**Ruby Big Code**

**Pal**

**Part A**

1- What are the most popular and least popular superpowers for superheroes with at least 1 superpower?

Approach I took:

I filter superheroes who have at least one superpower and then gather all these superpowers into a single array using the (flat\_map method)- GPT’s assistant. So now I group these superpowers by their values using group\_by resulting in a hash where each unique superpower is a key and the values are arrays containing all instances of that power. Ultimately the goal for me was that the code identified the most popular superpower by using max\_by to compare elements based on the count of occurrences of each power of the super heroes. The puts statement in the end provides the name of the most popular superpower along with the number of times it appears in the filtered data. And done yayyy :)

Code

| require 'json'  # # JSON data for superheroes and their powers based on Claudio Davi's Kaggle dataset # https://www.kaggle.com/datasets/claudiodavi/superhero-set/ # # Mainly I converted it from CSV to JSON #  class Hero  attr\_accessor :name, :gender, :eyecolor, :race, :haircolor, :height, :publisher, :skincolor, :alignment, :weight, :powers   def initialize(data)  data.each do |key, value|  instance\_variable\_set("@#{key.downcase}", value) if respond\_to?("#{key.downcase}=")  end  end   def to\_s  return instance\_variables.map { |var| "#{var} = #{instance\_variable\_get(var)}" }.join(',')  end  end  def sample heroes  # how many heroes have the power AcceleratedHealing?  puts heroes.filter { |hero| hero.powers.include?("AcceleratedHealing") }.count   # how many heroes have zero powers?  puts heroes.filter { |hero| hero.powers.size == 0 }.count   end  def filter\_heroes\_with\_powers(heroes)  # I am filtering the heroes with at least one power first (so power count could be greater than or equal to 1 anything works) heroes\_powers1ormore = heroes.filter { |hero| hero.powers.size >= 1 }  # So I got .flat\_map from GPT and what this is doing is basically combining all the powers into a single array # Now I have an array of all powers from all heroes\_with\_powers1ormore all\_powers = heroes\_powers1ormore.flat\_map { |hero| hero.powers }  # group\_by here groups all the powers by their value (so basically it is grouping all the same powers together)  # so result would be a hash in this case and where key is the same powers  grouped\_powers = all\_powers.group\_by { |power| power }  # maxby helping me find the most popular power by comparing power occu  most\_popular\_power = grouped\_powers.max\_by { |power, power\_occurrences| power\_occurrences.size }  puts"Pal's Popular Superhero power was:" puts most\_popular\_power[0] puts"It occurred:" puts most\_popular\_power[1].size puts"times"  end     def main  # read each row of heroes\_information.csv  heroes = Array.new  JSON.parse(File.read("heroes.json")).each do |row|  heroes << Hero.new(row)  end   filter\_heroes\_with\_powers(heroes)  # run the sample code   end  if \_\_FILE\_\_ == $0  main end |
| --- |

2- What is the average number of superpowers per superhero?

Approach I took:

The approach I took involves first calculating the total number of superpowers by summing the sizes of the powers array for each hero resulting in the totalPowers variable I had. And then I determined the total number of superheroes in the dataset using the totalHeroes variable which was easy cause I just found the size of the heroes.size. Next step was to compute the average number of superpowers by dividing the total number of powers by the total number of heroes. Calculated the average number of powers yay !!

