MSIS 2503 Project - Boba Tea Shop

Gaurvi Lal & Steven Wang

Business Description:

Our business is a Boba Shop. These shops are specialty tea shops that sell flavored teas. This particular store is located close to a large university. People order the items over the counter and the servers will call out orders when ready. Our store contains about 15 drink options and 4 snack options. Some of the drinks include: Milk Tea, Strawberry Tea, Thai Tea, Lemon Tea, Passion Fruit Tea, and many more. Customers have options to customize their drinks to their content. They have the option for choosing drink size, ice level, sugar amount, toppings, etc. The most popular drink is “Milk Tea”. As mentioned, in addition to selling teas, our shop sells a variety of small snacks such as fries, calamari, chicken bites and tofu. The shop has plenty of chairs and tables where people can hangout.

What data will be collected?

-Types of drink ordered. Help us see which drinks are popular and which are unpopular
-Orders per customer
-How many people order snacks compared to drinks
-How long people spend in the shop
-How much people tip
-Demographics of customers (Most were university students)
-Customer satisfaction rating (1 to 5 stars, or no response)

Questions we may want to find from data?

-Is there a time where there tends to be more customers? We could use this data to increase staffing at those times. This allows people to have a better experience, potentially increasing profits.
-What drinks are popular? What are unpopular? We can remove unpopular drinks from the menu, due to costs of buying those supplies/ingredients. For popular drinks, we can have promotions to further increase revenue.
-Is there a relationship/trend between how much customers tip depending on ratings/demographics/order type/time spent, etc. Is there a way we can improve customer experience to increase tips? How about customer ratings against the same criteria? This will also be useful for our store image as higher ratings could attract more customers.

Project Part 2

CREATE SCHEMA `Normalization`;
USE `Normalization`;

For the first part, we created two tables: One for Orders (Master Data) and one for Servers. We will normalize these two tables and popularize the data into those tables.

#1st Denormalized Table

#1st Denormalized Table

```
CREATE TABLE MasterDenorm
(
CustomerName VARCHAR(20) NOT NULL,
Gender VARCHAR(20),
Age INT,
OrderID INT NOT NULL PRIMARY KEY,
ServerName VARCHAR(20),
Orders VARCHAR(50),
OrderCategory VARCHAR(20),
Size VARCHAR(10) NOT NULL,
IceLevel VARCHAR(20),
Toppings VARCHAR(20),
SugarAmount VARCHAR(20),
Price FLOAT,
ReviewRating INT,
TimeSpent INT,
Tip FLOAT,
CustomerExperienceRating INT
);
```

#Normalized Tables

#Master Table 1NF

```
CREATE TABLE MasterNorm
(
CustomerName VARCHAR(20) NOT NULL,
Gender VARCHAR(20),
Age INT,
OrderID INT NOT NULL PRIMARY KEY,
ServerName VARCHAR(20),
ProductID INT,
ProductName VARCHAR(50),
ProductQuantity INT,
OrderCategory VARCHAR(20),
Size VARCHAR(10) NOT NULL,
IceLevel VARCHAR(20),
Toppings VARCHAR(20),
SugarAmount VARCHAR(20),
Price FLOAT,
```

```
ReviewRating INT,
TimeSpent INT,
Tip FLOAT,
CustomerExperienceRating INT
);

#Order Table 2NF
CREATE TABLE Orders
(
OrderID INT NOT NULL PRIMARY KEY,
CustomerName VARCHAR(20)
);

#Product Table 2NF
CREATE TABLE Product
(
ProductID INT NOT NULL PRIMARY KEY,
ProductName VARCHAR(50)
);

#OrderDetails Table 2NF
CREATE TABLE OrderDetails
(
OrderID INT NOT NULL PRIMARY KEY,
ProductID INT NOT NULL,
ProductQuantity INT
);

#Import Data from CSV Files

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Master.csv'
INTO TABLE MasterDenorm
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/MasterNorm.csv'
INTO TABLE MasterNorm
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Orders.csv'
INTO TABLE Orders
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Product.csv'
INTO TABLE Product
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/OrderDetails.csv'
INTO TABLE OrderDetails
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

#Populate Data
SELECT * FROM MasterDenorm;
Select * FROM MasterNorm;
Select * FROM Orders;
Select * FROM Product;
Select * FROM OrderDetails;

#2nd Denormalized Table

#2nd Denormalized Table
CREATE TABLE ServerDenormalized
(
ServerID VARCHAR(20) NOT NULL,
ServerName VARCHAR(20),
PayRate BIGINT NOT NULL,
ShiftInformation VARCHAR(50) NOT NULL,
OrdersTaken BIGINT NOT NULL,
Rating INT,
CONSTRAINT PKServerID PRIMARY KEY (ServerID)
);

#Normalized for Servers Table 2
CREATE TABLE Server1NF
(
ServerID VARCHAR(20) NOT NULL,
ServerName VARCHAR(20) NOT NULL,
PayRate BIGINT NOT NULL,
ShiftID BIGINT NOT NULL,
ShiftName VARCHAR(50) NOT NULL,
Hours BIGINT NOT NULL,
OrdersTaken BIGINT NOT NULL,
Rating INT
);
```

