

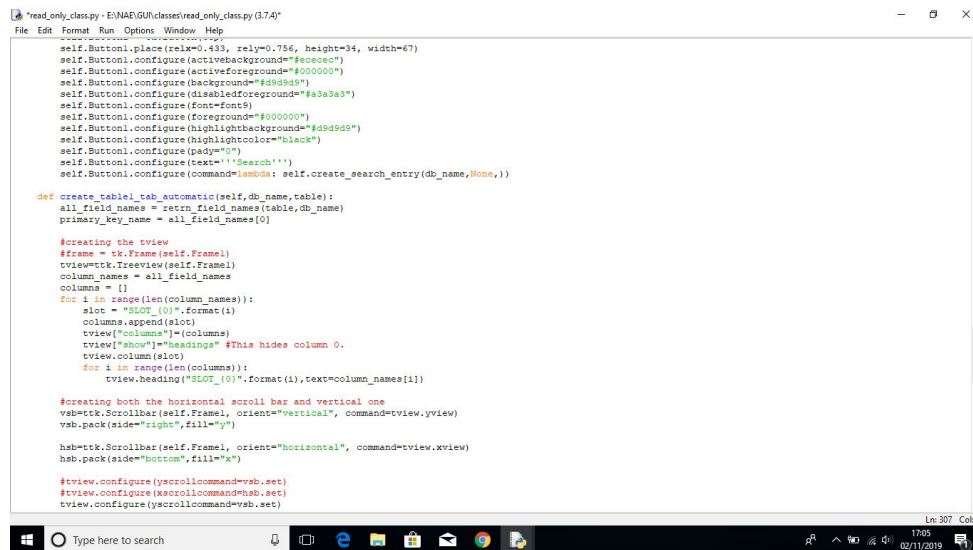
Section 7 - Development Testing

Problems	1
Problem 1 (Prototype) - Using notebook:	1
Problem 2 - Pop-Up window design:	2
Problem 3 - Notebook traversal:	3
Problem 4 -Missing records:	5
Problem 5 - Booking table not displaying correctly:	7
Problem 6 - Average salary:	8

Problems

Problem 1(Prototype) - Using notebook:

The problem below is that the function 'self.create_search_entry' requires either a notebook to get the name of the table or either the name of the table directly. As a notebook is not created when read only permission is given as only one table can be seen and no notebook can be passed.



```

read_only_class.py - E:\NAE\GUI\classes\read_only_class.py (3.7.4)
File Edit Format Run Options Window Help
self.Button1.place(relx=0.433, rely=0.756, height=34, width=67)
self.Button1.configure(activebackground="#f6f6f6")
self.Button1.configure(background="#d9d9d9")
self.Button1.configure(disabledforeground="#a3a3a3")
self.Button1.configure(font=font9)
self.Button1.configure(foreground="#000000")
self.Button1.configure(highlightbackground="#d9d9d9")
self.Button1.configure(highlightcolor="black")
self.Button1.configure(pady=0)
self.Button1.configure(text='Search')
self.Button1.configure(command=lambda: self.create_search_entry(db_name,None))

def create_table_tabAutomatic(self,db_name,table):
    all_field_names = retrn_field_names(table,db_name)
    primary_key_name = all_field_names[0]

#creating the tview
#frame = tk.Frame(self.Frame)
tview=tk.Treeview(self.Frame)
column_names = all_field_names
columns = []
for i in range(len(column_names)):
    slot = "SLOT_{}".format(i)
    columns.append(slot)
    tview["columns"]=(columns)
    tview["show"]="headings" #This hides column 0.
    tview.column(slot)
    for i in range(len(columns)):
        tview.heading("SLOT_{}".format(i),text=column_names[i])

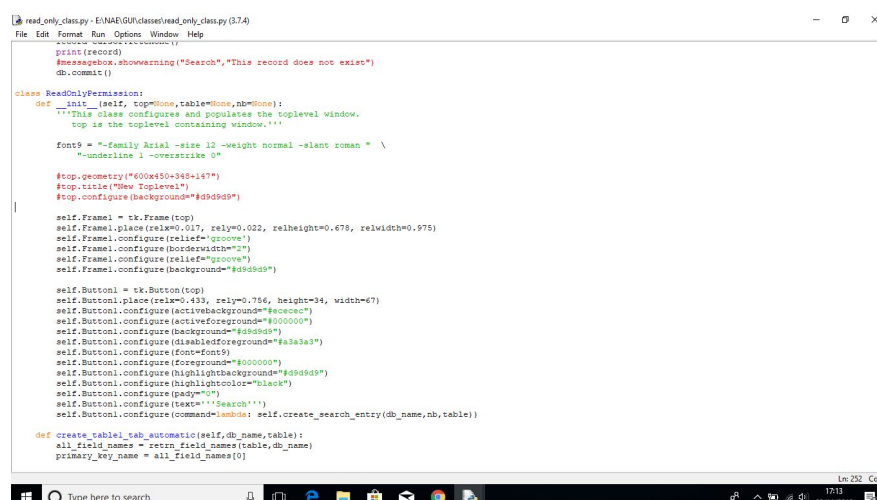
#creating both the horizontal scroll bar and vertical one
vsb=tk.Scrollbar(self.Frame, orient="vertical", command=tview.yview)
vsb.pack(side="right",fill="y")

hsb=tk.Scrollbar(self.Frame, orient="horizontal", command=tview.xview)
hsb.pack(side="bottom",fill="x")

#tview.configure(yscrollcommand=vsb.set)
#tview.configure(xscrollcommand=hsb.set)
tview.configure(yscrollcommand=vsb.set)

Ln: 307 Col: 0
  
```

I fixed this problem by changing the class declaration to include 2 new variables 'table' and 'nb' (this is a notebook). When declaring this class nb should be set to None. The reason why this was included is because the child class will inherit the search_button attribute and therefore the argument needs to be a variable instead of a hard-coded value so it can be changed in the child classes.



```

read_only_class.py - E:\NAE\GUI\classes\read_only_class.py (3.7.4)
File Edit Format Run Options Window Help
print(record)
#messagebox.showwarning("Search","This record does not exist")
db.commit()

class ReadOnlyPermissions:
    def __init__(self, top=None,table=None,nb=None):
        """This class configures and populates the toplevel window.
        top is the toplevel containing window."""
        font9 = "family Arial -size 12 -weight normal -slant roman " \
            "-underline 1 -overstrike 0"

        #top.geometry("600x450+345+147")
        #top.title("New Toplevel")
        #top.configure(background="#d9d9d9")

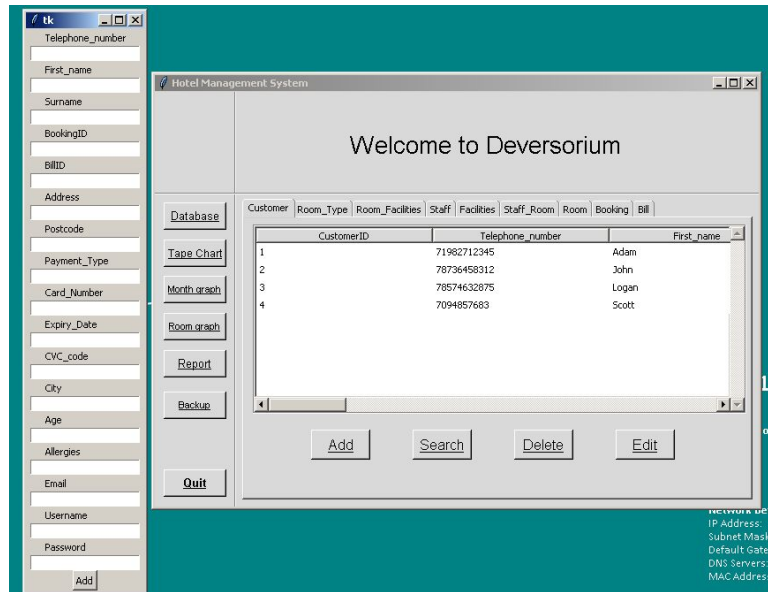
self.Frame = tk.Frame(top)
self.Frame.place(relx=0.017, rely=0.022, relheight=0.678, relwidth=0.975)
self.Frame.configure(relief="groove")
self.Frame.configure(borderwidth=1)
self.Frame.configure(background="#d9d9d9")

self.Button1 = tk.Button(top)
self.Button1.place(relx=0.433, rely=0.756, height=34, width=67)
self.Button1.configure(activebackground="#f6f6f6")
self.Button1.configure(activeforeground="#000000")
self.Button1.configure(background="#d9d9d9")
self.Button1.configure(disabledforeground="#a3a3a3")
self.Button1.configure(font=font9)
self.Button1.configure(foreground="#000000")
self.Button1.configure(highlightbackground="#d9d9d9")
self.Button1.configure(highlightcolor="black")
self.Button1.configure(pady=0)
self.Button1.configure(text='Search')
self.Button1.configure(command=lambda: self.create_search_entry(db_name,nb,table))

def create_table_tabAutomatic(self,db_name,table):
    all_field_names = retrn_field_names(table,db_name)
    primary_key_name = all_field_names[0]
  
