# Homework Assignment 2

[The Art of Analyzing Big Data - The Data Scientist's Toolbox]

By Dr. Michael Fire

## Dataset Collecting

**Question 1**: Write a function that collects all titles and number of votes for each title of a given hacker news page (15pt)

```
import requests

def get_hacker_news_titles_and_votes():
    url = 'https://hacker-news.firebaseio.com/v0/topstories.json?print=pretty'
    response = requests.get(url)

    if response.status_code == 200:
        top_story_ids = response.json()[:20]  # Fetching top 20 stories for demonstration
        titles_and_votes = []

        for story_id in top_story_ids:
            story_url = f'https://hacker-news.firebaseio.com/v0/item/{story_id}.json?print=pretty'
            story_response = requests.get(story_url)

            if story_response.status_code == 200:
                story_data = story_response.json()
                title = story_data.get('title', '')
                votes = story_data.get('score', 0)
                user = story_data.get('by', '')
                age = story_data.get('time', '')

                titles_and_votes.append({
                    'title': title,
                    'votes': votes,
                    'user': user,
                    'age': age
                })

        return titles_and_votes
    else:
        print(f"Error fetching top stories. Status code: {response.status_code}")
        return []

# Example usage
hacker_news_data = get_hacker_news_titles_and_votes()

for index, item in enumerate(hacker_news_data, start=1):
    print(f"{index}. {item['title']} - {item['votes']} points by {item['user']} {item['age']}")

```

```
1. Show HN: #!/usr/bin/env docker run - 331 points by adtac 1705202405
2. Citadel, a Calibre-compatible eBook management app - 177 points by phildenhoff 1705213360
3. Type information for faster Python C extensions - 43 points by ingve 1705225728
4. What Was ISDN? - 96 points by ecliptik 1705121345
5. OpenD, a D language fork that is open to your contributions - 174 points by mepian 1705196099
6. When "blocked indefinitely" is not indefinite - 17 points by g0xA52A2A 1705228885
7. The Ultimate Docker Cheat Sheet - 74 points by jpeer264 1705225524
8. Smart binoculars can identify 9k birds - 119 points by thunderbong 1705040052
9. BuildZoom (better way to build custom homes) Is hiring a Growth Associate - 1 points by the_economist 1705233860
10. Towards Modern Development of Cloud Applications (2023) - 96 points by signa11 1705216656
11. Building a fully local LLM voice assistant to control my smart home - 534 points by JohnTheNerd 1705183930
12. Posthog is closing their Slack community in favor of forum - 232 points by vmatsiiako 1705205576
13. Dynamic Programming is not Black Magic - 56 points by qsantos 1705225335
14. Ask HN: Does Cloudflare block HN comments if you have code blocks in a reply? - 245 points by zikduruqe 1705192250
15. Thinking in an array language - 238 points by tosh 1705164209
16. Looking back at Postgres (2019) - 14 points by infra_dev 1705038205
17. John Michell: Country Parson Described Black Holes in 1783 (2000) - 128 points by mikequinlan 1705189863
18. Nuclear battery produces power for 50 years without needing to charge - 26 points by prakhar897 1705233993
19. Free unexpected MIT courses to kick start the new year - 73 points by Anon84 1705225776
20. Stellarium: Software which renders realistic skies in real time - 660 points by tosh 1705162365
```

**Question 2**: Write a function that collects data on four Marvel movies from The Movie Database. You can use tmdbv3api (15pt)

```
pip install tmdbv3api
```

```
Collecting tmdbv3api
  Downloading tmdbv3api-1.9.0-py3-none-any.whl (25 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from tmdbv3api) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->tmdbv3api) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->tmdbv3api) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->tmdbv3api) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->tmdbv3api) (2023.11.17)
Installing collected packages: tmdbv3api
Successfully installed tmdbv3api-1.9.0
```

```
from tmdbv3api import TMDb, Movie

def collect_marvel_movie_data(api_key):
    tmdb = TMDb()
    tmdb.api_key = api_key

    movie = Movie()
    movie_id =[299536, 24428, 569094, 102382] # Example Marvel movie ID (Avengers: Endgame)
    for i in range(len(movie_id)):
      print(movie_id[i])
      m = movie.details(movie_id[i])

      print("\nMovie Details:")
      print(f"Title: {m.title}")
      print(f"Overview: {m.overview}")
      print(f"Popularity: {m.popularity}")
# Example usage
api_key = '06497a828f86e228abab0d482bf138d4'  # Replace with your actual API key
collect_marvel_movie_data(api_key)

```

```
299536
```

```
Movie Details:
Title: Avengers: Infinity War
Overview: As the Avengers and their allies have continued to protect the world from threats too large for any one hero to handle, a new danger has emerged from the cosmic shadows: Thanos. A desp
Popularity: 236.242
24428

Movie Details:
Title: The Avengers
Overview: When an unexpected enemy emerges and threatens global safety and security, Nick Fury, director of the international peacekeeping agency known as S.H.I.E.L.D., finds himself in need of
Popularity: 178.92
569094

Movie Details:
Title: Spider-Man: Across the Spider-Verse
Overview: After reuniting with Gwen Stacy, Brooklyn’s full-time, friendly neighborhood Spider-Man is catapulted across the Multiverse, where he encounters the Spider Society, a team of Spider-Pe
Popularity: 359.731
102382

Movie Details:
Title: The Amazing Spider-Man 2
Overview: For Peter Parker, life is busy. Between taking out the bad guys as Spider-Man and spending time with the person he loves, Gwen Stacy, high school graduation cannot come quickly enough.
Popularity: 100.049
```

