GSB 520 - Database Design Project

Nathan Diekema, Jared McMullen, Thomas Roske, Arash Akhavi

Part 1

Scope

The domain chosen for this project is the modeling of a restaurant. This includes the interactions between employees, customers, orders, order details, and reservations. The goal of this project is to accurately model the entities and relations of a generic restaurant, then create a database with relevant information to the model.

Business Rules

- A CUSTOMER can have zero to many ORDERS
- A CUSTOMER can have zero to many RESERVATIONS
- A CUSTOMER is limited to one RESERVATION a day
- An EMPLOYEE can be attached to zero to many ORDERS
- An EMPLOYEE can only be MANAGED by one person
- AN EMPLOYEE can only belong to one permanent ADDRESS
- An ORDER is associated with one and only one CUSTOMER
- An Order can have one to many MENU_ITEMS
- A MENU_ITEM can exist in zero to many ORDERS
- A MENU_ITEM must be of the item type FOOD or DRINK
- A CUSTOMER can only order an item if there is INVENTORY
- A CUSTOMER can only order as many of an item as there is INVENTORY
- RESERVATIONS cannot be canceled
- There are no returns or refunds

Entities

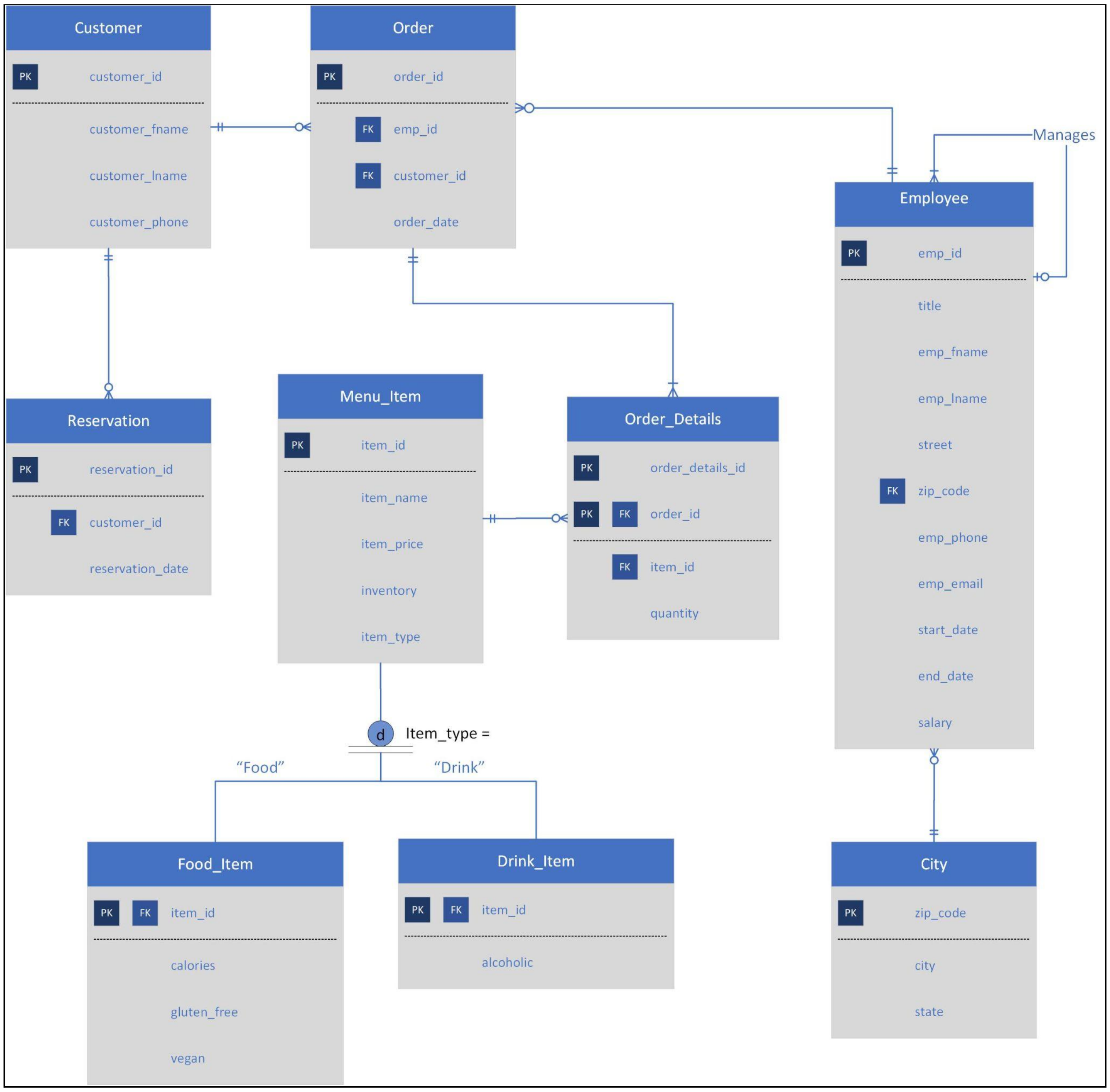
- Employees
    - <u>Emp_id</u>
    - fname
    - lname
    - title
    - street
    - city
    - zip_code
    - emp_phone

