Item_Type(Type) VALUES (‘Pasta’);
INSERT INTO Menu_Item_Type(Type) VALUES (‘Burgers’);
INSERT INTO Menu_Item_Type(Type) VALUES (‘Dim Sum’);
INSERT INTO Menu_Item_Type(Type) VALUES (‘Sushi’);

INSERT INTO Offers(BusinessNumber, Type) VALUES (123, ‘Pizza’);
INSERT INTO Offers(BusinessNumber, Type) VALUES (234, ‘Burgers’);
INSERT INTO Offers(BusinessNumber, Type) VALUES (234, ‘Pasta’);
INSERT INTO Offers(BusinessNumber, Type) VALUES (345, ‘Burgers’);
INSERT INTO Offers(BusinessNumber, Type) VALUES (345, ‘Dim Sum’);
INSERT INTO Offers(BusinessNumber, Type) VALUES (456, ‘Burgers’);
INSERT INTO Offers(BusinessNumber, Type) VALUES (999, ‘Dim Sum’);
INSERT INTO Offers(BusinessNumber, Type) VALUES (999, ‘Sushi’);
INSERT INTO Offers(BusinessNumber, Type) VALUES (909, ‘Pasta’);

INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (4.99, 1);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (7.99, 1);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (9.99, 1);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (12.99, 2);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (15.00, 2);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (19.99, 2);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (23.99, 2);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (25.00, 3);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (30.00, 3);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (34.99, 3);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (40.00, 4);
INSERT INTO Cost_Map(Cost, PriceCategory) VALUES (99.99, 5);
INSERT INTO Menu_Item(BusinessNumber, Name, Type, Cost) VALUES (123, ‘Pepperoni Pizza’, ‘Pizza’, 15.00);
INSERT INTO Menu_Item(BusinessNumber, Name, Type, Cost) VALUES (234, ‘Cheeseburger’, ‘Burgers’, 19.99);
INSERT INTO Menu_Item(BusinessNumber, Name, Type, Cost) VALUES (234, ‘Lasagna’, ‘Pasta’, 25.00);
INSERT INTO Menu_Item(BusinessNumber, Name, Type, Cost) VALUES (345, ‘Dumplings’, ‘Dim Sum’, 30.00);
INSERT INTO Menu_Item(BusinessNumber, Name, Type, Cost) VALUES (999, ‘Shrimp wrappers’, ‘Dim Sum’, 7.99);
INSERT INTO Menu_Item(BusinessNumber, Name, Type, Cost) VALUES (999, ‘California Roll’, ‘Sushi’, 4.99);
INSERT INTO Menu_Item(BusinessNumber, Name, Type, Cost) VALUES (909, ‘Premium Spaghetti’, ‘Pasta’, 40.00);

INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (1, ‘it sucks’, ‘user1);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (2, ‘it tastes good’, ‘user3’);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (3, ‘it’s too spicy’, ‘user6’);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (4, ‘excited to try more’, ‘user4’);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (5, ‘way too overpriced’, ‘user5’);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (6, ‘the service was okay’, ‘user1);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (7, ‘restaurant smelled bad’, ‘user2’);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (8, ‘i cannot seem to understand why they don’t have enough staff’,
‘user5’);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (9, ‘I really liked their service’, ‘user4’);
INSERT INTO Review(ReviewNumber, Content, UserId) VALUES (10, ‘The environment was nice’, ‘user6’);

INSERT INTO Restaurant_Rating(AmbienceRating, CleanlinessRating, ServiceRating, OverallRating) VALUES (5,5,5,5);
INSERT INTO Restaurant_Rating(AmbienceRating, CleanlinessRating, ServiceRating, OverallRating) VALUES (1,1,1,1);
INSERT INTO Restaurant_Rating(AmbienceRating, CleanlinessRating, ServiceRating, OverallRating) VALUES (3,4,2,3);
INSERT INTO Restaurant_Rating(AmbienceRating, CleanlinessRating, ServiceRating, OverallRating) VALUES (4,3,5,4);
INSERT INTO Restaurant_Rating(AmbienceRating, CleanlinessRating, ServiceRating, OverallRating) VALUES (1,2,5,2);