## Kickstarter Projects Dataset

Using the Kickstarter Projects Dataset and Pandas, please answer one of following questions:

**Note:** Use ks-projects-201801.csv data

### Please answer only **one** of the following questions according to your (ID number + 1) (use the formula **mod 3 +1**)

```
# which question to answer - put your ID number and run the code
your_id  = "207376187"
q = (int(your_id) + 1) % 3 + 1
print("You need to answer questions %s and 4" % q)
```

```
You need to answer questions 1 and 4
```

**Question 1:** On average which project category received the lowest number of backers? (15 pt)

```
!pip install kaggle
```

```
import pandas as pd
from google.colab import files
```

```
# Upload the Kaggle API key
files.upload()

# Move the uploaded Kaggle API key to the required directory
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

# Set permissions for the Kaggle API key
!chmod 600 ~/.kaggle/kaggle.json
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving ks-projects-201801.csv to ks-projects-201801.csv
cp: cannot stat 'kaggle.json': No such file or directory
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
```

```
# Download the dataset from Kaggle
!kaggle datasets download -d kemical/kickstarter-projects

# Unzip the downloaded dataset
!unzip kickstarter-projects.zip
```

```
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 5, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.10/dist-packages/kaggle/__init__.py", line 23, in <module>
    api.authenticate()
  File "/usr/local/lib/python3.10/dist-packages/kaggle/api/kaggle_api_extended.py", line 403, in authenticate
    raise IOError('Could not find {}. Make sure it\'s located in'
OSError: Could not find kaggle.json. Make sure it's located in /root/.kaggle. Or use the environment method.
unzip:  cannot find or open kickstarter-projects.zip, kickstarter-projects.zip.zip or kickstarter-projects.zip.ZIP.
```

```
# Read the dataset into a Pandas DataFrame
df = pd.read_csv('ks-projects-201801.csv')

# Display the first few rows of the DataFrame
df.head()
```

| | ID | name | category | main_category | currency | deadline | goal | launched | pledged | state | backers | country | usd pledged | u |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000002330 | The Songs of Adelaide & Abullah | Poetry | Publishing | GBP | 2015-10-09 | 1000.0 | 2015-08-11 12:12:28 | 0.0 | failed | 0 | GB | 0.0 | |
| **1** | 1000003930 | Greeting From Earth: ZGAC Arts Capsule For ET | Narrative Film | Film & Video | USD | 2017-11-01 | 30000.0 | 2017-09-02 04:43:57 | 2421.0 | failed | 15 | US | 100.0 | |
| **2** | 1000004038 | Where is Hank? | Narrative Film | Film & Video | USD | 2013-02-26 | 45000.0 | 2013-01-12 00:20:50 | 220.0 | failed | 3 | US | 220.0 | |
| | | ToshiCapital | | | | | | | | | | | | |

```
# Group the DataFrame by the 'category' column and calculate the mean number of backers for each category
average_backers_by_category = df.groupby('category')['backers'].mean()

# Find the category with the lowest average number of backers
lowest_average_backers_category = average_backers_by_category.idxmin()

# Display the results
print(f"The category with the lowest average number of backers is: {lowest_average_backers_category}")
print(f"Average number of backers for this category: {average_backers_by_category[lowest_average_backers_category]}")
```

```
The category with the lowest average number of backers is: Crochet
Average number of backers for this category: 7.851851851851852
```