| require 'json'  # # JSON data for superheroes and their powers based on Claudio Davi's Kaggle dataset # https://www.kaggle.com/datasets/claudiodavi/superhero-set/ # # Mainly I converted it from CSV to JSON #  class Hero  attr\_accessor :name, :gender, :eyecolor, :race, :haircolor, :height, :publisher, :skincolor, :alignment, :weight, :powers   def initialize(data)  data.each do |key, value|  instance\_variable\_set("@#{key.downcase}", value) if respond\_to?("#{key.downcase}=")  end  end   def to\_s  return instance\_variables.map { |var| "#{var} = #{instance\_variable\_get(var)}" }.join(',')  end  end  def sample heroes  # how many heroes have the power AcceleratedHealing?  puts heroes.filter { |hero| hero.powers.include?("AcceleratedHealing") }.count   # how many heroes have zero powers?  puts heroes.filter { |hero| hero.powers.size == 0 }.count   end  def filter\_heroes\_with\_powers(heroes)  # I am filtering the heroes with at least one power first (so power count could be greater than or equal to 1 anything works) heroes\_powers1ormore = heroes.filter { |hero| hero.powers.size >= 1 }  # So I got .flat\_map from GPT and what this is doing is basically combining all the powers into a single array # Now I have an array of all powers from all heroes\_with\_powers1ormore all\_powers = heroes\_powers1ormore.flat\_map { |hero| hero.powers }  # group\_by here groups all the powers by their value (so basically it is grouping all the same powers together)  # so result would be a hash in this case and where key is the same powers  grouped\_powers = all\_powers.group\_by { |power| power }  # maxby helping me find the most popular power by comparing power occu  most\_popular\_power = grouped\_powers.max\_by { |power, power\_occurrences| power\_occurrences.size }  puts"Pal's Popular Superhero power was:" puts most\_popular\_power[0] puts"It occurred:" puts most\_popular\_power[1].size puts"times"  end  def averageSuperpowers(heroes)  #I am finding the total number of superpowers by summing up the size of the powers array for each hero here  # and that all goes in totalPowers  totalPowers = heroes.sum { |hero| hero.powers.size }  #heroes in the data, finding their size  totalHeroes = heroes.size  if totalHeroes  #just dividing the total number of powers by the total number of heroes  averagePowers = totalPowers/ totalHeroes  puts "average number of powers for heroes:"  puts averagePowers  end  end  def main  # read each row of heroes\_information.csv  heroes = Array.new  JSON.parse(File.read("heroes.json")).each do |row|  heroes << Hero.new(row)  end   filter\_heroes\_with\_powers(heroes)  averageSuperpowers(heroes)  # run the sample code   end  if \_\_FILE\_\_ == $0  main end |
| --- |

3- What is the average number of superpowers for superheroes with at least 1 superpower?

\* remove superheroes with zero superpowers

Approach I took:

Approach I took involved first filtering out superheroes with at least one superpower using the heroes\_powers1ormore variable I made earlier in que 1. Then I checked if there is at least one superhero in this filtered array by using .any?- GPT told me about this.any?. So if there were superheroes I calculated the average number of superpowers among these filtered heroes and if no superheroes with at least one superpower meaning.any? is false I printed an error kind of message. It did work but the answer is 8 for both of them.

