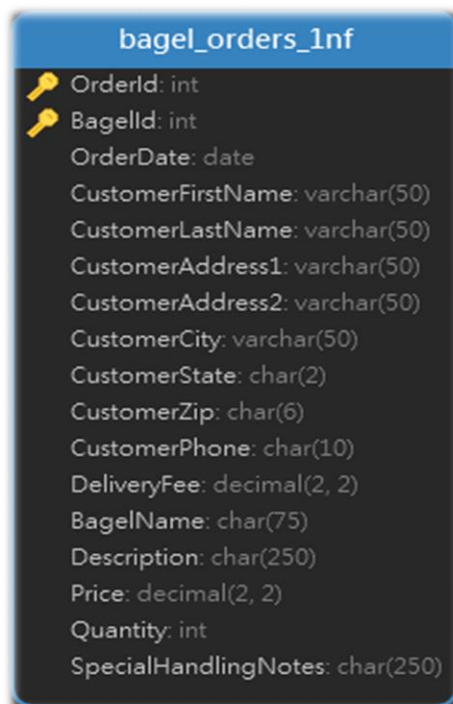


Data Management Project A

A. Construct a normalized physical database model to represent the ordering process for Nora's Bagel Bin by doing the following:

1. Complete the second normal form (2NF) section of the attached "Nora's Bagel Bin Database Blueprints" document by doing the following:
 - a. Assign each attribute from the 1NF table into the correct 2NF table.
 - b. Describe the relationship between the **two** pairs of 2NF tables by indicating their cardinality in each of the dotted cells: one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), or many-to-many (M:M).
 - c. Explain how you assigned attributes to the 2NF tables and determined the cardinality of the relationships between your 2NF tables.

First Normal Form (1NF)



bagel_orders_1nf	
OrderId: int	
BagelId: int	
OrderDate: date	
CustomerFirstName: varchar(50)	
CustomerLastName: varchar(50)	
CustomerAddress1: varchar(50)	
CustomerAddress2: varchar(50)	
CustomerCity: varchar(50)	
CustomerState: char(2)	
CustomerZip: char(6)	
CustomerPhone: char(10)	
DeliveryFee: decimal(2, 2)	
BagelName: char(75)	
Description: char(250)	
Price: decimal(2, 2)	
Quantity: int	
SpecialHandlingNotes: char(250)	