```

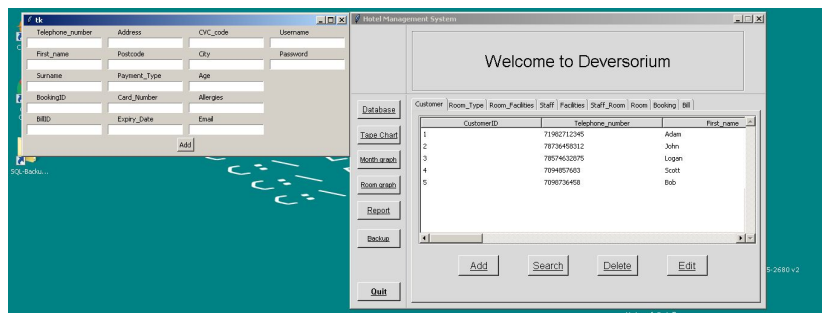
Problem 2 - Pop-Up window design:

The problem is that when the entry widget 'pop-up' window is created the entry widgets cover the length of the screen.



```
def create_entry(root, fields):
    """This will create an entry widget for every element in fields.
    It will also create a label with the text of the elements in fields."""
    entries = []
    for field in fields:
        row = tk.Frame(root) # all the entry widgets will be in this frame
        row.pack(padx=5)
        lab = tk.Label(row, width=15, text=field, anchor='w')
        ent = tk.Entry(row)
        lab.pack()
        ent.pack()
        entries.append(ent)
    return entries
```

A solution to this problem is to group the entry widgets together in groups of 5 and place these groups side by side.



