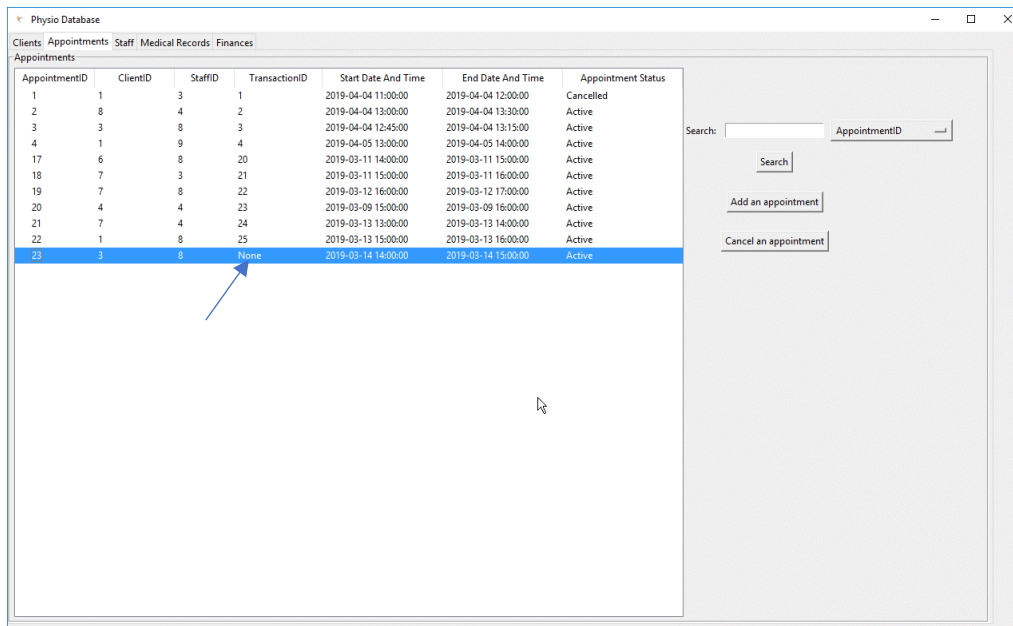# Developmental testing:

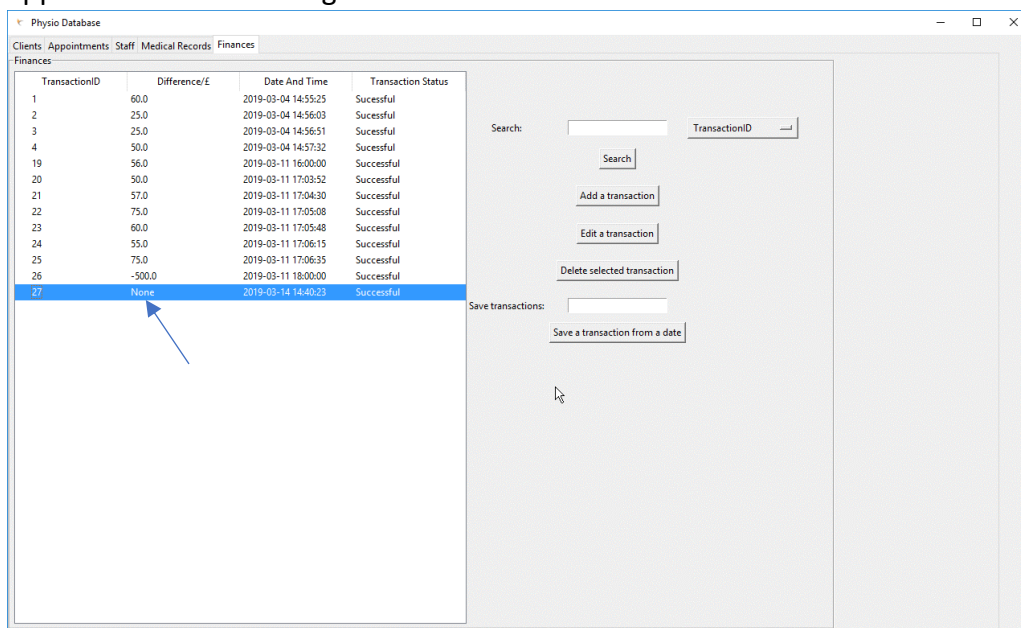## 7.1: Appointment and finance foreign key link

When adding a record into the appointments table, sometimes it would show 'None' under TransactionID, this shouldn't happen as the TransactionID should be created automatically when you make the program.