**Question 2:** On average which project category received the lowest pledged USD? (15 pt)

```

```

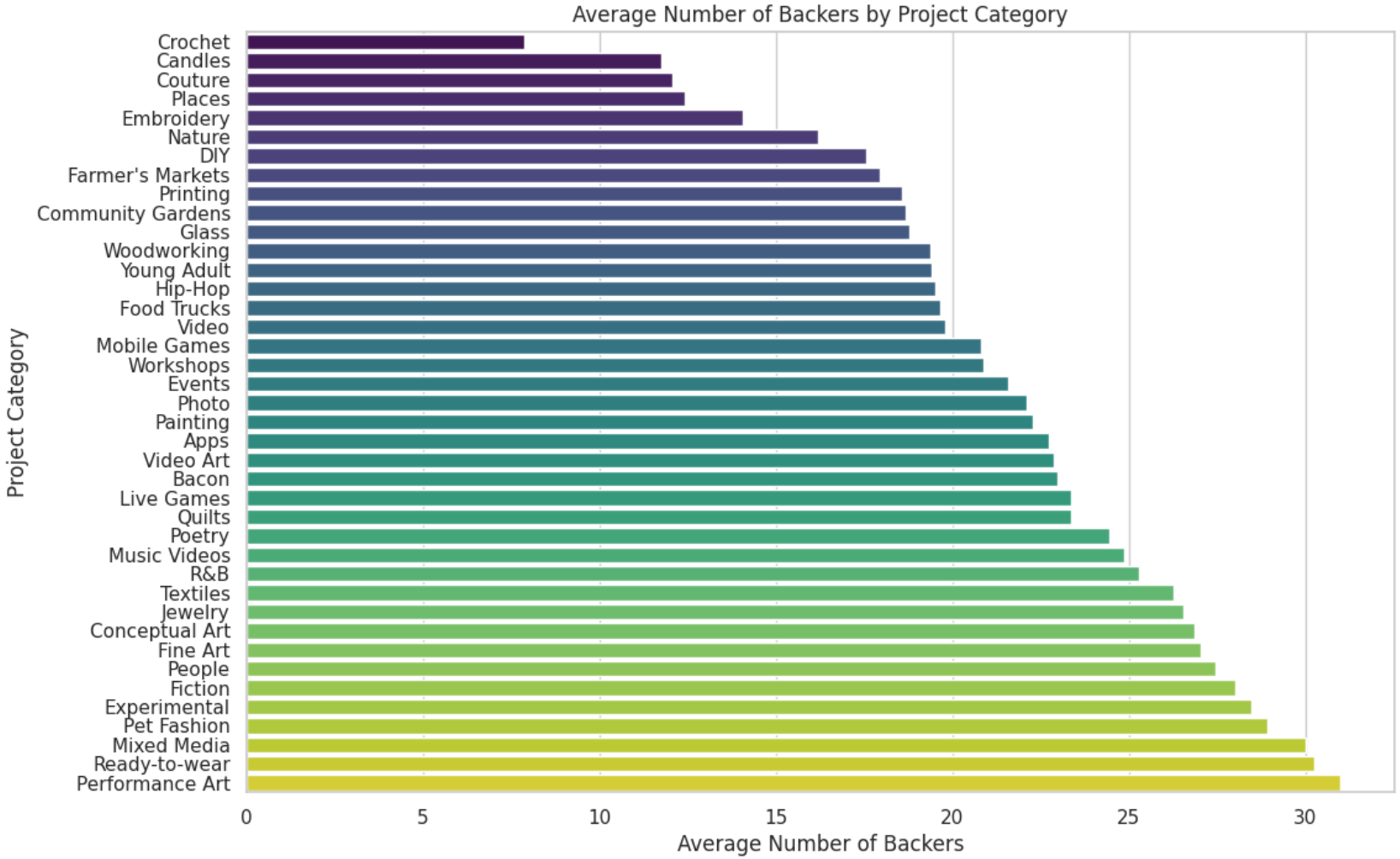
**Question 3:** In which month is there the lowest number of projects? (15 pt)

```

```

**Question 4 (for all):** Visualize your answer using matplotlib or seaborn (15pt)

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style for the plot
sns.set(style="whitegrid")

# Group the DataFrame by the 'category' column and calculate the mean number of backers for each category
average_backers_by_category = df.groupby('category')['backers'].mean().reset_index()

# Sort the DataFrame by the average number of backers in ascending order
average_backers_by_category = average_backers_by_category.sort_values(by='backers', ascending=True)

# Create a horizontal bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='backers', y='category', data=average_backers_by_category.head(40), palette='viridis')

# Set plot labels and title
plt.xlabel('Average Number of Backers')
plt.ylabel('Project Category')
plt.title('Average Number of Backers by Project Category')

# Show the plot
plt.show()
```

```
<ipython-input-12-78a13d0dae1e>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='backers', y='category', data=average_backers_by_category.head(40), palette='viridis')
```




## The Marvel Universe Social Network

Using the [The Marvel Universe Social Network] and **Pandas**, please answer the following questions:

**Question 1:** Write code which calculate the top-20 most friendly characters, i.e., characters with the highest number of friends. Please use *hero_network.csv* file (15pt).

**Note:** Not all the links in this dataset are symmetric.

```
import pandas as pd
from google.colab import files
# Upload the Kaggle API key
files.upload()

# Move the uploaded Kaggle API key to the required directory
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

# Set permissions for the Kaggle API key
!chmod 600 ~/.kaggle/kaggle.json
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving hero-network.csv to hero-network.csv
cp: cannot stat 'kaggle.json': No such file or directory
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
```

```
# Download the dataset from Kaggle
!kaggle datasets download -d csanhueza/the-marvel-universe-social-network
```

```
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 5, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.10/dist-packages/kaggle/__init__.py", line 23, in <module>
    api.authenticate()
  File "/usr/local/lib/python3.10/dist-packages/kaggle/api/kaggle_api_extended.py", line 403, in authenticate
    raise IOError('Could not find {}. Make sure it\'s located in'
OSError: Could not find kaggle.json. Make sure it's located in /root/.kaggle. Or use the environment method.
```

```
# Read the hero network dataset into a Pandas DataFrame
hero_network_df = pd.read_csv('hero-network.csv')

# Display the first few rows of the DataFrame
hero_network_df.head()
```

| | hero1 | hero2 |
|---|---|---|
| **0** | LITTLE, ABNER | PRINCESS ZANDA |
| **1** | LITTLE, ABNER | BLACK PANTHER/T'CHAL |
| **2** | BLACK PANTHER/T'CHAL | PRINCESS ZANDA |
| **3** | LITTLE, ABNER | PRINCESS ZANDA |
| **4** | LITTLE, ABNER | BLACK PANTHER/T'CHAL |

```
import pandas as pd

# Load the hero network dataset
hero_network_df = pd.read_csv('hero-network.csv')

# Drop duplicate connections
unique_connections_df = hero_network_df[['hero1', 'hero2']].drop_duplicates()

# Concatenate 'hero1' and 'hero2' columns
all_heroes = pd.concat([unique_connections_df['hero1'], unique_connections_df['hero2']])

# Count the unique occurrences of each hero
unique_connections_count = all_heroes.value_counts()

# Select the top 20 heroes with the highest number of unique connections
top_20_most_connected_characters = unique_connections_count.head(20)

# Display the result
print("Top 20 Most Connected Characters (Unique Friends):")
print(top_20_most_connected_characters)
```

