Retirement home management system

Mohammad Saeed Pourjafar

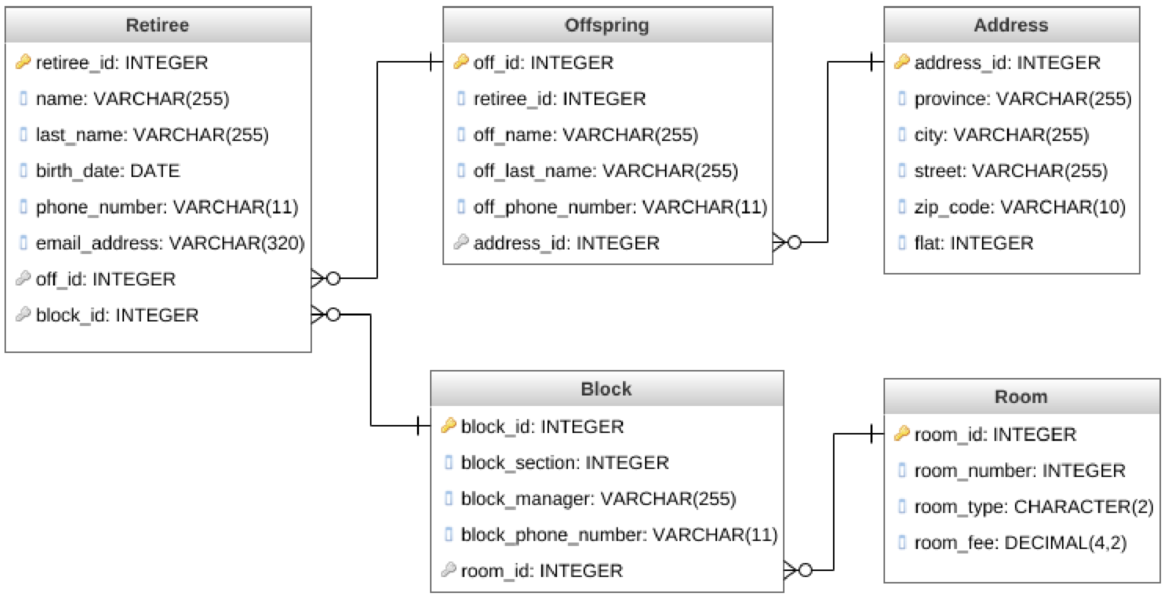
Student number: 436817

Introduction

The retirement home management system is a Graphical User Interface (GUI) for windows operating system which can be used to add, store and delete a retiree form the database. Also the administrator will be able to show the status quo of the retirees and update their information accordingly.

Database scheme

There will be 5 separate tables in the database. The diagram of the database schema is shown below.



Notes:

- Every retiree has offspring (children) and they have a corresponding addresses which has been stored in a different table.
- The type of email_address declared as VARCHAR (320) for these reasons:
    - 64 characters for the "local part" (username).
    - 1 character for the @ symbol.
    - 255 characters for the domain name.
- The type of room_type has been declared as CHARACTER (2) according to the table below and since it’s a one-value character (A,B,C,…)

```
Value    CHAR(2)   Storage Required   VARCHAR(2)   Storage Required
''       '  '      2 bytes            ''           1 byte
'a'      'a '      2 bytes            'a'          2 bytes
'ab'     'ab'      2 bytes            'ab'         3 bytes
```

- The type of room_fee has been declared as DECIMAL (4,2) which means it can take a price of 4 digits with 2 decimal points e.g. 1230.56 $

- Although the SQLite has only 5 types of datatype (Integer, Text, Blob, Real and Numeric), but in general it's a good practice to specified the exact datatype in our database.

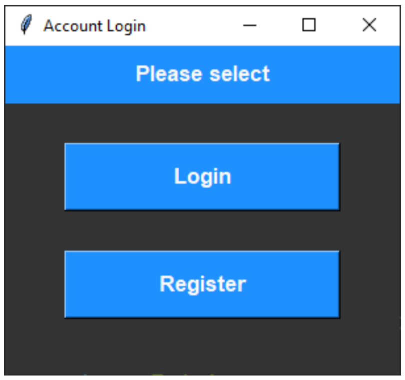
User manual

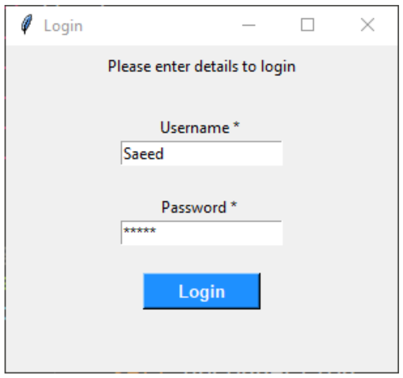
The application will be mainly used in retirement homes. Also, it can be extended and modified to any sort of application which deal with the different set of data from different segments e.g. day-care centers.

First for running the application you can either run the Retiree.ipynb file from Jupyter Notebook or you can open the file Retiree.py with Sublime text editor, hit *Ctrl+B* to run it more quickly (recommended). Next, you will have the homepage of the retirement management system which you can then interact with by the graphical user interface and each button that is associated with the specific function and will be discussed in the upcoming section.

Login and registration

Before the homepage, users must login in order to access the application. Alternatively they can register and then enter the application.