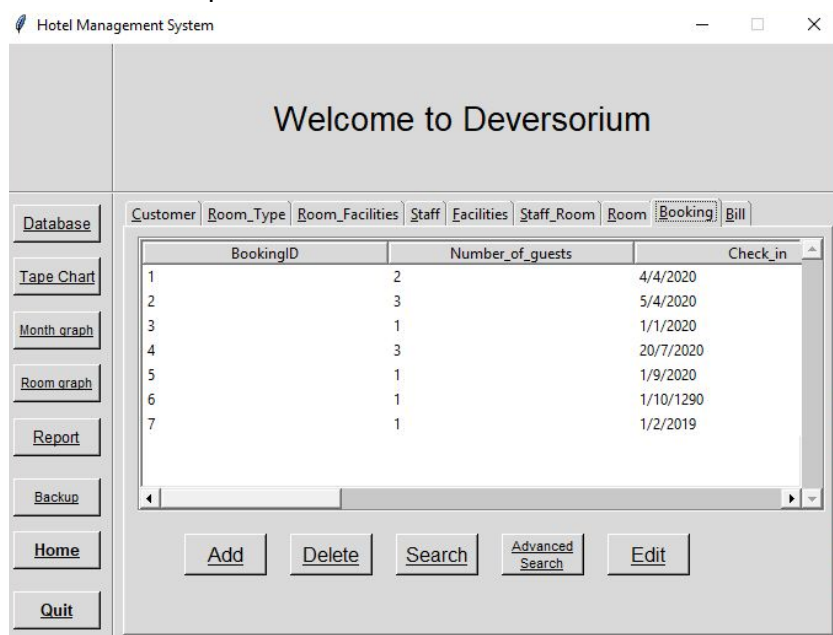
```
def create_entry(root, fields):
    """This will create an entry widget for every element in fields.
    It will also create a label with the text of the elements in fields."""
    master_frame=tk.Frame(root) # all the entry widgets will be in this frame
    counter=0 # counts how many groups of 5 there are (used to calculate the correct row and column the entry widgets should be placed in).
    entries = []
    for field in fields:
        row = tk.Frame(master_frame)
        if fields.index(field)%5==0 and fields.index(field)!=0:
            counter=counter+1
        row.grid(row=fields.index(field)-(counter*5),column=counter,padx=5)
        lab = tk.Label(row, width=15, text=field, anchor='w')
        ent = tk.Entry(row)
        lab.pack()
        ent.pack()
        entries.append(ent)
    master_frame.pack()
    return entries
```

Problem 3 - Notebook traversal:

To assist the user in traversing the notebook I have enabled traversal keys. These work by underlining a letter within the name of the tab and the user will be able to use 'Alt+X' (where 'X' is the underlined letter) to select the tab. The problem is that currently I am underlining the first letter of the tab but multiple tabs have the same first letter and therefore when the user uses 'Alt+X' it will select the first tab that appears.

As can be seen in the screenshot below when 'Alt+b' is pressed it will go to 'Booking' as this is the first tab that begins with 'B'. As 'Bill' also begins with 'B' the user will not be able to select this tab using this method.

When 'Alt+b' is pressed:



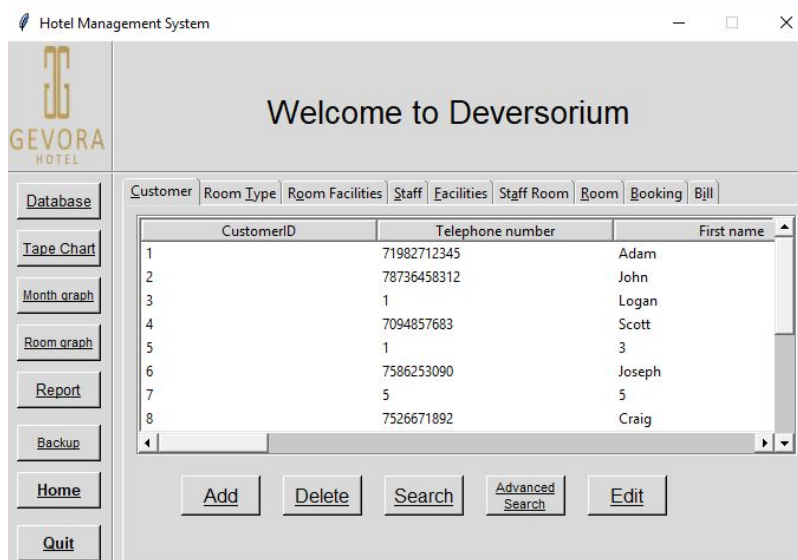
Code of function used to create notebook and underline the first character:

```
def get_database(self, db_name, tables):
    """This will create a notebook with each tab being a table.
    Each table has been given read and write permission."""
    clear_frame_or_window(self.Main_Frame)
    nb = ttk.Notebook(self.Main_Frame)
    ## Below is a for loop that will create a tab for each element in tables.
    for table in tables:
        frame=tk.Frame(nb,background="#d9d9d9")
        read_write = ReadAndWritePermission(frame,nb)
        read_write.create_treeview(db_name,table)

        nb.add(frame,text=table,underline=0)
        #nb.configure(cursor="hand2")
    nb.pack(expand=1,fill="both")
    nb.enable_traversal()
    return True
```

