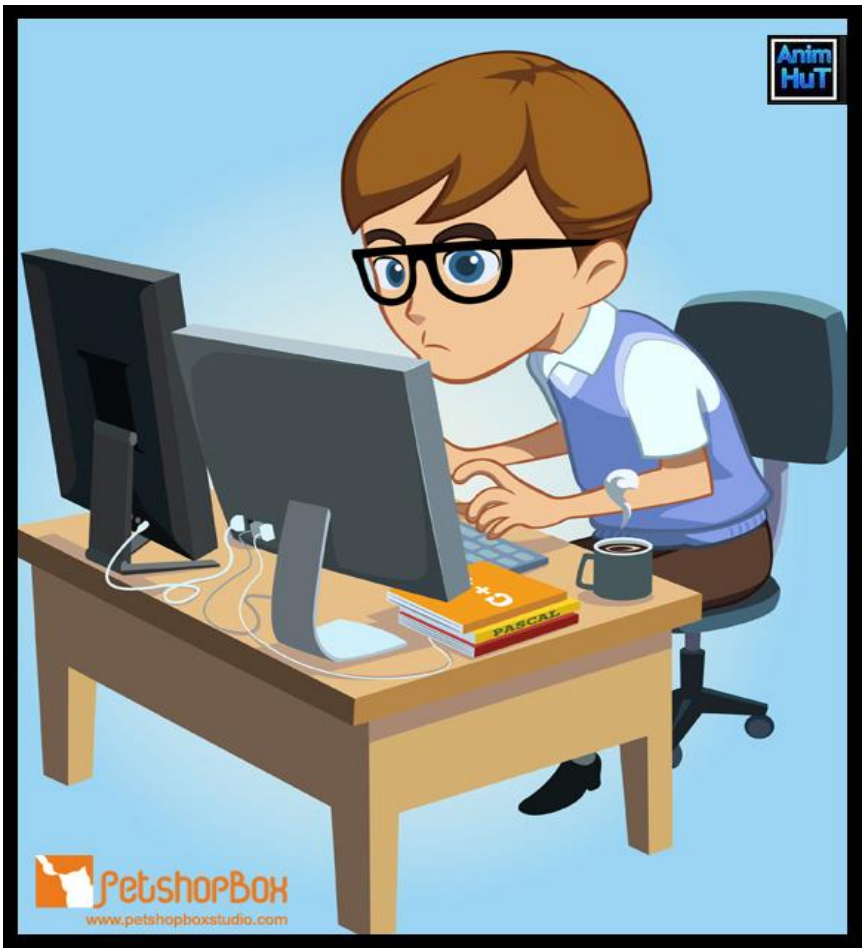
QAP 4 – Jul 16 – 26, 2023



- **Projects1, 2 and 4 will be completed individually in this QAP, and Project 3 will be done in your groups. You have over a week – plenty of time – don't leave it to the last minute.**

- **Each project files are to be submitted through the assignment portal by July 26 at 11:59 pm.**

- **Questions can be asked before any classes during the regularly scheduled times in the next two weeks.**

Project 1 – Python – Lists and Data Files

The **One Stop Insurance Company** needs a program to enter and calculate new insurance policy information for its customers. Create a data file called OSICDef.dat that contains the next policy number, the basic premium, the discount for additional cars, the cost of extra liability coverage, the cost of glass coverage, the cost for loaner car coverage, the HST rate, and the processing fee for monthly payments. The file will look as follows:

1944
869.00
.25
130.00
86.00
58.00
.15
39.99

As the program starts, read the values from the defaults file.

The user will input the *customer's* first and last name, address, city, province (validate using a list to make sure the province is valid), postal code, and phone number. They will also enter the number of cars being insured, and options for extra liability up to $1,000,000 (enter Y for Yes or N for No), optional glass coverage (Y or N), and optional loaner car (Y or N). Finally enter a value to indicate if they want to pay in full or monthly (Full or Monthly – use a list to validate). Convert the first and last name, the city, and the payment Method to title case and the Y/N values upper case. No validations required – other than those specified - but go for it if you want. Be careful when testing - enter values that are valid for each input.

Insurance premiums are calculated using a basic rate of $869.00 for the first automobile, with each additional automobile offered at a discount of 25%. If the user enters a Y for any of the options, the costs are $130.00 per car for extra liability, $86.00 per car for glass coverage, and $58.00 per car for the loaner car option. All these values are added together for the total extra costs. The total insurance premium is the premium plus the total extra costs. HST is calculated at 15% on the total insurance premium, and the total cost is the total insurance premium plus the HST. Customers can pay for their insurance in full or in 8 monthly payments. Calculate the monthly payment by adding a processing fee of $39.99 to the total cost and dividing the total cost by 8. The invoice date is the current date and the next payment date is the first day of the next month. Use the values from the defaults file in your processing.

