University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct 20, 2023

Group Number: 58

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Joshua Bhullar	21693668	k6i8a	jabhullar7@gmail.com
Jason Li	37727179	n2v7m	trust.jason@gmail.com
Bluze Chen	27153352	e3e2e	blu2eh@outlook.com

1. A completed cover page

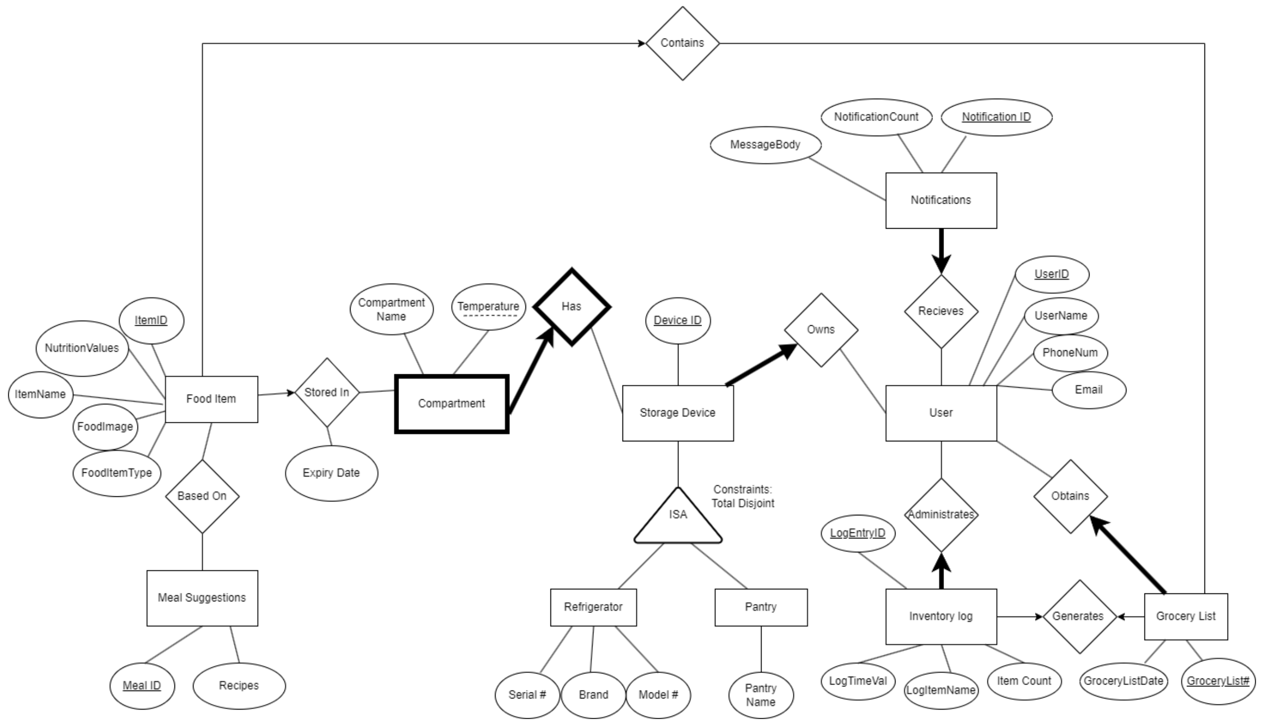
CPSC 304: Project Description
Milestone 2: Relational Schema, Normalization, SQL DDL, Tentative Queries

2. A brief summary of your project.

Our project is the ultimate food organizing application for university students and tech enthusiasts. The app will provide a user with meal suggestions and a grocery list, what food is stored in their fridge/pantry and where it's located inside the fridge/pantry, and also notifications for expiring food and an inventory log of their food.

3. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why.