The first thing I did was to check if the data was within the finances table, which it was. This means that the issue isn't with creating the new transaction data, but it is with the appointment table taking it from the transaction table.

I then decided to look at the code and noticed that the AppointmentID and TransactionID were in the wrong order:

```
self.cursor.execute("""UPDATE appointments SET TransactionID = ? WHERE AppointmentID = ?""",
                    (appointmentid, transactionid,))
```

Which I fixed by swapping them around:

```
self.cursor.execute("""UPDATE appointments SET TransactionID = ? WHERE AppointmentID = ?""",
                    (transactionid, appointmentid,))
```

I then again, tried to run the program and this time it worked:

7.2: Finances output

When making my finances output to my text file I got an error, the error came up after I wrote in the date of the transaction and it told me that the columns I'm trying to write in a text file could not be printed as they weren't a string.

When I checked my code, I found the following:

```
for row in rows:
    f.write('\n' + "          " + (row[0]) + "                     " + (row[1]) + "           " + row[3])
f.write('\n' + '\n' + "Overall difference: £" + (sum[0]))
```

So, to fix it I wrote str() in front of each of the rows which made:

```
for row in rows:
    f.write('\n' + "          " + str(row[0]) + "                     " + str(row[1]) + "           " + str(row[3]))
f.write('\n' + '\n' + "Overall difference: £" + str(sum[0]))
```

So now when I tried to output the finances I got:

```
--------------------2019-03-11--------------------
| TransactionID | Difference | TransactionStatus |
---------------------------------------------------
      19              56.0              Successful
      20              50.0              Successful
      21              57.0              Successful
      22              75.0              Successful
      23              60.0              Successful
      24              55.0              Successful
      25              75.0              Successful
      26             -500.0             Successful

Overall difference: £-72.0
```
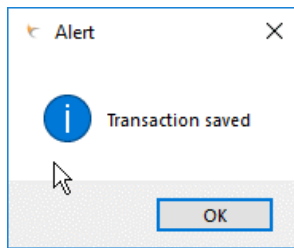
Alert — Transaction saved — OK

7.3: Table columns matched wrongly
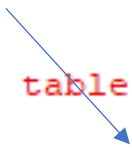
When creating my clients table, it would not match wrongly on the table (see address as a telephone number):

ClientID	MedicalRecordID	Prefix	First Name	Surname	DOB	Telephone	Address	Postcode
1	1	1	Mr	Kai	Holloway	1934-07-11	07863701077	23 Ermin Stre
2	2	2	Mr	Andrew	Rove	1938-04-12	07920991649	19 Middlewic
3	3	3	Mrs	Aaliyah	Price	1978-04-30	07935747494	34 Bishopthc
4	4	4	Ms	Paisley	Russel	2002-08-26	07830280919	12 Sutton Wi
5	5	5	Mr	Barack	Obama	1997-10-12	07702989653	98 Newport F
6	6	6	Mr	Cathal	Gorman	1954-09-01	07038789458	49 Ash Lane
7	7	7	Dr	Abigail	Dunn	1977-08-31	07777827240	5 Manor Clo:
8	8	8	Mr	Leo	Barker	1999-03-16	07956646717	69 Cefn Roac
9	9	9	Mrs	Jasmine	Powell	1941-01-04	07728974291	72 South Cre
10	10	10	Mr	Dayyan	OBrien	2001-01-31	07767862981	10 Windsor F
12	12	12	Dr	Steve	Jobs	1980-01-02	07605407812	5 Apple Lane

So, I decided to check update_table which is where the tables are outputted:

```python
def update_table(self):
    """
    updates the treeview and fills it with all records
    in the ClientID table
    """
    self.cursor.execute("""SELECT * FROM clients""")
    result = self.cursor.fetchall()
    self.tree.delete(*self.tree.get_children()) #clears table
    for item in result:
        self.tree.insert('', 'end', text=item[0], values=item[0:])
```
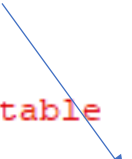
I noticed that the values were [0:] which is where the columns are set, so I changed it to 1:

```python
def update_table(self):
    """
    updates the treeview and fills it with all records
    in the ClientID table
    """
    self.cursor.execute("""SELECT * FROM clients""")
    result = self.cursor.fetchall()
    self.tree.delete(*self.tree.get_children()) #clears table
    for item in result:
        self.tree.insert('', 'end', text=item[0], values=item[1:])
```