| require 'json'  # # JSON data for superheroes and their powers based on Claudio Davi's Kaggle dataset # https://www.kaggle.com/datasets/claudiodavi/superhero-set/ # # Mainly I converted it from CSV to JSON #  class Hero  attr\_accessor :name, :gender, :eyecolor, :race, :haircolor, :height, :publisher, :skincolor, :alignment, :weight, :powers   def initialize(data)  data.each do |key, value|  instance\_variable\_set("@#{key.downcase}", value) if respond\_to?("#{key.downcase}=")  end  end   def to\_s  return instance\_variables.map { |var| "#{var} = #{instance\_variable\_get(var)}" }.join(',')  end  end  def sample heroes  # how many heroes have the power AcceleratedHealing?  puts heroes.filter { |hero| hero.powers.include?("AcceleratedHealing") }.count   # how many heroes have zero powers?  puts heroes.filter { |hero| hero.powers.size == 0 }.count   end  def filter\_heroes\_with\_powers(heroes)  # I am filtering the heroes with at least one power first (so power count could be greater than or equal to 1 anything works) heroes\_powers1ormore = heroes.filter { |hero| hero.powers.size >= 1 }  # So I got .flat\_map from GPT and what this is doing is basically combining all the powers into a single array # Now I have an array of all powers from all heroes\_with\_powers1ormore all\_powers = heroes\_powers1ormore.flat\_map { |hero| hero.powers }  # group\_by here groups all the powers by their value (so basically it is grouping all the same powers together)  # so result would be a hash in this case and where key is the same powers  grouped\_powers = all\_powers.group\_by { |power| power }  # maxby helping me find the most popular power by comparing power occu  most\_popular\_power = grouped\_powers.max\_by { |power, power\_occurrences| power\_occurrences.size }  puts"Pal's Popular Superhero power was:" puts most\_popular\_power[0] puts"It occurred:" puts most\_popular\_power[1].size puts"times"  end  def averageSuperpowers(heroes)  #I am finding the total number of superpowers by summing up the size of the powers array for each hero here  # and that all goes in totalPowers  totalPowers = heroes.sum { |hero| hero.powers.size }  #heroes in the data, finding their size  totalHeroes = heroes.size  if totalHeroes  #just dividing the total number of powers by the total number of heroes  averagePowers = totalPowers.to\_f/ totalHeroes  puts "average number of powers for superheroes:"  puts averagePowers  end  end def averageSuperpowers2(heroes)  # did the same for the first question so lets filter the zeroes(power count greater than or equal to 1)  heroes\_powers1ormore = heroes.select { |hero| hero.powers.size >= 1 }    if heroes\_powers1ormore.any?  #I am using .any? cause in ruby it tells us whether there is at least one superhero in the heroes\_powers1ormore  #so it will give true if the statemen is true  # amd now just calculating total number of superpowers for filtered heroes  totalPowers2 = heroes\_powers1ormore.sum { |hero| hero.powers.size}  totalHeroes2 = heroes\_powers1ormore.size  average\_powers = totalPowers2 / totalHeroes2    puts "average number of superpowers for superheroes with at least 1 superpower" puts average\_powers  else  # if I go in the else statement this means that there are no superheroes with at least 1 superpower puts "no heroes found with at least 1 superpower"  end  end   def main  # read each row of heroes\_information.csv  heroes = Array.new  JSON.parse(File.read("heroes.json")).each do |row|  heroes << Hero.new(row)  end   filter\_heroes\_with\_powers(heroes)  averageSuperpowers(heroes)  averageSuperpowers2(heroes)  # run the sample code   end  if \_\_FILE\_\_ == $0  main end |
| --- |

4- What are the most popular and least popular superpowers by publisher?

\* this helps us know things like whether super strength is more popular with DC Comics than with Marvel

Approach I took:

So this que was a lil tricky because we needed to find the most popular and least popular superpowers by each publisher, so my approach starts by grouping heroes based on their respective publishers. This grouping allowed me to analyze the superpowers within each publishers heroes separately. And then for each publisher I then flatten all the superpowers of their heroes into a single array using flat\_map (used this for my previous que#1 so it wasn't hard this time). I used max\_by to find the most popular superpower and min\_by to find the least popular one. My print was a little weird but I think I was successfully able to gain insights into the popularity of superpowers across different publishers for all the DC Comics and Marvel thingy you wanted.

| require 'json'  # # JSON data for superheroes and their powers based on Claudio Davi's Kaggle dataset # https://www.kaggle.com/datasets/claudiodavi/superhero-set/ # # Mainly I converted it from CSV to JSON #  class Hero  attr\_accessor :name, :gender, :eyecolor, :race, :haircolor, :height, :publisher, :skincolor, :alignment, :weight, :powers   def initialize(data)  data.each do |key, value|  instance\_variable\_set("@#{key.downcase}", value) if respond\_to?("#{key.downcase}=")  end  end   def to\_s  return instance\_variables.map { |var| "#{var} = #{instance\_variable\_get(var)}" }.join(',')  end  end  def sample heroes  # how many heroes have the power AcceleratedHealing?  puts heroes.filter { |hero| hero.powers.include?("AcceleratedHealing") }.count   # how many heroes have zero powers?  puts heroes.filter { |hero| hero.powers.size == 0 }.count   end  def filter\_heroes\_with\_powers(heroes)  # I am filtering the heroes with at least one power first (so power count could be greater than or equal to 1 anything works) heroes\_powers1ormore = heroes.filter { |hero| hero.powers.size >= 1 }  # So I got .flat\_map from GPT and what this is doing is basically combining all the powers into a single array # Now I have an array of all powers from all heroes\_with\_powers1ormore all\_powers = heroes\_powers1ormore.flat\_map { |hero| hero.powers }  # group\_by here groups all the powers by their value (so basically it is grouping all the same powers together)  # so result would be a hash in this case and where key is the same powers  grouped\_powers = all\_powers.group\_by { |power| power }  # maxby helping me find the most popular power by comparing power occu  most\_popular\_power = grouped\_powers.max\_by { |power, power\_occurrences| power\_occurrences.size }  puts"Pal's Popular Superhero power was:" puts most\_popular\_power[0] puts"It occurred:" puts most\_popular\_power[1].size puts"times"  end  def averageSuperpowers(heroes)  #I am finding the total number of superpowers by summing up the size of the powers array for each hero here  # and that all goes in totalPowers  totalPowers = heroes.sum { |hero| hero.powers.size }  #heroes in the data, finding their size  totalHeroes = heroes.size  if totalHeroes  #just dividing the total number of powers by the total number of heroes  averagePowers = totalPowers/ totalHeroes  puts "average number of powers for superheroes:"  puts averagePowers  end  end def averageSuperpowers2(heroes)  # did the same for the first question so lets filter the zeroes(power count greater than or equal to 1)  heroes\_powers1ormore = heroes.select { |hero| hero.powers.size >= 1 }    if heroes\_powers1ormore.any?  #I am using .any? cause in ruby it tells us whether there is at least one superhero in the heroes\_powers1ormore  #so it will give true if the statemen is true  # amd now just calculating total number of superpowers for filtered heroes  totalPowers2 = heroes\_powers1ormore.sum { |hero| hero.powers.size}  totalHeroes2 = heroes\_powers1ormore.size  average\_powers = totalPowers2 / totalHeroes2    puts "average number of superpowers for superheroes with at least 1 superpower" puts average\_powers  else  # if I go in the else statement this means that there are no superheroes with at least 1 superpower puts "no heroes found with at least 1 superpower"  end  end  def popularSuperpowersByPublisher(heroes) # so group\_by i am using again to group the heroes by their publisher  publishers = heroes.group\_by { |hero| hero.publisher }  # So for each publisher I am calculating the popularity of superpowers- used GPT to help me out with this statement cause I was getting this wrong  publishers.each do |publisher, publisher\_heroes| # again used fla\_map to flatten the powers of all heroes for this publisher ( have used this before )  allPowers = publisher\_heroes.flat\_map { |myhero| myhero.powers }  # now calc the most popular superpower for each publisher  famousPower = allPowers.max\_by { |powerofhero| allPowers.count(powerofhero) }    # Calculate the least popular superpower  screwYouPower = allPowers.min\_by { |powerofhero| allPowers.count(powerofhero) }    puts "For #{publisher}:"  puts "Most Popular Superpower:"  puts famousPower  puts "appears"  puts allPowers.count(famousPower)  puts"times"  puts "For #{publisher}:"  puts "Least Popular Superpower:"  puts screwYouPower  puts "appears"  puts allPowers.count(screwYouPower)  puts "times"  puts "For #{publisher}:"  end  end   def main  # read each row of heroes\_information.csv  heroes = Array.new  JSON.parse(File.read("heroes.json")).each do |row|  heroes << Hero.new(row)  end   filter\_heroes\_with\_powers(heroes)  averageSuperpowers(heroes)  averageSuperpowers2(heroes)  popularSuperpowersByPublisher(heroes)   # run the sample code   end  if \_\_FILE\_\_ == $0  main end |
| --- |

5- What is the breakdown by gender of the superheroes?

Approach I took:

For the breakdownByGender method I made, I first initialized counters for male and female superheroes and calc the total number of heroes in the dataset/sumOfHeroes. Then, I looped through the list of heroes using OR to check for both Male and male or Female and female for caps checking and then obv counting male and female heroes. After the loop I just calculated the percentages of male and female heroes based on the counts and the total number of heroes. The percentages were initially 0% but I changed it by using .to\_f thing which GPT helped me with. Anyways it works now but I still don’t know why the percentage doesn’t sum up to 100 lol.

| require 'json'  # # JSON data for superheroes and their powers based on Claudio Davi's Kaggle dataset # https://www.kaggle.com/datasets/claudiodavi/superhero-set/ # # Mainly I converted it from CSV to JSON #  class Hero  attr\_accessor :name, :gender, :eyecolor, :race, :haircolor, :height, :publisher, :skincolor, :alignment, :weight, :powers   def initialize(data)  data.each do |key, value|  instance\_variable\_set("@#{key.downcase}", value) if respond\_to?("#{key.downcase}=")  end  end   def to\_s  return instance\_variables.map { |var| "#{var} = #{instance\_variable\_get(var)}" }.join(',')  end  end  def sample heroes  # how many heroes have the power AcceleratedHealing?  puts heroes.filter { |hero| hero.powers.include?("AcceleratedHealing") }.count   # how many heroes have zero powers?  puts heroes.filter { |hero| hero.powers.size == 0 }.count   end  def filter\_heroes\_with\_powers(heroes)  # I am filtering the heroes with at least one power first (so power count could be greater than or equal to 1 anything works) heroes\_powers1ormore = heroes.filter { |hero| hero.powers.size >= 1 }  # So I got .flat\_map from GPT and what this is doing is basically combining all the powers into a single array # Now I have an array of all powers from all heroes\_with\_powers1ormore all\_powers = heroes\_powers1ormore.flat\_map { |hero| hero.powers }  # group\_by here groups all the powers by their value (so basically it is grouping all the same powers together)  # so result would be a hash in this case and where key is the same powers  grouped\_powers = all\_powers.group\_by { |power| power }  # maxby helping me find the most popular power by comparing power occu  most\_popular\_power = grouped\_powers.max\_by { |power, power\_occurrences| power\_occurrences.size }  puts"Pal's Popular Superhero power was:" puts most\_popular\_power[0] puts"It occurred:" puts most\_popular\_power[1].size puts"times"  end  def averageSuperpowers(heroes)  #I am finding the total number of superpowers by summing up the size of the powers array for each hero here  # and that all goes in totalPowers  totalPowers = heroes.sum { |hero| hero.powers.size }  #heroes in the data, finding their size  totalHeroes = heroes.size  if totalHeroes  #just dividing the total number of powers by the total number of heroes  averagePowers = totalPowers/ totalHeroes  puts "average number of powers for superheroes:"  puts averagePowers  end  end def averageSuperpowers2(heroes)  # did the same for the first question so lets filter the zeroes(power count greater than or equal to 1)  heroes\_powers1ormore = heroes.select { |hero| hero.powers.size >= 1 }    if heroes\_powers1ormore.any?  #I am using .any? cause in ruby it tells us whether there is at least one superhero in the heroes\_powers1ormore  #so it will give true if the statemen is true  # amd now just calculating total number of superpowers for filtered heroes  totalPowers2 = heroes\_powers1ormore.sum { |hero| hero.powers.size}  totalHeroes2 = heroes\_powers1ormore.size  average\_powers = totalPowers2 / totalHeroes2    puts "average number of superpowers for superheroes with at least 1 superpower" puts average\_powers  else  # if I go in the else statement this means that there are no superheroes with at least 1 superpower puts "no heroes found with at least 1 superpower"  end  end   def popularSuperpowersByPublisher(heroes)  # so group\_by i am using again to group the heroes by their publisher  publishers = heroes.group\_by { |hero| hero.publisher }    # So for each publisher I am calculating the popularity of superpowers- used GPT to help me out with this statement cause I was getting this wrong  publishers.each do |publisher, publisher\_heroes|  # again used fla\_map to flatten the powers of all heroes for this publisher ( have used this before )  allPowers = publisher\_heroes.flat\_map { |myhero| myhero.powers }  # now calc the most popular superpower for each publisher  famousPower = allPowers.max\_by { |powerofhero| allPowers.count(powerofhero) }    # Calculate the least popular superpower  screwYouPower = allPowers.min\_by { |powerofhero| allPowers.count(powerofhero) }    puts "For #{publisher}:"  puts "Most Popular Superpower:"  puts famousPower  puts "appears"  puts allPowers.count(famousPower)  puts"times"  puts "For #{publisher}:"  puts "Least Popular Superpower:"  puts screwYouPower  puts "appears"  puts allPowers.count(screwYouPower)  puts "times"  puts "For #{publisher}:"  end  end       def breakdownByGender(heroes)  # making a counter for male or female maleHeroesCount = 0 femaleHeroesCount = 0 sumOfHeroes=heroes.length  # I will just calc the total number of heroes first cause this will be using finidng the % # I am just going through the list of heroes and count superheroes by gender  heroes.each do |hero|  if hero.gender == "Male" || hero.gender == "male"  maleHeroesCount = maleHeroesCount +1  elsif hero.gender == "Female" || hero.gender == "female"  #i was doing 'else if' for so long and it wasn't working lol  femaleHeroesCount = femaleHeroesCount+1  else end end    male\_percentage = (maleHeroesCount.to\_f / heroes.length) \* 100  female\_percentage = (femaleHeroesCount.to\_f / heroes.length) \* 100   puts "gender bd:"  puts "sum of heroes: #{heroes.length}"  puts "male heroes: #{maleHeroesCount} \_ #{male\_percentage}"  puts "demale heroes: #{femaleHeroesCount} \_ #{female\_percentage}"  #using gpt found a way to put two print/puts ststement on one line using # so now this looks better when I PRINT THIS    end   def main  # read each row of heroes\_information.csv  heroes = Array.new  JSON.parse(File.read("heroes.json")).each do |row|  heroes << Hero.new(row)  end   filter\_heroes\_with\_powers(heroes)  averageSuperpowers(heroes)  averageSuperpowers2(heroes)  popularSuperpowersByPublisher(heroes)  breakdownByGender(heroes)   # run the sample code   end  if \_\_FILE\_\_ == $0  main end |
| --- |

**Part B**

**Question 1: How many superheroes have the power of flight?**

Explanation: I am just counting the heroes who possess the power of flight

| def superheroesWithFlightPower(heroes)  flightHeroes = heroes.filter { |pal| pal.powers.any? { |power| power == "Flight" } }  a=flightHeroes.size puts "# heroes with 'flight' power:" puts a  end |
| --- |

**Question 2: What is the average weight of the superheroes in the dataset?**

Explanation: more of calculating averages so for weight of all superheroes in the datast because it will provide insights into their physical characteristics kind of

| def averageWeight(heroes)  totalWeight = heroes.sum {|pal| pal.weight.to\_f}  totalHeroes = heroes.length  meanWeight = (totalWeight.to\_f / totalHeroes) puts "average weight heroes:" puts meanWeight puts "pounds/kilo idk"  end |
| --- |

**Part C** - Generating new superheroes with AI

1. **Using an AI Service such as ChatGPT or POE, generate new superheroes in the same JSON format. Tell the AI that you will give it JSON data about superheroes and that you want new superheroes in that format. Give the AI the JSON format as a prompt; it should acknowledge that is has received it**

Done

1. **Then ask for 20 new superheroes! Analyze these superheroes, refine your prompt with feedback or further instruction, and generate new superheroes. This is common when generating data! You usually must give the AI additional feedback or instruction to get better data.**

Yes. Here we go:  
  
[

{

"name": "Pal",

"gender": "Male",

"eyecolor": "emerald",

"race": "Android",

"haircolor": "None",

"height": "182.0",

"publisher": "FutureTech Comics",

"skincolor": "silver",

"alignment": "good",

"weight": "200.0",

"powers": [

"Super Intelligence",

"Energy Projection",

"Flight",

"Technopathy",

"Data Manipulation",

"Holographic Projection"

]

},

{

"name": "Crimson Comet",

"gender": "Female",

"eyecolor": "crimson",

"race": "Speedster",

"haircolor": "Auburn",

"height": "168.0",

"publisher": "Velocity Comics",

"skincolor": "fair",

"alignment": "good",

"weight": "54.0",

"powers": [

"Super Speed",

"Time Manipulation",

"Intangibility",

"Speed Force Aura",

"Molecular Acceleration"

]

},

{

"name": "Nebula Knight",

"gender": "Male",

"eyecolor": "black",

"race": "Celestial",

"haircolor": "White",

"height": "210.0",

"publisher": "Stellar Stories",

"skincolor": "dark blue",

"alignment": "good",

"weight": "120.0",

"powers": [

"Cosmic Awareness",

"Energy Manipulation",

"Gravity Control",

"Stellar Winds",

"Wormhole Creation"

]

},

{

"name": "Feral Spirit",

"gender": "Female",

"eyecolor": "amber",

"race": "Werewolf",

"haircolor": "Brown",

"height": "177.0",

"publisher": "Lunar Legends",

"skincolor": "tan",

"alignment": "neutral",

"weight": "75.0",

"powers": [

"Enhanced Senses",

"Regeneration",

"Animal Control",

"Superhuman Agility",

"Night Vision"

]

},

{

"name": "Arcane Mistress",

"gender": "Female",

"eyecolor": "silver",

"race": "Sorceress",

"haircolor": "Platinum",

"height": "165.0",

"publisher": "Mystic Arts Comics",

"skincolor": "light",

"alignment": "good",

"weight": "50.0",

"powers": [

"Spell Casting",

"Elemental Control",

"Teleportation",

"Illusion Casting",

"Enchantment"

]

},

{

"name": "Tempest Fury",

"gender": "Male",

"eyecolor": "stormy gray",

"race": "Storm Elemental",

"haircolor": "Dark Blue",

"height": "180.0",

"publisher": "Elemental Comics",

"skincolor": "pale",

"alignment": "good",

"weight": "85.0",

"powers": [

"Weather Manipulation",

"Electrokinesis",

"Flight",

"Super Strength",

"Atmokinesis"

]

},

{

"name": "Void Walker",

"gender": "Non-binary",

"eyecolor": "white",

"race": "Dimensional",

"haircolor": "Void Black",

"height": "190.0",

"publisher": "Cosmos Comics",

"skincolor": "gray",

"alignment": "neutral",

"weight": "0.0 (weightless)",

"powers": [

"Dimensional Travel",

"Invisibility",

"Telekinesis",

"Reality Warping",

"Intangibility"

]

},

{

"name": "Iron Sentinel",

"gender": "Male",

"eyecolor": "blue",

"race": "Cyborg",

"haircolor": "None",

"height": "198.0",

"publisher": "Cybernet Comics",

"skincolor": "metallic gray",

"alignment": "good",

"weight": "310.0",

"powers": [

"Super Strength",

"Durability",

"Technopathy",

"Flight

]

},

{

"name": "Mystic Shadow",

"gender": "Female",

"eyecolor": "violet",

"race": "Mutant",

"haircolor": "Black",

"height": "170.0",

"publisher": "Cosmic Publications",

"skincolor": "pale",

"alignment": "good",

"weight": "58.0",

"powers": [

"Telekinesis",

"Invisibility",

"Teleportation",

"Astral Projection",

"Psychic Shields"

]

},

{

"name": "Quantum Knight",

"gender": "Male",

"eyecolor": "green",

"race": "Alien",

"haircolor": "Silver",

"height": "200.0",

"publisher": "Galactic Guardians",

"skincolor": "metallic",

"alignment": "good",

"weight": "300.0",

"powers": [

"Time Travel",

"SuperStrength",

"Energy Blasts",

"Flight",

"Dimensional Travel"

]

},

{

"name": "Solar Flare",

"gender": "Non-binary",

"eyecolor": "gold",

"race": "Solar Entity",

"haircolor": "Flames",

"height": "183.0",

"publisher": "Nova Force Comics",

"skincolor": "orange",

"alignment": "good",

"weight": "80.0",

"powers": [

"Pyrokinesis",

"Flight",

"Energy Absorption",

"SuperStrength",

"Light Manipulation",

"Heat Generation"

]

},

{

"name": "Echo Phantom",

"gender": "Female",

"eyecolor": "indigo",

"race": "Ghostly Entity",

"haircolor": "White",

"height": "175.0",

"publisher": "Spectral Comics",

"skincolor": "translucent",

"alignment": "good",

"weight": "0.0 (intangible)",

"powers": [

"Intangibility",

"Invisibility",

"Sonic Scream",

"Flight",

"Ectoplasm Manipulation"

]

},

{

"name": "Lunar Blade",

"gender": "Male",

"eyecolor": "silver",

"race": "Moon-Born",

"haircolor": "Black",

"height": "182.0",

"publisher": "Celestial Warriors",

"skincolor": "pale",

"alignment": "good",

"weight": "77.0",

"powers": [

"Lunar Empowerment",

"Night Vision",

"Enhanced Agility",

"Weapon Master",

"Animal Communication"

]

},

{

"name": "Terra Firma",

"gender": "Non-binary",

"eyecolor": "green",

"race": "Earth Elemental",

"haircolor": "Grass Green",

"height": "190.0",

"publisher": "Gaia Comics",

"skincolor": "stone",

"alignment": "good",

"weight": "220.0",

"powers": [

"Geokinesis",

"Super Strength",

"Nature Control",

"Terrakinesis",

"Regeneration"

]

},

{

"name": "Quantum Spirit",

"gender": "Female",

"eyecolor": "hazel",

"race": "Quantum Apparition",

"haircolor": "Light Brown",

"height": "178.0",

"publisher": "Quantum Comics",

"skincolor": "ethereal",

"alignment": "good",

"weight": "0.0 (incorporeal)",

"powers": [

"Quantum Manipulation",

"Teleportation",

"Phasing",

"Time Travel",

"Quantum Tunneling"

]

