# Project Report

Team 1: Eunhee Kim, An Nguyen, Zach Shim, Max Nguyen, Kevin Nguyen
CSS 475 | Dr. Min Chen
University of Washington, Bothell

## Table of Contents

## Description

This application focuses on assisting the restaurant industry in providing services, specifically for chain family restaurants. For example, Red Robin is a national chain restaurant with many customers and reservations each day, along with a diverse menu. In addition, there may be many chain restaurants under the same brand name, so we must be able to distinguish between varying receipts, tables, and staff, and retrieve different information to maintain high quality business posture. The following document is an outline for the implementation of a SQL database that allows these many restaurants to query information from a central repository.

## Documentation

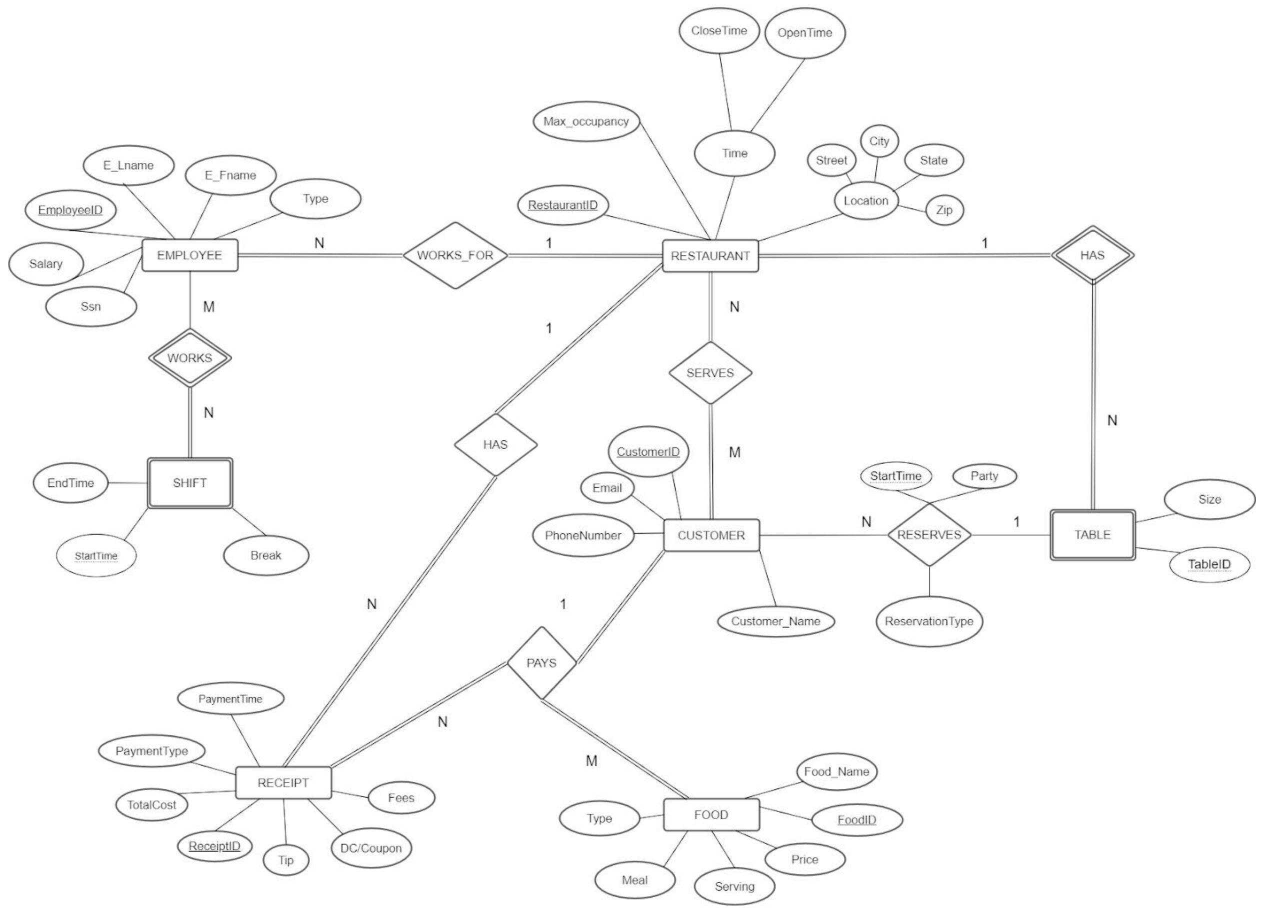
### Assumptions

- Each of RestaurantID, EmployeeID, CustomerID, FoodID, and ReceiptID is unique
- Restaurant:
  - This is a fast food restaurant where all restaurants at all locations have the same name
  - Restaurant must have at least 1 table, 1 customer, 1 employee, and 1 receipt.
  - Receipt has to be in a restaurant
  - A restaurant must minimally employ 1 employee
  - 1 restaurant can only have 1 value for each of max_occupancy, OpenTime, and CloseTime
- Receipt:
  - Receipt has to be in a restaurant
  - DC/Coupon is the amount of money off from a coupon