So now, when I opened the clients page it showed the following which fixed the table:

ClientID	MedicalRecordID	Prefix	First Name	Surname	DOB	Telephone	Address	Postcode
1	1	Mr	Kai	Holloway	1934-07-11	07863701077	23 Ermin Street	RH18 9WX
2	2	Mr	Andrew	Rove	1938-04-12	07920991649	19 Middlewich Road	TN5 9JE
3	3	Mrs	Aaliyah	Price	1978-04-30	07935747494	34 Bishopthorpe Road	SA44 7LU
4	4	Ms	Paisley	Russel	2002-08-26	07830280919	12 Sutton Wick Lane	CA14 1DQ
5	5	Mr	Barack	Obama	1997-10-12	07702989653	98 Newport Road	IV12 5WH
6	6	Mr	Cathal	Gorman	1954-09-01	07038789458	49 Ash Lane	EX23 7WL
7	7	Dr	Abigail	Dunn	1977-08-31	07777827240	5 Manor Close	LL54 0UD
8	8	Mr	Leo	Barker	1999-03-16	07956646717	69 Cefn Road	RG7 4FJ
9	9	Mrs	Jasmine	Powell	1941-01-04	07728974291	72 South Crescent	DY9 9XX
10	10	Mr	Dayyan	OBrien	2001-01-31	07767862981	10 Windsor Hill	BT34 1ER
12	12	Dr	Steve	Jobs	1980-01-02	07605407812	5 Apple Lane	RZ7 2GS