To fix this problem I have used a series of 'if statements' and the conditions of these 'if statements' is if the variable 'table' is equal to a specific table. Depending on the name of the table a different character will be underlined other than the first character.

As can be seen by the screenshot below depending on the table name a different character will be underlined and therefore solving the problem as the traversal is dependent on the underlined character.



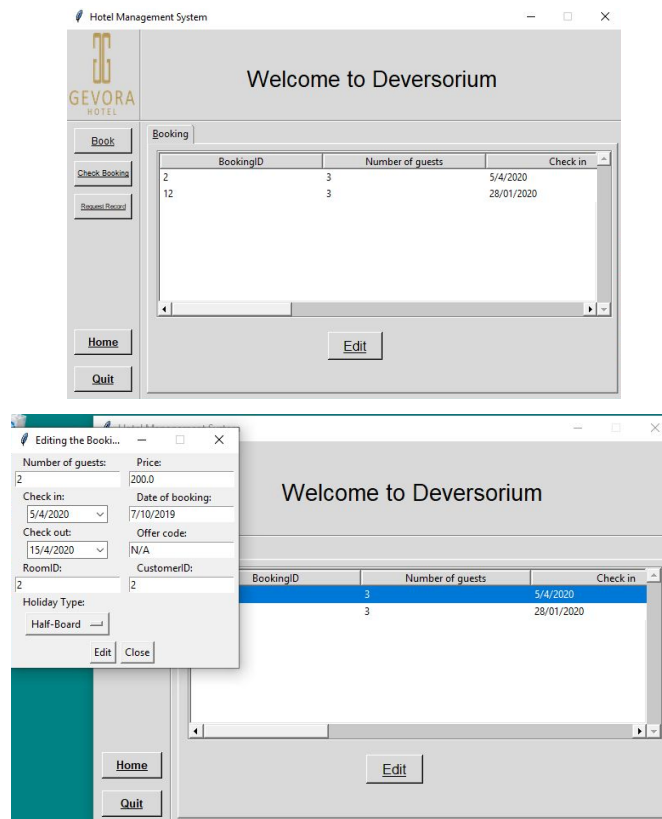
Corrected code:

```
def get_database(self,db_name,tables):
    """This will create a notebook with each tab being a table.
    Each table has been given read and write permission."""
    clear_frame_or_window(self.Main_Frame)
    nb = ttk.Notebook(self.Main_Frame)
    ## Below is a for loop that will create a tab for each element in tables.
    for table in tables:
        if table == "Room" and self.__class__.__name__ == "ReceptionistMainScreen":
            frame=tk.Frame(nb,background="#d9d9d9")
            read_only = ReadOnlyPermission(frame,nb)
            read_only.create_treeview(db_name,table)
        else:
            frame=tk.Frame(nb,background="#d9d9d9")
            read_write = ReadAndWritePermission(frame,nb)
            read_write.create_treeview(db_name,table)

        if table == "Bill" or table == "Room_Facilities":
            nb.add(frame,text=create_space(table),underline=1)
        elif table == "Room_Type":
            nb.add(frame,text=create_space(table),underline=5)
        elif table == "Staff_Room":
            nb.add(frame,text=create_space(table),underline=2)
        else:
            nb.add(frame,text=create_space(table),underline=0)
        #nb.configure(cursor="hand2")
    nb.pack(expand=1,fill="both")
    nb.enable_traversal()
    return True
```

Problem 4 -Missing records:

Shown below is the process of editing a record and the code responsible for this.