  - Each receipt must have only 1 Payment Type
  - Fee is the cost of delivery
- Restaurant_table:
  - TableID can be similar in different Restaurants, but TableID is unique inside each Restaurant
  - The total size of all table in a restaurant must equal the capacity of the restaurant
- Employee:
  - Employee must work for only 1 restaurant
  - Each Employee can work for only 1 type of role
  - All employees working for the same position have different salaries based on their raises
  - An employee works 260 days a year. [1]
- Customer:
  - Customer’s email and phone number can be null
  - This is a fast food restaurant, so customers do not have to have an account to buy/pay for their food. However, if they make a reservation, they need to have an account.
  - Customer has to be served by a restaurant

    - Customer does not have to reserves to a table in order to eat in a restaurant, but if they want to have a party in a restaurant, they must make a reservation with the specific table
- Pays:
    - Customer can only pay with one payment type (no splitting fees for using both cash and card for same bill)
- Shift:
    - Break's unit (in SHIFT) is minute
- Reserves:
    - Size of the party must be equal or smaller than Size of table

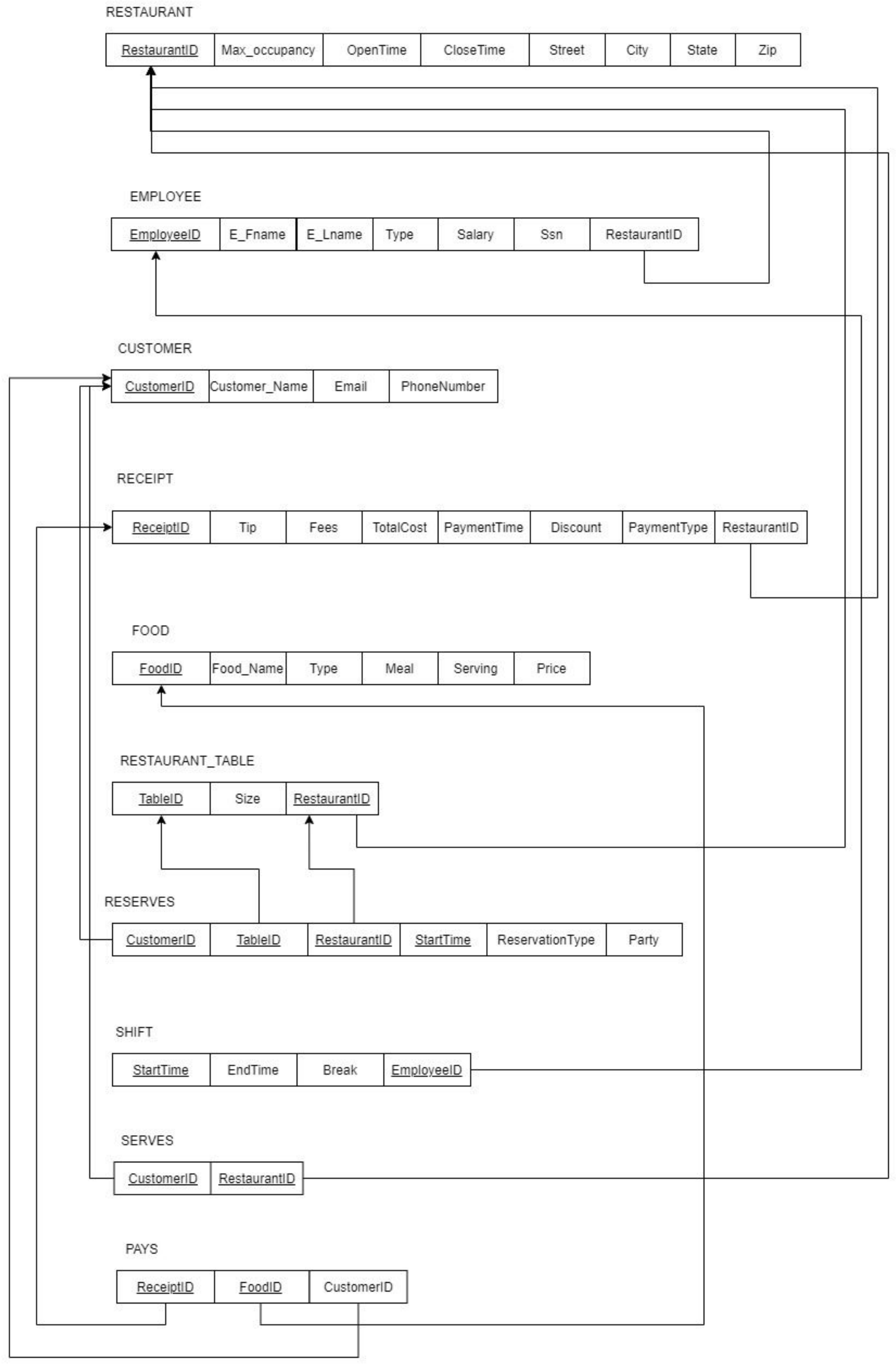
## Entity-Relationship Diagram

ER diagram - External Link



## Relational Data Model

Relational Data Model - External Link



## SQL Statements (create database)

Please refer to project_sql_statements.txt.