7.4: Logging in going blank

When logging in, the frame would switch to something blank:
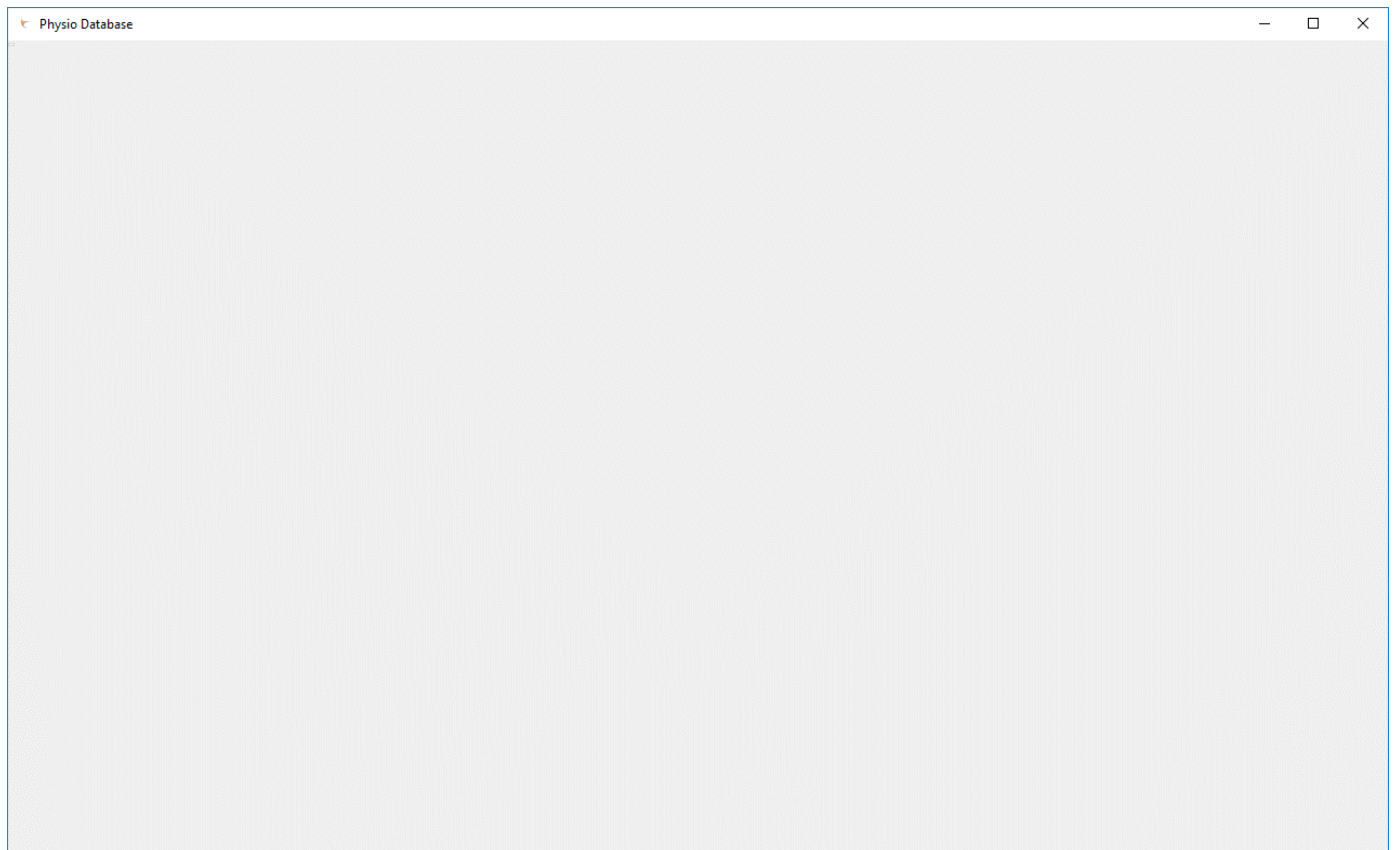
So, I checked the code that created tabs:

```python
class TabFrame(tk.Frame):
    """
    this is the frame which holds all the tabs, this allows users
    to select which tab they want to use
    """

    def __init__(self, parent, *args, **kwargs):
        tk.Frame.__init__(self, parent, *args, **kwargs)

        notebook = ttk.Notebook(parent)

        cursor.execute("""SELECT Position FROM staff WHERE LoggedIn = ?""",(True,))
        pos = cursor.fetchone()[0] #does views to see which tabs to show
        if pos == 'IT'  or pos == 'Owner' or pos == 'Nurse' or pos == 'Physiotherapist' or pos == 'Receptionist':
            notebook.add(ClientsFrame(notebook), text='Clients') #shows client tab
        if pos == 'Owner' or pos == 'Nurse' or pos == 'Physiotherapist' or pos == 'Receptionist':
            notebook.add(AppointmentsFrame(notebook), text='Appointments')
        if pos == 'IT'  or pos == 'Owner':
            notebook.add(StaffFrame(notebook), text='Staff')
        if pos == 'Owner' or pos == 'Nurse' or pos == 'Physiotherapist':
            notebook.add(MedicalRecordsFrame(notebook), text='Medical Records')
        if pos == 'Owner':
            notebook.add(FinancesFrame(notebook), text='Finances')

        notebook.grid(row=0, column=0)
```

As you can see the pos doesn't contain spaces, the actual positions have lots of spaces, so I decided to add spaces to each of the positions:

```python
class TabFrame(tk.Frame):
    """
    this is the frame which holds all the tabs, this allows users
    to select which tab they want to use
    """

    def __init__(self, parent, *args, **kwargs):
        tk.Frame.__init__(self, parent, *args, **kwargs)

        notebook = ttk.Notebook(parent)

        cursor.execute("""SELECT Position FROM staff WHERE LoggedIn = ?""",(True,))
        pos = cursor.fetchone()[0] #does views to see which tabs to show
        if pos == 'IT                  ' or pos == 'Owner               ' or pos == 'Nurse               ' or pos == 'Physiotherapist' or pos == 'Receptionist      ':
            notebook.add(ClientsFrame(notebook), text='Clients') #shows client tab
        if pos == 'Owner               ' or pos == 'Nurse               ' or pos == 'Physiotherapist' or pos == 'Receptionist      ':
            notebook.add(AppointmentsFrame(notebook), text='Appointments')
        if pos == 'IT                  ' or pos == 'Owner               ':
            notebook.add(StaffFrame(notebook), text='Staff')
        if pos == 'Owner               ' or pos == 'Nurse               ' or pos == 'Physiotherapist':
            notebook.add(MedicalRecordsFrame(notebook), text='Medical Records')
        if pos == 'Owner               ':
            notebook.add(FinancesFrame(notebook), text='Finances')

        notebook.grid(row=0, column=0)
```