#Normalized for Servers Table 2NF

```
CREATE TABLE Server2NF1
(
ServerID VARCHAR(20) NOT NULL,
ServerName VARCHAR(20) NOT NULL,
CONSTRAINT PKServerID PRIMARY KEY (ServerID)
);
```

#Normalized for Servers Table 2NF #2

```
CREATE TABLE Server2NF2
(
ShiftID VARCHAR(20) NOT NULL,
ShiftName VARCHAR(20) NOT NULL,
CONSTRAINT PKShiftID PRIMARY KEY (ShiftID)
);
```

#Normalized for Servers Table 2NF #2

```
#(Server ID, Shift ID) --> Hours
CREATE TABLE Server2NF3
(
ServerID VARCHAR(20) NOT NULL,
ShiftID VARCHAR(20) NOT NULL,
Hours BIGINT NOT NULL
);
```

#Normalized for Servers Table 2NF #3
#(Server ID, Shift ID) --> Number of Orders Taken

```
CREATE TABLE Server2NF4
(
ServerID VARCHAR(20) NOT NULL,
ShiftID VARCHAR(20) NOT NULL,
OrdersTaken BIGINT NOT NULL
);
```

#Import Data from CSV Files

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Denormalized 2.csv'
INTO TABLE ServerDenormalized
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Normalized 2.1.csv'
INTO TABLE Server1NF
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Normalized 2.2.csv'
INTO TABLE Server2NF1
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Normalized 2.3.csv'
INTO TABLE Server2NF2
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Normalized 2.4.csv'
INTO TABLE Server2NF3
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Normalized 2.5.csv'
INTO TABLE Server2NF4
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

#Populate Data
SELECT * FROM ServerDenormalized;
Select * FROM Server1NF;
Select * FROM Server2NF1;
Select * FROM Server2NF2;
Select * FROM Server2NF3;
Select * FROM Server2NF4;
```

Project Part 3

For This part, we will create a new Schema to avoid errors with Normalization tables. This table will be used for Part 3 and 4

CREATE SCHEMA `Project - Boba`;
USE `Project - Boba`;

```
CREATE TABLE Orders
(
CustomerName VARCHAR(20) NOT NULL,
Gender VARCHAR(20),
Age INT,
OrderID INT NOT NULL PRIMARY KEY,
ServerName VARCHAR(20),
ProductID INT,
ProductName VarChar(20),
ProductQuantity INT,
Size VARCHAR(10) NOT NULL,
IceLevel VARCHAR(20),
Toppings VARCHAR(20),
SugarAmount VARCHAR(20),
Price FLOAT,
ReviewRating INT,
TimeSpent INT,
Tip FLOAT,
CustomerExperienceRating INT
);

CREATE TABLE OrderDescr
(
ProductID INT,
ProductName VARCHAR(50),
OrderCategory VARCHAR(20),
DefaultPrice INT,
SugarDefault VARCHAR(10)
);

CREATE TABLE Servers
(
ServerID VARCHAR(20) NOT NULL PRIMARY KEY,
OrderID INT,
ServerName VARCHAR(20) NOT NULL,
PayRate BIGINT NOT NULL,
ShiftID BIGINT NOT NULL,
ShiftName VARCHAR(50) NOT NULL,
Hours BIGINT,
OrdersTaken BIGINT,
Rating INT
);
```

#Views

#Our First view is a list of all the products we sell to see how many items are priced low, medium or high. If there is too many or few items in one category, we might have to adjust prices for certain items.