```
Top 20 Most Connected Characters (Unique Friends):
CAPTAIN AMERICA         2854
SPIDER-MAN/PETER PAR    2563
IRON MAN/TONY STARK     2238
WOLVERINE/LOGAN         2036
THING/BENJAMIN J. GR    2023
MR. FANTASTIC/REED R    1958
HUMAN TORCH/JOHNNY S    1939
SCARLET WITCH/WANDA     1932
THOR/DR. DONALD BLAK    1913
BEAST/HENRY &HANK& P    1840
VISION                  1814
INVISIBLE WOMAN/SUE     1796
HAWK                    1709
CYCLOPS/SCOTT SUMMER    1616
STORM/ORORO MUNROE S    1581
HULK/DR. ROBERT BRUC    1571
ANGEL/WARREN KENNETH    1563
WASP/JANET VAN DYNE     1557
SHE-HULK/JENNIFER WA    1538
DR. STRANGE/STEPHEN     1530
dtype: int64
```

**Question 2:** Using Pandas and Networkx create a graph object of The Marvel Universe Social Network with the 150 most "friendly" characters (10pt). The vertices in that graph need to be relative to the size of each character's number of links (also referred to as the vertex degree) (10pt). Please color each node in the graph according to character type according to data in the *nodes.csv* file (5pt)

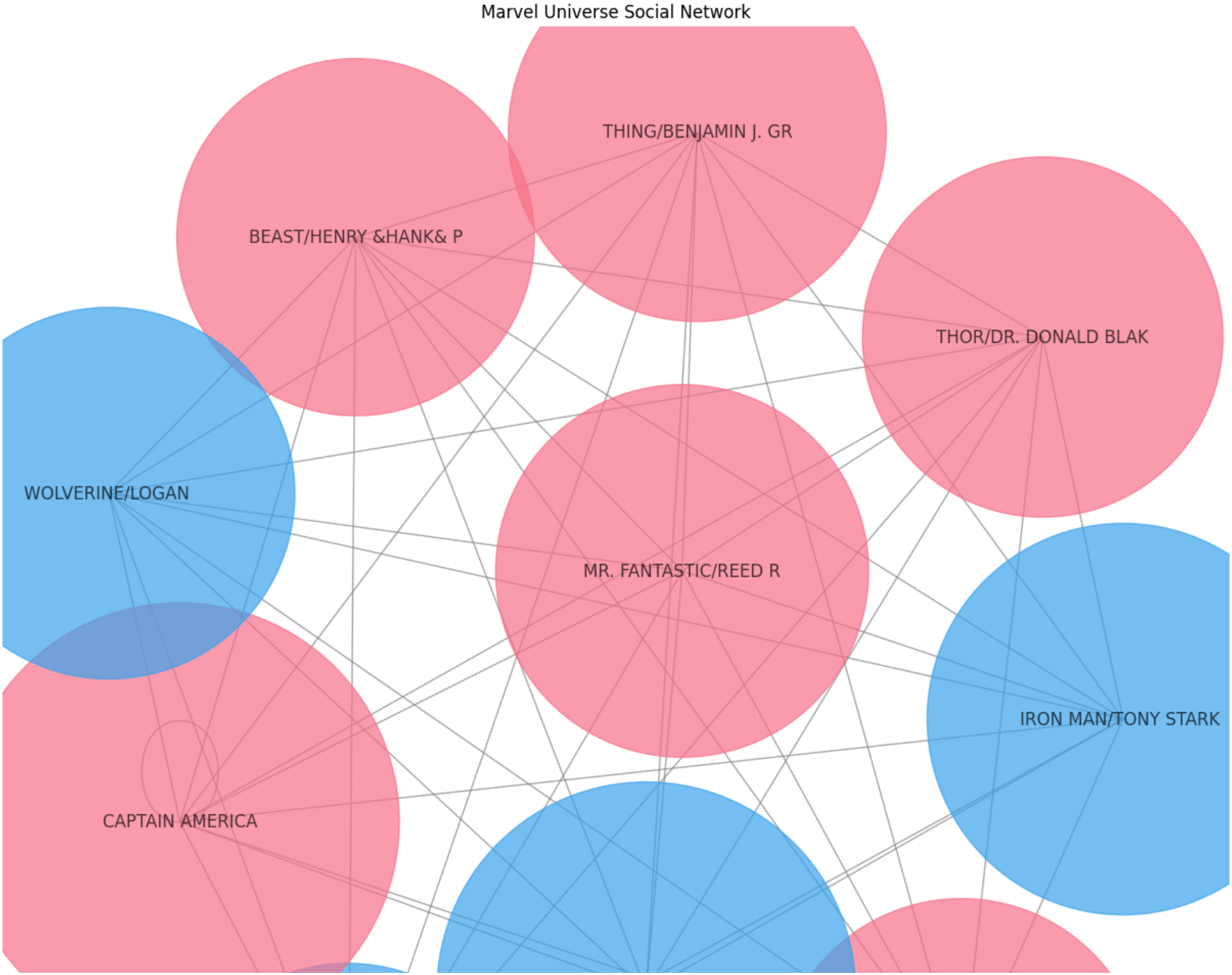
```
# Upload the Kaggle API key
files.upload()

# Move the uploaded Kaggle API key to the required directory
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

# Set permissions for the Kaggle API key
!chmod 600 ~/.kaggle/kaggle.json
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