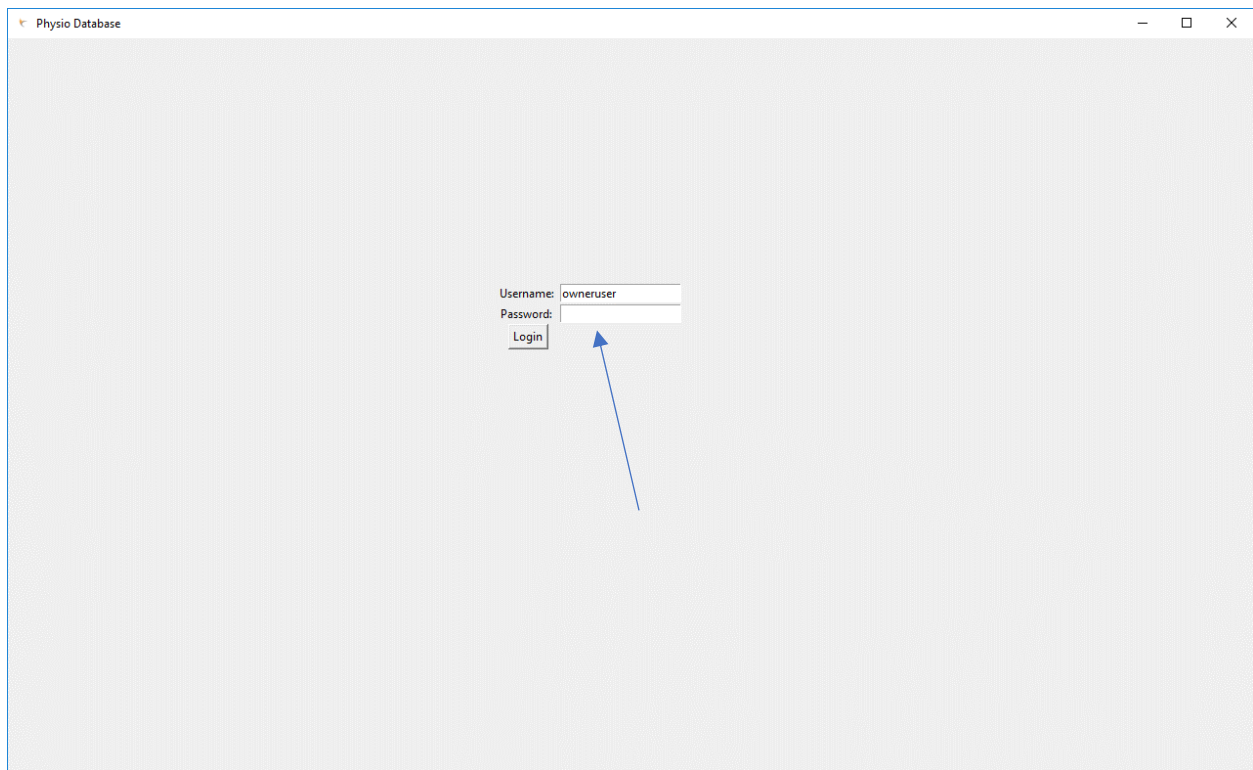I then tried logging in and the tables now showed:

7.5: Password box being blank

When writing my password into the search box is showed blank instead of *


So, I decided to look at the code:

```python
def __init__(self, master):
    tk.Frame.__init__(self, master)
    self.master = master
    tk.Label(self, text="Username: ").grid(row=0, column=0, padx = (500,0),
    tk.Label(self, text="Password: ").grid(row=1, column=0, padx = (500,0))
    self.un_entry = tk.Entry(self)
    self.un_entry.grid(row=0, column=1, padx = (0,500), pady = (250,0))
    self.pw_entry = tk.Entry(self, show = ' * ') #use stars for password
    self.pw_entry.grid(row=1, column=1, padx = (0,500))
```