ER Diagram:



Note:

1. We added two attributes (Compartment name, GroceryList#) to the entities (Compartment, Grocery List). We decided to change GroceryList# because if two users, or even the same user, wanted to create a grocery list on the same day, the system would not support it. Compartment name was changed for the same reason.
2. Changed the primary key for Grocery List to GroceriyList# because we need to uniquely identify each list with a number.
3. We did not add an additional attribute to the entity "Storage Device" as stated by the TA in the Milestone 1 rubric grading. The reason is because we want to ensure that each record in the table is unique based on that attribute alone. Therefore in this situation we

want a more simplistic entity with a clear, unique identifier. (We also couldn't come up with any other attributes "Storage Device" needed practically other than Device ID.)

4. The following attribution name is revised for clarification: Image → FoodImage, Count → NotificationCount, Date → GroceryListDate, Name → UserName.
5. The following attribution is added for clarification: Compartment Name on Compartment
6. Added LogicItemName to InventoryLog to make tracking more specific

4. The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:

a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.
b. Specify the primary key (PK), candidate key (CK), ***foreign keys (FK)***, and other constraints (e.g., not null, unique, etc.) that the table must maintain.
   i. FoodItem (ItemID: int, NutritionValue: String, ItemName: String, FoodItemType: String, FoodImage: Image file), ItemName not null unique, Food ItemType not null, FoodImage unique
   ii. StorageDevice (Device ID: int)
   iii. Refrigerator (Device ID: int, Serial#: int, Brand: string, Model #: int), Serial# unique
   iv. Pantry (Device ID: int, PantryName: String)
   v. Stored in (***ItemID: int***, ***Device ID: int***, ***Temperature: double***, ***CompartmentName: String***, expiry date: int (DDMMYY))
   vi. Has (***Temperature: double***, ***Device ID: int***, ***CompartmentName: String***)
   vii. User (UserID: int, UserName: String, PhoneNum: int, Email: String), UserName not null unique, PhoneNum not null unique, Email not null unique
   viii. Owns (***DeviceID: int***, ***UserID:int***)
   ix. Notifications (MessageBody: string, NotificationCount: int, Notification ID: int), MessageBody not null
   x. Receives (**UserID: int**, **Notification ID: int**)
   xi. Inventory log (LogEntryID: int, LogTimeVal: int(DDMMYY), ItemCount: int, LogItemName: String), LogTimeVal not null, ItemCount not null
   xii. Administrates (***UserID: int***, ***LogEntryID: int***)
   xiii. Obtains (***UserID: int, GroceryList#: int***)
   xiv. Grocery list (GroceryListDate: int(DDMMYY), GroceryList#: int), GroceryListDate not null
   xv. Contains (***GroceryListDate: int(DDMMYY)***, ***GroceryList#: int***, ItemID: int), GroceryListDate not null
   xvi. MealSuggestion (Meal ID: int, Recipes: string), Recipes not null
   xvii. BasedOn (MealID: int, ItemID: int)

5. Functional Dependencies (FDs)

a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key). PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A → A.

Note: In your list of FDs, there must be some kind of valid FD other those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process.

FoodItem:
- PK: ItemID
- FDs:
    - ItemID → NutritionValue, ItemName, FoodItemType, FoodImage

Stored in:
- PK: (ItemID, DeviceID)
- CK: Temperature, CompartmentName, Expiry Date
- FDs:
    - (ItemID, DeviceID) → Temperature, CompartmentName, Expiry Date

Has:
- PK: (Temperature, DeviceID)
- FDs:
    - (Temperature, DeviceID) → CompartmentName

Refrigerator:
- PK: DeviceID
- FDs:
    - DeviceID → Serial#, Brand, Model#

Pantry:
- PK: DeviceID
- FDs:
    - DeviceID → PantryName

User:
- PK: UserID
- FDs:
    - UserID → UserName, PhoneNumber, Email

Owns:
- PK: (DeviceID, UserID)
- DeviceID → UserID

Notifications:
- PK: NotificationID
- FDs:
    - NotificationID → MessageBody, NotificationCount

Receives:
- PK: (UserID, NotificationID)
- FDs:
    - NotificationID → UserID

Inventory log:
- PK: LogEntryID

- FDs:
  - LogEntryID → LogTimeVal, ItemCount, LogItemName

Administrates
- PK: (UserID, LogEntryID)
- FDs:
  - LogEntryID → UserID

Obtains
- PK: (UserID, GroceryList#)
- FDs:
  - GroceryList# → UserID

Grocery list:
- PK: GroceryList#: int
  - GroceryList# → GroceryListDate

Contains:
- PK: (GroceryListDate, ItemID)
- No additional FDs other than the PK.