| **Purpose** | **SQL statement** |
|---|---|
| Clearing all existing tables from team_project.txt | drop table restaurant;<br>drop table employee;<br>drop table shift;<br>drop table food;<br>drop table customer;<br>drop table receipt;<br>drop table restaurant_table;<br>drop table pays;<br>drop table serves;<br>drop table reserves; |

### Create table statements

| **Purpose** | **SQL statement** |
|---|---|
| Creating a restaurant table with attributes containing the RestaurantID, Max_occupancy, Open_time, Close_time, Street, City, State, and Zip.<br>Assigning primary key to RestaurantID | create table restaurant (<br>    RestaurantID int not null,<br>    Max_occupancy int check (Max_occupancy > 0),<br>    Open_time time not null,<br>    Close_time time not null,<br>    Street varchar(30) not null,<br>    City varchar(30) not null,<br>    State char(2) not null,<br>    Zip int not null check (zip >= 10000 and zip <= 99999),<br>    primary key(restaurantID)<br>); |
| Creating an employee table with attributes containing EmployeeID, E_Fname, E_Lname, Wage, Type (limited to 8 types), Ssn, RestaurantID.<br>Assigning primary key to the EmployeeID with secondary key as Ssn.<br>Connecting cascades within the foreign key of RestaurantID with the | create table employee (<br>    EmployeeID      int not null check (EmployeeID >= 1000 and EmployeeID <= 9999),<br>    E_Fname        varchar(15) not null,<br>    E_Lname        varchar(15) not null,<br>    Wage           decimal(10,2),<br>    Type           varchar(15) not null check (Type in ('Waiter', 'Waitress', 'Chef', 'Manager', 'Owner', 'Host', 'Hostess', 'Janitor')), |

| | |
|---|---|
| RestaurantID coming from the restaurant table. | Ssn char(9) not null,<br>RestaurantID int not null,<br>primary key (EmployeeID),<br>unique (Ssn),<br>foreign key(RestaurantID) references restaurant(RestaurantID) on delete cascade on update cascade<br>check (Wage > 0)<br>); |
| Creating a shift table with attributes containing the StartTime, EndTime, Break, and EmployeeID.<br>Assigning the primary key to the StartTime and EmployeeId.<br>Connecting cascades within the foreign key of EmployeeID with the EmployeeID coming from the employee table. | create table shift (<br>StartTime timestamp not null,<br>EndTime timestamp not null,<br>Break int,<br>EmployeeID int not null,<br>primary key(StartTime, EmployeeID),<br>foreign key(EmployeeID) references employee(EmployeeID) on delete cascade on update cascade<br>); |
| Creating a food table with attributes containing the FoodID, Food_Name, Type, Meal, Serving, and Price.<br>Assigning the primary key to the FoodID. | create table food (<br>FoodID int not null check(FoodID >= 100000 and FoodID <= 999999),<br>Food_Name varchar(30) not null,<br>Type varchar(10) not null,<br>Meal varchar(15) not null,<br>Serving int,<br>Price decimal(4,2) not null,<br>primary key (FoodId),<br>check (Type in ('Drink', 'Dessert', 'Entree', 'Appetizer', 'Side', 'Any')),<br>check (Meal in ('Breakfast', 'Lunch', 'Dinner', 'Any'))<br>); |
| Creating a customer table with attributes containing the CustomerID, Customer_Name, Email, and Phone_number.<br>Assigning the primary key to the CustomerID. | create table customer (<br>CustomerID int not null,<br>Customer_Name varchar (20),<br>Email varchar(30),<br>Phone_number char(12),<br>primary key (CustomerID),<br>check (CustomerID >= 1000 and CustomerID <= 9999),<br>check (email is not null OR phone_number is not null) |