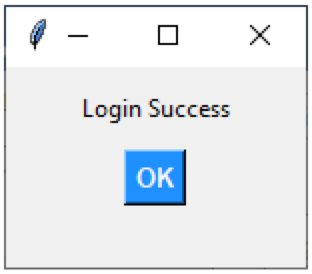


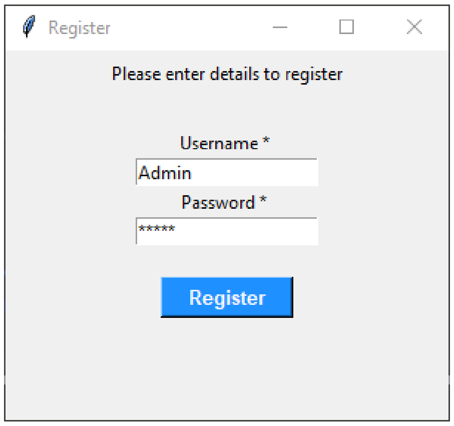

Login

Please enter details to login

Username * Saeed

Password * *****

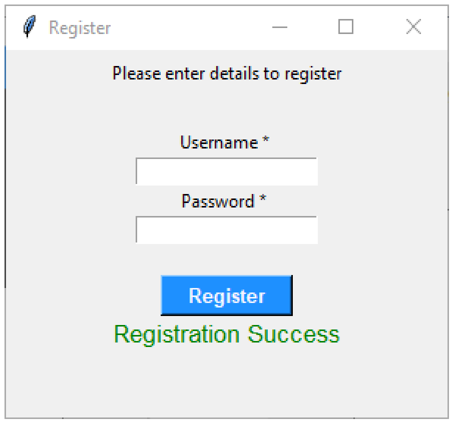

Login Success


Register

Please enter details to register

Username * Admin

Password * *****


Register

Please enter details to register

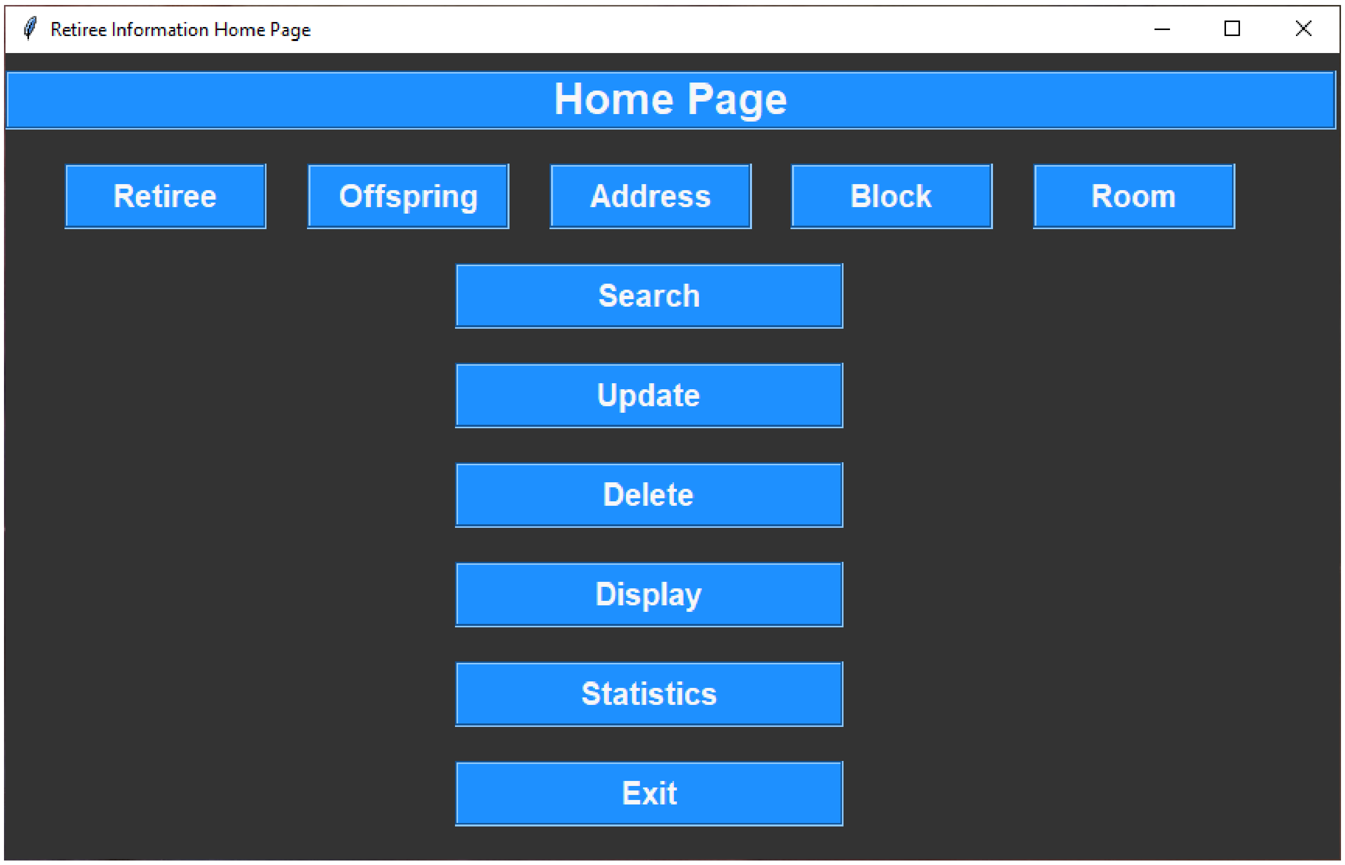
Username *

Password *

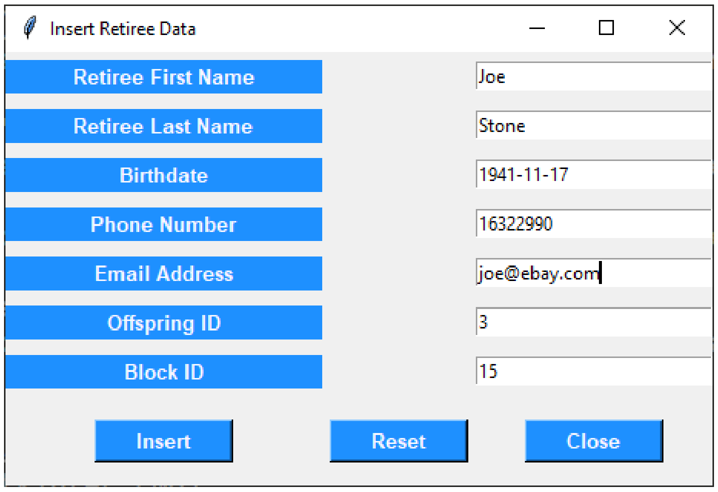
Registration Success

Screenshots

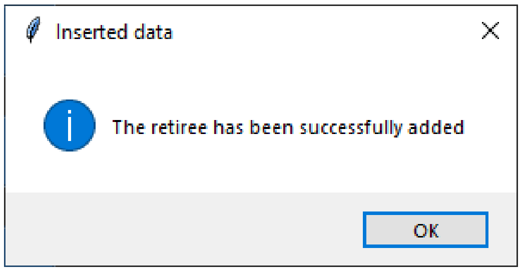
Homepage: The front page of the application



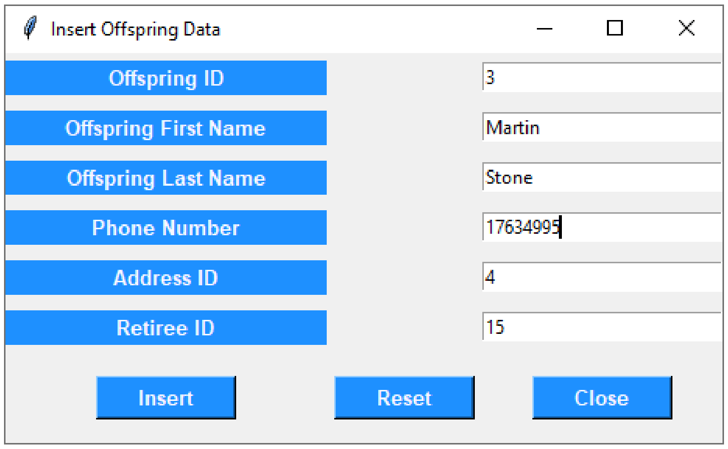
Inserts: They are designed in a separate buttons and therefore windows in order to give the user the ability to enter the information from top to bottom (Retiree to Room) or from bottom to top (Room to Retiree)


Insert Retiree Data

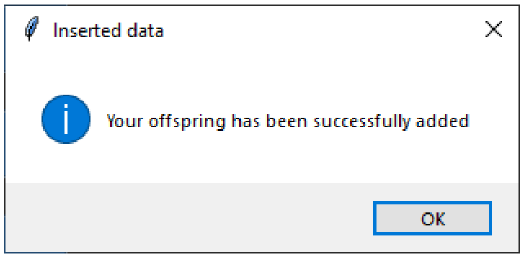
Retiree First Name	Joe
Retiree Last Name	Stone
Birthdate	1941-11-17
Phone Number	16322990
Email Address	joe@ebay.com
Offspring ID	3
Block ID	15


Inserted data

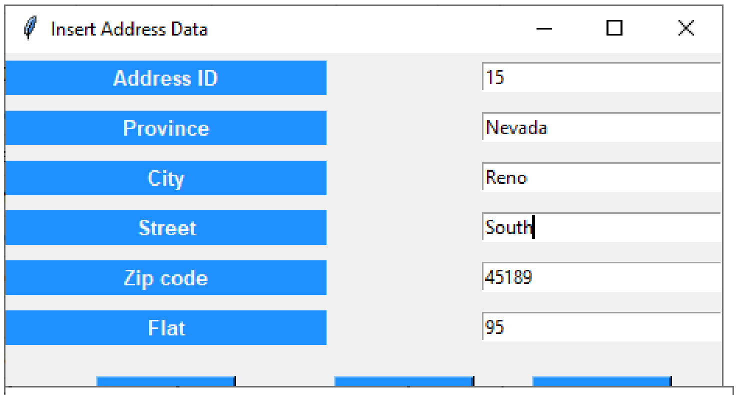
The retiree has been successfully added


Insert Offspring Data

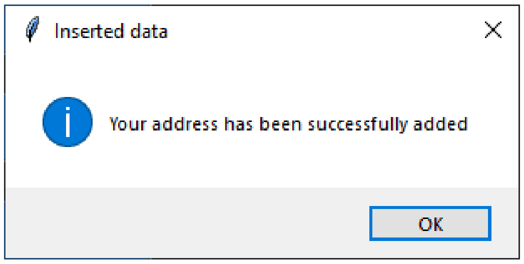
Offspring ID	3
Offspring First Name	Martin
Offspring Last Name	Stone
Phone Number	17634995
Address ID	4
Retiree ID	15


Inserted data

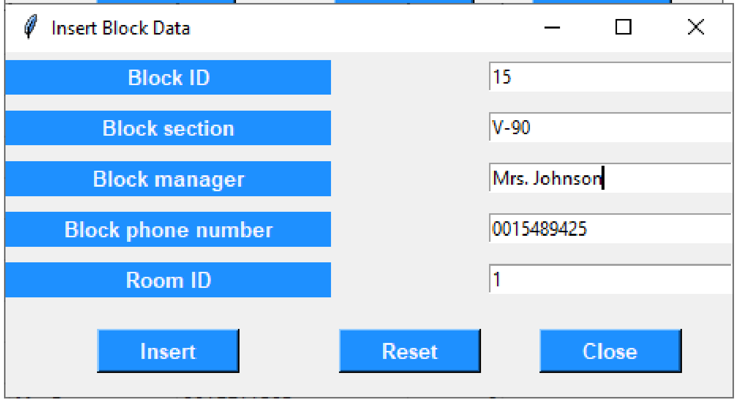
Your offspring has been successfully added


Insert Address Data

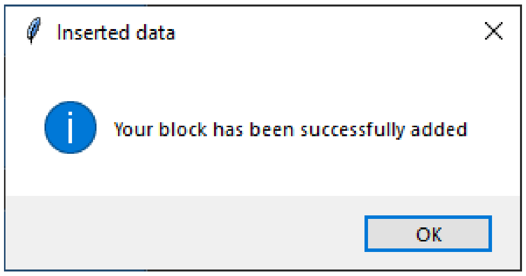
Address ID	15
Province	Nevada
City	Reno
Street	South
Zip code	45189
Flat	95


Inserted data

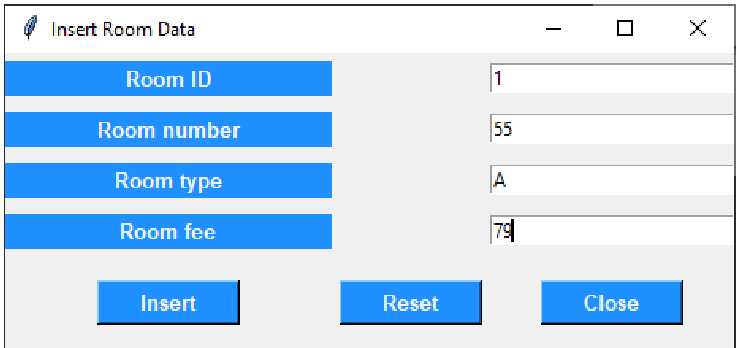
Your address has been successfully added


Insert Block Data

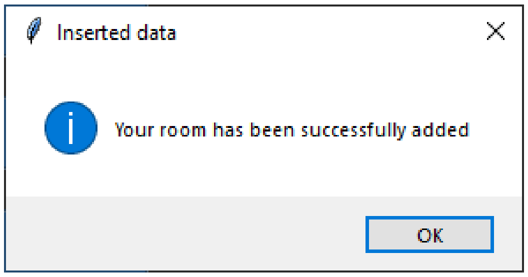
Block ID	15
Block section	V-90
Block manager	Mrs. Johnson
Block phone number	0015489425
Room ID	1


Inserted data

Your block has been successfully added

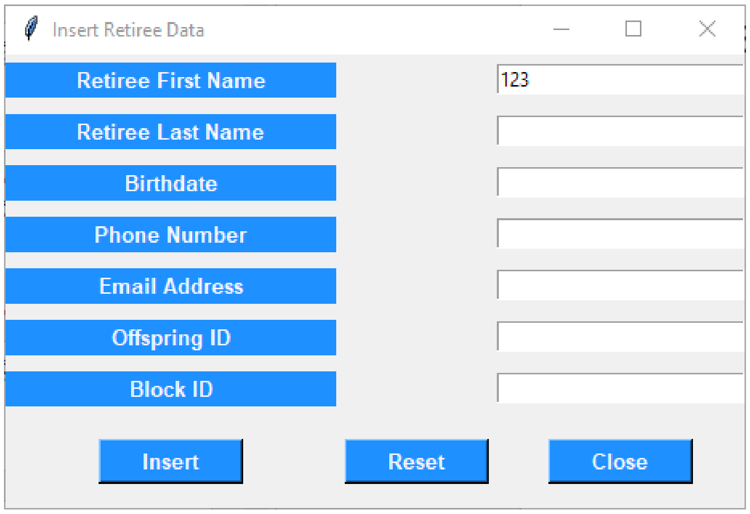

Insert Room Data

Room ID	1
Room number	55
Room type	A
Room fee	79


Inserted data

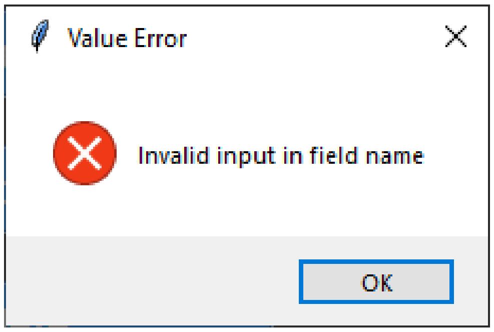
Your room has been successfully added

Inserts: Will raise value error for wrong inputs



Insert Retiree Data

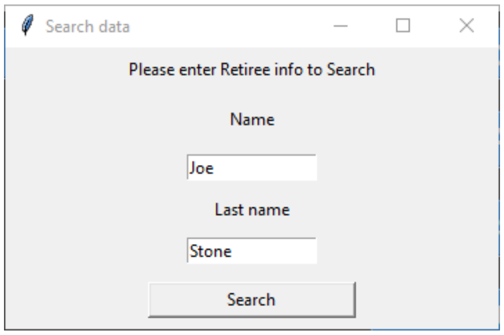
Retiree First Name	123
Retiree Last Name	
Birthdate	
Phone Number	
Email Address	
Offspring ID	
Block ID	



Value Error

Invalid input in field name

Search: By clicking on this button, we will be able to search the retirees based on their name and last name



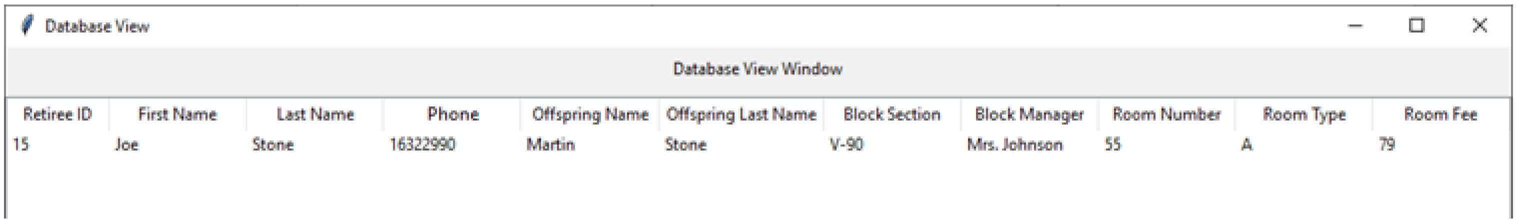
Search data

Please enter Retiree info to Search

Name: Joe

Last name: Stone

And here's the result from the retiree we had just added (Joe Stone)

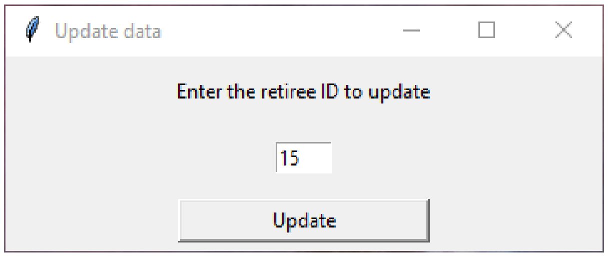


Database View

Database View Window

Retiree ID	First Name	Last Name	Phone	Offspring Name	Offspring Last Name	Block Section	Block Manager	Room Number	Room Type	Room Fee
15	Joe	Stone	16322990	Martin	Stone	V-90	Mrs. Johnson	55	A	79

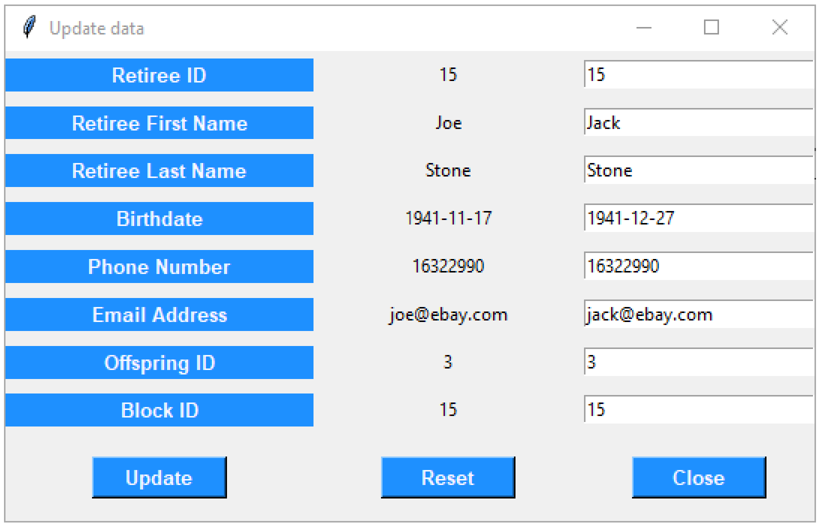
Update: Will update the information in Retiree table according to their ID. Previous values also shown in order to make it easier to compare the old and the new value.



Update data

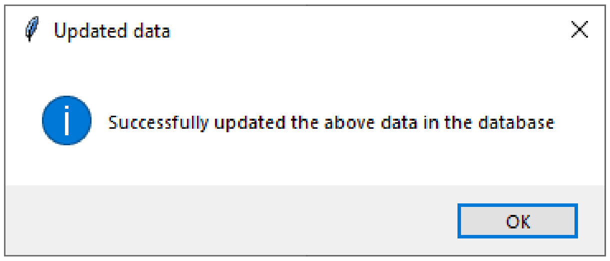
Enter the retiree ID to update

15



Update data

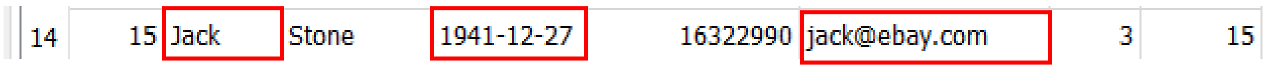
Retiree ID	15	15
Retiree First Name	Joe	Jack
Retiree Last Name	Stone	Stone
Birthdate	1941-11-17	1941-12-27
Phone Number	16322990	16322990
Email Address	joe@ebay.com	jack@ebay.com
Offspring ID	3	3
Block ID	15	15



Updated data

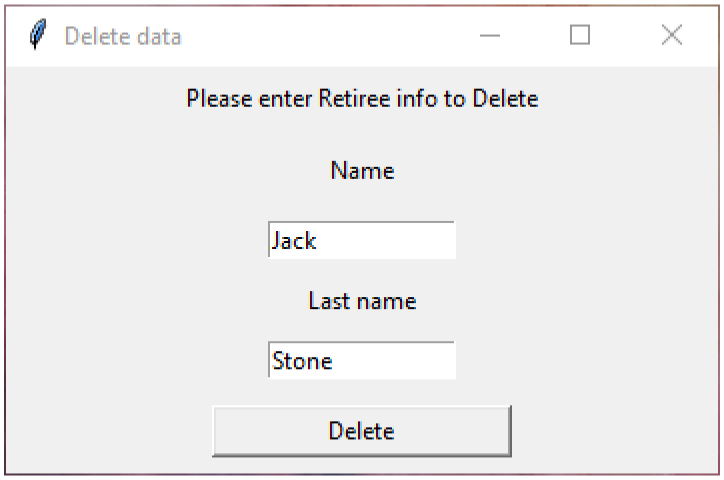
Successfully updated the above data in the database

Just to make sure if it worked:



14	15	Jack	Stone	1941-12-27	16322990	jack@ebay.com	3	15

Delete: Will delete a retirees from Retiree table based on their name and surname



Delete data

Please enter Retiree info to Delete

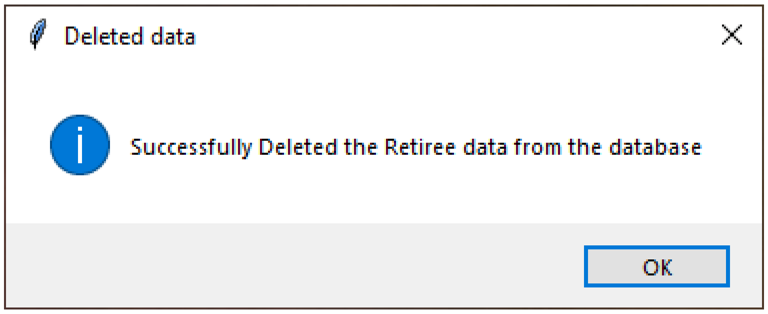
Name

Jack

Last name

Stone

Delete

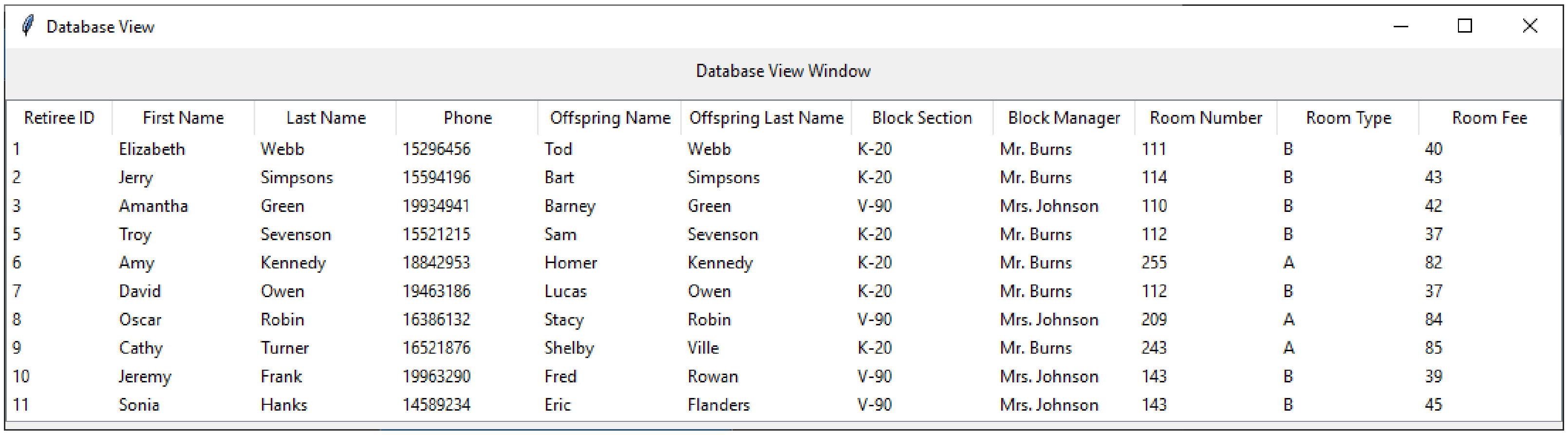


Deleted data

Successfully Deleted the Retiree data from the database

OK

Display: Will display the scrollable information about retiree ID, name, last name, phone number, offspring name, offspring last name, block section, block manager, room number, room type and room fee.



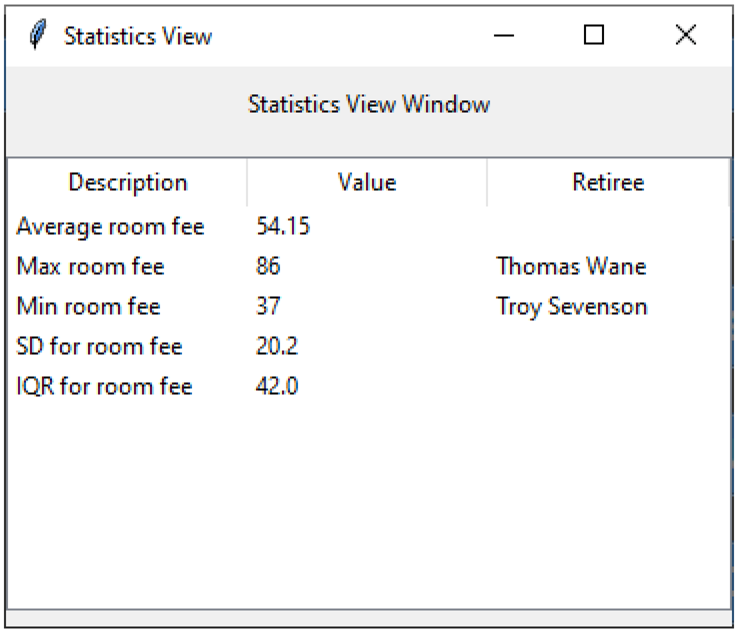
Database View

Database View Window

Retiree ID	First Name	Last Name	Phone	Offspring Name	Offspring Last Name	Block Section	Block Manager	Room Number	Room Type	Room Fee
1	Elizabeth	Webb	15296456	Tod	Webb	K-20	Mr. Burns	111	B	40
2	Jerry	Simpsons	15594196	Bart	Simpsons	K-20	Mr. Burns	114	B	43
3	Amantha	Green	19934941	Barney	Green	V-90	Mrs. Johnson	110	B	42
5	Troy	Sevenson	15521215	Sam	Sevenson	K-20	Mr. Burns	112	B	37
6	Amy	Kennedy	18842953	Homer	Kennedy	K-20	Mr. Burns	255	A	82
7	David	Owen	19463186	Lucas	Owen	K-20	Mr. Burns	112	B	37
8	Oscar	Robin	16386132	Stacy	Robin	V-90	Mrs. Johnson	209	A	84
9	Cathy	Turner	16521876	Shelby	Ville	K-20	Mr. Burns	243	A	85
10	Jeremy	Frank	19963290	Fred	Rowan	V-90	Mrs. Johnson	143	B	39
11	Sonia	Hanks	14589234	Eric	Flanders	V-90	Mrs. Johnson	143	B	45

Statistics: Will display the statistics from the database based on the room fee and return the following values:

- Average room fee
- Max/Min room fee: Which retiree paid the maximum/minimum price for his/her?
- Standard deviation for the room fee
- Inter quartile range for the room fee



Statistics View Window

Description	Value	Retiree
Average room fee	54.15	
Max room fee	86	Thomas Wane
Min room fee	37	Troy Sevenson
SD for room fee	20.2	
IQR for room fee	42.0	

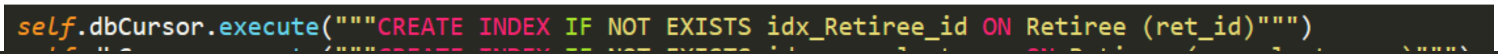
Exit: Will close the application

SQL queries

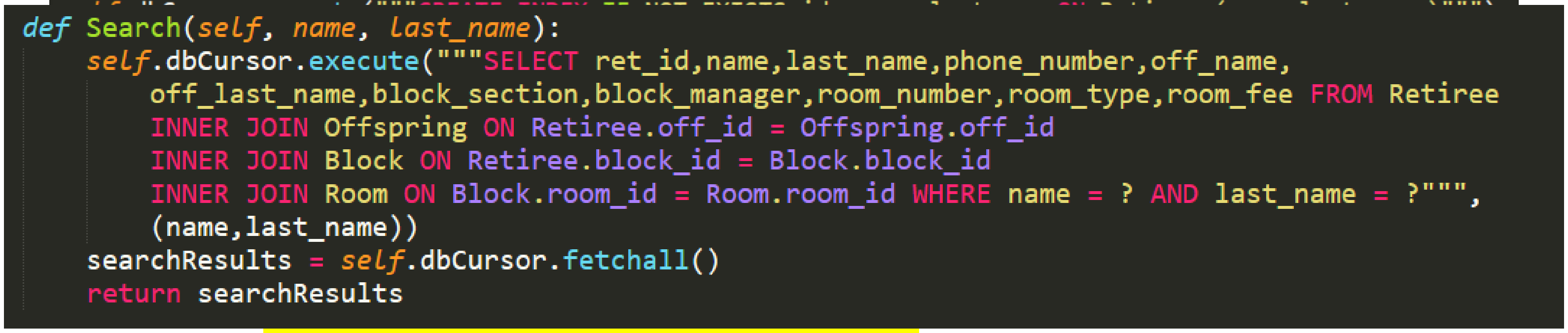
- Creating tables such as Retiree

```
self.dbCursor.execute("""CREATE TABLE IF NOT EXISTS Retiree (ret_id INTEGER PRIMARY KEY AUTOINCREMENT,
                            name TEXT NOT NULL,
                            last_name TEXT NOT NULL,
                            birth_date TEXT,
                            phone_number INTEGER,
                            email_address TEXT,
                            off_id INTEGER,
                            block_id INTEGER,
                            FOREIGN KEY (off_id)
                            REFERENCES Offspring (off_id),
                            FOREIGN KEY (block_id)
                            REFERENCES Block (block_id))""")
```

- Creating indices



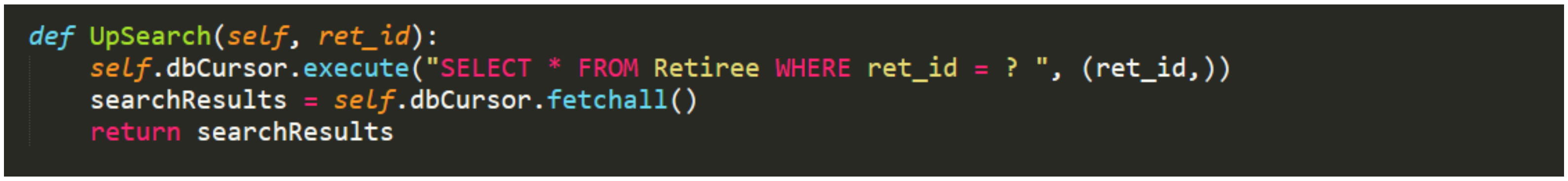
```
self.dbCursor.execute("""CREATE INDEX IF NOT EXISTS idx_Retiree_id ON Retiree (ret_id)""")
```



```
def Search(self, name, last_name):
    self.dbCursor.execute("""SELECT ret_id,name,last_name,phone_number,off_name,
        off_last_name,block_section,block_manager,room_number,room_type,room_fee FROM Retiree
        INNER JOIN Offspring ON Retiree.off_id = Offspring.off_id
        INNER JOIN Block ON Retiree.block_id = Block.block_id
        INNER JOIN Room ON Block.room_id = Room.room_id WHERE name = ? AND last_name = ?""",
        (name,last_name))
    searchResults = self.dbCursor.fetchall()
    return searchResults
```

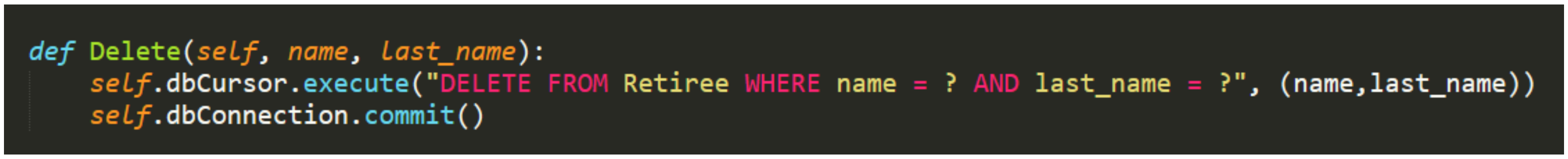
- Search function with SQL query based on inner join

- Search for Update function based on the retiree ID



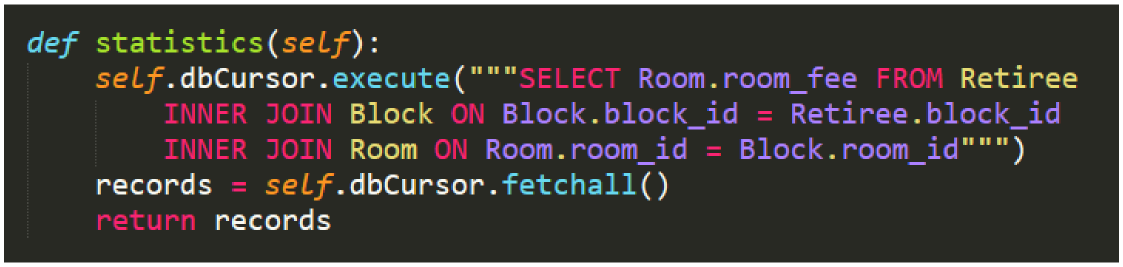
```
def UpSearch(self, ret_id):
    self.dbCursor.execute("SELECT * FROM Retiree WHERE ret_id = ? ", (ret_id,))
    searchResults = self.dbCursor.fetchall()
    return searchResults
```

- Delete function based on name and last name



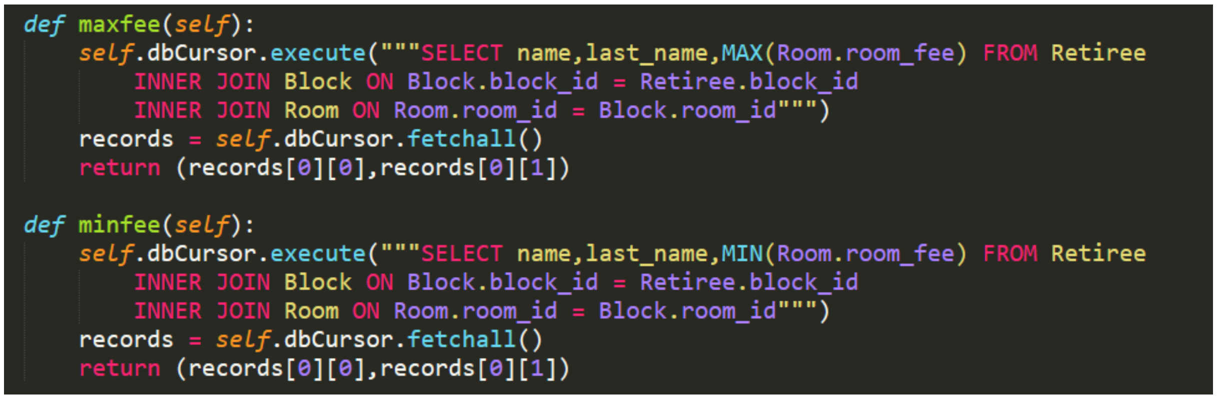
```
def Delete(self, name, last_name):
    self.dbCursor.execute("DELETE FROM Retiree WHERE name = ? AND last_name = ?", (name,last_name))
    self.dbConnection.commit()
```

- Statistics function SQL query for retrieving the value of room fee based on three tables



```
def statistics(self):
    self.dbCursor.execute("""SELECT Room.room_fee FROM Retiree
        INNER JOIN Block ON Block.block_id = Retiree.block_id
        INNER JOIN Room ON Room.room_id = Block.room_id""")
    records = self.dbCursor.fetchall()
    return records
```

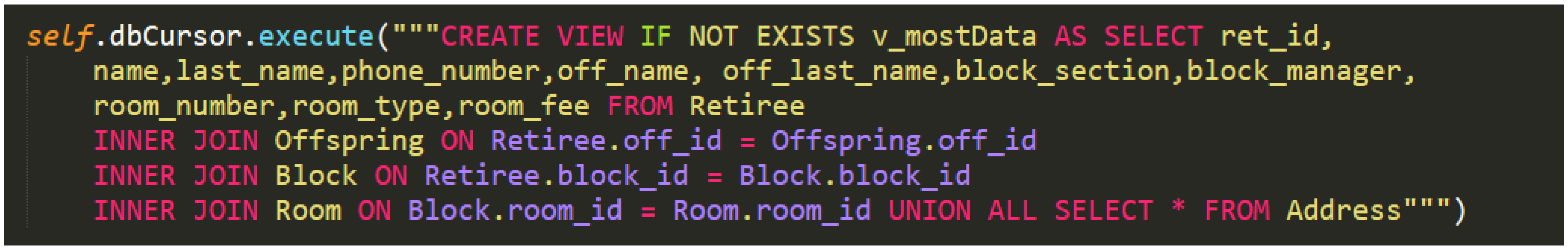
- Max and Min query for room fee



```python
def maxfee(self):
    self.dbCursor.execute("""SELECT name,last_name,MAX(Room.room_fee) FROM Retiree
        INNER JOIN Block ON Block.block_id = Retiree.block_id
        INNER JOIN Room ON Room.room_id = Block.room_id""")
    records = self.dbCursor.fetchall()
    return (records[0][0],records[0][1])

def minfee(self):
    self.dbCursor.execute("""SELECT name,last_name,MIN(Room.room_fee) FROM Retiree
        INNER JOIN Block ON Block.block_id = Retiree.block_id
        INNER JOIN Room ON Room.room_id = Block.room_id""")
    records = self.dbCursor.fetchall()
    return (records[0][0],records[0][1])
```

- Creating a view named *v_mostData* for the most important fields of database



```python
self.dbCursor.execute("""CREATE VIEW IF NOT EXISTS v_mostData AS SELECT ret_id,
    name,last_name,phone_number,off_name, off_last_name,block_section,block_manager,
    room_number,room_type,room_fee FROM Retiree
    INNER JOIN Offspring ON Retiree.off_id = Offspring.off_id
    INNER JOIN Block ON Retiree.block_id = Block.block_id
    INNER JOIN Room ON Block.room_id = Room.room_id UNION ALL SELECT * FROM Address""")
```

External packages

For this project, the Python's de facto standard GUI interface **Tkinter** is used which is a toolkit for developing applications in different operating systems also the **numpy** package is used for some statistical inference such as standard deviation and inter quartile range for the room fee.