Design

Description of the system

I am going to make a sales system that can handle multiple sales made by an employee which should necessarily be a salesperson and each salesperson has been linked to a manager. The manager will get a specific amount of profit which is 3.5% on each sale made by a salesman under that particular manager and also a profit of 6.5% will be given to the salesman that will make the sale. Our system will be able to perform basic crud operations on employees, both salesman and managers, along with cars, and the system will be able to keep records of the sales that have been made and can display the sales done by a particular employee.

Assumptions

I have made the following few assumptions while designing this system

- In one sale only one car can be sold
- A sale can be made by only one sales man, more than one sales man can not be associated with one sale.
- Manager can not be deleted unless there are sales men associated to him. Either we need to modify sales man to assign another manager or to delete all sales men under that particular manager in order to align this system with real world scenario.
- Manager can not make a sale directly

Description of classes

Each class has an init() function which act as an constructor for that particular class.

1. **Employee class**
   This class represents an employee with attributes such as
- **name**: A string representing the name of the employee.
- **salary**: An integer representing the salary of the employee.
- **department**: A string representing the department in which the employee works.
- **passport**: A string representing the passport number of the employee.
- **age**: An integer representing the age of the employee.
- **email**: A string representing the email address of the employee.
- **profit**: A float representing the profit made by the employee. It has a default value of 0.

2. **Manager class**
   This class is a subclass of Employee or we can say it inherits from Employee class and represents a manager with additional attributes such as
- **managerId**: An integer representing the ID of the manager.
- **isDeleted**: A boolean representing whether the manager has been deleted or not. It has a default value of False.

3. **SalesMan class**
   This class is a subclass of Employee or we can say it inherits from Employee class and represents a salesperson with additional attributes such as

- **empId**: An integer representing the ID of the salesperson.
- **managerId**: An integer representing the ID of the manager.
- **isDeleted**: A boolean representing whether the salesperson has been deleted or not. It has a default value of False.

4. **Car class**
   This class represents a car with attributes such as

- **name**: A string representing the name of the car.
- **carId**: An integer representing the ID of the car.
- **type**: A string representing the type of the car.
- **price**: An integer representing the price of the car.
- **model**: A string representing the model of the car.
- **isDeleted**: A boolean representing whether the car has been deleted or not. It has a default value of False.

5. **Sales class**
   This class represents a sale transaction with attributes such as

- **carId**: An integer representing the ID of the car being sold.
- **empId**: An integer representing the ID of the salesperson who made the sale.
- **date**: A datetime object representing the date and time when the sale was made.

6. **Driver**
   This class acts as the main driver of the program, and contains lists of other classes in order to manage those classes. These lists are:

- **saleList**: A list containing all the sales made by the salespeople.
- **managersList**: A list containing all the managers.
- **salesManList**: A list containing all the salespeople.
- **carList**: A list containing all the cars.

This class also have functions to manipulate these classes such as

- **addSales(self, sale)**: Adds a Sales object to the list of sales.
- **searchSalesByEmpId(self, empId)**: Searches for sales made by a SalesMan with the given ID.
- **addCar(self, car)**: Adds a Car object to the list of cars.

- **deleteCar(self, carId)**: Deletes a Car object from the list of cars with the given ID.
- **modifyCar(self, carId, newPrice,newName,newModel,newType)**: Modifies the attributes of a Car object with the given ID.
- **searchCar(self, carId)**: Searches for a Car object with the given ID.
- **addSalesMan(self, salesMan)**: Adds a SalesMan object to the list of salesmen.
- **deleteSalesMan(self, empId)**: Deletes a SalesMan object from the list of salesmen with the given ID.
- **modifySalesMan(self, empId, newSalary,newAge,newDep,newName,newPass)**: Modifies the attributes of a SalesMan object with the given ID.
- **searchSalesMan(self, empId)**: Searches for a SalesMan object with the given ID.
- **addManager(self, manager)**: Adds a Manager object to the list of managers.
- **deleteManager(self, managerId)**: Deletes a Manager object from the list of managers with the given ID.
- **modifyManager(self, managerId, newSalary,newAge,newDep,newName,newPosition)**: This function modifies the attributes of an existing Manager instance in the manager list.
- **search_manager_by_id(self, manid):** This function searches for a manager in the managers list based on their ID. It takes a manager ID as input, iterates over the managers in the list, and returns True if a manager with a matching ID is found, otherwise False.
- **validate_string(self, s):** This function is used to validate a string by checking if it is empty or contains only alphabets, spaces, and digits. It takes a string as input, iterates over each character in the string, and returns False if any character is not an alphabet, space, or digit. Otherwise, it returns True.
- **is_valid_price(self, price)**: This function is used to validate a price by checking if it is a positive number. It takes a price as input, converts it to a float, and returns False if the resulting number is negative. Otherwise, it returns True.
- **validate_car_id(self, car_id):** This function is used to validate a car ID by checking if it already exists in the car list and is not marked as deleted. It takes a car ID as input, iterates over the cars in the list, and returns False if a car with a matching ID is found and is not marked as deleted. Otherwise, it returns True.

All the functions have their doc strings which define the functionality of each function in details that can be seen in the code implementation section.