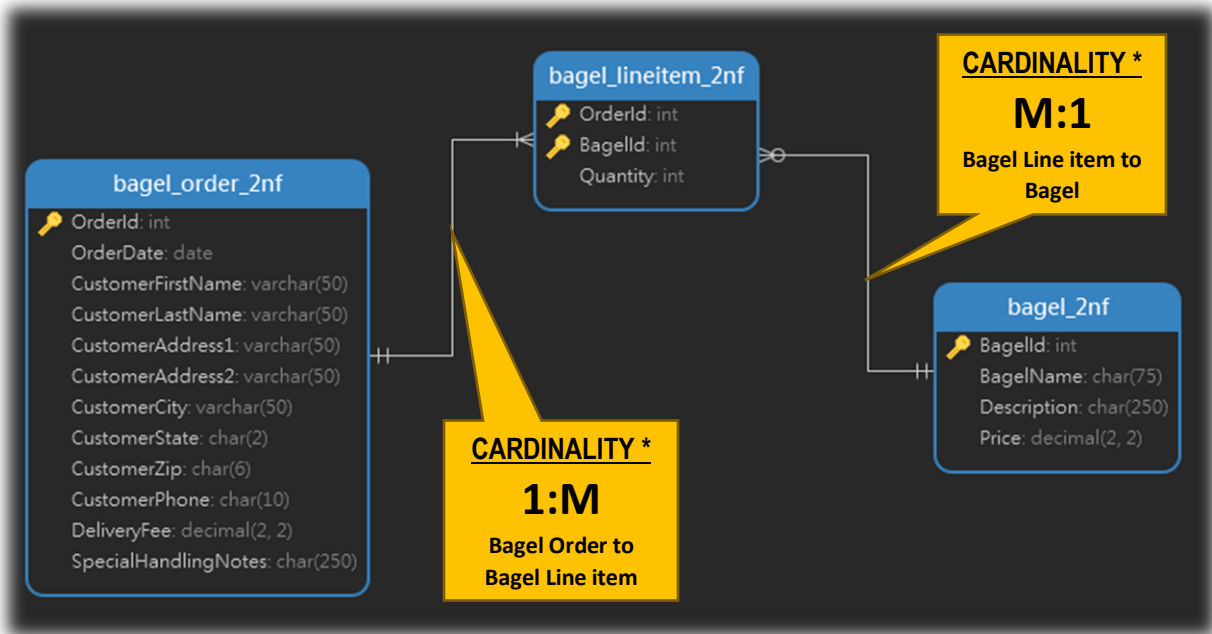
MySQL Code

```
CREATE TABLE Bagel_Orders_INF
(
  OrderId INT NOT NULL AUTO_INCREMENT,
  BagelId INT NOT NULL,
  OrderDate DATE NOT NULL,
  CustomerFirstName NVARCHAR(50) NULL,
  CustomerLastName NVARCHAR(50) NULL,
  CustomerAddress1 NVARCHAR(50) NULL,
  CustomerAddress2 NVARCHAR(50) NULL,
  CustomerCity NVARCHAR(50) NULL,
  CustomerState CHAR(2) NULL,
  CustomerZip CHAR(6) NULL,
  CustomerPhone CHAR(10) NULL,
  DeliveryFee DECIMAL(2,2) NULL,
  BagelName CHAR(75) NOT NULL,
  Description CHAR(250) NULL,
  Price DECIMAL(2,2) NULL,
  Quantity INT NOT NULL,
  SpecialHandlingNotes CHAR(250) NULL,
  CONSTRAINT PK_Bagel_Orders_INF PRIMARY KEY (OrderId, BagelId)
);
```

To start I was able to form a table with every single bit of data I could find, ranging from name, address, phone number to order id and bagel price. From this I was able to construct a basic normalized table (**1NF**) and made sure each column name was self-

describing and obvious as to what it was for, so value designation would be obvious too. I was also able to determine the data type of each column easily. The primary key construction was also an easy choice. id data usually make for good primary keys but to ensure total uniqueness I created a composite primary key from the **OrderId** and **BagelId** data.

Second Normal Form (2NF)



- ***CARDINALITY** - relationship between those two tables: one-to-one (**1:1**), one-to-many (**1:M**), many-to-one (**M:1**), or many-to-many (**M:M**), read from left to right.

MySQL code for the corresponding tables

```

CREATE TABLE Bagel_Order_2NF
(
  OrderId INT NOT NULL AUTO_INCREMENT,
  OrderDate DATE NOT NULL,
  CustomerFirstName NVARCHAR(50) NULL,
  CustomerLastName NVARCHAR(50) NULL,
  CustomerAddress1 NVARCHAR(50) NULL,
  CustomerAddress2 NVARCHAR(50) NULL,
  CustomerCity NVARCHAR(50) NULL,
  CustomerState CHAR(2) NULL,
  CustomerZip CHAR(6) NULL,
  CustomerPhone CHAR(10) NULL,
  DeliveryFee DECIMAL(2,2) NULL,
  SpecialHandlingNotes CHAR(250) NULL,
  CONSTRAINT PK_Bagel_Order_2NF PRIMARY
  KEY (OrderId)
);

```

```

CREATE TABLE Bagel_LineItem_2NF
(
  OrderId INT NOT NULL AUTO_INCREMENT,
  BagelId INT NOT NULL,
  Quantity INT NOT NULL,
  CONSTRAINT FK_Bagel_LineItem_2NF_Order
  FOREIGN KEY (OrderId) REFERENCES
  Bagel_Order_2NF(OrderId),
  CONSTRAINT FK_Bagel_LineItem_2NF_Bagel
  FOREIGN KEY (BagelId) REFERENCES
  Bagel_2NF(BagelId),
  CONSTRAINT PK_Bagel_LineItem_2NF PRIMARY
  KEY (OrderId, BagelId)
);