```
CREATE VIEW PriceCategory AS
SELECT ProductID, ProductName, DefaultPrice,
CASE
        WHEN DefaultPrice > 7 THEN 'High'
        WHEN DefaultPrice > 3 THEN 'Medium'
        WHEN DefaultPrice > 0 THEN 'Low'
        ELSE 'N/A'
END AS PriceRange
FROM OrderDescr;
SELECT * FROM PriceCategory;
```

#View displays what type of hours employees worked for that day. Here we can see if they worked a lot or few hours by classifying hours as Double OT, OT, regular or reduced shift
#This data can be used to redistribute hours to the employees if necessary. We can also use this as a reference when employees request hours change.

```
CREATE VIEW EmployeeWorked AS
SELECT ServerID, ServerName, PayRate, Hours,
CASE
        WHEN Hours > 10 THEN 'Double OT'
        WHEN Hours > 6 THEN 'OT'
        WHEN 4 > 0 THEN 'Regular'
        ELSE 'Reduced Shift'
END AS Worktype
FROM Servers;

SELECT * FROM EmployeeWorked;
```

#SELECT Queries

 # Most popular drink/snack and how many of those items were ordered.

```
SELECT ProductID, ProductName, COUNT(ProductID) FROM Orders ORDER BY
COUNT(ProductID) DESC;
```

 # The Order Category that is most appreciated as seen by the rating and amount of tips received

```
SELECT ProductID, Tip, ReviewRating FROM Orders ORDER BY ReviewRating DESC;
```

 # Most expensive order categories, and average price.

```
SELECT OrderCategory, AVG(DefaultPrice) AS AveragePrice FROM OrderDescr
GROUP BY OrderCategory
ORDER BY AveragePrice DESC;
```

 # Find out what age spend the most time in the store.

```
SELECT Age, AVG(TimeSpent) AS AverageTimeSpent FROM Orders WHERE Age IN(18, 19,
20, 21, 22, 23, 24)
GROUP BY Age ORDER BY AverageTimeSpent DESC;
```

#Server who had the highest ratings.

```
SELECT ServerID, ServerName, Rating FROM Servers ORDER BY Rating LIMIT 5;
```

Project Part 4

For Part/Week 4, We are going to use the same table as week 3 to Create Joins and Subqueries as well as Triggers and Procedures. We will create 1 more table and insert items into OrderDescr.

```
DROP TRIGGER IF EXISTS ProductInsertTrigger;
DROP PROCEDURE IF EXISTS FindCommissionDiff;

INSERT INTO OrderDescr
VALUES
 (100,'Milk Boba Tea','Drinks',10,'Brown Sugar'),
 (101,'Strawberry Boba Tea','Drinks',7,'Light Sugar'),
 (102,'Honey Boba Tea','Drinks',8,'No Sugar'),
 (103,'Tofu Bites','Snacks',4,'NA'),
 (104,'Calamari','Snacks',4,'NA'),
 (105,'Thai Tea','Drinks',6,'Brown Sugar');

CREATE TABLE OrderFrequency
(
Product                 CHAR(20),
NumOrders     INT
);
```

#Joins and Subqueries

#Find Server, Payrate, and ShiftID of Server who took the most orders, sorted by Total orders.
#This will let us see which servers do most/least amount of work. Also let's us see how they are compensated.

```
SELECT Servers.ServerID, Servers.PayRate, Servers.ShiftID,
COUNT(Orders.OrderID) AS OrdersTotal
FROM Servers
INNER JOIN Orders
ON Servers.ServerName = Orders.ServerName
GROUP BY  Servers.ServerName
Order BY OrdersTotal DESC;
```

#Find what shifts are the most pouplar among college students and see orders and ratings

```
SELECT Servers.ShiftName, SUM(Servers.OrdersTaken) AS TotOrders, AVG(Servers.Rating)
AS AvgRate
FROM Servers WHERE
Servers.ServerName IN
        (SELECT Orders.ServerName FROM Orders
   WHERE Orders.Age = 18 OR Orders.Age = 19 OR Orders.Age = 20
        OR Orders.Age = 21 OR Orders.Age = 22)
ORDER BY AvgRate;
```

#There are usually more women who visits Boba Shops than men. Are there specific drinks or food they prefer.
#Subquery shows which drinks or food women might prefer.

