Kwajo Ansong

CIS 344 Final Project Report

Restaurant Portal

MySQL Restaurant_Reservations database

In my restaurant reservations database, mysql code starts by creating the database and giving it the required tables for running customers, reservations, and dining preferences. I used restaurant_reservations to create a new database which allowed me to set it as a current database for the entire sql script. The code: **create database restaurant_reservations;**

use restaurant_reservations;

For second step, I created three tables for Customers, Reservations, and Dining_Preferences. The first table "**Customers**" stores details of the customer in the table like the unique identifier named customer_id, along with the customer's name and contact info. The second table "**Reservations**" is like customers table where it stores everything like the reservation time, number of guests, and special requests. Also, this table has its own unique identifier named reservation_id and customer_id. It links the ID's of customer_id and reservation_id to each reservation. The third table "**Dining_Preferences**" holds the details of favorite table and dietary restrictions. Also, it has links customer_id to customer. The code:

```
create table Customers
(
   customer_id int not null unique auto_increment,
   customer_name varchar(45) not null,
   contact_info varchar(200),
   primary key (customer_id)
```

```
);
create table Reservations
(
    reservation_id int not null unique auto_increment,
    customer_id int not null,
    reservation_time datetime not null,
    number_of_guests int not null,
    special_requests varchar(200),
    primary key (reservation_id),
    foreign key (customer_id) references Customers(customer_id)
);
create table Dining_Preferences
(
    preference_id int not null unique auto_increment,
    customer_id int not null,
    favorite_table varchar(45),
    dietary_restrictions varchar(200),
    primary key (preference_id),
    foreign key (customer_id) references Customers(customer_id)
);
```

For third step, I started to insert data into the three tables by using the insert function, values, and select. For customers, I connected the customers tables using the functions to

populate the customers names and contact info 5 times. For example, with the data from customers tables, inserted customers info with the values function, and use select * from customers to show what the output looks like. I did the same to reservations and dining preferences.

The code: **insert into Customers(customer_name, contact_info)**

values("Kwajo Ansong", 9145603690), ("Jaylen Brown", 5121123944), ("Grant Gustin", 9294560993),

("Trevor Noah", 8120984226), ("Boris Kodjoe", 7183451002);

select * from Customers;

insert into Reservations(customer_id, reservation_time, number_of_guests, special_requests)

values(1, "2024-05-10 8:00:00", 1, "None"), (2, "2024-05-10 8:30:00", 2, "None"), (3, "2024-05-10 9:00:00", 3, "None"),

(4, "2024-05-10 9:30:00", 1, "None"), (5, "2024-05-10 10:00:00", 0, "None");

select * from Reservations;

insert into Dining_Preferences(customer_id, favorite_table, dietary_restrictions)

values(1, "Table by no widow", "None"), (2, "Table by the widow", "Low-carb diet"), (3, "private table", "Lactose intolerance"),

(4, "Table near the entrance", "None"), (5,"Table near the bar", "None");

select * from Dining_Preferences;

For the fourth step, I created procedures using the procedures function along with delimiters. For **findReservation** procedure, I use the delimiter function to start and end the statement. Then I created the procedure for findReservations to get all reservations for the

customer_id. I put in **begin** and **end** to contain the select statement in the procedure. Finally, I used the select * from reservations to tell the database to select everything in the columns from reservations table. Also, the "where" clause is used to treat all records on a condition. For **addSpecialRequest** procedure, I create procedure where I put two parameters from reservations table to get the requests from that table. I used update, set, and where to update the special requests in reservations. For **addReservation** procedure, I created the procedure for addreservation where it creates another reservation. For declare, select, from, and where, a check is created for that part to see if the customer exists. If the customer doesn't exist, then if the customerID is null leaving a new created customer. Furthermore, the insert function is used for reservations where you bring all parameters from addReservation to add the reservation.

The code: **delimiter //**

```
create procedure findReservations(in customer_id int)
begin
   select * from Reservations where customer_id = customer_id;
end //
delimiter ;
delimiter //
create procedure addSpecialRequest(in reservation_id int, in requests varchar(200))
begin
   update Reservations
   set special_requests = requests
   where reservation_id = reservation_id;
end //
```

```
delimiter ;
delimiter //
create procedure addReservation(in customerName varchar(45), in special_requests
varchar(200), in reservationTime datetime)
begin
   declare customerID int;
   select customer_id into customerID
   from Customers
   where customer_name = customerName;

   if customerID is null then
      insert into Customers(customer_name) values(customerName);
      set customerID = last_insert_id();
   end if;

   insert into Reservations(customer_id, special_requests, reservation_time)
   values(customerID, special_requests, reservationTime);
end//
delimiter ;
```

Restaurant Database.py

In the python restaurant database, the code starts off by **import mysql.connector** which means to connect mysql from workbench to python (installed) through terminal. For the restaurant database class, it contains all the practicality needed to work with the sql database. With "**_init_**" **(initialization**), it starts the database connection of the parameters and sets up a connection to the database. As for the parameters: **host, port, database, user, and password** are used to highlight the connect method to show a connection. For the **connect** method, this part is used to connect to the database using the previous parameters from class restaurant database. The if **self.connection.is_connected** method is to show that the **def connect(self)** part can be successfully connected and functional. But if you encounter an error with the connection then **except Error as e** is the method to be used for this situation.

The **addReservation** method is used to insert a new reservation into the reservation table. The parameters: **customer_id, reservation_time, number_of_guests, and special_requests** are brought over from the sql database. Also, the parameters are used in the method to connect and carry out the printed insert statement to add the reservation. The **getAllReservation** method is used to recover everything from the reservations table. Also, the function of **getAllReservation** method is to connect to the database and get all records from reservations and returns them back. For **addcustomer** method, it gets all the information from the dining preferences for the customer from the table. Lastly, **getCustomerPreferences** method, it connects the database and shows the select statement to get the dinning preferences info of the customer.

Restaurant server.py

In the python restaurant server, the code starts off with **_init_ (initialization**) and it's used to start off the database interactions. It uses **Restaurant Database** and **BaseHTTPRequestHandler** to start the process of the code. For **do_post** method, it starts off by checking if the request path is addReservation. Afterwards, form = **cgi.FieldStorage** is used to get form data from Post requests. For **do_get** method, it contains **self.path** which checks the request path. The **getAllReservations** method, it recovers all reservations records which is why it starts with records = self.database.getallReservations(). Lastly, for **run** function, **HTTPServer** and **RestaurantPortalHandler** are the parameters. Theses parameters are linked to the port number (8000), which is used to connect your server to the database. HTTPServer is created to combine everything to the 8000 port or preferred number.

https://github.com/Kwajo21/Cis-344-final-project.git