MGK
Hotel Resource Management System

By Anthony **M**essana, Michael **G**iordano, and Leah **K**azenmayer

Plan

- Hotel Management Resource System
  - Keep records of all the customers, employees, inventory, and so forth to make sure that the hotel staff focus on their task at hand, and worry less about the data retrieval
  - We decided to organize all of our thoughts and brainstorms by creating the EER diagram for the management system, formulating the Relational Schema based on that, and normalizing the relation

EER Diagram

https://app.lucidchart.com/documents/edit/6a40e6d7-c9d8-407f-96f5-0259b335fec3/0_0

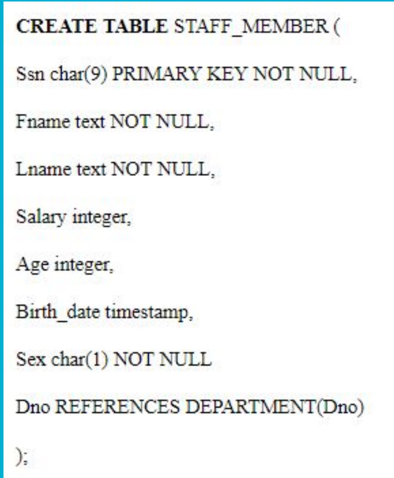
Relational Schema

https://app.lucidchart.com/documents/edit/fa8f112a-11ff-47d9-bfed-8da88f4fadc8/0_0

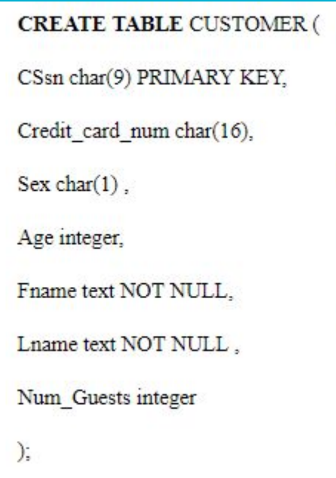
BCNF

https://app.lucidchart.com/documents/edit/fa7a8364-ba39-4c1c-a7cb-f863195da12e/0_0

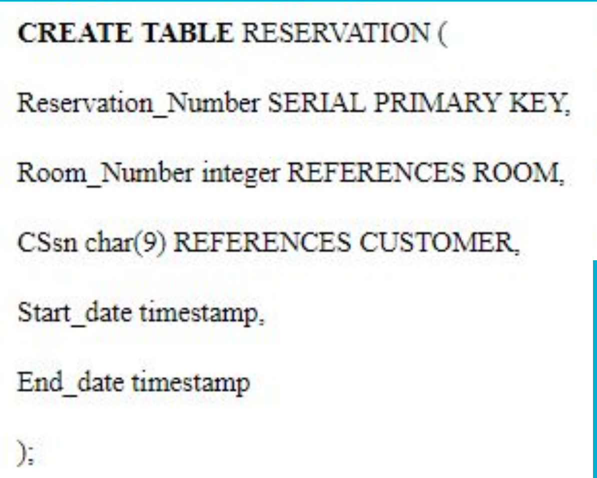
Queries (Create Table)



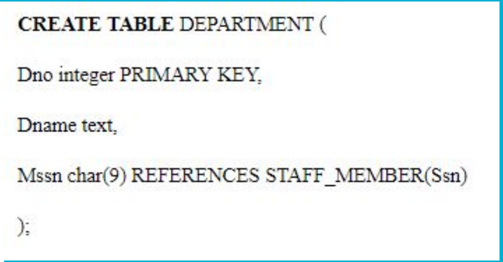
```
CREATE TABLE STAFF_MEMBER (

Ssn char(9) PRIMARY KEY NOT NULL,

Fname text NOT NULL,

Lname text NOT NULL,

Salary integer,

Age integer,

Birth_date timestamp,

Sex char(1) NOT NULL

Dno REFERENCES DEPARTMENT(Dno)

);
```



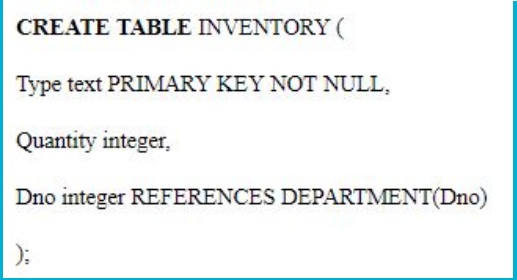
```
CREATE TABLE CUSTOMER (

CSsn char(9) PRIMARY KEY,

Credit_card_num char(16),

Sex char(1) ,

Age integer,

Fname text NOT NULL,

Lname text NOT NULL ,

Num_Guests integer

);
```