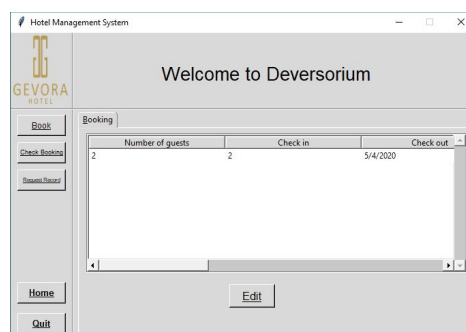


```
def edit_item(tview,data,table,db_name,primary_key_name,fields,record,cls): #This will edit a selected item
    fields=create_list(fields)
    sql_code_for_fields=""
    for i in range(len(fields)):
        sql_code_without_comma=fields[i]+"="
        sql_code_for_fields=sql_code_for_fields+sql_code_without_comma
    try:
        rogue_value=fields[i+1]
        sql_code_for_fields=sql_code_for_fields+" "
    except:
        next

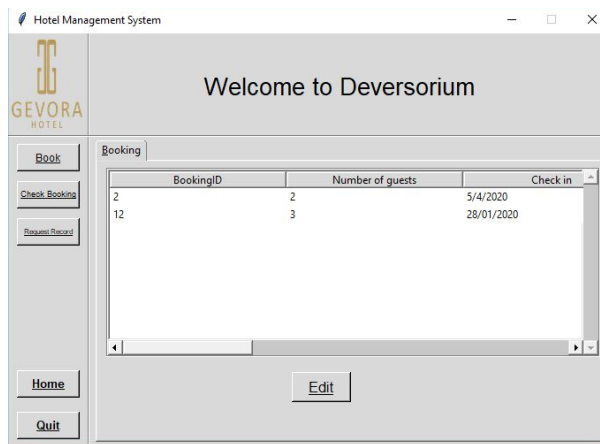
    primary_key=record[0] # The reason it is here is because it needs to have the original index value (i.e

    with sqlite3.connect(db_name) as db:
        cursor=db.cursor()
        cursor.execute("select * from (1) where (2)=(0)".format(str(primary_key),table,primary_key_name))#Th
        record=cursor.fetchone()
        edited_record=str(primary_key),data
        if messagebox.askyesno("Edit","Are you sure you want to edit (0) to (1)?".format(record,edited_recor
        sql="update (2) set (1) where (3)=(0)".format(str(primary_key),sql_code_for_fields,table,primary
        data=create_list(data)
        cursor.execute(sql,data)
        db.commit()
```

As we can see below 2 problems occur. One of these problems is that only the edited record is shown and the other problem is that primary key of the table has been removed as a column.



