Employee Program Evaluation

I have created a program that helps sale managers calculate the amount of commission paid to employees and display the amount of sold properties from highest to lowest. My program saves user input; number of employees, names, IDs, and properties sold. I think my program meets some of the scenario requirements but does not meet the bonus to the commissions of the employee with most sales. Other than the program not calculating the bonus of most sales, the program works fine and does not crash at all. The program is filled with data validation that stops the program from crashing when the user inputs the wrong data type, the program gives the user another chance to input the correct data type.

The program is fast and efficient, for example the program takes 2-3 seconds to boot up and uses less than 8 MB of processing memory. The program is well optimized using the least amount of code and using correct data types and variables, for example the 'employeeIdArray' is an array that stores integers only and if the user tries to store anything but integers, the program shows an error message and allows the user to input the correct data type. The program performs well when used on a low spec computer, this means the program can work on any computer device no matter how old or slow the computer is.

I choose to use the 'byte' data type for the number of employees, because 'byte' can range is 0-255 which is more than suitable for this program and scenario. Before choosing to use 'byte' data type, I used 'int' data type which has an approximate range of -2 million to 2 million, therefore this data type isn't necessary because the number of employees can never be negative. During the development stage of the program, I changed the local variables of the 'while' loops which were confusing and inconsistent. The 'while' loops variables were: selectedNumber1, selectednumber2, numberselected2 and numberselected3. Now the 'while' loops variables are: validationLoop1, validationLoop2, validationLoop3 and validationLoop4. I also used a 'TryParse' method to validate user input, this is because it required less code than the 'try-catch' method and works well with the 'while' loop.

Overall, the program works fine and handles wrong user input well, gives the user a chance to retry inputting data types till they are valid, but still doesn't meet most of the scenario's requirements. The program also allows the user to restart and exit the program by pressing a single or multiple buttons making quitting and restarting the program easy and efficient.