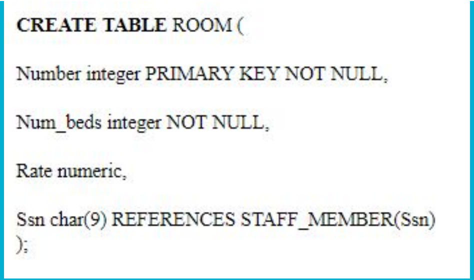
```
CREATE TABLE RESERVATION (

Reservation_Number SERIAL PRIMARY KEY,

Room_Number integer REFERENCES ROOM,

CSsn char(9) REFERENCES CUSTOMER,

Start_date timestamp,

End_date timestamp

);
```



```
CREATE TABLE DEPARTMENT (

Dno integer PRIMARY KEY,

Dname text,

Mssn char(9) REFERENCES STAFF_MEMBER(Ssn)

);
```

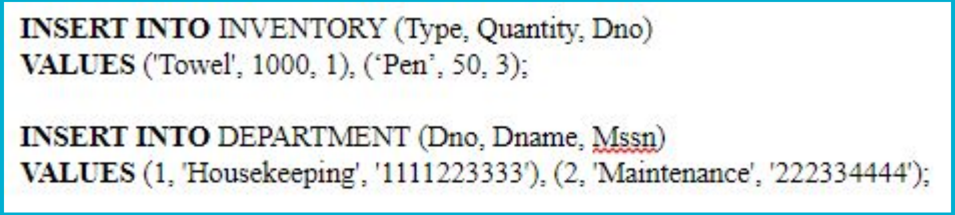


```
CREATE TABLE INVENTORY (

Type text PRIMARY KEY NOT NULL,

Quantity integer,

Dno integer REFERENCES DEPARTMENT(Dno)

);
```

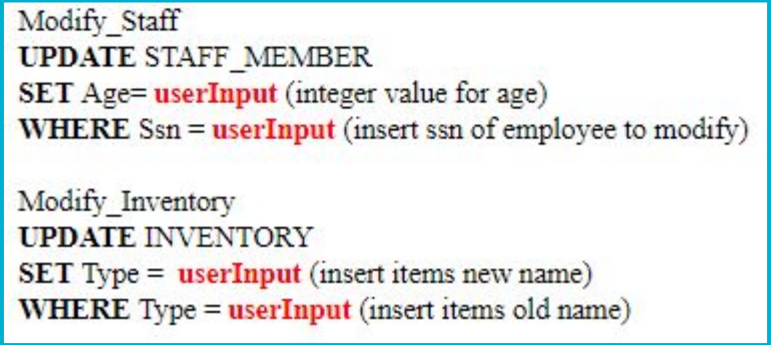


```
CREATE TABLE ROOM (

Number integer PRIMARY KEY NOT NULL,

Num_beds integer NOT NULL,

Rate numeric,

Ssn char(9) REFERENCES STAFF_MEMBER(Ssn)
);
```

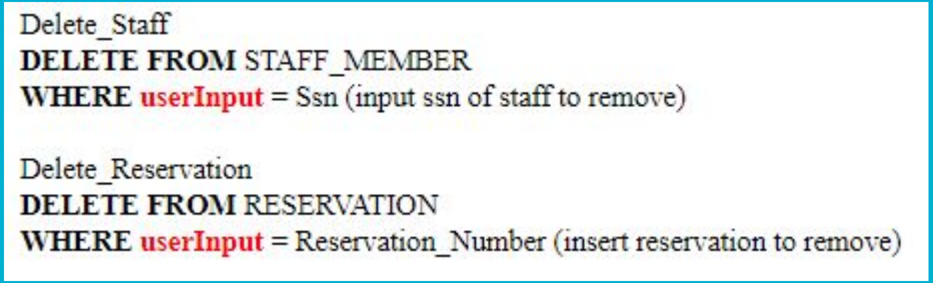
Queries (Insert/Update/Delete)



```
INSERT INTO INVENTORY (Type, Quantity, Dno)
VALUES ('Towel', 1000, 1), ('Pen', 50, 3);

INSERT INTO DEPARTMENT (Dno, Dname, Mssn)
VALUES (1, 'Housekeeping', '1111223333'), (2, 'Maintenance', '222334444');
```

