```elixir
#Seventh Homework
#
#Rodrigo Benavente García
#18/10/21

defmodule CSVdata do

  def read_data(filename) do
    filename
    |> File.stream!()
    |> Enum.map(&String.trim/1)
    # Using the 'capture' syntax
    |> Enum.map(&(String.split(&1, ",")))
  end

  def make_numeric([header | data], types) do
    # The first list comprehension works on each row of the matrix
    new_data = for row <- data do
                  # The second comprehension works on each column of the row
                  # pairs the columns with the types, using zip
                  for {value, type} <- Enum.zip(row, types) do
                    # Convert the value in the column depending on the type
                    case type do
                      # Change the type of data
                      :int -> String.to_integer(value)
                      :float -> String.to_float(value)
                      _ -> value
                    end
                  end
                end
    # Join back the header that was separated when getting the arguments
    [header | new_data]
  end

  def write_data(data, filename) do#Recieves the data and the path of a file
in order to save the matrix in the file
    {:ok, out_file} = File.open(filename, [:write])
    for row <- data do
      IO.puts(out_file, Enum.join(row, ","))
    end
    File.close(out_file)
  end

  def sum_column(data, identifier) when identifier == 0 do #Runs until we
reach the desired column
```

```
    #We get the desired column
    [head | tail] = data #Divide data
    res = Tuple.to_list(head) #Erase the tail
    [headR | tailR] = res #Divide the desired column
    total = Enum.sum(tailR)
    res = [headR] ++ [total]
  end

  def sum_column(data, identifier) do#Recieves a matrix and identifier of
the column to use
    [head | tail] = data #Divided data between head and tail
    total = List.delete(data, head) #Deletes head of the data
    sum_column(total, identifier - 1) #Runs sum again until the desired
column is reached
  end

  def separate(data, identifier) when identifier == 0 do #Runs until we
reach the desired column
    #We get the desired column
    [head | tail] = data #Divide data
    res = Tuple.to_list(head) #Erase the tail
    [headR | tailR] = res #Divide the desired column
  end

  def separate(data, identifier) do#Recieves a matrix and identifier of the
column to use
    [head | tail] = data #Divided data between head and tail
    total = List.delete(data, head) #Deletes head of the data
    separate(total, identifier - 1) #Runs sum again until the desired column
is reached
  end

  def aggregate(data, fidentifier, lidentifier) do
    firstList = separate(data, fidentifier)#Gets the lists we want to work
with
    secondList = separate(data, lidentifier)#Gets the lists we want to work
with
    total = Enum.zip(firstList, secondList)#Combines both lists into a list
of tuples

    final = for x <- total do #Make the tuple lists
      Tuple.to_list(x)
    end
    [head | tail] = final#Divide the head and the tail
```

```
    fin = for x <- tail do
      if Enum.at(x, 1) == 1, do: x #Search if the list has 1s
    end

    fin = Enum.filter(fin, & !is_nil(&1)) #Erase Nil values
        |> Enum.frequencies() #Search the frequency of the years in the list
        |> Map.to_list() #Make the map into a list
        |> transform() #Runs transform

    fin = for x <- fin do #Flatten the elements of the matrix
      List.flatten(x)
    end

    fin = for x <- fin do #Delete the second value of the lists inside the
matrix
      List.delete(x, 1)
    end
    fin = Enum.sort(fin, :asc) #Sort the years of the matrix in ascendant
order

    #[head | fin] = final

  end

  def transform(data) do #Transform tuples into a list
    for {x, y} <- data do
      [x, y]
    end
  end

  def main() do #Main function that will run every other function

    filename = "MoviesOnStreamingPlatforms_11_cols_short.csv"
    # The list of types for each column, in the order they appear in the
file
    types = [:int, :int, :str, :int, :str, :str, :str, :int, :int, :int,
:int]

    # Elixir allows using the pipe operator for better readability
    #read_data(filename)
    data = filename
           |> read_data()
           |> make_numeric(types)

    #Create a list to save the lists of the total movies
```

```
    list = []
    #We pass the data as a zip of the elements, and the identifier is the
ninght column
    #Starts counting by 0
    filename2 = "total.csv" #CSV en donde se guarda el total de películas
por plataforma
    #-------------------------------------------------------------------------
----------------------------
    list = [sum_column(Enum.zip(data), 7)] #Add a new list to the matrix
    list = list ++ [sum_column(Enum.zip(data), 8)] #Add second list of total
movies
    list = list ++ [sum_column(Enum.zip(data), 9)] #Add third list of total
movies
    list = list ++ [sum_column(Enum.zip(data), 10)] #Add fourth list of
total movies
          |> write_data("" <> filename2) #Creates a csv with the existing
matrix

    filename3 = "totalNetflix.csv"
    net = aggregate(Enum.zip(data), 3, 7)
        |> write_data("" <> filename3) #Creates a csv with the existing
matrix

    filename3 = "totalHulu.csv"
    net = aggregate(Enum.zip(data), 3, 8)
        |> write_data("" <> filename3) #Creates a csv with the existing
matrix

    filename3 = "totalPrimeVid.csv"
    net = aggregate(Enum.zip(data), 3, 9)
        |> write_data("" <> filename3) #Creates a csv with the existing
matrix

    filename3 = "totalDisney.csv"
    net = aggregate(Enum.zip(data), 3, 10)
        |> write_data("" <> filename3) #Creates a csv with the existing
matrix

  end

end
```