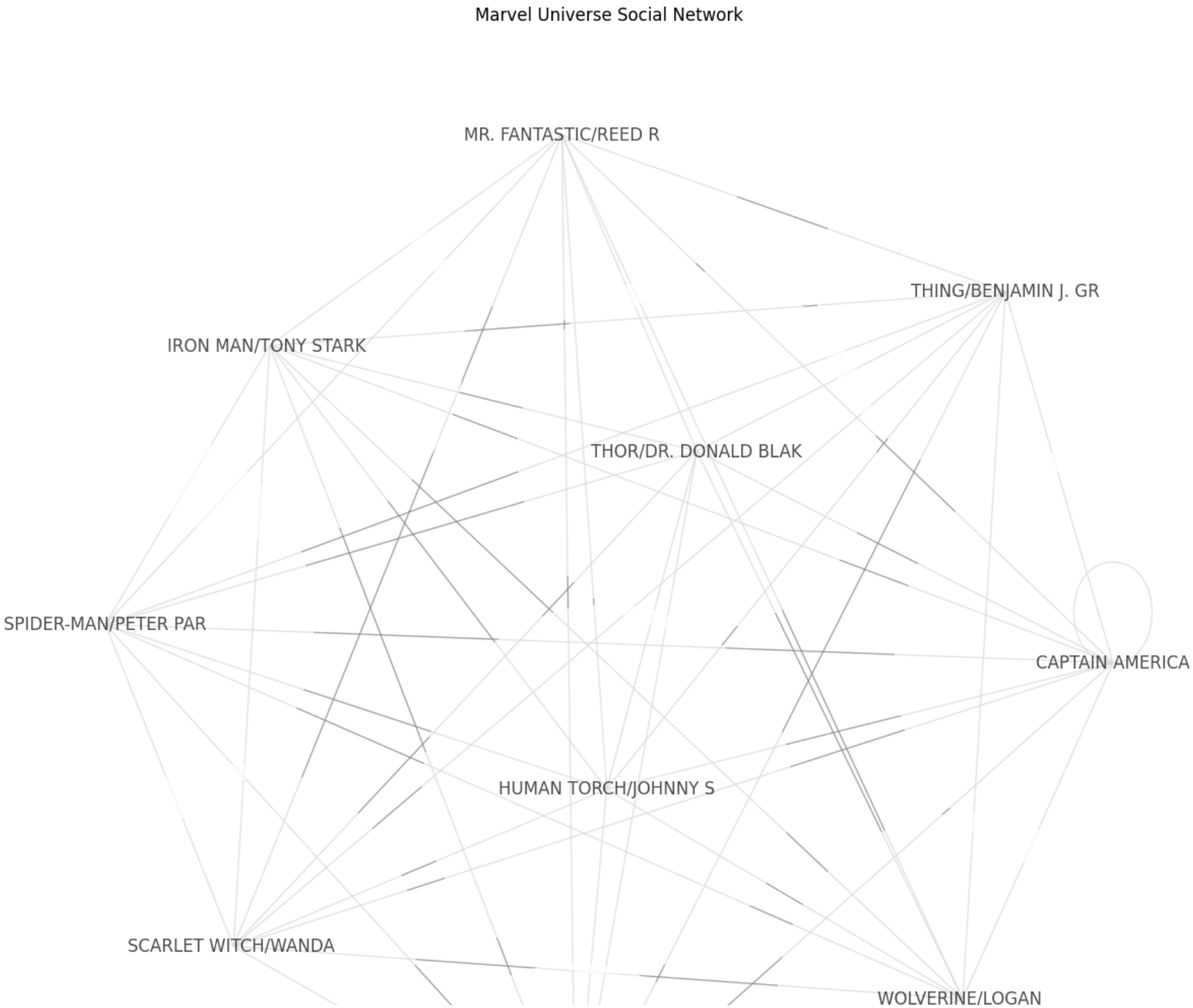
```
Saving nodes.csv to nodes.csv
cp: cannot stat 'kaggle.json': No such file or directory
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
```

```python
# Most 10 friendly colored. only for easy demonstraion.

import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns

# Load the hero network dataset
hero_network_df = pd.read_csv('hero-network.csv')

# Drop duplicate connections
unique_connections_df = hero_network_df[['hero1', 'hero2']].drop_duplicates()

# Create a graph object
G = nx.Graph()

# Add edges to the graph
G.add_edges_from(unique_connections_df.values)

# Calculate degree for each node
degrees = G.degree()

# Get the 150 most friendly characters based on degree
top_characters = sorted(degrees, key=lambda x: x[1], reverse=True)[:10]

# Create a subgraph with the top characters
subgraph = G.subgraph([character for character, degree in top_characters])

# Get node types from nodes.csv
nodes = pd.read_csv('nodes.csv')

# Assuming 'node' column contains identifiers and 'type' column contains node types
node_types = dict(zip(nodes['node'], nodes['type']))

# Get unique node types including 'Unknown'
unique_node_types = list(set(node_types.values())) + ['Unknown']

# Get colors based on node types
color_palette = sns.color_palette("husl", n_colors=len(unique_node_types))
colors = [color_palette[unique_node_types.index(node_types.get(node, 'Unknown'))] for node in subgraph.nodes()]

# Draw the graph with node sizes based on degree and a single color for all edges
plt.figure(figsize=(12, 12))
nx.draw(subgraph, with_labels=True, node_size=[degrees[node] * 50 for node in subgraph.nodes()], node_color=colors, cmap=plt.cm.rainbow, edge_color='grey', alpha=0.7)
plt.title("Marvel Universe Social Network")
plt.show()