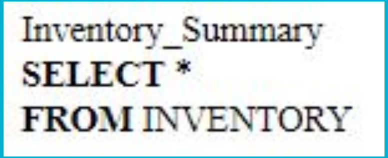


```
Modify_Staff
UPDATE STAFF_MEMBER
SET Age= userInput (integer value for age)
WHERE Ssn = userInput (insert ssn of employee to modify)

Modify_Inventory
UPDATE INVENTORY
SET Type =  userInput (insert items new name)
WHERE Type = userInput (insert items old name)
```

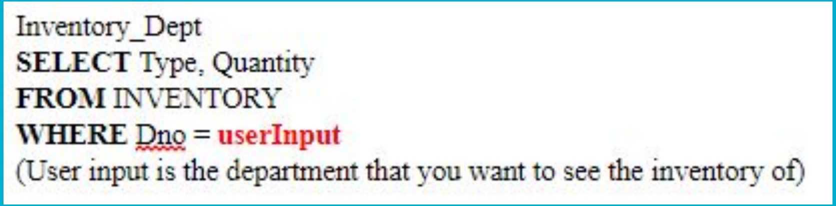


```
Delete_Staff
DELETE FROM STAFF_MEMBER
WHERE userInput = Ssn (input ssn of staff to remove)

Delete_Reservation
DELETE FROM RESERVATION
WHERE userInput = Reservation_Number (insert reservation to remove)
```

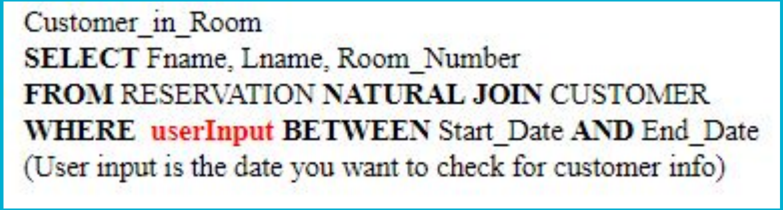
Queries (Select)



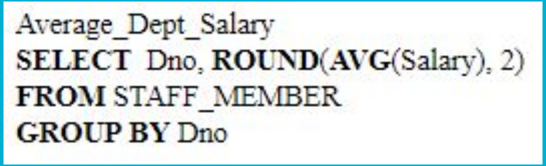
```
Inventory_Summary
SELECT *
FROM INVENTORY
```



```
Inventory_Dept
SELECT Type, Quantity
FROM INVENTORY
WHERE Dno = userInput
(User input is the department that you want to see the inventory of)
```

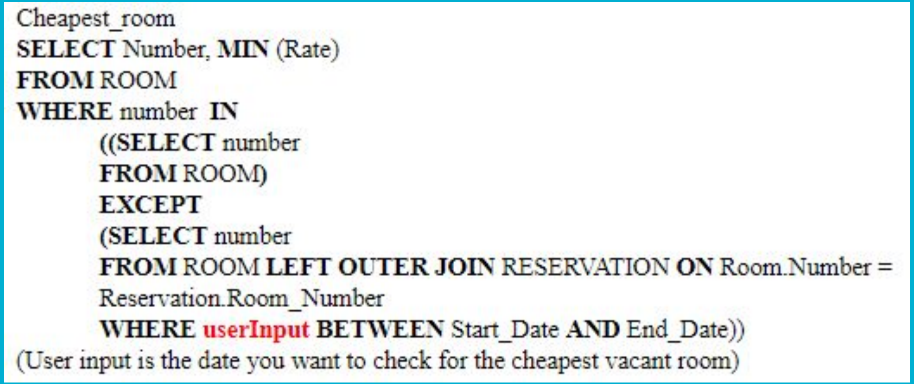


```
Customer_in_Room
SELECT Fname, Lname, Room_Number
FROM RESERVATION NATURAL JOIN CUSTOMER
WHERE  userInput BETWEEN Start_Date AND End_Date
(User input is the date you want to check for customer info)
```