| | |
|---|---|
| | ); |
| Creating a receipt table with attributes containing the ReceiptID, Tip, Fees, TotalCost, PaymentTime, Discount, PaymentType, and RestaurantID. Assigning the primary key to the ReceiptID. Connecting cascades within the foreign key of RestaurantID with the RestaurantID coming from the restaurant table. | create table receipt( ReceiptID int not null check(receiptID >= 1000000 and receiptID <= 9999999), Tip decimal(5,2) not null, Fees decimal(5,2) not null, TotalCost decimal(5,2) not null, PaymentTime timestamp not null, Discount decimal(10,2) not null, PaymentType varchar(15) not null check(PaymentType in ('Cash', 'Card', 'GiftCard', 'Check')), RestaurantID int not null check (RestaurantID >= 100 and RestaurantID <= 999), primary key (ReceiptID), foreign key(RestaurantID) references restaurant(RestaurantID) on delete set null on update cascade ); |
| Creating a restaurant_table table with attributes containing TableID, Size, RestaurantID. Assigning the primary key to TableID and RestaurantID. Connecting cascades within the foreign key of RestaurantID with the RestaurantID coming from the Restaurant table. | create table restaurant_table ( TableID int not null, Size int not null, RestaurantID int not null , primary key (TableID, RestaurantID), foreign key (RestaurantID) references restaurant(RestaurantID) on delete cascade on update cascade, check (RestaurantID >= 100 and RestaurantID <= 999), check (Size > 0), check (TableID >= 10 and TableID <= 99) ); |
| Creating a pays table with attributes containing the ReceiptID, FoodID, and CustomerID. Assigning the primary key to the ReceiptID and FoodID. Connecting cascades within the foreign key of ReceiptID with the ReceiptID coming from the receipt table, FoodID with the FoodID coming from the food table, | create table pays ( ReceiptID int not null check(receiptID >= 1000000 and receiptID <= 9999999), FoodID int not null check(FoodID >= 100000 and FoodID <= 999999), CustomerID int not null check(CustomerID >= 1000 and CustomerID <= 9999), primary key(ReceiptID, FoodID), |

| | |
|---|---|
| CustomerID with the CustomerID coming from the customer table. | foreign key(ReceiptID) references receipt(receiptID) on delete cascade on update cascade,<br>foreign key(FoodID) references food(FoodID) on delete cascade on update cascade,<br>foreign key(CustomerID) references customer(CustomerID) on delete set null on update cascade<br>); |
| Creating a reserves table with attributes containing CustomerID, TableID, RestaurantID, StartTime, ReservationType, Party. Assigning the primary key to CustomerID, TableID, RestaurantID, and StartTime. Connecting cascades within the foreign key of CustomerID with the CustomerID coming from the Restaurant table. Also connecting cascades within the foreign key of TableID and RestaurantID with TableID, RestaurantID coming from the restaurant_table table. | create table reserves (<br>CustomerID int not null check (CustomerID >= 1000 and CustomerID <= 9999),<br>TableID int not null check (TableID >= 10 and TableID <= 99),<br>RestaurantID int not null check (RestaurantID >= 100 and RestaurantID <= 999),<br>StartTime timestamp not null,<br>ReservationType varchar(20),<br>Party int not null check (party >= 1),<br>primary key(CustomerID, TableID, RestaurantID, StartTime),<br>foreign key(CustomerID) references customer(CustomerID) on delete cascade on update cascade,<br>foreign key(TableID, RestaurantID) references restaurant_table(TableID, RestaurantID) on delete cascade on update cascade<br>); |
| Creating a serves table with attributes containing the CustomerID and RestaurantID. Assigning the primary key to the CustomerID and RestaurantID. Connecting cascades within the foreign key of RestaurantID with the RestaurantID coming from the restaurant table, CustomerID with the CustomerID coming from the customer table. | create table serves (<br>CustomerID int not null check (CustomerID >= 1000 and CustomerID <= 9999),<br>RestaurantID int not null check (RestaurantID >= 100 and RestaurantID <= 999),<br>primary key(CustomerID, RestaurantID),<br>foreign key(CustomerID) references customer(CustomerID) on delete cascade on update cascade,<br>foreign key(RestaurantID) references restaurant(RestaurantID) on delete cascade on update cascade); |

## Insertion statements

| Purpose | SQL statement |
|---|---|
| Populating the restaurant table with values | insert into restaurant values (100, 200, '8:00', '22:00', '152 53th Ave S', 'Renton', 'WA',  98134);<br>insert into restaurant values (101, 50, '6:00', '22:00', '555 Campus Way N', 'Bothell', 'WA',  97546);<br>insert into restaurant values (102, 100, '7:00', '21:00', '202 1st Ave N', 'Tacoma', 'WA',  98121);<br>insert into restaurant values (103, 150, '8:00', '21:30', '1001 Pine St', 'Seattle', 'WA',  92012);<br>insert into restaurant values (104, 75, '5:00', '22:30', '152 2nd Ave', 'Woodenville', 'WA',  98254); |
| Populating the customer table with values | insert into customer values(1111, 'Emma Watson', 'emma.watson@gmail.com', '2065554999');<br>insert into customer values(2222, 'Daniel Radcliffe', 'iamharrypotter@gmail.com', '2516667878');<br>insert into customer values(3333, 'John Smith', 'johns@gmail.com', '2065558888');<br>insert into customer values(4444, 'Rupert Grint', 'iamron@gmail.com', '5554049999'); |
| Populating the food table with values | insert into food values(111111, 'Chicken Wings', 'Entree', 'Any', 3, 11.99);<br>insert into food values(111112, 'Cheese Burger', 'Entree', 'Any', 2, 5.99);<br>insert into food values(111113, 'Cheesecake', 'Dessert', 'Any', 1, 4.99);<br>insert into food values(111114, 'Cheese Sticks', 'Appetizer', 'Any', 1, 3.99);<br>insert into food values(111115, 'Fries', 'Appetizer', 'Any', 1, 5.49);<br>insert into food values(111116, 'Coke', 'Drink', 'Any', 1, 0.99);<br>insert into food values(111117, 'Sprite', 'Drink', 'Any', 1, 0.99);<br>insert into food values(111118, 'Pancake', 'Entree', 'Breakfast', 1, 4.99); |
| Populating the restaurant_table table with values | insert into restaurant_table values(10, 4, 100);<br>insert into restaurant_table values(11, 5, 100);<br>insert into restaurant_table values(12, 4, 100);<br>insert into restaurant_table values(13, 10, 100);<br>insert into restaurant_table values(10, 4, 101);<br>insert into restaurant_table values(11, 3, 101); |

| | |
|---|---|
| | insert into restaurant_table values(12, 4, 101);<br>insert into restaurant_table values(13, 2, 101);<br>insert into restaurant_table values(10, 4, 102);<br>insert into restaurant_table values(11, 5, 102);<br>insert into restaurant_table values(12, 10, 102);<br>insert into restaurant_table values(13, 15, 102);<br>insert into restaurant_table values(10, 4, 103);<br>insert into restaurant_table values(11, 2, 103);<br>insert into restaurant_table values(12, 4, 103);<br>insert into restaurant_table values(13, 6, 103);<br>insert into restaurant_table values(10, 2, 104);<br>insert into restaurant_table values(11, 4, 104);<br>insert into restaurant_table values(12, 8, 104);<br>insert into restaurant_table values(13, 8, 104); |
| Populating the employee table with values | insert into employee values(1001, 'Jane', 'Whiteman', 17.50, 'Chef', '321654789', 100);<br>insert into employee values(1002, 'Joe', 'Gatto', 18.50, 'Chef', '436278743', 100);<br>insert into employee values(1003, 'Jessica', 'Carpenter', 16.00, 'Waitress', '123456789', 100);<br>insert into employee values(1004, 'Flo', 'Florence', 16.00, 'Waitress', '534782748', 101);<br>insert into employee values(1005, 'Luke', 'Rawlings', 14.00, 'Janitor', '258962156', 101);<br>insert into employee values(1006, 'Bob', 'Peng', 31.25, 'Manager', '751235485', 101);<br>insert into employee values(1007, 'Pat', 'Williamson', 16.00, 'Waitress', '12345999', 102);<br>insert into employee values(1008, 'Alex', 'Connor', 16.00, 'Host', '953827478', 102);<br>insert into employee values(1009, 'Tom', 'Brady', 30.50, 'Manager', '198309403', 102);<br>insert into employee values(1010, 'Twilight', 'Sparkle', 100.00, 'Owner', '234234859', 103);<br>insert into employee values(1011, 'Jeff', 'Bezos', 16.50, 'Waiter', '159452170', 104);<br>insert into employee values(1012, 'Elon', 'Musk', 18.20, 'Janitor', '847264729', 104); |
| Populating the shift table with values | insert into shift values('2021-01-10 07:30:00', '2021-01-10 15:00:00', 30, 1001);<br>insert into shift values('2021-01-11 13:30:00', '2021-01-11 23:00:00', 30, 1001); |

| | |
|---|---|
| | insert into shift values('2021-01-10 13:00:00', '2021-01-10 23:30:00', 30, 1002); insert into shift values('2021-01-11 07:00:00', '2021-01-11 14:00:00', 40, 1002); insert into shift values('2021-01-10 07:30:00', '2021-01-10 14:30:00', 30, 1003); insert into shift values('2021-01-11 14:00:00', '2021-01-11 22:30:00', 30, 1003); insert into shift values('2021-01-14 05:30:00', '2021-01-14 13:00:00', 45, 1004); insert into shift values('2021-01-13 12:30:00', '2021-01-13 22:30:00', 30, 1004); insert into shift values('2021-01-13 20:30:00', '2021-01-13 23:00:00', 50, 1005); insert into shift values('2021-01-14 20:00:00', '2021-01-14 23:00:00', 30, 1005); insert into shift values('2021-01-13 05:00:00', '2021-01-13 16:00:00', 60, 1006); insert into shift values('2021-01-14 12:30:00', '2021-01-14 22:30:00', 30, 1006); insert into shift values('2021-01-16 06:00:00', '2021-01-16 14:00:00', 30, 1007); insert into shift values('2021-01-15 13:30:00', '2021-01-15 22:00:00', 100, 1007); insert into shift values('2021-01-15 06:30:00', '2021-01-15 15:00:00', 30, 1008); insert into shift values('2021-01-16 14:30:00', '2021-01-16 22:30:00', 30, 1008); insert into shift values('2021-01-16 06:30:00', '2021-01-16 13:00:00', 35, 1009); insert into shift values('2021-01-15 12:30:00', '2021-01-15 22:00:00', 30, 1009); insert into shift values('2021-01-17 07:30:00', '2021-01-17 22:00:00', 45, 1010); insert into shift values('2021-01-18 07:00:00', '2021-01-18 22:00:00', 30, 1010); insert into shift values('2021-01-19 04:30:00', '2021-01-19 14:00:00', 40, 1011); insert into shift values('2021-01-20 13:00:00', '2021-01-20 23:00:00', 50, 1011); insert into shift values('2021-01-20 19:30:00', '2021-01-20 23:00:00', 45, 1012); |