```

```

CREATE TABLE Bagel_2NF
(
  BagelId INT NOT NULL,
  BagelName CHAR(75) NOT NULL,
  Description CHAR(250) NULL,
  Price DECIMAL(2,2) NULL,
  CONSTRAINT PK_Bagel_2NF
  PRIMARY KEY (BagelId)
);

```

To convert the original table from first order form (**1NF**) to second order form (**2NF**), the table will have to be separated into 3 parts; Bagel, Bagel Order, Bagel Line item, from which each will be their own individual table. The Bagel data will store information on the various bagel types and prices. The Order data will store information on each individual transaction. Finally, Line item will store specific line-item information within the order. When a line item is created the resulting composite primary key will be made up of two foreign keys configured from the bagelid and orderid data, to ensure total uniqueness.

The **Cardinality** represents the relationship between the pairs of tables, and in this case read left to right between the bagel orders and bagel line item, cardinality would be 1 to many (1:M) representing the fact that an order can have a minimum of one or maximum of many line items. For the Bagel line item and Bagel tables, the cardinality would be many to 1 (M:1), a line item can have at most one bagel, each bagel has many bagel line items.

The attributes were assigned based on information from the Bagel order form and what category the data fell into; order transaction information, order line-item information and product information, with line-item information as the central point connecting transaction details to the product. It was from this general premise that the basis of the cardinality relationships was built. They were also selected based on partial dependency of the orderid for the order information and bagelid for the bagel information.

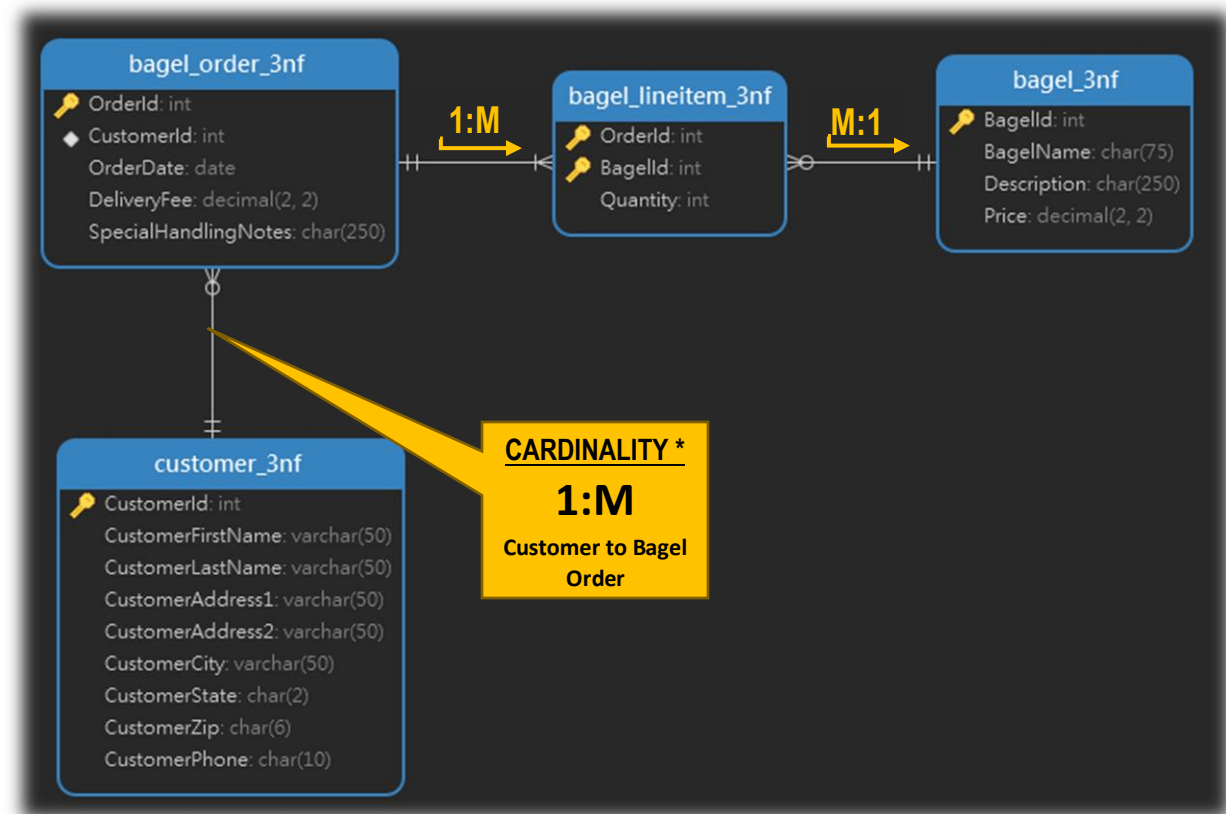
2. Complete the third normal form (3NF) section of the attached "Nora's Bagel Bin Database Blueprints" document by doing the following:
 - a. Assign each attribute from your 2NF "Bagel Order" table into one of the new 3NF tables.
Copy all other information from your 2NF diagram into the 3NF diagram.
 - b. Provide each 3NF table with a name that reflects its contents.
 - c. Create a new field that will be used as a key linking the **two** 3NF tables you named in part A2b. Ensure that your primary key (PK) and foreign key (FK) fields are in the correct locations in the 3NF diagram.
 - d. Describe the relationships between the 3NF tables by indicating their cardinality in each of the dotted cells: one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), or many-to-many (M:M).
 - e. Explain how you assigned attributes to the 3NF tables and determined the cardinality of the relationships between your 3NF tables.