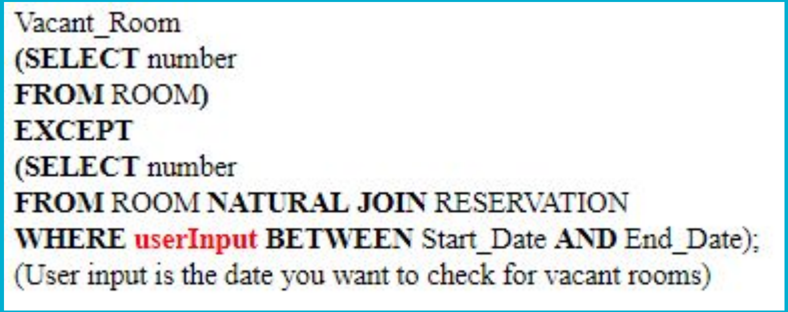


```
Average_Dept_Salary
SELECT  Dno, ROUND(AVG(Salary), 2)
FROM STAFF_MEMBER
GROUP BY Dno
```

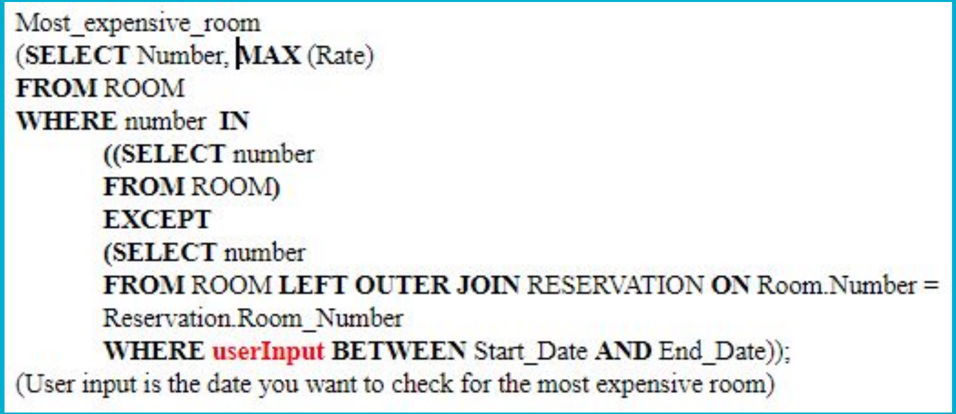
Queries (Select)



```
Cheapest_room
SELECT Number, MIN (Rate)
FROM ROOM
WHERE number  IN
          ((SELECT number
          FROM ROOM)
          EXCEPT
          (SELECT number
          FROM ROOM LEFT OUTER JOIN RESERVATION ON Room.Number =
          Reservation.Room_Number
          WHERE userInput BETWEEN Start_Date AND End_Date))
(User input is the date you want to check for the cheapest vacant room)
```



```
Vacant_Room
(SELECT number
FROM ROOM)
EXCEPT
(SELECT number
FROM ROOM NATURAL JOIN RESERVATION
WHERE userInput BETWEEN Start_Date AND End_Date);
(User input is the date you want to check for vacant rooms)
```