| | |
|---|---|
| | insert into shift values('2021-01-19 19:00:00', '2021-01-19 23:00:00', 30, 1012); |
| Populating the reserves table with values | insert into reserves values(1111, 10, 100, '2021-02-12 15:30:00', 'family reunion', 30);<br>insert into reserves values(2222, 11, 101, '2021-02-12 11:30:00', 'date',3);<br>insert into reserves values(3333, 10, 103, '2021-02-12 09:30:00', 'business',5);<br>insert into reserves values(2222, 12, 100, '2021-03-16 08:30:00', 'wedding', 30);<br>insert into reserves values(1111, 13, 100, '2021-06-23 15:30:00', 'wedding', 30);<br>insert into reserves values(2222, 13, 101, '2021-01-12 18:00:00', 'birthday', 5);<br>insert into reserves values(3333, 12, 102, '2021-08-08 19:30:00', 'business',3);<br>insert into reserves values(4444, 10, 104, '2021-05-19 13:45:00', null, 10); |
| Populating the serves table with values | insert into serves values(1111, 100);<br>insert into serves values(2222, 101);<br>insert into serves values(3333, 102);<br>insert into serves values(4444, 104); |
| Populating the receipt table with values | insert into receipt values(1000000, 5.00, 6.00, 40.00, '2021-01-10 12:53:02', 2.00, 'Card', 100);<br>insert into receipt values(1000001, 10.00, 20.00, 85.23, '2021-01-10 22:59:51', 10.00, 'Card', 100);<br>insert into receipt values(1000002, 0.00, 10.29, 21.67, '2021-01-11 11:11:21', 0.00, 'Cash', 101);<br>insert into receipt values(1000003, 10.11, 0.89, 11.21, '2021-01-11 12:45:56', 1.40, 'GiftCard', 101);<br>insert into receipt values(1000004, 10.56, 20.89, 153.23, '2021-01-13 23:45:56', 20.40, 'GiftCard', 102);<br>insert into receipt values(1000005, 20.01, 5.39, 368.23, '2021-01-13 10:45:56', 48.22, 'Check', 102);<br>insert into receipt values(1000006, 0.00, 0.00, 15.86, '2021-02-01 11:22:33', 0.00, 'Cash', 103);<br>insert into receipt values(1000007, 0.00, 0.00, 26.11, '2021-02-01 22:22:22', 0.00, 'Card', 103);<br>insert into receipt values(1000008, 0.00, 0.00, 12423.88, '2021-02-01 11:22:33', 0.00, 'Cash', 104); |

| | |
|---|---|
| | insert into receipt values(1000009, 0.00, 0.00, 4.99, '2021-02-01 22:22:22', 0.00, 'Card', 104);<br>insert into receipt values(1000010, 5.00, 6.00, 40.00, '2021-01-10 12:53:02', 2.00, 'Card', 100);<br>insert into receipt values(1000011, 10.00, 20.00, 85.23, '2021-01-10 22:59:51', 10.00, 'Card', 100);<br>insert into receipt values(1000012, 0.00, 10.29, 21.67, '2021-01-11 11:11:21', 0.00, 'Cash', 101);<br>insert into receipt values(1000013, 10.11, 0.89, 11.21, '2021-01-11 12:45:56', 1.40, 'GiftCard', 101);<br>insert into receipt values(1000014, 10.56, 20.89, 153.23, '2021-01-13 23:45:56', 20.40, 'GiftCard', 102);<br>insert into receipt values(1000015, 20.01, 5.39, 368.23, '2021-01-13 10:45:56', 48.22, 'Check', 102);<br>insert into receipt values(1000016, 0.00, 0.00, 15.86, '2021-02-01 11:22:33', 0.00, 'Cash', 103);<br>insert into receipt values(1000017, 0.00, 0.00, 26.11, '2021-02-01 22:22:22', 0.00, 'Card', 103);<br>insert into receipt values(1000018, 1.00, 2.00, 12423.88, '2021-02-01 11:22:33', 0.00, 'Cash', 104);<br>insert into receipt values(1000019, 1.23, 1.89, 27.99, '2021-02-01 22:22:22', 0.00, 'Card', 104); |
| Populating the pays table with values | insert into pays values(1000000, 111111, 1111);<br>insert into pays values(1000001, 111112, 1112);<br>insert into pays values(1000002, 111113, 1113);<br>insert into pays values(1000003, 111114, 4444);<br>insert into pays values(1000004, 111111, 2222);<br>insert into pays values(1000004, 111112, 2222);<br>insert into pays values(1000004, 111113, 2222);<br>insert into pays values(1000004, 111114, 2222);<br>insert into pays values(1000004, 111115, 2222);<br>insert into pays values(1000005, 111116, 3333);<br>insert into pays values(1000006, 111117, 4444);<br>insert into pays values(1000007, 111118, 4444);<br>insert into pays values(1000008, 111111, 4444);<br>insert into pays values(1000009, 111111, 3333);<br>insert into pays values(1000010, 111111, 4444);<br>insert into pays values(1000011, 111112, 3333);<br>insert into pays values(1000012, 111113, 4444);<br>insert into pays values(1000013, 111114, 4444);<br>insert into pays values(1000014, 111115, 3333);<br>insert into pays values(1000015, 111116, 3333);<br>insert into pays values(1000016, 111117, 3333); |