Solution



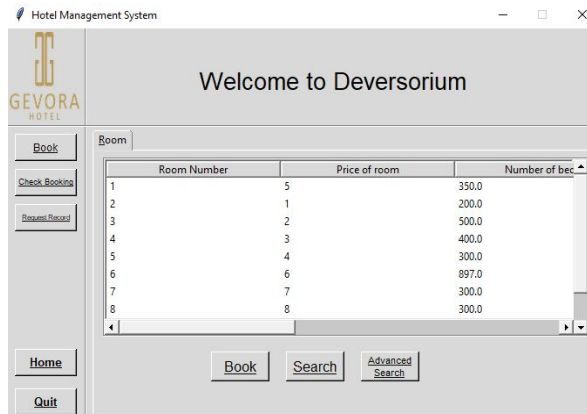
Below is a screenshot of the code used to fix this problem. The 'if statement' checks if the current class is 'RequestRecord' so this does not interfere with other tables being edited. The code below will get the updated record directly from the database and remove the old record from the list of records linked to the class. It will then add the updated record to the list of records and sort this list depending on the primary key. To solve the column issue I simply added the primary key name to the start of the 'fields' variable which solved the issue.

```
with sqlite3.connect(db_name) as db:
    cursor=db.cursor()
    cursor.execute("select * from {1} where {2}={0}".format(str(primary_key), table, primary_key_name))#The
    record=cursor.fetchone()
    edited_record=str(primary_key),data
    if messagebox.askyesno("Edit","Are you sure you want to edit {0} to {1}?".format(record,edited_recor
        sql="update {2} set {1} where {3}={0}".format(str(primary_key),sql_code_for_fields,table,primary
        data=create_lst(data)
        cursor.execute(sql,data)
        db.commit()

if cls.__class__.__name__ == "RequestRecord":
    with sqlite3.connect(db_name) as db:
        cursor=db.cursor()
        cursor.execute("select * from {1} where {2}={0}".format(str(primary_key), table, primary_key_n
        new_record=cursor.fetchall()
        record = list(record)
        new_record = list(new_record[0])
        class.users_record.remove(record)
        class.users_record.append(new_record)
        fields.insert(0,primary_key_name)
        clear_frame_or_window(class.Treeview_Frame)
        class.users_record.sort(key=lambda x:x[0])
        make_tview(fields,class.users_record,class.Treeview_Frame)
else:
    refresh(tview,db_name,table)
```


Problem 5 - Booking table not displaying correctly:

The problem is that when a customer is booking a holiday the room table is not displaying correctly. It seems that the field names and the column names do not match.



The problem therefore needs to be from the 'create_treeview' method which is below:

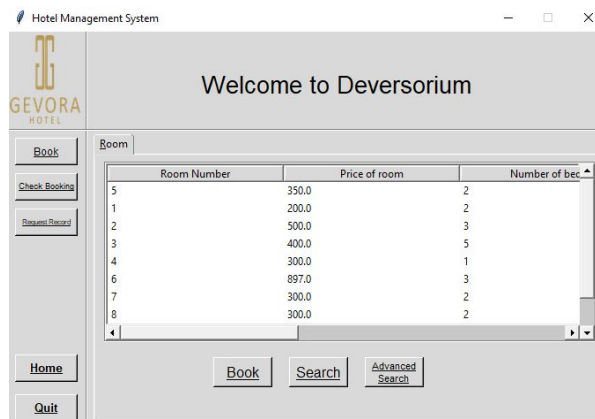
```
def create_treeview(self,db_name,table): # db_name needs to be a string. table needs to be a string.
    """This will create a treeview for a particular table in a database and populate it."""
    column_names = retrn_field_names(table,db_name) # gets the field names from the table.

    with sqlite3.connect(db_name) as db:
        cursor=db.cursor()
        cursor.execute("select * from {}".format(table))
        records=cursor.fetchall()

    if self.__class__.__name__ == "BookHoliday":
        column_names.remove("RoomID")
        column_names.remove("RoomTypeID")
        column_names.remove("Staff_RoomID")
```