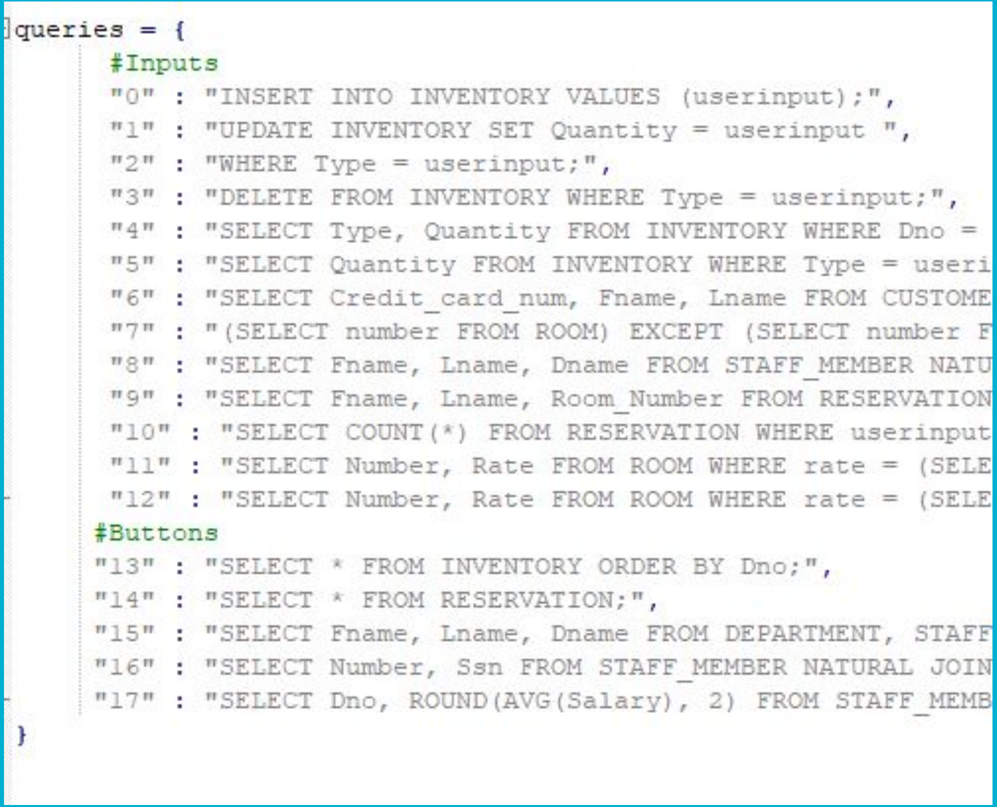
```
Most_expensive_room
(SELECT Number, MAX (Rate)
FROM ROOM
WHERE number  IN
          ((SELECT number
          FROM ROOM)
          EXCEPT
          (SELECT number
          FROM ROOM LEFT OUTER JOIN RESERVATION ON Room.Number =
          Reservation.Room_Number
          WHERE userInput BETWEEN Start_Date AND End_Date));
(User input is the date you want to check for the most expensive room)
```

Implementation

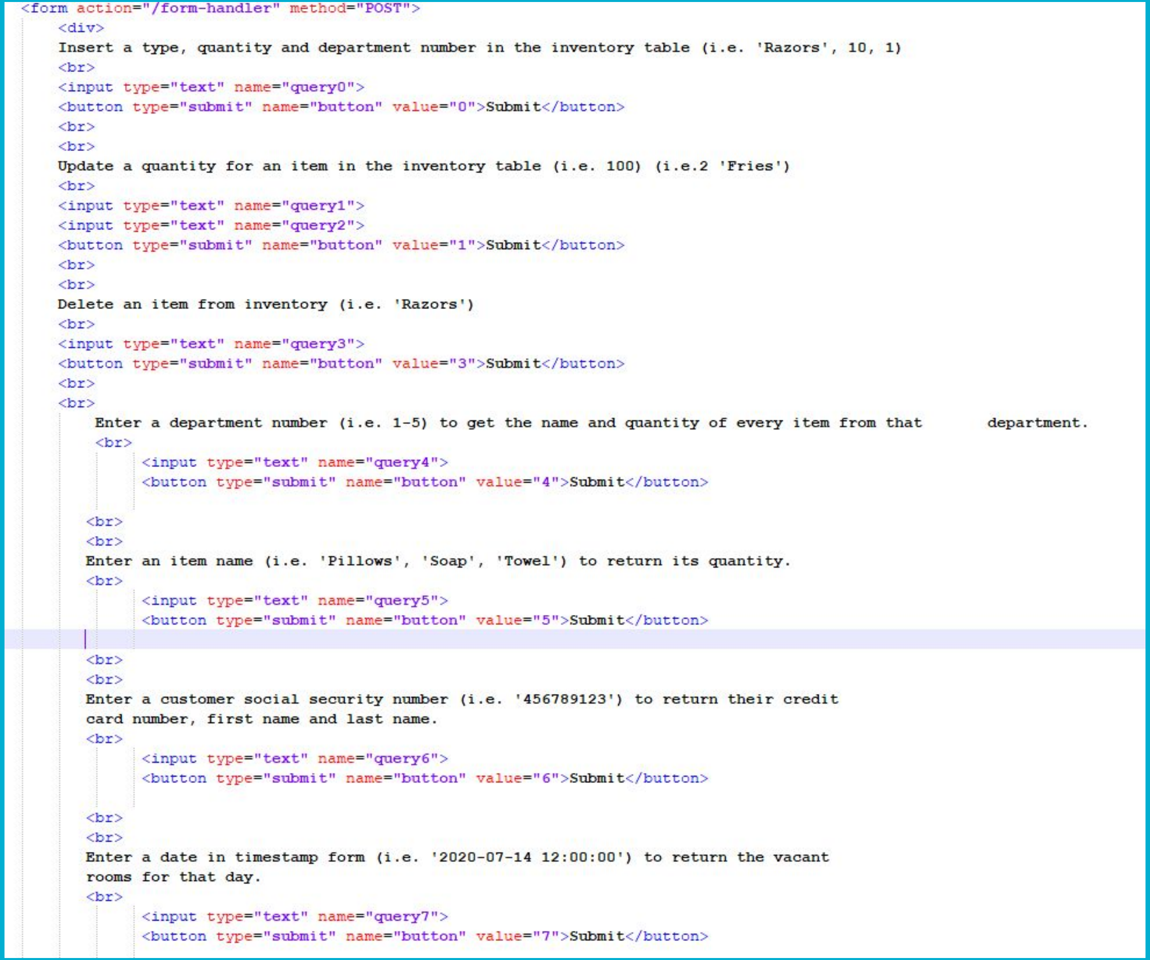
- Tools
  - Python programming language
  - PostgreSQL, an open source DDL for the database
  - Psycopg, a PostgreSQL database adapter
  - **Flask**, a web application framework for python