```

```
/usr/local/lib/python3.10/dist-packages/networkx/drawing/nx_pylab.py:437: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  node_collection = ax.scatter(
```



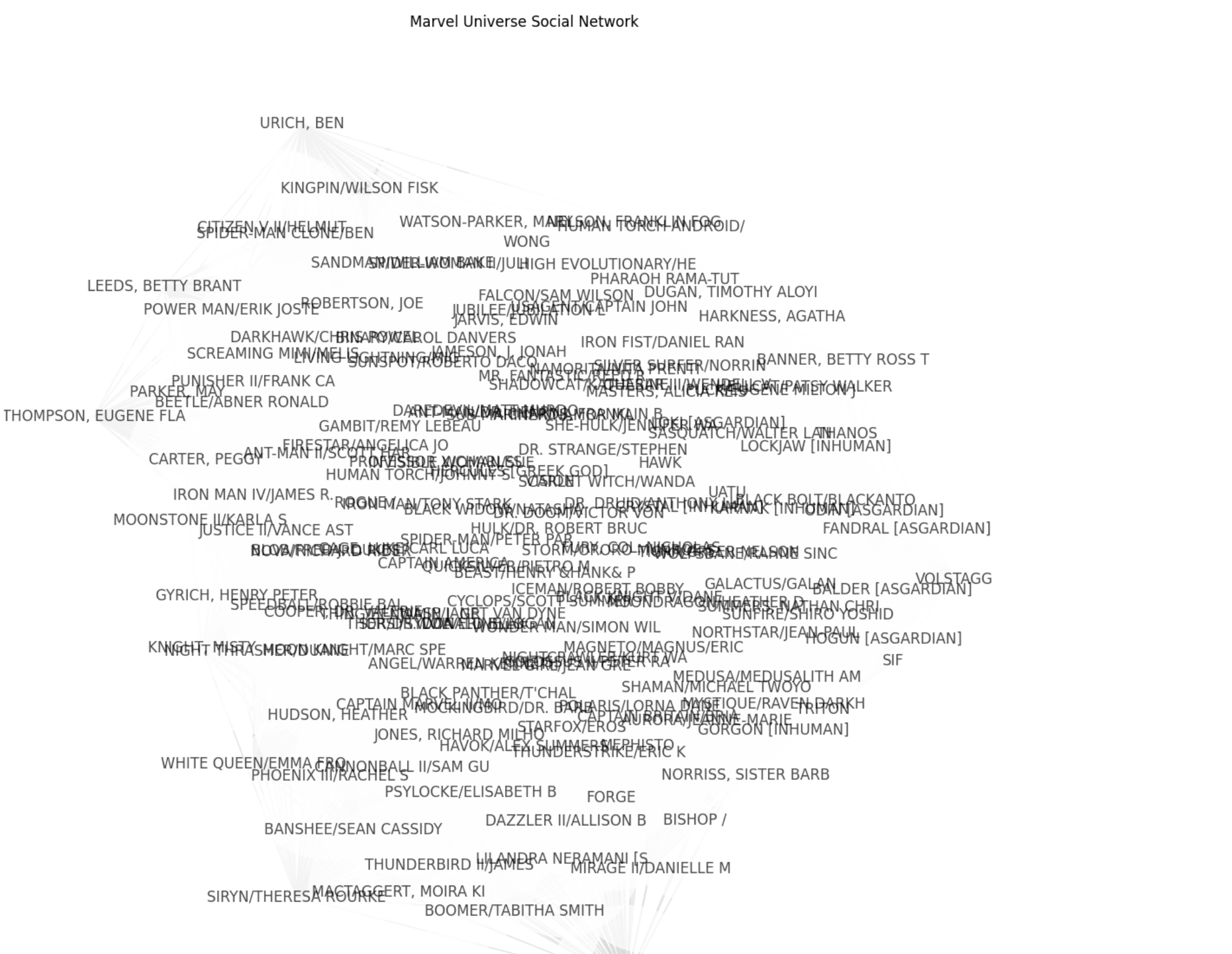
```python
# Most 10 friendly not colored. only for easy demonstraion.

import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Load the hero network dataset
hero_network_df = pd.read_csv('hero-network.csv')

# Drop duplicate connections
unique_connections_df = hero_network_df[['hero1', 'hero2']].drop_duplicates()

# Create a graph object
G = nx.Graph()

# Add edges to the graph
G.add_edges_from(unique_connections_df.values)

# Calculate degree for each node
degrees = G.degree()

# Get the 150 most friendly characters based on degree
top_characters = sorted(degrees, key=lambda x: x[1], reverse=True)[:10]

# Create a subgraph with the top characters
subgraph = G.subgraph([character for character, degree in top_characters])

# Draw the graph with node sizes based on degree without any colors
plt.figure(figsize=(12, 12))
nx.draw(subgraph, with_labels=True, node_size=[degrees[node] * 50 for node in subgraph.nodes()], node_color='white', edge_color='grey', alpha=0.7)
plt.title("Marvel Universe Social Network")
plt.show()
```


Marvel Universe Social Network
MR. FANTASTIC/REED R
THING/BENJAMIN J. GR
IRON MAN/TONY STARK
THOR/DR. DONALD BLAK
SPIDER-MAN/PETER PAR
CAPTAIN AMERICA
HUMAN TORCH/JOHNNY S
SCARLET WITCH/WANDA
WOLVERINE/LOGAN

```
# Most 150 friendly colored.

import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns

# Load the hero network dataset
hero_network_df = pd.read_csv('hero-network.csv')

# Drop duplicate connections
unique_connections_df = hero_network_df[['hero1', 'hero2']].drop_duplicates()

# Create a graph object
G = nx.Graph()

# Add edges to the graph
G.add_edges_from(unique_connections_df.values)

# Calculate degree for each node
degrees = G.degree()

# Get the 150 most friendly characters based on degree
top_characters = sorted(degrees, key=lambda x: x[1], reverse=True)[:150]

# Create a subgraph with the top characters
subgraph = G.subgraph([character for character, degree in top_characters])

# Get node types from nodes.csv
nodes = pd.read_csv('nodes.csv')

# Assuming 'node' column contains identifiers and 'type' column contains node types
node_types = dict(zip(nodes['node'], nodes['type']))

# Get unique node types including 'Unknown'
unique_node_types = list(set(node_types.values())) + ['Unknown']

# Get colors based on node types
color_palette = sns.color_palette("husl", n_colors=len(unique_node_types))
colors = [color_palette[unique_node_types.index(node_types.get(node, 'Unknown'))] for node in subgraph.nodes()]

# Draw the graph with node sizes based on degree and a single color for all edges
plt.figure(figsize=(12, 12))
nx.draw(subgraph, with_labels=True, node_size=[degrees[node] * 50 for node in subgraph.nodes()], node_color=colors, cmap=plt.cm.rainbow, edge_color='grey', alpha=0.7)
plt.title("Marvel Universe Social Network")
plt.show()

```


BEETLE/ABNER RONALD
HUMAN TORCH AND THE
SANDMAN/WILLIAM BAKE
ODIN [ASGARDIAN]
BANNER, BETTY ROSS T
URICH, BEN
SIF
GORGON [INHUMAN]
STARFOX/EROS
FANDRAL [ASGARDIAN]
TRITON
MOONSTONE II/KARLA S
SERSI/SYLVIA
FALCON/SAM WILSON
JARVIS, EDWIN
THANOS
BLACK BOLT/BLACKANTO
DR. DOOM/VICTOR VON
LOKI [ASGARDIAN]
CITIZEN V II/HELMUT
MEDUSA/MEDUSALITH AM
LIVING LIGHTNING/MIC
CAPTAIN MARVEL II/MO
HOGUN [ASGARDIAN]
KARNAK [INHUMAN]
DR. STRANGE/STEPHEN
JUBILEE II/VANCE AST
POWER MAN/ERIK JOSTE
HELLCAT/PATSY WALKER
CRYSTAL [INHUMAN]
TIGRA/GREER NELSON
BALDER [ASGARDIAN]
IRON FIST/DANIEL RAN
ROBERTSON, JOE
WONG
SILVER SURFER/NORRIN
BLACK WIDOW/NATASHA
LOCKJAW [INHUMAN]
JAMESON, J. JONAH
HUMAN TORCH/JOHNNY S
RICHARDS, FRANKLIN B
PUNISHER II/FRANK CA
DAREDEVIL/MATT MURDO
SHE-HULK/JENNIFER WA
QUASAR III/WENDELL V
WASP/JANET VAN DYNE
IRON MAN/TONY STARK
MEPHISTO
HULK/DR. ROBERT BRUC
THING/BENJAMIN J GRIM
CAGE, LUKE/CARL LUCA
SCREAMING MIMI/MELIS
PHARAOH RAMA TUT
DR. DRUID/ANTHONY D
DUGAN, TIMOTHY/SEAN
SPIDER-MAN/PETER PAR
SCARLET WITCH/WANDA
CAPTAIN AMERICA
CAPTAIN BRITAIN/BRIA
QUICKSILVER/PIETRO
SPIDER-WOMAN II/JULI
NOVA/RICHARD RIDER
BLOB/FRED J. DUKES
THUNDERSTRIKE/ERIC K
CYCLOPS/SCOTT SUMMER
BINARY/CAROL DANVERS
UATU
WONDER MAN/SIMON WIL
WOLVERINE/LOGAN
SASQUATCH/WALTER LAN
FIRESTAR/ANGELICA JO
PROFESSOR X/CHARLES
GYRICH, HENRY PETER
MOON KNIGHT/MARG SPE
NORRISS, SISTER BARB
BLACK KNIGHT V/DANE
NAMORITA/NITA PRENTI
SUNFIRE/SHIRO YOSHID
COLOSSUS II/PETER RA
NORTHSTAR/JEAN-PAUL
SHAMAN/MICHAEL TWOYO
COOPER, DR. VALERIE
BISHOP /
ROGUE /
HAVOK/ALEX SUMMERS
NIGHTCRAWLER/KURT WA
CANNONBALL II/SAM GU
HIGH EVOLUTIONARY/HE
AURORA/JEANNE-MARIE
USAGENT/CAPTAIN JOHN
DAZZLER II/ALLISON B
NATHAN CHRI
GAMBIT/REMY LEBEAU
SHADOWCAT/KATHERINE
MAGNETO/MAGNUS/ERIC
SUNSPOT/ROBERTO DACO
BOOMER/TABITHA SMITH
KNIGHT, MISTY
POLARIS/LORNA DANE
PSYLOCKE/ELISABETH B
PUCK/EUGENE MILTON J
FORGE
WHITE QUEEN/EMMA FRO
HUDSON, HEATHER
PHOENIX III/RACHEL S
MIRAGE II/DANIELLE M
BANSHEE/SEAN CASSIDY
MYSTIQUE/RAVEN DARKH
THUNDERBIRD II/JAMES
JUBILEE/JUBILATION L
MACTAGGERT, MOIRA KI
SIRYN/THERESA ROURKE
DOMINO III/BEATRICE/

```
# Most 150 friendly not colored.

import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Load the hero network dataset
hero_network_df = pd.read_csv('hero-network.csv')

# Drop duplicate connections
unique_connections_df = hero_network_df[['hero1', 'hero2']].drop_duplicates()

# Create a graph object
G = nx.Graph()

# Add edges to the graph
G.add_edges_from(unique_connections_df.values)

# Calculate degree for each node
degrees = G.degree()

# Get the 150 most friendly characters based on degree
top_characters = sorted(degrees, key=lambda x: x[1], reverse=True)[:150]

# Create a subgraph with the top characters
subgraph = G.subgraph([character for character, degree in top_characters])

# Draw the graph with node sizes based on degree without any colors
plt.figure(figsize=(12, 12))
nx.draw(subgraph, with_labels=True, node_size=[degrees[node] * 50 for node in subgraph.nodes()], node_color='white', edge_color='grey', alpha=0.7)
plt.title("Marvel Universe Social Network")
plt.show()
```


Marvel Universe Social Network
URICH, BEN
KINGPIN/WILSON FISK
CITIZEN V II/HELMUT
SPIDER-MAN CLONE/BEN
WATSON-PARKER, MAY
WONG
HUMAN TORCH ANDROID/
SANDMAN/WILLIAM BAKE
HIGH EVOLUTIONARY/HE
PHARAOH RAMA-TUT
LEEDS, BETTY BRANT
ROBERTSON, JOE
FALCON/SAM WILSON
DUGAN, TIMOTHY ALOYI
POWER MAN/ERIK JOSTE
JUBILEE/JUBILATION L
CAPTAIN JOHN
HARKNESS, AGATHA
JARVIS, EDWIN
DARKHAWK/CHRIS POWER
CAROL DANVERS
IRON FIST/DANIEL RAN
JAMESON, J. JONAH
SCREAMING MIMI/MELIS
SUNSPOT/ROBERTO DACO
SILVER SURFER/NORRIN
BANNER, BETTY ROSS T
PUNISHER II/FRANK CA
MR. FANTASTIC/REED R
MASTERS, ALICIA REIS
PARKER, MAY
BEETLE/ABNER RONALD
THOMPSON, EUGENE FLA
GAMBIT/REMY LEBEAU
SASQUATCH/WALTER LA
THANOS
FIRESTAR/ANGELICA JO
DR. STRANGE/STEPHEN
LOCKJAW [INHUMAN]
ANT-MAN II/SCOTT LAR
CARTER, PEGGY
HAWK
HUMAN TORCH/JOHNNY S
IRON MAN IV/JAMES R.
BLACK BOLT/BLACKANTO
MOONSTONE II/KARLA S
FANDRAL [ASGARDIAN]
JUSTICE II/VANCE AST
HULK/DR. ROBERT BRUC
SPIDER-MAN/PETER PAR
CAPTAIN AMERICA
BEAST/HENRY &HANK& P
GALACTUS/GALAN
VOLSTAGG
BALDER [ASGARDIAN]
GYRICH, HENRY PETER
ICEMAN/ROBERT BOBBY
CYCLOPS/SCOTT
SUNFIRE/SHIRO YOSHID
WONDER MAN/SIMON WIL
NORTHSTAR/JEAN PAUL
HOGUN [ASGARDIAN]
KNIGHT, MISTY
MAGNETO/MAGNUS/ERIC
SIF
ANGEL/WARREN
MEDUSA/MEDUSALITH AM
SHAMAN/MICHAEL TWOYO
BLACK PANTHER/T'CHAL
CAPTAIN MARVEL
MYSTIQUE/RAVEN DARKH
HUDSON, HEATHER
TRITON
STARFOX/EROS
GORGON [INHUMAN]
JONES, RICHARD MILHO
HAVOK/ALEX SUMMERS
MEPHISTO
THUNDERSTRIKE/ERIC K
WHITE QUEEN/EMMA FRO
CANNONBALL II/SAM GU
NORRISS, SISTER BARB
PHOENIX III/RACHEL S
PSYLOCKE/ELISABETH B
FORGE
DAZZLER II/ALLISON B
BISHOP /
BANSHEE/SEAN CASSIDY
LILANDRA NERAMANI [S
THUNDERBIRD II/JAMES
MIRAGE II/DANIELLE M
MACTAGGERT, MOIRA KI
SIRYN/THERESA ROURKE
BOOMER/TABITHA SMITH


**Bonus:** Visualize the above network using Cytoscape or Gephi or D3 (10pt)

```
!pip install py4cytoscape
```

```
# I succeeded only when i downloaded Cytoscape manualy on my pc.
```

```
_PY4CYTOSCAPE = 'git+https://github.com/cytoscape/py4cytoscape@0.0.11'
import requests
exec(requests.get("https://raw.githubusercontent.com/cytoscape/jupyter-bridge/master/client/p4c_init.py").text)
IPython.display.Javascript(_PY4CYTOSCAPE_BROWSER_CLIENT_JS) # Start browser client
```

```
import py4cytoscape as p4c
p4c.cytoscape_ping()
p4c.cytoscape_version_info()
```