Meal Suggestion:
- PK: MealID
- FDs:
  - MealID → Recipes

Based on:
- PK: MealID: int, ItemID: int
- FDs:
  - MealID → ItemID

6. Normalization

a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown. The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only the ones post normalization.

To normalize the relations to 3NF or BCNF, we need to make sure:

CPSC 304: Project Description
Milestone 2: Relational Schema, Normalization, SQL DDL, Tentative Queries

Note: <u>primary key (PK)</u>, candidate key (CK), ***foreign keys (FK)***

1. FoodItem (<u>ItemID: int</u>, NutritionValue: String, ItemName: String, FoodItemType: String, FoodImage: Image file), ItemName not null unique, Food ItemType not null, FoodImage unique
    a. ItemID += {NutritionValue, ItemName, FoodItemType, FoodImage}
    FoodItem is in BCNF

2. StorageDevice (<u>Device ID: int</u>)
    a. StorageDevice is in BCNF

3. Refrigerator (<u>Device ID: int</u>, Serial#: int, Brand: string, Model #: int)
    a. DeviceID += {Serial#, Brand, Model#}
    Refrigerator is in BCNF

4. Pantry (<u>Device ID: int</u>, PantryName: String)
    a. DeviceID += {PantryName}
    Pantry is in BCNF

5. Stored in (***<u>ItemID: int</u>***, ***<u>Device ID: int</u>***, ***Temperature: double***, ***CompartmentName: String***, expiry date: int (DDMMYY))
    a. {itemID, DeviceID} += {Temperature, CompartmentName, ExpiryDate}
    This is not in BCNF. We need to decompose it into:
    ItemStorage(ItemID, Device ID, Temperature, Expiry date)
    StorageDetail(Device ID, Temperature, CompartmentName)

6. Has (***<u>Temperature: double</u>***, ***<u>Device ID: int</u>***, ***CompartmentName: String***)
    a. This table is redundant since we have already had the same table when we were breaking down StoredIn. Thus, this table will be dropped

7. User (<u>UserID: int</u>, UserName: String, PhoneNum: int, Email: String)
    a. UserID += {UserName: String, PhoneNum: int, Email: String}
    User is in BCNF
8. Owns (***<u>DeviceID: int</u>***, ***<u>UserID:int</u>***), Receives (**<u>UserID: int</u>**, **<u>Notification ID: int</u>**), Administrates (***<u>UserID: int</u>***, ***<u>LogEntryID: int</u>***), Obtains (***<u>UserID: int, GroceryList#: int</u>***), Contains (***GroceryListDate: int(DDMMYY)***, ***<u>GroceryList#: int</u>***, <u>ItemID: int</u>), BasedOn (<u>MealID: int</u>, <u>ItemID: int</u>) all represents some relationships between two entities and they are all in the form of BCNF
9. Notifications (MessageBody: string, NotificationCount: int, <u>Notification ID: int</u>), MessageBody not null
    a. NotificationID += {MessageBody, NotificationCount}
    Notifications is in BCNF
10. Inventory log (<u>LogEntryID: int</u>, LogTimeVal: int(DDMMYY), ItemCount: int, LogItemName: String), LogTimeVal not null, ItemCount not null
    a. LogEntryID += {LogTimeVal, ItemCount, LogItemName}
    InventoryLog is in BCNF

11. Grocery list (GroceryListDate: int(DDMMYY), GroceryList#: int), GroceryListDate not null
    a. GrocerList# += GroceryListDate
       GroceryList is in BCNF
12. MealSuggestion (Meal ID: int, Recipes: string), Recipes not null
    a. MealID += Recipes
       MealSuggestion is in BCNF