And I noticed that there were spaces around the *, so I changed it to:

```python
def __init__(self, master):
    tk.Frame.__init__(self, master)
    self.master = master
    tk.Label(self, text="Username: ").grid(row=0, column=0, padx = (500,0),
    tk.Label(self, text="Password: ").grid(row=1, column=0, padx = (500,0))
    self.un_entry = tk.Entry(self)
    self.un_entry.grid(row=0, column=1, padx = (0,500), pady = (250,0))
    self.pw_entry = tk.Entry(self, show = '*') #use stars for password
    self.pw_entry.grid(row=1, column=1, padx = (0,500))
```
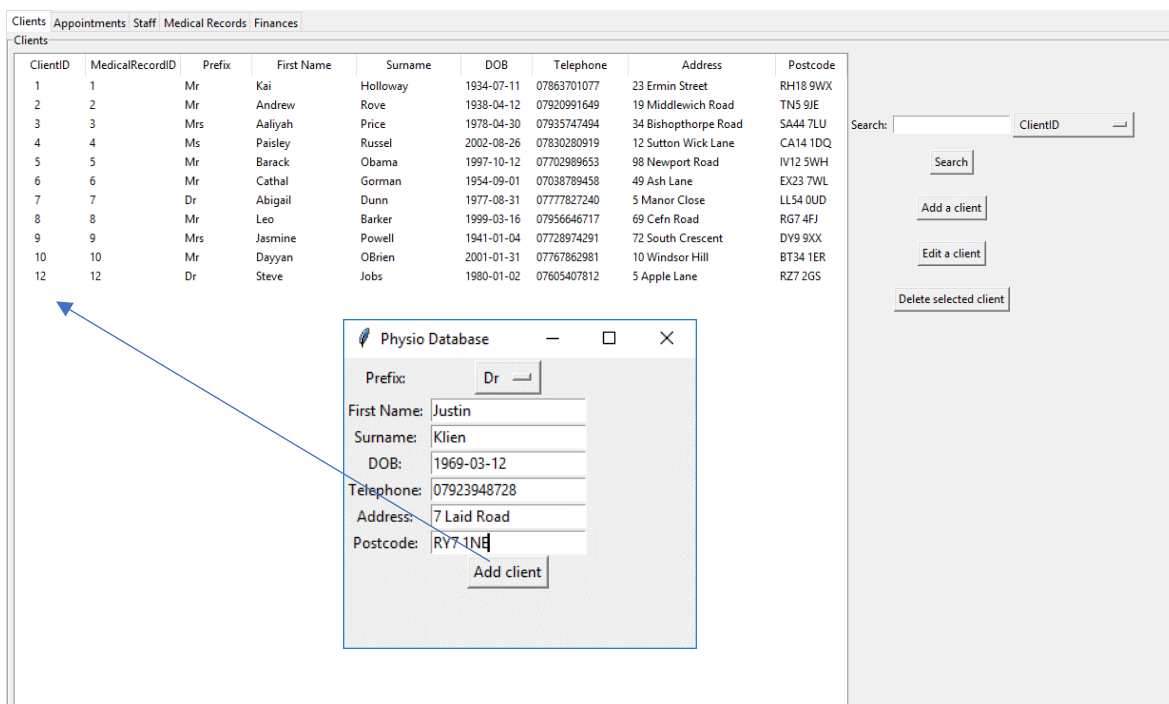
So now writing out the password looked like:
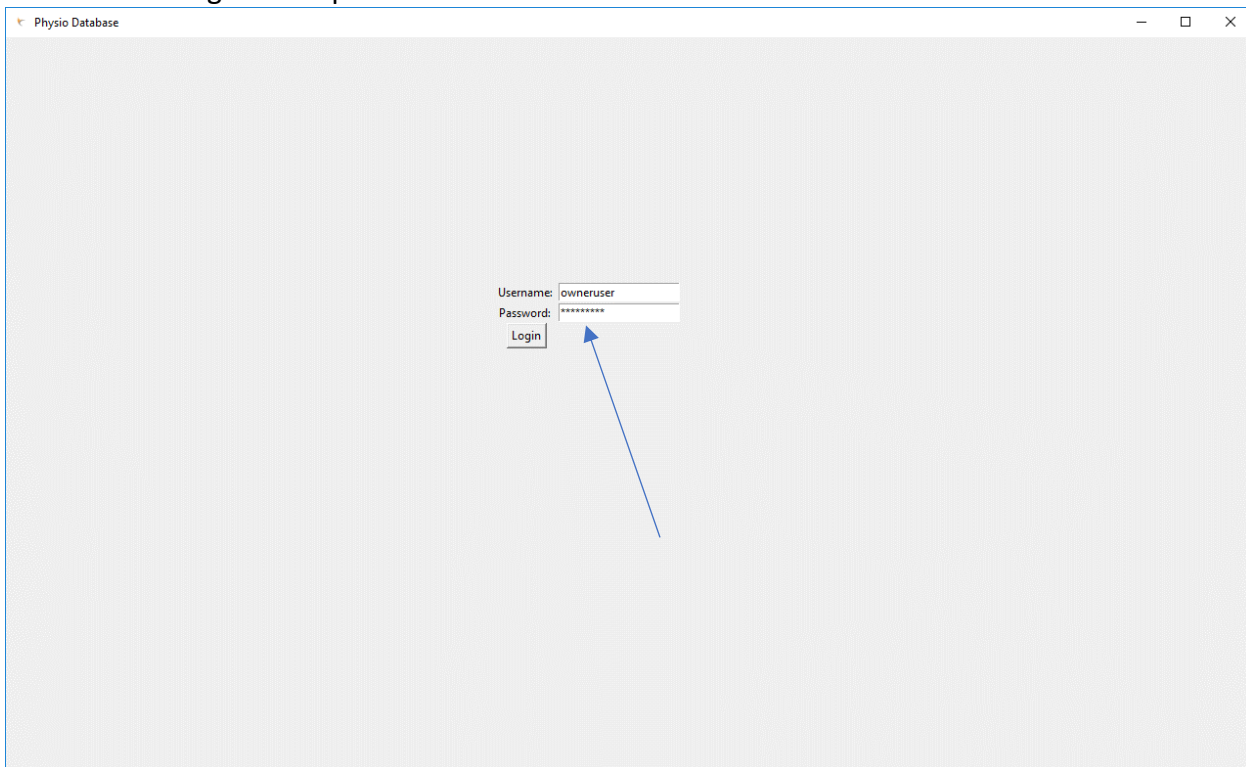
7.6: Records not updating automatically when a client is added

