```
from tkinter import *
from tkinter import ttk, messagebox


class Client:
    def __init__(self, name, id, age):
        self.name = name
        self.id = id
        self.age = age

    def __str__(self):
        return f"{self.name} ({self.age})"

    def __repr__(self):
        return str(self)


class BookingError(Exception):
    pass


class MovieNotFoundError(Exception):
    pass

"""
Start Area for Question 1
"""
class Movie:
    def __init__(self, id, name, day, month, year, hour, duration, num_of_tickets):
        self.id = id
        self.name = name
        self.day = day
        self.month = month
        self.year = year
        self.hour = hour
        self.duration = duration
        self.num_of_tickets = num_of_tickets
        self.booked = []

    def __str__(self):
        return f"ID: {self.id} | {self.name} | {self.day}.{self.month}.{self.year} - {self.hour}:00 |
{self.duration} minutes | {self.num_of_tickets}"

    def __repr__(self):
        return str(self)

    def is_available(self):
        return len(self.booked) < self.num_of_tickets

    def book(self, client):
```

```
        if self.is_available():
            self.booked.append(client)
        else:
            raise BookingError(f"The movie ({self.name}) is fully booked")
"""
End Area for Question 1
"""


"""
Start Area for Question 2
"""
class LimitedMovie(Movie):
    def __init__(self, id, name, day, month, year, hour, duration, num_of_tickets, limit_age):
        super().__init__(id, name, day, month, year, hour, duration, num_of_tickets)
        self.limit_age = limit_age

    def __str__(self):
        return f"{super().__str__()} | +{self.limit_age}"

    def book(self, client):
        if client.age < self.limit_age:
            raise BookingError(f"Sorry, It's a great film but not for people under {self.limit_age}")

        super().book(client)

"""
End Area for Question 2
"""

"""
Start Area for Question 3
"""
class Cinema:
    def __init__(self, name):
        self.name = name
        self.movies = []

    def add_movie(self, movie):
        self.movies.append(movie)

    def book_movie(self, movie_id, client):
        for movie in self.movies:
            if movie_id == movie.id:
                movie.book(client)
                return movie

        raise MovieNotFoundError(f"Movie {movie_id} not exist!")

"""
End Area for Question 3
"""
```

```
# creating instances of the classes
cinema = Cinema("Afeka Cinema")

m_1 = Movie(1, "Shrek", 23, 2, 2023, 17, 90, 2)
m_2 = LimitedMovie(2, "The Godfather", 23, 2, 2023, 19, 175, 4, limit_age=16)
m_3 = Movie(3, "Forrest Gump",  24, 2, 2023, 19, 142, 4)
m_4 = LimitedMovie(4, "Fulp Fiction", 24, 2, 2023, 22, 154, 2, 18)

# adding book to the library
cinema.add_movie(m_1)
cinema.add_movie(m_2)
cinema.add_movie(m_3)
cinema.add_movie(m_4)

# List of members
client_1 = Client("John Smith", 1, 15)
client_2 = Client("Barbara Mary", 2, 20)
client_3 = Client("Robert James", 3, 21)
client_4 = Client("Patricia Roberts", 4, 17)
clients = [client_1, client_2, client_3, client_4]


# helper function that return a client from chosen Combobox
def get_client_from_combo():
  name = clients_combo.get().split("(")[0].strip()
  for client in clients:
      if client.name == name:
          return client
  return None


# Define a Tkinter GUI Application
window = Tk()
window.title(cinema.name)

window.geometry("750x300")

lbl = Label(window, text=cinema.name, font=("Ariel bold", 40), fg="blue")
lbl.grid(column=0, row=0)

lbl = Label(window, text="Movies list")
lbl.grid(column=0, row=1)

tasks_list = Listbox(window, width=45, height=5)
tasks_list.grid(row=2, column=0)
for m in cinema.movies:
  tasks_list.insert(END, m)
tasks_list["state"] = "disabled"

clients_lbl = Label(window, text="Client")
clients_lbl.grid(row=3, column=1)
```

```
clients_combo = ttk.Combobox(window, values=clients)
clients_combo.grid(row=4, column=1)

movie_lbl = Label(window, text="Movie ID")
movie_lbl.grid(row=3, column=2)

movie_entry_txt = Entry(window, width=10)
movie_entry_txt.grid(row=4, column=2)
movie_entry_txt.focus()

"""
Start Area for Question 4
"""
def handel_book_it_click():
    movie_id = movie_entry_txt.get()
    if not movie_id.isnumeric():
        messagebox.showinfo(message=f"Movie ID must be a number ({movie_id})")
    movie_id = int(movie_id)
    client = get_client_from_combo()

    try:
        movie = cinema.book_movie(movie_id, client)
        messagebox.showinfo(message=f"{client} booked a ticket to the movie: {movie}!")
    except (BookingError, MovieNotFoundError) as e:
        messagebox.showinfo(message=str(e))


btn = Button(window, text="Book it", width=10, height=2, command=handel_book_it_click)
btn.grid(row=5, column=2)
"""
End Area for Question 4
"""

window.mainloop()
```

```python
import numpy as np
from matplotlib import pyplot as plt

# אל תמחקו את השורה הזו
np.set_printoptions(suppress=True)  # כדי לא לקבל תוצאות ברישום מדעי ( scientific notation)

# Q1
driver_fault = np.genfromtxt("driverfault.csv", delimiter=",", skip_header=1)
print(driver_fault)

# Q2
rows = driver_fault.shape[0]
print(f"There are {rows} rows")

# Q3
sum_all = driver_fault[:, 1:].sum(axis=1)
sum_all = sum_all.reshape(rows, 1)
driver_fault = np.hstack([driver_fault, sum_all])
print(driver_fault)

# Q4
print(f"The max is {driver_fault[:, 8].max()}")

# Q5
regions = {1: 'south', 2: 'east', 3: 'north', 4: 'west', 5: 'center'}
reglist_names = list(regions.values())
# reglist=['south', 'east', 'north', 'west', 'center']
speedlist = []

for k in regions:
    current_region = np.around(driver_fault[driver_fault[:, 0] == k, 1].mean(), 2)
    speedlist.append(current_region)

print(reglist_names)
print(speedlist)

# Q6
plt.bar(reglist_names, speedlist)
plt.show()

# Bonus
mean_accident = driver_fault[:, 8].mean()
norths = driver_fault[(driver_fault[:, 0] == 3) & (driver_fault[:, 8] > mean_accident)]
print(norths)
```