INSERT INTO Restaurant_Review(ReviewNumber, BusinessNumber, AmbienceRating, CleanlinessRating, ServiceRating) VALUES
(6,999,3,4,2);
INSERT INTO Restaurant_Review(ReviewNumber, BusinessNumber, AmbienceRating, CleanlinessRating, ServiceRating) VALUES
(7,234,1,2,5);
```

```
INSERT INTO Restaurant_Review(ReviewNumber, BusinessNumber, AmbienceRating, CleanlinessRating, ServiceRating) VALUES
(8,456,1,1,1);
INSERT INTO Restaurant_Review(ReviewNumber, BusinessNumber, AmbienceRating, CleanlinessRating, ServiceRating) VALUES
(9,234,4,3,5);
INSERT INTO Restaurant_Review(ReviewNumber, BusinessNumber, AmbienceRating, CleanlinessRating, ServiceRating) VALUES
(10,123,5,5,5);

INSERT INTO Menu_Item_Rating(PresentationRating, TasteRating, PortionSizeRating, OverallRating) VALUES (5,5,5,5);
INSERT INTO Menu_Item_Rating(PresentationRating, TasteRating, PortionSizeRating, OverallRating) VALUES (2,2,2,2);
INSERT INTO Menu_Item_Rating(PresentationRating, TasteRating, PortionSizeRating, OverallRating) VALUES (1,2,1,1);
INSERT INTO Menu_Item_Rating(PresentationRating, TasteRating, PortionSizeRating, OverallRating) VALUES (4,2,3,3);
INSERT INTO Menu_Item_Rating(PresentationRating, TasteRating, PortionSizeRating, OverallRating) VALUES (4,3,5,4);

INSERT INTO Menu_Item_Review(ReviewNumber, BusinessNumber, MenuItemName, PresentationRating, TasteRating,
PortionSizeRating) VALUES (1, 999, ‘Shrimp wrappers’, 2,2,2);
INSERT INTO Menu_Item_Review(ReviewNumber, BusinessNumber, MenuItemName, PresentationRating, TasteRating,
PortionSizeRating) VALUES (2, 234, ‘Cheeseburger’, 5,5,5);
INSERT INTO Menu_Item_Review(ReviewNumber, BusinessNumber, MenuItemName, PresentationRating, TasteRating,
PortionSizeRating) VALUES (3, 345, ‘Dumplings’, 4,2,3);
INSERT INTO Menu_Item_Review(ReviewNumber, BusinessNumber, MenuItemName, PresentationRating, TasteRating,
PortionSizeRating) VALUES (4, 123, ‘Pepperoni Pizza’, 4,3,5);
INSERT INTO Menu_Item_Review(ReviewNumber, BusinessNumber, MenuItemName, PresentationRating, TasteRating,
PortionSizeRating) VALUES (5, 909, ‘Premium Spaghetti’, 1,2,1);

INSERT INTO Helpful_Tag(TagID, HelpfulnessRating, ReviewNumber, UserId) VALUES(1, 1, 3, ‘user2’);
INSERT INTO Helpful_Tag(TagID, HelpfulnessRating, ReviewNumber, UserId) VALUES(2, 0, 1, ‘user6’);
INSERT INTO Helpful_Tag(TagID, HelpfulnessRating, ReviewNumber, UserId) VALUES(3, 1, 7, ‘user4’);
INSERT INTO Helpful_Tag(TagID, HelpfulnessRating, ReviewNumber, UserId) VALUES(4, 1, 7, ‘user6’);
INSERT INTO Helpful_Tag(TagID, HelpfulnessRating, ReviewNumber, UserId) VALUES(5, 0, 10, ‘user1’);
```