7. The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc. Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.

```
CREATE DATABASE LibraryDB;
USE LibraryDB;
CREATE TABLE FoodItem (
ItemID INT,
NutritionValue VARCHAR(255),
ItemName VARCHAR(100) NOT NULL UNIQUE,
FoodItemType VARCHAR(50) NOT NULL,
FoodImage IMAGE,
PRIMARY KEY(ItemID)
)

CREATE TABLE StorageDevice (
DeviceID INT,
PRIMARY KEY(DeviceID),
)

CREATE TABLE Refrigerator (
DeviceID INT,
Serial# VARCHAR(255) UNIQUE,
Brand VARCHAR(100),
Model# INT UNIQUE,
PRIMARY KEY(DeviceID),
UNIQUE (Serial#)
FOREIGN KEY(DeviceID) REFERENCES StorageDevice
)

CREATE TABLE Pantry (
DeviceID INT,
PantryName VARCHAR(100),
PRIMARY KEY(DeviceID)
FOREIGN KEY(DeviceID) REFERENCES StorageDevice
)

CREATE TABLE ItemStorage (
ItemID INT
DeviceID INT,
Temperature FLOAT,
ExpiryDate int,
PRIMARY KEY(DeviceID),
FOREIGN KEY(DeviceID) REFERENCE StorageDevice
FOREIGN KEY(ItemID) REFERENCE FoodItem
)

CREATE TABLE StorageDetail (
DeviceID INT,
Temperature FLOAT,
CompartmentName VARCHAR(100),
PRIMARY KEY(DeviceID),
FOREIGN KEY(DeviceID) REFERENCE StorageDevice
)

CREATE TABLE Has (
DeviceID INT,
Temperature FLOAT,
CompartmentName VARCHAR(100),
PRIMARY KEY(DeviceID),
FOREIGN KEY(DeviceID) REFERENCE StorageDevice,
FOREIGN KEY(Temperature, CompartmentName) REFERENCE Compartment
)

CREATE TABLE StorageFood (
ItemID INT,
DeviceID INT,
ExpiryDate INT NOT NULL,
PRIMARY KEY(ItemID, DeviceID),
FOREIGN KEY(ItemID) REFERENCES FoodItem,
FOREIGN KEY(DeviceID) REFERENCES StorageDevice
)


CREATE TABLE User (
UserID INT,
UserName VARCHAR(100) NOT NULL UNIQUE,
PhoneNumber INT NOT NULL UNIQUE,
Email VARCHAR(100) NOT NULL UNIQUE,
PRIMARY KEY(UserID)
)

CREATE TABLE Owns (
DeviceID INT,
UserID INT,
PRIMARY KEY(UserID, DeviceID),
FOREIGN KEY(UserID)REFERENCES User,
FOREIGN KEY(DeviceID)REFERENCES StorageDevice
)

CREATE TABLE Notifications (
NotificationID INT,
MessageBody VARCHAR(10000) NOT NULL,
NotificationCount INT,
PRIMARY KEY(NotificationID)
)

CREATE TABLE Receives (
UserID INT,
NotificationID INT,
PRIMARY KEY(UserID, NotificationID),
FOREIGN KEY(UserID)REFERENCES User,
FOREIGN KEY(NotificationID)REFERENCES Notifications
)

CREATE TABLE InventoryLog (
LogEntryID INT,
LogTimeVal INT NOT NULL UNIQUE,
ItemCount INT NOT NULL,
LogItemName VARCHAR(50),
PRIMARY KEY(LogEntryID)
)

CREATE TABLE Administrates (
UserID INT,
LogEntryID INT,
PRIMARY KEY(UserID, LogEntryID),
FOREIGN KEY(UserID)REFERENCES User,
FOREIGN KEY(LogEntryID)REFERENCES InventoryLog
)

CREATE TABLE GroceryList (
GroceryListDate INT NOT NULL,
GroceryList# INT,
PRIMARY KEY(GroceryList#)
)

CREATE TABLE Obtains (
UserID INT,
GroceryList# INT,
PRIMARY KEY(UserID, GroceryList#),
FOREIGN KEY(UserID)REFERENCES User,
FOREIGN KEY(GroceryList#)REFERENCES GroceryList
)

CREATE TABLE Contains (
GroceryList# INT,
ItemID INT,
PRIMARY KEY(ItemID, GroceryList#),
FOREIGN KEY(ItemID)REFERENCES FoodItem,
FOREIGN KEY(GroceryList#)REFERENCES GroceryList
)

CREATE TABLE MealSuggestion (
MealID INT,
Recipes VARCHAR(300) NOT NULL,
PRIMARY KEY(MealID)
)

CREATE TABLE BasedOn (
MealID INT,
ItemID INT,
PRIMARY KEY(MealID, ItemID),
FOREIGN KEY(MealID)REFERENCES MealSuggestion,
FOREIGN KEY(ItemID)REFERENCES FoodItem
)
```

8. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.

```
INSERT INTO FoodItem VALUES (1, '250 Calories', 'Apple',
'Fruit', NULL);

INSERT INTO FoodItem VALUES (2, '350 Calories', 'Bread',
'Grain', NULL);

INSERT INTO FoodItem VALUES (3, '150 Calories', 'Milk', 'Dairy',
NULL);

INSERT INTO FoodItem VALUES (4, '200 Calories', 'Eggs',
'Protein', NULL);

INSERT INTO FoodItem  VALUES (5, '100 Calories', 'Orange',
'Fruit', NULL);

INSERT INTO StorageDevice VALUES (1);

INSERT INTO Refrigerator VALUES (1, 'SN001', 'Whirlpool', 10');

INSERT INTO StorageDevice VALUES (2);

INSERT INTO Refrigerator VALUES (2, 'SN002', 'Samsung', 11);

INSERT INTO StorageDevice VALUES (3);

INSERT INTO Refrigerator VALUES (3, 'SN003', 'LG', 12);

INSERT INTO StorageDevice VALUES (4);

INSERT INTO Refrigerator VALUES (4, 'SN004', 'GE', 13);

INSERT INTO StorageDevice VALUES (5);

INSERT INTO Refrigerator VALUES (5, 'SN005', 'Frigidaire', 14);

INSERT INTO StorageDevice VALUES (6);

INSERT INTO Pantry VALUES (6, 'Main Pantry');

INSERT INTO StorageDevice VALUES (7);

INSERT INTO Pantry VALUES (7, 'Secondary Pantry');

INSERT INTO StorageDevice VALUES (8);
INSERT INTO Pantry VALUES (8, 'Tertiary Pantry');

INSERT INTO StorageDevice VALUES (9);

INSERT INTO Pantry VALUES (9, 'Quaternary Pantry');

INSERT INTO StorageDevice VALUES (10);

INSERT INTO Pantry VALUES (10, 'Quinary Pantry');

INSERT INTO ItemStorage VALUES (1, 1, 5.5, 211203);

INSERT INTO ItemStorage VALUES (2, 1, 5.5, 251023);

INSERT INTO ItemStorage VALUES (3, 2, -18, 301223);

INSERT INTO ItemStorage VALUES (4, 2, -18, 150124);

INSERT INTO ItemStorage VALUES (5, 3, 20, 281023);

INSERT INTO StorageDetail VALUES (1, 5.5, 'Fridge Lower Shelf');

INSERT INTO StorageDetail VALUES (3, 20, 'Pantry Top Shelf');

INSERT INTO StorageDetail VALUES (2, -18, 'Freezer Bottom Bin');

INSERT INTO StorageDetail VALUES (2, 4, 'Fridge Middle Shelf');

INSERT INTO StorageDetail VALUES (1, -20, 'Freezer Bottom Bin');

INSERT INTO Has VALUES (1, 25.5, 'Freezer');

INSERT INTO Has VALUES (2, 20.0, 'Refrigerator');

INSERT INTO Has VALUES (3, 15.3, 'Pantry');

INSERT INTO Has VALUES (4, 30.2, 'Freezer');

INSERT INTO Has VALUES (5, 18.8, 'Refrigerator');

INSERT INTO StorageFood VALUES (1, 1, '20241025');

INSERT INTO StorageFood VALUES (2, 2, '20231105');

INSERT INTO StorageFood VALUES (3, 3, '20231020');
INSERT INTO StorageFood VALUES (4, 4, '20231030');

INSERT INTO StorageFood VALUES (5, 5, '20241018');

INSERT INTO User VALUES (1, 'John Doe', 1234567890,
'john.doe@email.com');

INSERT INTO User VALUES (2, 'Jane Smith', 0987654321,