Solution

The solution was to remove the offending fields from the record therefore placing the correct fields with the correct column names.



```
def create_treeview(self,db_name,table): # db_name needs to be a string. table needs to be a string
    """This will create a treeview for a particular table in a database and populate it"""
    column_names = retrn_field_names(table,db_name) # gets the field names from the table

    with sqlite3.connect(db_name) as db:
        cursor=db.cursor()
        cursor.execute("select * from {}".format(table))
        records=cursor.fetchall()

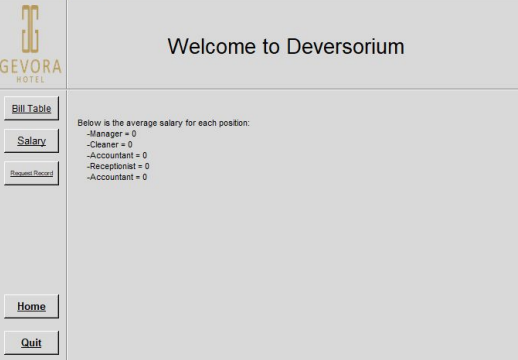
    if self.__class__.__name__ == "BookHoliday" and table == "Room":
        column_names.remove("RoomID")
        column_names.remove("RoomTypeID")
        column_names.remove("Staff_RoomID")

    for record in records:
        lst_record = list(record) # Converts record into list as you cannot use .pop() on tuples
        lst_record.pop(0) # Removes RoomID
        lst_record.pop(2) # Removes RoomTypeID
        lst_record.pop(7) # Removes Staff_RoomID
        lst_record = tuple(lst_record) # Converts the record to be added into records back into tuple
        ## Next 2 lines replace the old record with the new record
        records.insert(records.index(record),lst_record)
        records.remove(record)

    make_tview(column_names,records,self.Treeview_Frame)
```


Problem 6 - Average salary:

The average salary function is not working correctly as the average salary for each position is displaying as 0.



Hotel Management System

Welcome to Deversorium

Below is the average salary for each position:

- Manager = 0
- Cleaner = 0
- Accountant = 0
- Receptionist = 0
- Accountant = 0

Buttons: Bill Table, Salary, Payment Received, Home, Quit

```
def get_av_salary(position):
    print(position)
    total_salary = 0
    with sqlite3.connect("Hotel_Management_system.db") as db:
        cursor=db.cursor()
        cursor.execute("select Salary from Staff where Job = '{}'.format(position)")
        salaries=cursor.fetchall()
        db.commit()
    for salary in salaries:
        try:
            total_salary = total_salary+salary[0]
        except TypeError:
            pass
    try:
        average = total_salary/len(salaries)
    except ZeroDivisionError:
        average = 0
    return average

def get_salaries(self):
    all_salaries = []
    positions = ["Manager", "Cleaner", "Accountant", "Receptionist", "Accountant"]
    for position in positions:
        all_salaries.append(get_av_salary(position))

    text = """
Below is the average salary for each position:
-Manager = {0}
-Cleaner = {1}
-Accountant = {2}
-Receptionist = {3}
-Accountant = {4}

""".format(all_salaries[0],all_salaries[1],all_salaries[2],all_salaries[3],all_salaries[4])

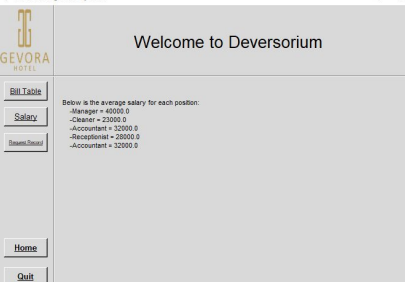
    font = "family Arial -size 8 -weight normal -slant roman " \
    "-underline 0 -overstrike 0"
    self.welcome_message(text, font)
```

Solution

The solution to this problem was very simple. As shown in the screenshot below the result returned from the sql statement (the result after the position) is an empty list. This then led me to inspect my sql statement. The issue was that I was searching for the positions with an uppercase letter at the start while the database stored these as all lower case. To fix this issue I edited the database so that it will have uppercase letters at the beginning.

```
Manager
[]
0
Cleaner
[]
0
Accountant
[]
0
Receptionist
[]
0
Accountant
[]
0
```

Fixed result



Hotel Management System

Welcome to Deversorium

Below is the average salary for each position:

- Manager = 40000.0
- Cleaner = 25000.0
- Accountant = 32000.0
- Receptionist = 28000.0
- Accountant = 32000.0

Buttons: Bill Table, Salary, Payment Received, Home, Quit