Display all input and calculated values to the screen in a **well-designed receipt**.

Save the Policy number, all input values and the total insurance premium to a file called Policies.dat for future reference the invoice date will be the current date. Display the message "Policy information processed and saved." – you may also include a type of progress bar at this point. And increase the next policy number by 1. A sample line in the file might appear as follows:

1944, 2023-03-10, John, Smith, 12 Main St., St. John's, NL, A1A8H4, 123-123-1234, 2, Y, N, Y, Full, 1329.00

Allow the user to enter as many policies as they need. When the user is done entering policies and exits the program, write the current values back to the defaults file.

Create a second Python program that will allow the user to enter the total amount of sales for each month from January to December – note that if it not the end of the year some of the monthly values may be 0. Place the values in a list representing the y-axis. Values for the x-axis will be the months of the year in the form [“Jan”, “Feb”, “Mar”, …]

Use Matplotlib to create a graph of the total sales ($) against the months using the two lists created. Add an appropriate title and other options that you see fit.

Project 2 – GitHub

Create a GitHub repository called QAP 4 Files XX – where XX are your initials. Make the repository public and add a ReadMe file. Add the two Python Programs, the defaults file (OSICDef.dat), and the data file (Policies.dat) to the repository.

In a separate chat, send me the URL for your repository so I can clone and run your project. Also include these files in the assignment portal as well as a backup just in case I cannot access your repository.

Project 3 – Entity / Relationship Diagram

The following is to be completed in your groups. It is strongly suggested that you get together with your team and plan the solution over several meetings – so not to have one person complete it all.

Sleep-Tite Motel requires a program to track rooms, bookings, and revenues for the hotel. Note there is no restaurant since there is a Tim's on one side and a steak house on the other side of the parking lot. They do have a coffee station and snacks set up in the corner of the reception area for guests – purchases are just paid for at the front reception. Create the ERD for the following system.

A **Defaults** file called STDef.dat will be used to store current values for the invoice number (1856), room rate ($75.00), the HST rate (15%), the Early check-in rate ($12.00), the Extra bed rate ($7.00 per night), the Extra key rate ($2.00), and the Late check-out rate ($12.00). This can be set up as a table / file in the ERD but it will be stand alone, not have a primary key, and not have relationships to any other tables.

The **Room Status** file will only have 26 records since there are 26 rooms in the hotel (101 - 126). Included will be the Room number, the Customer name, the full mailing address for the customer, the customer phone number, the customer credit card number and expiry date, the check-in date, the check-out date, and options for an early check-in, an extra bed in the room, extra key for the room, late check out. If the room is empty the customer's name will contain the value "Vacant".

The **Bookings** file has information for each advanced booking made by a customer. These bookings are usually made by phone, or through the company web page. Included in this table are the booking ID which is generated by the system, the customer's name and phone, the check-in and the check-out dates.

As a customer checks out, information will be saved to a **Revenue** file. Based on the information in the room status table, the system will create, display, and print an invoice for the customer. This will include the charges based on the # nights and the room rate, any extra charges such as the early check-in, extra bed, extra key, or the late check-out. The revenue table includes the invoice number, the invoice date, the customer's name, the payment method (Cash, Debit or Credit card), the room charge, the extras charge (based on early check-in, extra bed, extra key, and late check-out), the subtotal, taxes and total.

The hotel has a storage room with all supplies for cleaning, office supplies, extra room towels and face clothes, and supplies for the coffee station and snack bar. They would like to put a computer in the room to track items brought in and items taken out. Set up a **Supplies** file that will allow them to track inventory and allow for ordering.

Identify 2 or 3 extra field that you could add to the ERD. Include a reason why you feel each field would be useful to the system. Explain your additions in a separate Word file and explain the reason / purpose for the extra fields.

Project 4 – JavaScript

Create a reasonably complicated Object in JavaScript that contains attributes and methods for the **Motel** customer. The customer attributes should include (and not be limited to), the customer's name, birth date, gender, and room preferences (as an array), payment method, mailing address (as a sub-object), phone number, check-in and check-out datetime (as a sub-object), and object methods to determine there age and duration of stay. The JavaScript code should also create a template literal string, or properly formatted html, that describes the customer. In other words, writes a paragraph describing the customer with many of their personal attributes dynamically embedded in the string / html coming from the above object definition.