'jane.smith@email.com');

INSERT INTO User VALUES (3, 'Alice Johnson', 1112233445,
'alice.johnson@email.com');

INSERT INTO User VALUES (4, 'Bob Brown', 2223344556,
'bob.brown@email.com');

INSERT INTO User VALUES (5, 'Charlie White', 3334455667,
'charlie.white@email.com');

INSERT INTO Owns VALUES (1, 1);

INSERT INTO Owns VALUES (2, 2);

INSERT INTO Owns VALUES (3, 3);

INSERT INTO Owns VALUES (4, 4);

INSERT INTO Owns VALUES (5, 5);

INSERT INTO Notifications VALUES (1, 'Your milk is expiring
tomorrow!', 1);

INSERT INTO Notifications VALUES (2, 'You have 5 items expiring
this week!', 2);

INSERT INTO Notifications VALUES (3, 'Your eggs are expired!',
1);

INSERT INTO Notifications VALUES (4, 'Grocery list has been
updated.', 1);

INSERT INTO Notifications VALUES (5, 'A new meal suggestion is
available.', 1);

INSERT INTO Receives VALUES (1, 1);
INSERT INTO Receives VALUES (1, 2);

INSERT INTO Receives VALUES (1, 3);

INSERT INTO Receives VALUES (1, 4);

INSERT INTO Receives VALUES (1, 5);

INSERT INTO InventoryLog VALUES (1, 20231001, 5, ‘Apple’);

INSERT INTO InventoryLog VALUES (2, 20231002, 4, ‘Bread’);

INSERT INTO InventoryLog VALUES (3, 20231003, 6, ‘Milk’);

INSERT INTO InventoryLog VALUES (4, 20231004, 5, ‘Eggs’);

INSERT INTO InventoryLog VALUES (5, 20231005, 7, ‘Orange’);

INSERT INTO Administrates VALUES (1, 1);

INSERT INTO Administrates VALUES (2, 2);

INSERT INTO Administrates VALUES (3, 3);

INSERT INTO Administrates VALUES (4, 4);

INSERT INTO Administrates VALUES (5, 5);

INSERT INTO GroceryList VALUES (20231001, 1);

INSERT INTO GroceryList VALUES (20231002, 2);

INSERT INTO GroceryList VALUES (20231003, 3);

INSERT INTO GroceryList VALUES (20231004, 4);

INSERT INTO GroceryList VALUES (20231005, 5);

INSERT INTO Obtains VALUES (1, 1);

INSERT INTO Obtains VALUES (2, 2);

INSERT INTO Obtains VALUES (3, 3);

INSERT INTO Obtains VALUES (4, 4);
INSERT INTO Obtains VALUES (5, 5);

INSERT INTO Contains VALUES (1, 1);

INSERT INTO Contains VALUES (2, 2);

INSERT INTO Contains VALUES (3, 3);

INSERT INTO Contains VALUES (4, 4);

INSERT INTO Contains VALUES (5, 5);

INSERT INTO MealSuggestion VALUES (1, 'Apple Pie');

INSERT INTO MealSuggestion VALUES (2, 'Bread Sandwich');

INSERT INTO MealSuggestion VALUES (3, 'Milkshake');

INSERT INTO MealSuggestion VALUES (4, 'Fried Eggs');

INSERT INTO MealSuggestion VALUES (5, 'Orange Juice');

INSERT INTO BasedOn VALUES (1, 1);

INSERT INTO BasedOn VALUES (2, 2);

INSERT INTO BasedOn VALUES (3, 3);

INSERT INTO BasedOn VALUES (4, 4);

INSERT INTO BasedOn VALUES (5, 5);
```