  - gender
  - salary
  - start_date
  - end_date (NULL if still employed)
- City
  - zip_code
  - city
  - state
- Customers
  - customer_id
  - customer_fname
  - customer_lname
  - customer_phone
- Orders
  - order_id
  - emp_id
  - customer_id
  - order_date
- Order Details
  - order_details_id
  - order_id
  - item_id
  - quantity
- Reservations
  - reservation_id
  - customer_id
  - reservation_date
- Menu_Item
  - item_id
  - item_name
  - item_price
  - inventory
  - item_type
- Food_Item
  - Item_id
  - Calories
  - Gluten_free
  - Vegan
- Drink_Item
  - Item_id
  - alcoholic

Definitions

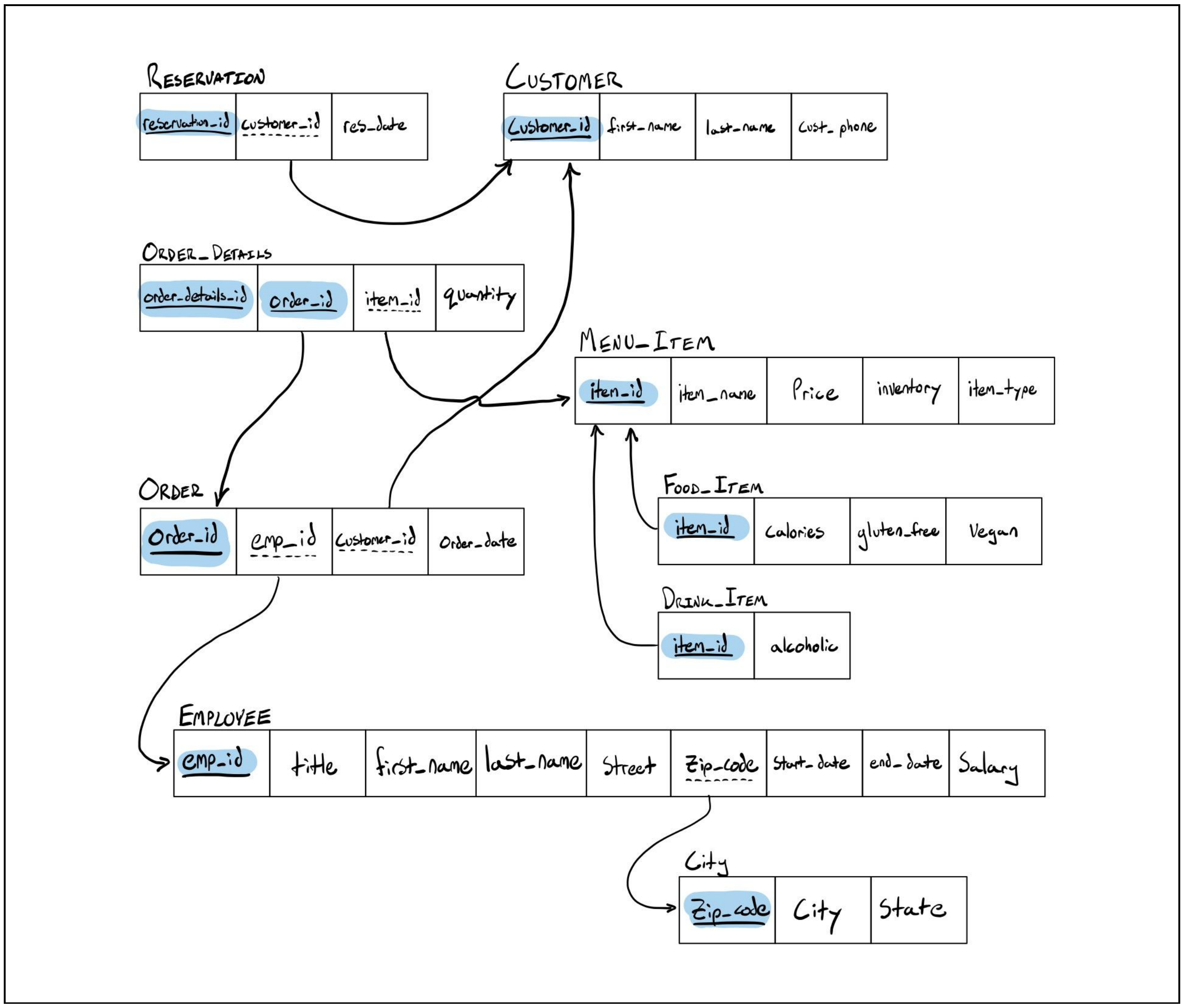
- An **Employee** is someone who is on payroll for the restaurant. These are individuals who are currently working, or have previously worked, for the restaurant.
- A **Customer** is an individual who interacts with the restaurant in the form of placing an order.
- **Zip** is a table containing address information for Employees per their associated postal codes. It keeps track of the city and state associated with each postal code.
- **Orders** are taken by Employees and placed by Customers. They contain unique Order IDs which are used to find the corresponding details of the order placed.
- **Order Details** are the items ordered and quantity ordered by the Customer.
- **Reservations** are made by Customers wishing to dine-in on a specific date at a specific time.
- **Menu Items** are the items available for Order by Customers at the restaurant. This entity keeps track of price, inventory, and item_type. The inventory variable keeps track of how many items the restaurant has in stock.
- **Food Items** are a subtype of menu items and keep track of the food items and associated food information.
- **Drink Items** are a subtype of menu items and keep track of the drink items and whether they are alcoholic or not.

EER Diagram


Customer
PK customer_id
customer_fname
customer_lname
customer_phone
Order
PK order_id
FK emp_id
FK customer_id
order_date
Employee
PK emp_id
title
emp_fname
emp_lname
street
FK zip_code
emp_phone
emp_email
start_date
end_date
salary
Manages
Reservation
PK reservation_id
FK customer_id
reservation_date
Menu_Item
PK item_id
item_name
item_price
inventory
item_type
Order_Details
PK order_details_id
PK FK order_id
FK item_id
quantity
d
Item_type =
"Food"
"Drink"
Food_Item
PK FK item_id
calories
gluten_free
vegan
Drink_Item
PK FK item_id
alcoholic
City
PK zip_code
city
state

Part 2

Normalized Relations



For the normalization process we made one minor edit to the table relationships. The Zip table was added in order to eliminate transitive dependencies and to achieve the 3rd normal form. The Zip table used zip_code as the foreign and primary key with city and state as attributes.

SQL Code

Database Creation

```
CREATE DATABASE if not exists restaurantdb;
USE restaurantdb;

CREATE TABLE zip
(
zip_code CHAR(7),
city VARCHAR(15),
state VARCHAR(15),
PRIMARY KEY(zip_code)
);

CREATE TABLE employee
(
emp_id INT NOT NULL AUTO_INCREMENT,
fname VARCHAR(20),
lname VARCHAR(20),
title CHAR(15),
street VARCHAR(30),
zip_code CHAR(7),
salary FLOAT,
start_date DATE,
end_date DATE,
PRIMARY KEY(emp_id),
FOREIGN KEY(zip_code) REFERENCES zip (zip_code)
        ON DELETE CASCADE
  ON UPDATE CASCADE
);

CREATE TABLE customer
(
customer_id INT NOT NULL AUTO_INCREMENT,
customer_fname VARCHAR(15),
customer_lname VARCHAR(15),
customer_phone CHAR(10),
PRIMARY KEY(customer_id)
);


CREATE TABLE orders
```

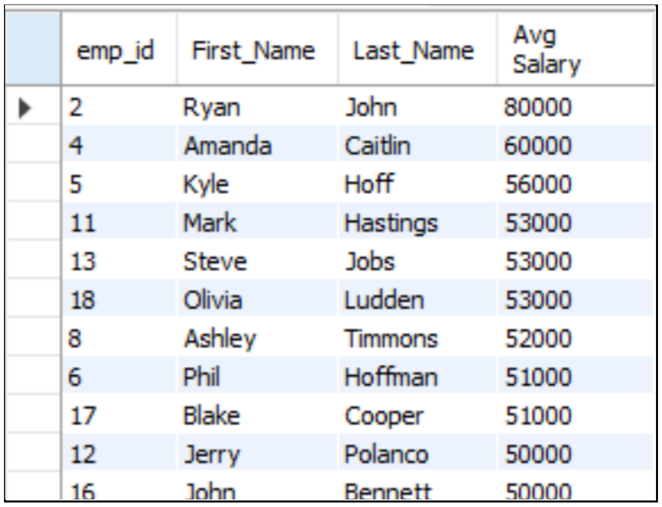
```
(
order_id INT NOT NULL AUTO_INCREMENT,
emp_id INT,
customer_id INT,
order_date DATE,
PRIMARY KEY(order_id),
FOREIGN KEY(emp_id) REFERENCES employee (emp_id)
            ON DELETE CASCADE
   ON UPDATE CASCADE,
FOREIGN KEY(customer_id) REFERENCES customer (customer_id)
            ON DELETE CASCADE
   ON UPDATE CASCADE
);

CREATE TABLE menu_item
(
item_id INT NOT NULL AUTO_INCREMENT,
item_name VARCHAR(20),
item_price FLOAT,
inventory CHAR(4),
item_type CHAR(5),
PRIMARY KEY(item_id)
);

CREATE TABLE order_details
(
order_details_id INT NOT NULL AUTO_INCREMENT,
order_id INT,
item_id INT,
quantity CHAR(3),
PRIMARY KEY(order_details_id, order_id),
FOREIGN KEY(order_id) REFERENCES orders (order_id)
            ON DELETE CASCADE
   ON UPDATE CASCADE,
FOREIGN KEY(item_id) REFERENCES menu_item (item_id)
            ON DELETE CASCADE
   ON UPDATE CASCADE
);


CREATE TABLE food_item
(
item_id INT,
calories SMALLINT,
gluten_free BOOLEAN,
vegan BOOLEAN,
PRIMARY KEY(item_id),
FOREIGN KEY(item_id) REFERENCES menu_item (item_id)
            ON DELETE CASCADE
   ON UPDATE CASCADE
);

CREATE TABLE drink_item
(
item_id INT,
alcoholic BOOLEAN,
PRIMARY KEY(item_id),
FOREIGN KEY(item_id) REFERENCES menu_item (item_id)
            ON DELETE CASCADE
   ON UPDATE CASCADE
);

CREATE TABLE reservation
(
reservation_id INT NOT NULL AUTO_INCREMENT,
customer_id INT,
reservation_date DATE,
PRIMARY KEY(reservation_id),
FOREIGN KEY(customer_id) REFERENCES customer (customer_id)
            ON DELETE CASCADE
   ON UPDATE CASCADE
);
```