| | |
|---|---|
| | insert into pays values(1000017, 111118, 4444);<br>insert into pays values(1000018, 111111, 4444);<br>insert into pays values(1000019, 111111, 1111);<br>insert into pays values(1000019, 111113, 1111);<br>insert into pays values(1000019, 111115, 1111);<br>insert into pays values(1000019, 111117, 1111);<br>insert into pays values(1000019, 111118, 1111); |

## Statements to show all content

| Purpose | SQL statement |
|---|---|
| Showing all content from all tables | select * from restaurant;<br>select * from customer;<br>select * from food;<br>select * from employee;<br>select * from restaurant_table;<br>select * from shift;<br>select * from reserves;<br>select * from serves;<br>select * from pays;<br>select * from receipt; |

## Query statements

Note that the following SQL may be found in project_sql_statements.txt.

| Purpose | SQL statement |
|---|---|
| Retrieve a receipt based on receiptID from a given restaurant to bill a customer for their food.<br>To process transactions for food orders, a restaurant needs to be able to generate a receipt for a customer based on the food they ordered.<br>For example: retrieve a receipt for a customer from a restaurant with restaurantID = 100 and receipt = 1000000. | select customer_name, p.receiptID, TotalCost from pays p, customer c, restaurant r, receipt where p.CustomerID = c.CustomerID and receipt.RestaurantID = r.RestaurantID and receipt.ReceiptID = p.ReceiptID and r.restaurantID = 100 and p.receiptID = 1000000; |
| Retrieve the salary (get the sum of all shifts worked in a (user) given year by an employee, for example, 2021) for all employees. | select e.employeeID, e.E_Fname, e.E_Lname, e.type, sum((((strftime('%H', s.endTime) - strftime('%H',s.startTime)) + ((strftime('%M', s.endTime)/60) - |

| | |
|---|---|
| | (strftime('%M',s.startTime)/60))) * e.wage))<br>from employee e, shift s<br>where e.employeeID = s.employeeID and s.startTime like ('%2021%')<br>group by e.employeeID; |
| Allow a specific employee to see their salary given their employeeID.<br>For example, retrieve the salary for employee 1001. | select e.employeeID, e.E_Fname, e.E_Lname, e.type, sum((((strftime('%H', s.endTime) - strftime('%H',s.startTime)) + ((strftime('%M', s.endTime)/60) - (strftime('%M',s.startTime)/60))) * e.wage))<br>from employee e, shift s<br>where e.employeeID = s.employeeID and s.startTime like ('%2021%') and e.employeeID = 1001<br>group by e.employeeID; |
| Retrieve an employee's earnings for a certain day based on the hours they worked in their shift multiplied by their hourly wage given a date and an EmployeeID.<br>For example, retrieve the earnings for employee 1001 on the 10th of January in 2021 | select e.E_Fname, e.E_Lname, e.type, (((strftime('%H', s.endTime) - strftime('%H',s.startTime)) + ((strftime('%M', s.endTime)/60) - (strftime('%M',s.startTime)/60))) * e.wage)<br>from employee e, shift s<br>where e.employeeID = s.employeeID and s.startTime like ('%2021-01-10%') and e.employeeID = 1001; |
| Retrieve all receipts that include information about each order (food ordered). | select pays.ReceiptID, food.Food_Name, food.price<br>from food, pays<br>where food.FoodID = pays.FoodID; |
| List all items (food) a customer ordered given a receiptID. | select f.Food_Name, f.price<br>from food f, pays p<br>where f.FoodID = p.FoodID and p.receiptID = 1000019; |
| Retrieve all table reservation times for a certain day to find the table that a customer registered for.<br>When a customer registered for a reservation either online or called it in, a host/hostess should be able to find and bring it up. | select Customer_name, TableID, StartTime<br>from CUSTOMER c, RESERVES r<br>where c.CustomerID = r.CustomerID and StartTime like ('%2021-02-12%'); |
| Retrieve the food menu for all restaurants (The menu is similar for all restaurants) | select Food_Name, Type, Meal, Serving, Price<br>from food<br>order by Food_Name ASC; |
| Retrieve a restaurant based on location to | select rt.RestaurantID, rsv.TableID, |