Implementation(cont.)

- The Code:
  - Each SQL query was stored in a dictionary. In the key value pair, a numerical string was used as a key and the SQL query as a string was the value.
  - In the HTML, each button returns a string value that corresponded to a key in the dictionary.



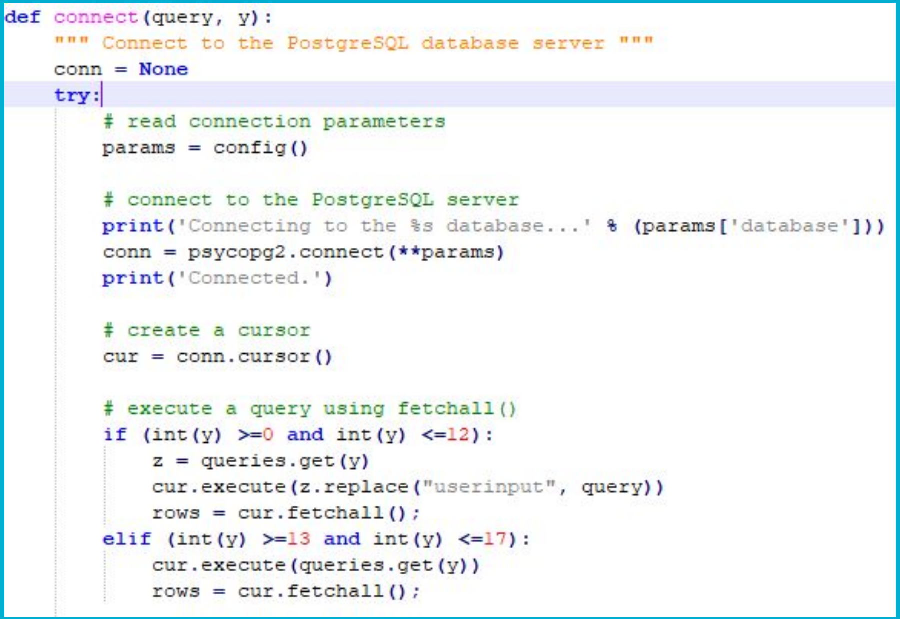
```
queries = {
        #Inputs
        "0" : "INSERT INTO INVENTORY VALUES (userinput);",
        "1" : "UPDATE INVENTORY SET Quantity = userinput ",
        "2" : "WHERE Type = userinput;",
        "3" : "DELETE FROM INVENTORY WHERE Type = userinput;",
        "4" : "SELECT Type, Quantity FROM INVENTORY WHERE Dno =
        "5" : "SELECT Quantity FROM INVENTORY WHERE Type = useri
        "6" : "SELECT Credit_card_num, Fname, Lname FROM CUSTOME
        "7" : "(SELECT number FROM ROOM) EXCEPT (SELECT number F
        "8" : "SELECT Fname, Lname, Dname FROM STAFF_MEMBER NATU
        "9" : "SELECT Fname, Lname, Room_Number FROM RESERVATION
        "10" : "SELECT COUNT(*) FROM RESERVATION WHERE userinput
        "11" : "SELECT Number, Rate FROM ROOM WHERE rate = (SELE
        "12" : "SELECT Number, Rate FROM ROOM WHERE rate = (SELE
       #Buttons
       "13" : "SELECT * FROM INVENTORY ORDER BY Dno;",
       "14" : "SELECT * FROM RESERVATION;",
       "15" : "SELECT Fname, Lname, Dname FROM DEPARTMENT, STAFF
       "16" : "SELECT Number, Ssn FROM STAFF_MEMBER NATURAL JOIN
       "17" : "SELECT Dno, ROUND(AVG(Salary), 2) FROM STAFF_MEMB
}
```



```
<form action="/form-handler" method="POST">
    <div>
    Insert a type, quantity and department number in the inventory table (i.e. 'Razors', 10, 1)
    <br>
    <input type="text" name="query0">
    <button type="submit" name="button" value="0">Submit</button>
    <br>
    <br>
    Update a quantity for an item in the inventory table (i.e. 100) (i.e.2 'Fries')
    <br>
    <input type="text" name="query1">
    <input type="text" name="query2">
    <button type="submit" name="button" value="1">Submit</button>
    <br>
    <br>
    Delete an item from inventory (i.e. 'Razors')
    <br>
    <input type="text" name="query3">
    <button type="submit" name="button" value="3">Submit</button>
    <br>
    <br>
        Enter a department number (i.e. 1-5) to get the name and quantity of every item from that       department.
        <br>
             <input type="text" name="query4">
             <button type="submit" name="button" value="4">Submit</button>

       <br>
       <br>
       Enter an item name (i.e. 'Pillows', 'Soap', 'Towel') to return its quantity.
       <br>
             <input type="text" name="query5">
             <button type="submit" name="button" value="5">Submit</button>

       <br>
       <br>
       Enter a customer social security number (i.e. '456789123') to return their credit
       card number, first name and last name.
       <br>
             <input type="text" name="query6">
             <button type="submit" name="button" value="6">Submit</button>

       <br>
       <br>
       Enter a date in timestamp form (i.e. '2020-07-14 12:00:00') to return the vacant
       rooms for that day.
       <br>
             <input type="text" name="query7">
             <button type="submit" name="button" value="7">Submit</button>
```

Implementation(cont.)

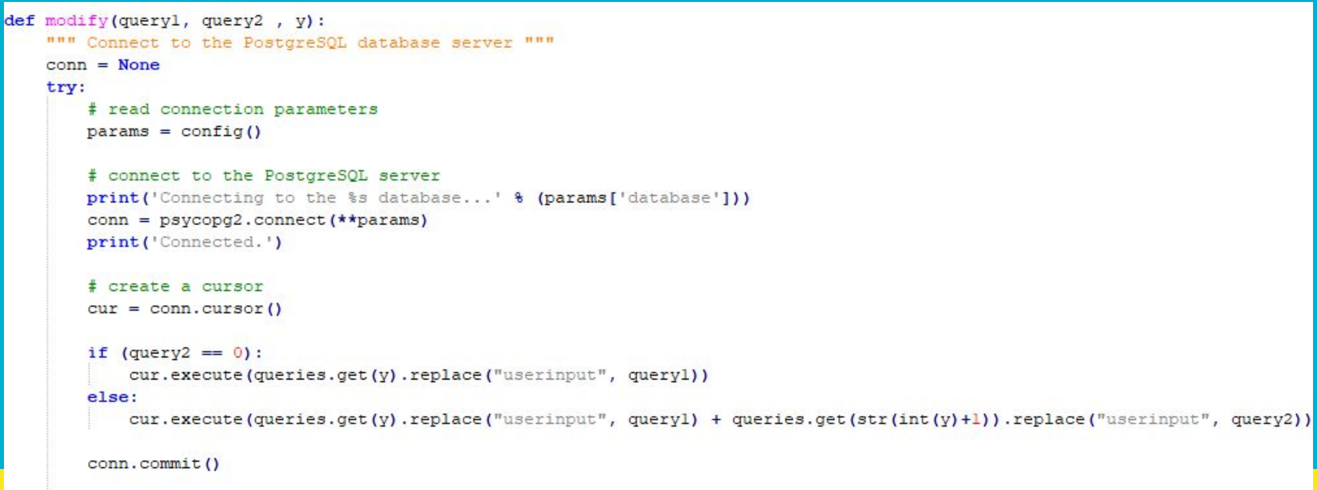
- Connect Function:
    - Took two parameters, one represents the input in the textbox form the user and the other the key value returned by a button press in the HTML code.
    - A combination of these values would be used to create the full SQL query.
    - Some queries did not need user input and were accomplished solely by a button press.



```python
def connect(query, y):
    """ Connect to the PostgreSQL database server """
    conn = None
    try:
        # read connection parameters
        params = config()

        # connect to the PostgreSQL server
        print('Connecting to the %s database...' % (params['database']))
        conn = psycopg2.connect(**params)
        print('Connected.')

        # create a cursor
        cur = conn.cursor()

        # execute a query using fetchall()
        if (int(y) >=0 and int(y) <=12):
            z = queries.get(y)
            cur.execute(z.replace("userinput", query))
            rows = cur.fetchall();
        elif (int(y) >=13 and int(y) <=17):
            cur.execute(queries.get(y))
            rows = cur.fetchall();
```