},

{

"name": "Chrono Warden",

"gender": "Male",

"eyecolor": "timeless bronze",

"race": "Temporal Guardian",

"haircolor": "Sandy Blonde",

"height": "188.0",

"publisher": "Temporal Tales",

"skincolor": "bronze",

"alignment": "good",

"weight": "88.0",

"powers": [

"Time Manipulation",

"Age Control",

"Super Speed",

"Temporal Lock",

"Chronokinesis"

]

},

{

"name": "Gale Force",

"gender": "Female",

"eyecolor": "sky blue",

"race": "Wind Spirit",

"haircolor": "Silver",

"height": "169.0",

"publisher": "Aero Comics",

"skincolor": "light",

"alignment": "good",

"weight": "57.0",

"powers": [

"Aerokinesis",

"Flight",

"Wind Generation",

"Sound Nullification",

"Atmospheric Adaptation"

]

},

{

"name": "Astral Knight",

"gender": "Male",

"eyecolor": "galactic black",

"race": "Starborn",

"haircolor": "Midnight Blue",

"height": "193.0",

"publisher": "Cosmic Knights",

"skincolor": "dark",

"alignment": "good",

"weight": "100.0",

"powers": [

"Astrokinesis",

"Space Survival",

"Gravity Control",

"Stellar Beam",

"Cosmic Awareness"

]

},

{

"name": "Inferno Witch",

"gender": "Female",

"eyecolor": "fire red",

"race": "Pyro-Sorcerer",

"haircolor": "Blazing Orange",

"height": "167.0",

"publisher": "Arcane Fire Comics",

"skincolor": "red",

"alignment": "neutral",

"weight": "59.0",

"powers": [

"Pyrokinesis",

"Spell Casting",

"Fire Manipulation",

"Thermal Resistance",

"Flight",

"Mystic Arts Mastery"

]

}  
]

1. **Once you are happy with your new superheroes, use your ability to read the JSON file, plus Ruby code from Part 1 and 2, and possibly some new Ruby code, to analyze these new superheroes**

So I made a different json format file in VS code for my new 20 superheroes. I put that in heroes2.json and then ran my main for heroes2.json rather than heroes.json to do data analytics on the new dataset I have.

| def main  # read each row of heroes\_information.csv  heroes = Array.new  JSON.parse(File.read("heroes2.json")).each do |row|  heroes << Hero.new(row)  end |
| --- |

I checked all the things I did for the heroes.json. For example I figured out that for these 20 superheroes. Look at the results for a few things now:  
  
**1-** # heroes with 'flight' power = 8  
**2**- average weight heroes = 101.73684210526316 pounds/kilo idk ( the average weight is pretty high compared to the heroes.json data

**3-** male heroes: \_ 42.1052631578947% | female heroes: \_ 42.10526315789473%

1. **What was the "best" superhero generated, and why?**

For the best I would say   
  
Nebula Knight stands out for me because his superpowers are really good ngl. He's like a space wizard who knows everything happening in space and can move things with his mind. Not just that that dude can even control gravity, that's like deciding how heavy or light things should be. He can also create space whirlpools to jump long distances in space in a snap which I personally think is really cool. Imagine him stopping an asteroid from hitting Earth just by thinking about it or taking you on a space trip without needing a spaceship. He's got the kind of powers that aren't just about fighting bad guys but also about exploring and protecting the universe which makes him pretty special .He could be a great friend :)

1. **What was the "worst" superhero generated, and why?**

LOL, Pal might not be the best superhero because he's like a robot (android eww) and robots can have a hard time understanding people's feelings. This makes it tough for him to figure out the best way to help someone who is upset or scared because sometimes you need to get the feelings part right. Because he's all about technology, if someone messes with his programming or if there's no technology around he might not be very helpful. Imagine if your phone stops working when there's no signal and that's Pal in places without machines or gadgets. Plus, Pal doesn't have hair or a regular human look, which might make it harder for people to feel close to him or think of him as their favorite hero. He's super smart and strong but he might not be the hero you'd want to hang out with :(

1. **What did you have to do with your prompts to get better superheroes?**

I just told GPT to generate superheroes with more relatability so like more human-like qualities or a backstory that people can connect with emotionally so it gave me more human things and I think that would make superheroes better. We don’t want only robots, I need someone to relate with.

1. **Be sure to submit a link to your AI transcript!**

[SuperHeroes Generated](https://chat.openai.com/share/91e4ba97-3d5e-46dc-ac73-30ef0c810fdb)