| | |
|---|---|
| see their reservation times.<br>For example, to book a reservation, a customer may want to see all tables that are currently booked by a local restaurant. | rsv.StartTime<br>from reserves rsv, restaurant rt<br>where rt.City like ('%Bothell%')<br>order by rt.RestaurantID ASC; |
| Calculate the gross profit and number of sales of each restaurant by adding up all receipts a restaurant made in a year.<br>For example, calculate the total profit made by each restaurant in 2021 to see net profit margins. | select RestaurantID, sum(TotalCost), count(ReceiptID)<br>from receipt<br>where PaymentTime like ('%2021%')<br>group by RestaurantID; |
| Add new food during the holiday season, delete when it's over. | insert into food values (111120, 'Baked Holiday Pie', 'Dessert', 'Any', 1, 4.99);<br>insert into food values (111121, 'Baked Peach Pie', 'Dessert', 'Any', 1, 5.99);<br>insert into food values (111122, 'Baked Pumpkin Pie', 'Dessert', 'Any', 1, 3.99);<br>insert into food values (111123, 'Baked Strawberry Pie', 'Dessert', 'Any', 1, 4.99);<br>delete from food where foodID in (111120, 111121, 111122, 111123); |
| Retrieve Employee name and type (job) and restaurant for every restaurant = Washington. | select E_Fname, E_Lname, type, RestaurantID from employee<br>where RestaurantID in (select RestaurantID from restaurant where State = 'WA')<br>order by RestaurantID ASC; |

## Normal Form

Restaurant relation satisfies 3NF because of this assumption:

- 1 restaurant can only have 1 max_occupancy, OpenTime, and CloseTime

Employee relation satisfies 3NF because of these assumptions:

- Each Employee works for only 1 Restaurant
- Each Employee can work for only 1 type of role
- All employees working for the same position have different salaries based on their raises

Customer relation satisfies 3NF, because customer's email and phone number can be null, so those two attributes cannot be FDs.

Receipt relation satisfies 3NF, because of this assumption:

- Customer can only pay with one payment type (no splitting fees for using both cash and card for same bill)

Food, restaurant_table, reserves, shift, works, serves, pays relations all satisfy 3NF.

## SQL Statements (support functions)

- Retrieve a receipt based on receiptID from a given restaurant to bill a customer for their food
    - To process transactions for food orders, a restaurant needs to be able to generate a receipt for a customer based on the food they ordered
    - For example: to retrieve a receipt for a customer from restaurant with restaurantID = 100
- Retrieve the salary (get the sum of all shifts worked in a (user) given year, for example, 2021) for all employees
    - Allow employees to see history of payments and their annual salary
- Retrieve employee’s earnings for a day based on the hours they worked in their shift multiplied by their hourly wage
- Retrieve the items on the receipt
- Retrieve a reservation time that a customer registered for to a table
    - When a customer registered for a reservation either online or called it in, a host/hostess should be able to find and bring it up
- Retrieve the food menu for all restaurants (The menu is similar for all restaurants)
- Retrieve a restaurant based on location to see their reservation times
- Calculate the gross profit and number of sale of each restaurant by adding up all receipts a restaurant made in a year
- Add new food during holiday season, delete when it’s over
- Retrieve Employee name and type (job) and restaurant for every restaurant = Washington

Please refer to the text file submitted or the Query Statements section above for SQL codes.

## Project Evaluation

On the team project, we worked a total of 5 meetings on Discord where each took from 2.5 to 5 hours. During meetings, the majority of the team's work was done collaboratively. Our work included writing project proposals, building a Entity-Relation diagram and a relational data model, checking normal forms, and writing SQL statements.

During the building process of the Entity-Relationship Diagram, we gave careful consideration to each attribute, each table, and each relationship. For the relational data model, we also considered how to make the table convenient to use and not too fat. Furthermore, we provided detailed assumptions for each possible case to be considered. Because of those, we did not find any violations of the normalization during the later process. With the diagrams completed and all tables created with SQL code, we were able to populate the data and create the queries easily.

The model and SQL tables had to be altered multiple times because we made minor mistakes as we designed. These issues weren't major and weren't time consuming to repair, but it affected the process in general. Besides that, we encountered some minor bugs like SQL constraint errors and missing 1 column in a table. As soon as they were found, those were fixed immediately.

If we could do it again, we would spend time conducting the design process more thoroughly so that we wouldn't run into any conflicts in the future. In addition, we learned normalization in class after completing the diagram and the relational data model, so the model is good, but not yet in its best form. If we could re-build this database, we would pay more attention to normalization while building the diagram and the model.

## Reference

[1] “How Many Working Days Are In A Year?” Symmetry Software, Nov 13, 2017. https://www.symmetry.com/payroll-tax-insights/how-many-working-days-are-in-a-year.