MySQL code for 3nf Customer table and Bagel order table

```
CREATE TABLE Customer_3NF
(
  CustomerId INT NOT NULL,
  CustomerFirstName NVARCHAR(50) NULL,
  CustomerLastName NVARCHAR(50) NULL,
  CustomerAddress1 NVARCHAR(50) NULL,
  CustomerAddress2 NVARCHAR(50) NULL,
  CustomerCity NVARCHAR(50) NULL,
  CustomerState CHAR(2) NULL,
  CustomerZip CHAR(6) NULL,
  CustomerPhone CHAR(10) NULL,
  CONSTRAINT PK_Customer_3NF PRIMARY
  KEY (CustomerId)
);
```

```
CREATE TABLE Bagel_Order_3NF
(
  OrderId INT NOT NULL AUTO_INCREMENT,
  CustomerId INT NOT NULL,
  OrderDate DATE NOT NULL,
  DeliveryFee DECIMAL(2,2) NULL,
  SpecialHandlingNotes CHAR(250) NULL,
  CONSTRAINT FK_Order_Customer_3NF
  FOREIGN KEY (CustomerId) REFERENCES
  Customer_3NF(CustomerId),
  CONSTRAINT PK_Bagel_Order_3NF
  PRIMARY KEY (OrderId)
);
```

Third Normal Form (3NF)