When adding a client, I found that nothing refreshed from the treeview. You can see that nothing is added below:

So, I decided to check the code that adds a client:

```python
self.cursor.execute(
    """INSERT INTO clients(Prefix, FirstName, Surname, DOB, Telephone, Address, Postcode) VALUES (?,?,?,?,?,?,?)""",
    (self.variable.get(), self.first_name_box.get(), self.surname_box.get(),
     self.dob_box.get(), self.telephone_box.get(), self.address_box.get(),
     self.postcode_box.get(),))
self.cursor.execute(
    """SELECT ClientID FROM clients ORDER BY ClientID DESC LIMIT 1"""
)
clientid = self.cursor.fetchone()
clientid = clientid[0]
self.cursor.execute(
    """INSERT INTO medicalrecords(ClientID) VALUES (?)""", (clientid,))
self.cursor.execute(
    """SELECT MedicalRecordID FROM medicalrecords ORDER BY MedicalRecordID DESC LIMIT 1"""
)
medicalrecordid = self.cursor.fetchone()
medicalrecordid = medicalrecordid[0]
self.cursor.execute(
    """UPDATE clients SET MedicalRecordID = ? WHERE ClientID = ?""",
    (medicalrecordid, clientid,))
self.db.commit()
self.create_client_window.destroy()
ClientsFrame.update_table()
```

I noticed that the ClientsFrame.update_table() is what called the update table function and in this case the function was in a different class, so I added self inside the brackets:

```python
self.cursor.execute(
    """INSERT INTO clients(Prefix, FirstName, Surname, DOB, Telephone, Address, Postcode) VALUES (?,?,?,?,?,?,?)""",
    (self.variable.get(), self.first_name_box.get(), self.surname_box.get(),
     self.dob_box.get(), self.telephone_box.get(), self.address_box.get(),
     self.postcode_box.get(),))
self.cursor.execute(
    """SELECT ClientID FROM clients ORDER BY ClientID DESC LIMIT 1"""
)
clientid = self.cursor.fetchone()
clientid = clientid[0]
self.cursor.execute(
    """INSERT INTO medicalrecords(ClientID) VALUES (?)""", (clientid,))
self.cursor.execute(
    """SELECT MedicalRecordID FROM medicalrecords ORDER BY MedicalRecordID DESC LIMIT 1"""
)
medicalrecordid = self.cursor.fetchone()
medicalrecordid = medicalrecordid[0]
self.cursor.execute(
    """UPDATE clients SET MedicalRecordID = ? WHERE ClientID = ?""",
    (medicalrecordid, clientid,))
self.db.commit()
self.create_client_window.destroy()
ClientsFrame.update_table(self)
```

I then added a client and this time the table automatically refreshed: