Mark Santiago

Final Project Restaurant Reservation Report

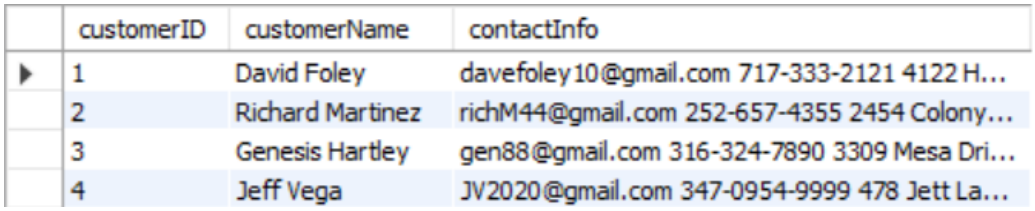
MySQL:

The first thing that needs to be done is to create the database. I did this by creating it in MySQL and creating tables that could showcase what the database would perform. Each table had its attributes and some primary and/or foreign keys. For example, in the Customers table the 'customerID' is the primary key. It would also be a foreign key in the Reservations table and the DiningPreferences table. Those 2 tables would also have a primary key being the 'reservationID' and 'preferenceID' respectively. To populate the tables I used the 'insert into' "*tablename"* values() where I added 4 rows of data.

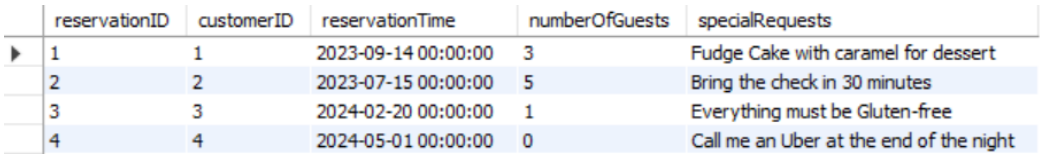
Stored Procedures:

Then I created the procedures that would manipulate the data of the tables or eliminate some data from the tables. 5 procedures were created called findReservations(), addSpecialRequest(), addReservation(), deleteReservation(), and searchPreferences(). In the findReservations procedure, you can find all the reservations of one customer based on their customerID. In the addSpecialRequests procedure, I used a parameter to indicate that based on a specific customer's reservationID, they would be able to make a new special request. In the deleteReservation procedure, a reservation can be deleted from the Reservations table based on the customer's customerID. In the procedure searchPreferences the preference of the customer can be found based on their customerID as well. Lastly, is the addReservation procedure. This procedure took me 7 attempts to get right [reason it's called addReservation7 in mysql] and it is updating multiple tables at once. To make it work for ME I had to use IF THEN ELSE within my procedure along with multiple parameters with unique nicknames. This procedure allows the user to add a new customer into these 3 tables to datatypes that are not null. Meaning that it asks for some sort of input from all attributes that have a not null in them. And if it satisfies the IF THEN statement it adds a new customer. If not then ELSE returns that the customer already exists.

Tables Populated screenshots:



customerID	customerName	contactInfo
1	David Foley	davefoley10@gmail.com 717-333-2121 4122 H...
2	Richard Martinez	richM44@gmail.com 252-657-4355 2454 Colony...
3	Genesis Hartley	gen88@gmail.com 316-324-7890 3309 Mesa Dri...
4	Jeff Vega	JV2020@gmail.com 347-0954-9999 478 Jett La...



reservationID	customerID	reservationTime	numberOfGuests	specialRequests
1	1	2023-09-14 00:00:00	3	Fudge Cake with caramel for dessert
2	2	2023-07-15 00:00:00	5	Bring the check in 30 minutes
3	3	2024-02-20 00:00:00	1	Everything must be Gluten-free
4	4	2024-05-01 00:00:00	0	Call me an Uber at the end of the night

	preferenceID	customerID	favoriteTable	dietaryRestrictions
▶	1	1	Round table facing each other	None
	2	2	Rectangular glass table	Vegetarian
	3	3	Outside table	Gluten-free diet
	4	4	Booth table	None

Stored Procedures Working:

findReservation:

```
55    delimiter //
56 •  create procedure findReservations(in id int)
57    begin
58        select *
59        from Reservations
60        where customerID=id;
61    end //
62    delimiter ;
63 •  call findReservations(2);
64
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	reservationID	customerID	reservationTime	numberOfGuests	specialRequests
▶	2	2	2023-07-15 00:00:00	5	Bring the check in 30 minutes
	7	2	2023-07-15 00:00:00	5	3 Bottles of Pinot Noir wine

addSpecialRequests:

```
65    delimiter //
66 •  create procedure addSpecialRequest(in res_id int, in new_req varchar(200))
67    begin
68        update Reservations
69        set specialRequests=new_req
70        where reservationID=res_id;
71    end //
72    delimiter ;
73 •  call addSpecialRequest(1, "Bring breadsticks every 20 minutes");
74
75    delimiter //
76 •  create procedure addReservation7(in cust_id int, in cust_name varchar(45), i
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

	reservationID	customerID	reservationTime	numberOfGuests	specialRequests
	1	1	2023-09-14 00:00:00	3	Bring breadsticks every 20 minutes

addReservation7:

```
78      if not exists(select 1 from Reservations where customerID=cust_id)then
79          insert into Customers(customerID, customerName)
80          values(cust_id, cust_name);
81          insert into Reservations(reservationID, customerID, reservationTime, numberOfGuests)
82          values(res_id, cust_id, res_date, num_gst);
83          insert into DiningPreferences(preferenceID, customerID)
84          values(pref_id, cust_id);
85
86      else
87      select 'Customer already has a reservation.';
88
89      end if;
90
91  end //
92  delimiter ;
93  call addReservation7(5, "James Franko", 5, '2024-05-13', 2, 5);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customerID	customerName	contactInfo
1	David Foley	davefoley10@gmail.com 717-333-2121 4122 H...
2	Richard Martinez	richM44@gmail.com 252-657-4355 2454 Colony...
3	Genesis Hartley	gen88@gmail.com 316-324-7890 3309 Mesa Dri...
4	Jeff Vega	JV2020@gmail.com 347-0954-9999 478 Jett La...
5	James Franko	NULL

deleteReservation:

```
96  create procedure deleteReservation(in cus_id int)
97  begin
98      delete from Reservations
99      where customerID=cus_id;
100 end //
101 delimiter ;
102 call deleteReservation(5);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

reservationID	customerID	reservationTime	numberOfGuests	specialRequests
1	1	2023-09-14 00:00:00	3	Bring breadsticks every 20 minutes
2	2	2023-07-15 00:00:00	5	Bring the check in 30 minutes
3	3	2024-02-20 00:00:00	1	Everything must be Gluten-free
4	4	2024-05-01 00:00:00	0	Call me an Uber at the end of the night
6	1	2023-09-14 00:00:00	3	Fudge Cake with caramel for dessert

searchPreferences:

```
105 •  create procedure searchPreferences(in c_id int)
106    begin
107        select *
108        from DiningPreferences
109        where customerID=c_id;
110    end //
111    delimiter ;
112 •  call searchPreferences(2);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

preferenceID	customerID	favoriteTable	dietaryRestrictions
2	2	Rectangular glass table	Vegetarian

Python:

The first thing I had to do was make sure that all my information and data matched up with what I put into Python. Unfortunately, all I was able to get was the initial connect and front demo page. The database and its location are accurately stated in the database.py and server.py which in turn used some data values from my tables in SQL. However, from there on out nothing else translates. After interacting with the other methods/procedures in the nav bar it states that "localhost didn't send any data". I thought my understanding of Python and what it was asking was clear, but it seems that is not the case. In the 3 days working on it, I've tried doing multiple things to try to get it to work but it still gives me the same result. Maybe it is something very simple that I am missing and not seeing but I can't seem to figure it out.

Restaurant Portal

Home| Add Reservation| View Reservations

All Reservations

Reservation ID	Customer ID	Reservation Time	Number of Guests	Special Requests
1	1	2023-09-14 00:00:00	3	Bring breadsticks every 20 minutes
2	2	2023-07-15 00:00:00	5	Bring the check in 30 minutes
3	3	2024-02-20 00:00:00	1	Everything must be Gluten-free
4	4	2024-05-01 00:00:00	0	Call me an Uber at the end of the night

This page isn't working

localhost didn't send any data.

ERR_EMPTY_RESPONSE

Reload

However, I think the other procedures in MySQL translated well into Python. Again, I'm not sure if I implemented them correctly but here are the other methods in both the database.py and server.py. I simply used the code written beforehand as a guide or skeleton to structure them.

```
# Add more methods as needed for restaurant operations
def addSpecialRequest(self, reservationID, specialRequests):
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = "UPDATE Reservations set specialRequests = new_req WHERE reservationID = %s"
        self.cursor.execute(query, (reservationID, specialRequests))
        self.connection.commit()
        print("New Special request added successfully")
        return


def findReservations(self, customerID):
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = "SELECT * FROM Reservations WHERE customerID = %s"
        self.cursor.execute(query, (customerID))
        records = self.cursor.fetchall()
        return records


def deleteReservation(self, customerID):
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        self.cursor.callproc("deleteReservation")
        return


def searchPreferences(self, customerID):
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        self.cursor.callproc("searchPreferences")
        return
```

```
if self.path =='/addSpecialRequest':
    return

if self.path =='findReservations':
    return

if self.path =='deleteReservation':
    return

if self.path =='searchPreferences':
    return
```

GitHub Repository Link:
https://github.com/Mark-SantiagoLC/CIS344-FinalProject