Database Population

```
INSERT INTO zip
VALUES
("93405", "San Luis Obispo", "California"),
("93446", "Paso Robles", "California");


INSERT INTO employee (fname, lname, title, street, zip_code, salary, start_date, end_date)
VALUES
   ("Daniel", "Solomon", "waiter", "dover", "93405", 45000, "2016-02-01", NULL),
   ("Ryan", "John", "manager", "jeffrey", "93405", 80000, "2014-02-01", NULL),
   ("Shane", "Taylor", "waiter", "Frogley", "93405", 40000, "2019-02-01", NULL),
   ("Amanda", "Caitlin", "manager", "Lemon",  "93405", 60000, "2018-02-01", NULL),
   ("Kyle", "Hoff", "waiter", "acute",  "93405", 56000, "2018-02-01", "2019-03-05"),
   ("Phil", "Hoffman", "waiter", "Simmons", "93405", 51000, "2018-04-09", "2019-09-01"),
   ("Tim", "Phillips", "waiter", "Foothill",  "93405", 49000, "2015-05-02", "2020-01-05"),
   ("Ashley", "Timmons", "waiter", "California", "93405", 52000, "2015-01-06", NULL),
   ("Alex", "Evans", "waiter", "Ticket",  "93405", 45000, "2019-08-01", "2019-11-05"),
   ("Mary", "Sano", "waiter", "Pasture", "93405", 44000, "2016-02-01", "2019-03-08"),
   ("Mark", "Hastings", "waiter", "acute", "93405", 53000, "2018-02-04", NULL),
   ("Jerry", "Polanco", "waiter", "los osos", "93405", 50000, "2015-02-01", NULL),
   ("Steve", "Jobs", "waiter", "Hillman", "93405", 53000, "2017-03-01", "2019-08-04"),
   ("Mati", "Kepler", "waiter", "Santa Rosa", "93405", 45000, "2020-01-04", NULL),
   ("Chris", "Masters", "waiter", "gilbert",  "93405", 41000, "2019-02-01", "2020-03-07"),
   ("John", "Bennett", "waiter", "carlton",  "93405", 50000, "2015-04-04", "2019-09-21"),
   ("Blake", "Cooper", "waiter", "Orange",  "93405", 51000, "2018-04-022", "2019-05-22"),
   ("Olivia", "Ludden", "waiter", "Winderemere", "93405", 53000, "2015-02-27", "2017-03-16"),
   ("Will", "Ferguson", "waiter", "shelby", "93405", 41000, "2019-12-03", "2020-07-03"),
   ("Will", "Merriman", "dish washer", "Sand Point", "93446", 45000, "2014-08-12", NULL);

select * from employee;

INSERT INTO customer
VALUES
(NULL, "Tim", "Salmon", "4539221348"),
   (NULL, "Angela", "Trout", "8056432358"),
   (NULL, "Mike", "Scott", "3045443218"),
   (NULL, "Ann", "Taylor", "8059843651"),
