1.py

```
import tkinter as tk
from tkinter import ttk
import sqlite3


##Класс главного окна
class Main(tk.Frame):
    def __init__(self, root):
        super().__init__(root)
        self.init_main()
        self.db = db
        # Для отображения данных:
        self.view_records()

    def init_main(self):
        toolbar = tk.Frame(bg='#d7d8e0', bd=2)
        toolbar.pack(side=tk.TOP, fill=tk.X)

        self.add_img = tk.PhotoImage(file="add.gif")
        btn_open_dialog = tk.Button(toolbar, text='Добавить перевозчика', command=self.open_dialog, bg='#d7d8e0', bd=0,
                                    compound=tk.TOP, image=self.add_img)
        btn_open_dialog.pack(side=tk.LEFT)

        self.update_img = tk.PhotoImage(file='update.gif')
        btn_edit_dialog = tk.Button(toolbar, text="Редактировать выделенную запись", bg='#d7d8e0', bd=0,
                                    image=self.update_img,
                                    compound=tk.TOP, command=self.open_update_dialog)
        btn_edit_dialog.pack(side=tk.LEFT)

        self.delete_img = tk.PhotoImage(file='delete.gif')
        btn_delete = tk.Button(toolbar, text='Удалить выделенные записи', bg='#d7d8e0', bd=0, image=self.delete_img,
                               compound=tk.TOP, command=self.delete_records)
        btn_delete.pack(side=tk.LEFT)

        self.search_img = tk.PhotoImage(file='search.gif')
        btn_search = tk.Button(toolbar, text='Поиск по ФИО', bg='#d7d8e0', bd=0, image=self.search_img,
                               compound=tk.TOP, command=self.open_search_dialog)
        btn_search.pack(side=tk.LEFT)

        self.refresh_img = tk.PhotoImage(file='refresh.gif')
        btn_refresh = tk.Button(toolbar, text='Обновить', bg='#d7d8e0', bd=0, image=self.refresh_img,
                                compound=tk.TOP, command=self.view_records)
        btn_refresh.pack(side=tk.LEFT)

        self.tree = ttk.Treeview(self, columns=('ID_Driver', 'DriverName', 'FIO_Driver', 'PassportData',
                                                'Car', 'Trailer', 'Driver_Phone', 'Available'),
                                 height=15, show='headings')
        self.tree.column("ID_Driver", width=30, anchor=tk.CENTER)
        self.tree.column("DriverName", width=365, anchor=tk.CENTER)
        self.tree.column("FIO_Driver", width=365, anchor=tk.CENTER)
        self.tree.column("PassportData", width=365, anchor=tk.CENTER)
        self.tree.column("Car", width=150, anchor=tk.CENTER)
        self.tree.column("Trailer", width=150, anchor=tk.CENTER)
        self.tree.column("Driver_Phone", width=100, anchor=tk.CENTER)
        self.tree.column("Available", width=100, anchor=tk.CENTER)

        self.tree.heading("ID_Driver", text='ID')
        self.tree.heading("DriverName", text='Название перевозчика')
        self.tree.heading("FIO_Driver", text='ФИО водителя')
        self.tree.heading("PassportData", text='Паспортные данные')
        self.tree.heading("Car", text='Марка и номер тягача')
        self.tree.heading("Trailer", text='Номер полуприцепа')
        self.tree.heading("Driver_Phone", text='Номер телефона')
        self.tree.heading("Available", text='Доступность')

        self.tree.pack()

    # Функция записи данных
    def records(self, drivername, fio, passdata, car, trailer, driver_phone, available):
        self.db.insert_data(drivername, fio, passdata, car, trailer, driver_phone, available)
        # Для отображения данных:
        self.view_records()

    def update_record(self, DriverName, FIO_Driver, PassportData, Car, Trailer, Driver_Phone, Available):
        self.db.c.execute('UPDATE carriers SET DriverName=?, FIO_Driver=?, PassportData=?, Car=?, Trailer=?, '
                          'Driver_Phone=?, Available=? WHERE ID=?',
                          [DriverName, FIO_Driver, PassportData, Car, Trailer, Driver_Phone, Available,
                           self.tree.set(self.tree.selection()[0], '#1')])

        self.db.conn.commit()  # сохранить изменения
        self.view_records()  # отобразить изменения

    def view_records(self):
        self.db.c.execute('''SELECT * FROM carriers''')  # Отображение в тревью
        [self.tree.delete(i) for i in self.tree.get_children()]  # Обновление данных
        [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]

    def delete_records(self):
```

```
        for selection_item in self.tree.selection():  # Нужен цикл для удаления нескольких полей
            self.db.c.execute('''DELETE FROM carriers WHERE id=?''', (self.tree.set(selection_item, '#1'),))

        self.db.conn.commit()
        self.view_records()

    def search_records(self, FIO_Driver):
        FIO_Driver = ('%' + FIO_Driver + '%',)
        self.db.c.execute('''SELECT * FROM carriers WHERE FIO_Driver LIKE ?''',
                          FIO_Driver)  # Обращаемся к базе данных и формируем SQL запрос
        ##Отображаем строки в виджете Treeview
        #1. Очистим содержимое виджета путём использования генератора списка
        #2. В цикле будем получать строки из метода treeview с помощью метода get.children
        #3. После удалять, применяя метод Delete
        [self.tree.delete(i) for i in self.tree.get_children()]
        #Отображаем результаты поиска:
        [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]

    def open_dialog(self):
        Child()

    def open_update_dialog(self):
        Update()

    def open_search_dialog(self):
        Search()


class Child(tk.Toplevel):
    def __init__(self):
        super().__init__(root)
        self.init_child()
        self.view = app

    def init_child(self):
        self.title('Добавить перевозчика')
        self.geometry('500x400+400+300')
        self.resizable(False, False)

        # Виджет ввода данных:

        label_drivername = tk.Label(self, text="Наименование перевозчика")
        label_drivername.place(x=50, y=50)
        self.entry_drivername = ttk.Entry(self)
        self.entry_drivername.place(x=250, y=50)  # Наименование перевозчика

        label_drivername = tk.Label(self, text="ФИО водителя")
        label_drivername.place(x=50, y=90)
        self.entry_fio = ttk.Entry(self)
        self.entry_fio.place(x=250, y=90)  # ФИО Водителя

        label_drivername = tk.Label(self, text="Паспортные данные")
        label_drivername.place(x=50, y=130)
        self.entry_passdata = ttk.Entry(self)
        self.entry_passdata.place(x=250, y=130)  # Паспортные данные

        label_drivername = tk.Label(self, text="Номер и марка тягача")
        label_drivername.place(x=50, y=170)
        self.entry_car = ttk.Entry(self)
        self.entry_car.place(x=250, y=170)  # Марка и номер тягача

        label_drivername = tk.Label(self, text="Номер прицепа")
        label_drivername.place(x=50, y=210)
        self.entry_trailer = ttk.Entry(self)
        self.entry_trailer.place(x=250, y=210)  # Номер прицепа

        label_drivername = tk.Label(self, text="Номер телефона")
        label_drivername.place(x=50, y=250)
        self.entry_driver_phone = ttk.Entry(self)
        self.entry_driver_phone.place(x=250, y=250)

        label_drivername = tk.Label(self, text='Доступность')
        label_drivername.place(x=50, y=290)
        self.combobox = ttk.Combobox(self, values=[u'Доступен', u'Не доступен'])
        self.combobox.current(0)
        self.combobox.place(x=250, y=290)  # Доступность экипажа

        # Кнопка закрытия окна:
        btn_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)
        btn_cancel.place(x=50, y=350)

        # Кнопка "Добавить"
        self.btn_ok = ttk.Button(self, text='Добавить')
        self.btn_ok.place(x=300, y=350)
        self.btn_ok.bind('<Button-1>', lambda event: self.view.records(self.entry_drivername.get(),
                                                                       self.entry_fio.get(),
                                                                       self.entry_passdata.get(),
                                                                       self.entry_car.get(),
                                                                       self.entry_trailer.get(),
                                                                       self.entry_driver_phone.get(),
                                                                       self.combobox.get()))
```

```
        ##END##

        self.grab_set()
        self.focus_set()

        # Создадим класс update, который будет наследоваться от класса child


class Update(Child):
    def __init__(self):
        super().__init__()
        self.init_edit()  # Для отображения пользователю изменений. Вызываем функцию из конструктора класса update.
        self.view = app

    def init_edit(self):
        self.title('Редактировать запись')
        btn_edit = ttk.Button(self, text='Редактировать')
        btn_edit.place(x=300, y=350)
        btn_edit.bind('<Button-1>', lambda event: self.view.update_record(self.entry_drivername.get(),
                                                                          self.entry_fio.get(),
                                                                          self.entry_passdata.get(),
                                                                          self.entry_car.get(),
                                                                          self.entry_trailer.get(),
                                                                          self.entry_driver_phone.get(),
                                                                          self.combobox.get()))
        self.btn_ok.destroy()  # убрали кнопку "OK", потому что заменили на кнопку "Редактировать"

        # Класс базы данных


class Search(tk.Toplevel):
    def __init__(self):
        super().__init__()
        self.__init_search()
        self.view = app

    def __init_search(self):
        self.title('Поиск')
        self.geometry('300x100+400+300')
        self.resizable(False, False)

        label_search = tk.Label(self, text='Поиск')
        label_search.place(x=50, y=20)

        self.entry_search = ttk.Entry(self)
        self.entry_search.place(x=105, y=20, width=150)

        btn_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)
        btn_cancel.place(x=185, y=50)

        btn_search = ttk.Button(self, text='Поиск')
        btn_search.place(x=105, y=50)
        btn_search.bind('<Button-1>', lambda event: self.view.search_records(self.entry_search.get()))
        #Чтобы после нажатия кнопки "Поиск" окно закрывалось:
        btn_search.bind('<Button-1>', lambda event: self.destroy(), add='+')


class DB:
    def __init__(self):
        self.conn = sqlite3.connect('trucking.db')
        self.c = self.conn.cursor()
        self.c.execute(
            '''CREATE TABLE IF NOT EXISTS carriers (id integer primary key, DriverName text, FIO_Driver text,
            PassportData text, Car text, Trailer text, Driver_Phone text, Available text)''')
        self.conn.commit()

    def insert_data(self, drivername, fio, passdata, car, trailer, driver_phone, available):
        self.c.execute(
            '''INSERT INTO carriers(DriverName, FIO_Driver, PassportData, Car, Trailer, Driver_Phone, Available)
            VALUES (?, ?, ?, ?, ?, ?, ?)''',
            (drivername, fio, passdata, car, trailer, driver_phone, available))
        self.conn.commit()


if __name__ == "__main__":
    root = tk.Tk()
    # Для того чтобы обращаться к функицям класса db из класса main. Необходимо создать экземпляр класса db:
    db = DB()
    app = Main(root)
    app.pack()
    root.title("Перевозчики")
    root.geometry("1650x450+300+200")
    root.resizable(False, False)
    root.mainloop()
```