```
SELECT COUNT(OrderDescr.ProductID) AS ItemCount, OrderDescr.ProductName,
OrderDescr.OrderCategory, OrderDescr.DefaultPrice
FROM OrderDescr WHERE
OrderDescr.ProductID IN
        (SELECT Orders.ProductID FROM Orders
   WHERE Gender = 'Female')
GROUP BY OrderDescr.ProductName;
```

#Find top 10 drinks and the total amount paid and tips total
#We can have special promotions for these drinks or adjust pricings.

```
SELECT Orders.ProductName, SUM(Orders.Price) AS TotalPaid, SUM(Orders.Tip) AS
TotalTips
FROM Orders WHERE
Orders.ProductID IN
        (SELECT OrderDescr.ProductID FROM OrderDescr
        WHERE OrderDescr.OrderCategory = 'Drink')
GROUP BY Orders.ProductName
ORDER BY TotalPaid DESC LIMIT 10;
```

#List Employees amount of Revenue (Excluding Tips) collected for employees who worked 5 or more hours.
#Revenue must exceed $500 total

```
SELECT Servers.ServerID, Servers.ServerName, Servers.OrdersTaken, SUM(Orders.Price) AS
Revenue
FROM Servers
INNER JOIN Orders
ON Servers.ServerName = Orders.ServerName
WHERE Servers.Hours >= 5
GROUP BY Servers.ServerName
HAVING Revenue >= 500
ORDER BY Revenue;
```

#1 A query to initialize the OrderFrequency table by inserting one row for each ProductID already in the OrderDescr table, and having the
 # accurate count of the number of orders for that product.

```
INSERT INTO OrderFrequency
SELECT OrderDescr.ProductName, COUNT(OrderDescr.ProductName) AS "#Orders" FROM
OrderDescr GROUP BY OrderDescr.ProductName ORDER BY "#Orders";

SELECT * FROM OrderFrequency;
```

#2 Write a trigger to update NumEmployeesByLastName whenever a new record is inserted into the Employee table.

```
DELIMITER //
CREATE TRIGGER ProductInsertTrigger AFTER INSERT ON OrderDescr
FOR EACH ROW
BEGIN
IF
((SELECT COUNT(*) FROM OrderFrequency WHERE Product = NEW.ProductName) = 0)
THEN
INSERT INTO OrderFrequency SELECT ProductName, COUNT(ProductName) FROM
OrderDescr
WHERE ProductName = NEW.ProductName GROUP BY ProductName;
ELSE
UPDATE OrderFrequency SET NumOrders = (SELECT COUNT(ProductName) FROM
OrderDescr
WHERE ProductName = NEW.ProductName GROUP BY ProductName)
WHERE Product=New.ProductName;
END IF;
END //
DELIMITER ;

SET SQL_SAFE_UPDATES = 0;
INSERT INTO OrderDescr VALUES (102,'Honey Boba Tea','Drinks',8,'No Sugar');
INSERT INTO OrderDescr VALUES (201, 'Fries', 'Snacks',4,'NA');
SELECT * FROM OrderFrequency;
```

#3 Create a procedure CompareCustomerRating that takes 2 products as input. The procedure returns the difference in the customer rating
# of these two products

```
DELIMITER $$
CREATE PROCEDURE CompareCustomerRating (IN ProdID1 INT, IN ProdID2 INT, OUT
CustRatingDiff INT )
BEGIN
   DECLARE CustRating1 INT;
   DECLARE CustRating2 INT;
   SELECT CustomerExperienceRating INTO CustRating1 FROM Orders where
Orders.ProductID = ProdID1;
   SELECT CustomerExperienceRating INTO CustRating2 FROM Orders where
Orders.ProductID = ProdID2;
        SET CustRatingDiff = CustRating1 - CustRating2;
END $$

DELIMITER ;

CALL CompareCustomerRating(100, 101, @df);
SELECT @df;
```