   (NULL, "Scott", "Hanson", "2069229653"),
   (NULL, "Tim", "Roberts", "8053316519"),
   (NULL, "Roger", "Butkus", "4251916531"),
   (NULL, "Lily", "Roberts", "3431219482"),
   (NULL, "Selena", "Hayek", "3873239853"),
   (NULL, "Mike", "Haniger", "8053439845");

INSERT INTO orders
VALUES
(NULL, 1, 1, "2020-01-16"),
   (NULL, 3, 2, "2020-01-18"),
   (NULL, 1, 3, "2020-01-19"),
   (NULL, 3, 4, "2020-01-19"),
   (NULL, 3, 1, "2020-01-20"),
   (NULL, 11, 5, "2020-01-20"),
   (NULL, 12, 1, "2020-01-20"),
   (NULL, 3, 6, "2020-01-20"),
   (NULL, 1, 7, "2020-01-20"),
   (NULL, 11, 5, "2020-01-21"),
   (NULL, 11, 8, "2020-01-21"),
   (NULL, 12, 9, "2020-01-21");

INSERT INTO menu_item
VALUES
(NULL, "Burger", 7.50, 134, "food"),
   (NULL, "Vegan Burger", 8.50, 89, "food"),
   (NULL, "Chicken Burger", 9.00, 177, "food"),
   (NULL, "Fries", 4.50, 402, "food"),
   (NULL, "Burrito", 9.00, 129, "food"),
   (NULL, "Veggie Burrito", 8.00, 185, "food"),
   (NULL, "Coke", 2.50, 134, "drink"),
   (NULL, "Sprite", 2.50, 128, "drink"),
   (NULL, "Dr. Pepper", 2.50, 150, "drink"),
   (NULL, "Beer", 5, 90, "drink"),
   (NULL, "Wine", 7, 48, "drink"),
   (NULL, "Alcoholic Seltzer", 5, 180, "drink");


INSERT INTO food_item
VALUES
(1, 700, FALSE, FALSE),
(2, 450, FALSE, FALSE),
   (3, 850, FALSE, FALSE),
   (4, 550, FALSE, TRUE),
   (5, 900, FALSE, FALSE),
   (6, 400, FALSE, TRUE);

INSERT INTO drink_item
VALUES
(7, FALSE),
   (8, FALSE),
(9, FALSE),
   (10, TRUE),
   (11, TRUE),
   (12, TRUE);

INSERT INTO reservation
VALUES
(NULL, 1, "2020-01-16"),
   (NULL, 2, "2020-01-18"),
   (NULL, 3, "2020-01-18"),
   (NULL, 4, "2020-01-19"),
   (NULL, 5, "2020-01-19"),
   (NULL, 5, "2020-01-20"),
   (NULL, 6, "2020-01-20"),
   (NULL, 1, "2020-01-21"),
   (NULL, 8, "2020-01-22"),
   (NULL, 9, "2020-01-23");

INSERT INTO order_details
VALUES
(NULL, 1, 2, 1),
   (NULL, 1, 7, 1),
   (NULL, 2, 3, 2),
   (NULL, 2, 9, 1),
   (NULL, 3, 3, 1),
   (NULL, 4, 5, 2),
   (NULL, 5, 7, 1),
   (NULL, 5, 1, 1),
   (NULL, 6, 3, 1),
   (NULL, 6, 10, 1),
   (NULL, 7, 1, 1),
   (NULL, 8, 8, 3),
   (NULL, 9, 2, 2),
   (NULL, 10, 2, 1),
   (NULL, 10, 10, 1);
```