Implementation(cont.)

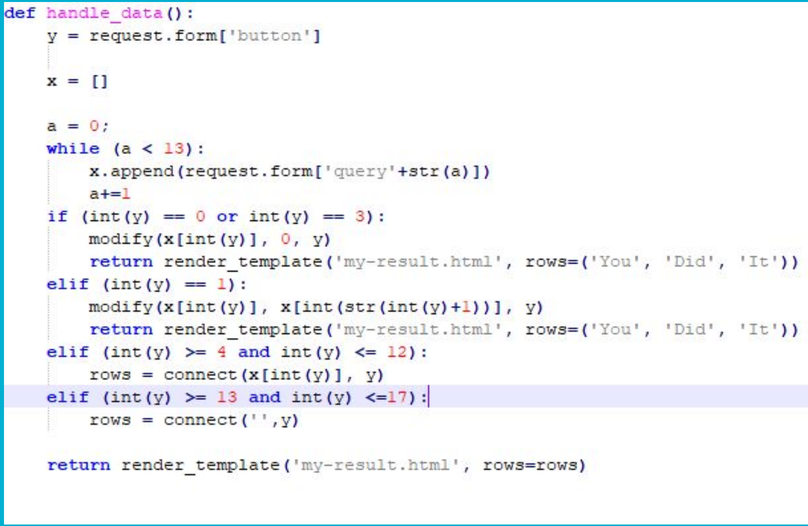
- Modify Function
    - Works in a similar fashion to the connect. A third parameter is added to specifically handle updates, which need 2 inputs at once rather than one.
    - This function also will not return a row variable. Instead row is handled in the handle_data function, and it outputs a message confirming that the transaction committed.



```
def modify(query1, query2 , y):
    """ Connect to the PostgreSQL database server """
    conn = None
    try:
        # read connection parameters
        params = config()

        # connect to the PostgreSQL server
        print('Connecting to the %s database...' % (params['database']))
        conn = psycopg2.connect(**params)
        print('Connected.')

        # create a cursor
        cur = conn.cursor()

        if (query2 == 0):
            cur.execute(queries.get(y).replace("userinput", query1))
        else:
            cur.execute(queries.get(y).replace("userinput", query1) + queries.get(str(int(y)+1)).replace("userinput", query2))

        conn.commit()
```

Implementation(cont.)

- Handle_Data Function:
    - Values of all the textbox inputs were stored in a list, while the value of the button press was stored in a single variable.
    - The index of each input stored in the list also correlates with its key value and the value returned by its submit button. This ensures that each button press executes the correct query on its corresponding input from the textbox.
    - IF ELIF statement determines based on the key value whether the query was a single button press, a select user input query, or an insert/modify/delete. If a query had no user input, then instead of passing a textbox input from the list, a blank string and the key value were passed.



```
def handle_data():
    y = request.form['button']

    x = []

    a = 0;
    while (a < 13):
        x.append(request.form['query'+str(a)])
        a+=1
    if (int(y) == 0 or int(y) == 3):
        modify(x[int(y)], 0, y)
        return render_template('my-result.html', rows=('You', 'Did', 'It'))
    elif (int(y) == 1):
        modify(x[int(y)], x[int(str(int(y)+1))], y)
        return render_template('my-result.html', rows=('You', 'Did', 'It'))
    elif (int(y) >= 4 and int(y) <= 12):
        rows = connect(x[int(y)], y)
    elif (int(y) >= 13 and int(y) <=17):
        rows = connect('',y)

    return render_template('my-result.html', rows=rows)
```