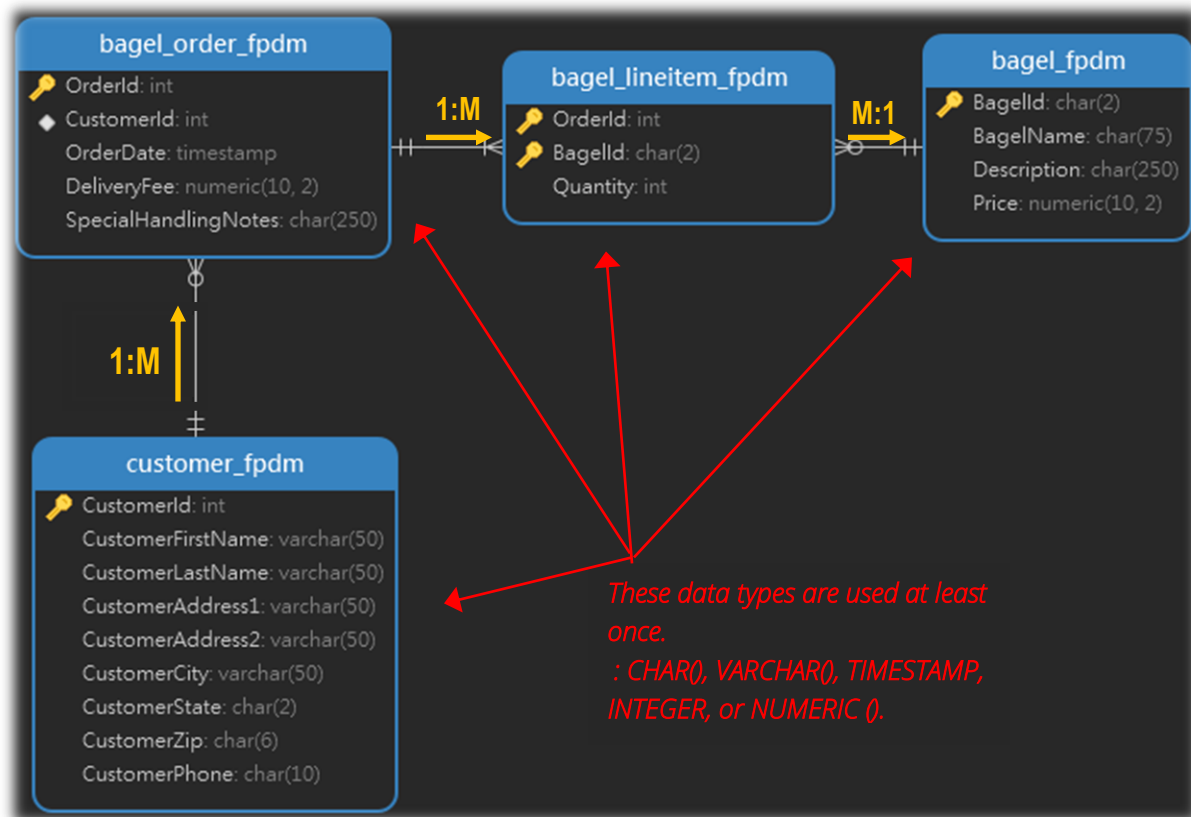


- ***CARDINALITY** - relationship between those two tables: one-to-one (**1:1**), one-to-many (**1:M**), many-to-one (**M:1**), or many-to-many (**M:M**), zero-to many (**0:M**), read from left to right.

To setup the Third normal form (**3NF**) the tables will need to be further broken down. The Bagel order table can be further separated to form another table so we can move the repeated data to a new table and name it Customer and create a primary key for it and call it **CustomerId**. This new key will be used as a Foreign key in the Bagel order table. This can be seen in the above diagram. The original Cardinalities between the other tables remain unchanged. For the Customer and Bagel order table, it makes sense that it would be a one to many relationship as a customer can have at most many orders, and an order belongs to at most one customer.

3. Complete the "Final Physical Database Model" section of the attached "Nora's Bagel Bin Database Blueprints" document by doing the following:
 - a. Copy the table names and cardinality information from your 3NF diagram into the "Final Physical Database Model" and rename the attributes.
 - b. Assign **one** of the following **five** data types to each attribute in your 3NF tables: CHAR(), VARCHAR(), TIMESTAMP, INTEGER, or NUMERIC(). Each data type must be used at least once.

Final Physical Database Model



- ***CARDINALITY** - relationship between those two tables: one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), or many-to-many (M:M), zero-to many (0:M), read from left to right.

For the Final Physical Database Model, the table names and cardinality information have been maintained. There are no spaces in the attribute name to avoid formatting issues with other tools and also an increased chance of errors in the SQL code where quotation marks are required. All five datatypes : CHAR(), VARCHAR(), TIMESTAMP, INTEGER, or NUMERIC() have been used at least once in the tables.

Okunta Braide
Student ID: #002450037
C170: Data Management – Applications