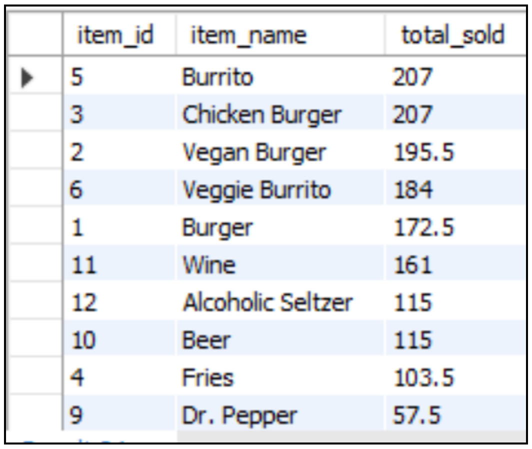
Database Testing

```
#Find employees with average salary > 45000
SELECT emp_id, fname AS "First_Name", lname AS "Last_Name", AVG(salary) AS "Avg Salary"
FROM employee
GROUP BY emp_id
HAVING AVG(salary) >= 45000
ORDER BY AVG(salary) DESC;
```



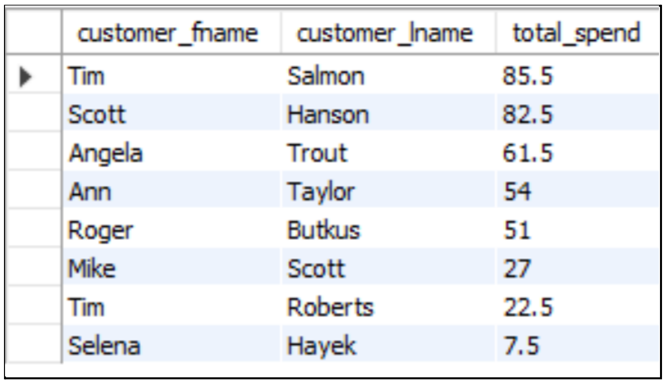
emp_id	First_Name	Last_Name	Avg Salary
2	Ryan	John	80000
4	Amanda	Caitlin	60000
5	Kyle	Hoff	56000
11	Mark	Hastings	53000
13	Steve	Jobs	53000
18	Olivia	Ludden	53000
8	Ashley	Timmons	52000
6	Phil	Hoffman	51000
17	Blake	Cooper	51000
12	Jerry	Polanco	50000
16	John	Bennett	50000

```
#Find the most profitable item on the menu
SELECT m.item_id, m.item_name, SUM(m.item_price * od.quantity) As total_sold
FROM menu_item m JOIN order_details od
GROUP BY m.item_id
ORDER BY total_sold DESC;
```



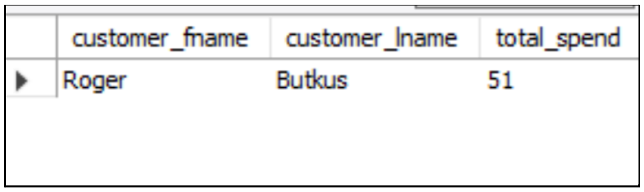
item_id	item_name	total_sold
5	Burrito	207
3	Chicken Burger	207
2	Vegan Burger	195.5
6	Veggie Burrito	184
1	Burger	172.5
11	Wine	161
12	Alcoholic Seltzer	115
10	Beer	115
4	Fries	103.5
9	Dr. Pepper	57.5

```
#Find the customers who spend the most amount of money
SELECT c.customer_fname, c.customer_lname, sum(mi.item_price*od.quantity) AS "total_spend"
FROM customer c JOIN menu_item mi JOIN order_details od JOIN orders o
ON od.order_id = o.order_id
AND o.customer_id = c.customer_id
AND od.item_id = mi.item_id
GROUP BY c.customer_fname, c.customer_lname
ORDER BY total_spend DESC;
```



customer_fname	customer_lname	total_spend
Tim	Salmon	85.5
Scott	Hanson	82.5
Angela	Trout	61.5
Ann	Taylor	54
Roger	Butkus	51
Mike	Scott	27
Tim	Roberts	22.5
Selena	Hayek	7.5

```
#Display the information of customers who have never placed reseravations
SELECT c.customer_fname, c.customer_lname, sum(mi.item_price*od.quantity) AS "total_spend"
FROM customer c JOIN menu_item mi JOIN order_details od JOIN orders o
ON od.order_id = o.order_id
AND o.customer_id = c.customer_id
AND od.item_id = mi.item_id
WHERE c.customer_id NOT IN
(
SELECT customer_id
FROM reservation
)
GROUP BY c.customer_fname, c.customer_lname
ORDER BY total_spend DESC;
```



customer_fname	customer_lname	total_spend
Roger	Butkus	51

*There was only one customer that did not make a reservation in the data we provided