In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()
```

In [2]:
```
from bs4 import BeautifulSoup
```

In [3]:
```
import requests
```

In [4]:
```
url = 'https://www.goodreads.com/list/show/153860.Goodreads_Top_100_Highest_Rated_Books_on_Goodreads_with_at_le
```

In [5]:
```
headers = {"User-Agent": "Mozilla/5.0"}
```

In [6]:
```
response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.text, "html.parser")
```

In [7]:
```
client = requests.get(url)
client
```

Out[7]:
```
<Response [403]>
```

In [8]:
```
client.text
```

Out[8]:
```
'\n<!-- This is a random-length HTML comment: wyjtlfaxyduxsxgbgtsxsieggututbilvewhawvtfsnfdr -->'
```

In [9]:
```
books = soup.find_all("a", class_="bookTitle")
```

In [10]:
```
html_code = response.text
soup = BeautifulSoup(html_code, "html.parser")
```

In [11]:
```
soup = BeautifulSoup(html_code, "html.parser")
soup
```

Out[11]:
```
<!DOCTYPE html>

<html class="desktop withSiteHeaderTopFullImage">
<head>
<title>Goodreads Top 100 - Highest Rated Books on Goodreads with at least 10,000 Ratings (106 books)</title>
<meta content="106 books based on 1085 votes: Words of Radiance by Brandon Sanderson, Harry Potter and the Deat
hly Hallows by J.K. Rowling, Fourth Wing by Rebecca Yarro..." name="description"/>
<meta content="telephone=no" name="format-detection"/>
<link href="https://www.goodreads.com/list/show/153860.Goodreads_Top_100_Highest_Rated_Books_on_Goodreads_with_
at_least_10_000_Ratings" rel="canonical"/>
<script type="text/javascript"> var ue_t0=window.ue_t0||+new Date();
 </script>
<script type="text/javascript">
    var ue_mid = "A1PQBFHBHS6YH1";
    var ue_sn = "www.goodreads.com";
    var ue_furl = "fls-na.amazon.com";
    var ue_sid = "345-8230264-3058968";
    var ue_id = "0AX08R0MGKTZB9M7BDQ3";

    (function(e){var c=e;var a=c.ue||{};a.main_scope="mainscopecsm";a.q=[];a.t0=c.ue_t0||+new Date();a.d=g;func
tion g(h){return +new Date()-(h?0:a.t0)}function d(h){return function(){a.q.push({n:h,a:arguments,t:a.d()})}}fu
nction b(m,l,h,j,i){var k={m:m,f:l,l:h,c:""+j,err:i,fromOnError:1,args:arguments};c.ueLogError(k);return false}
b.skipTrace=1;e.onerror=b;function f(){c.uex("ld")}if(e.addEventListener){e.addEventListener("load",f,false)}el
se{if(e.attachEvent){e.attachEvent("onload",f)}}a.tag=d("tag");a.log=d("log");a.reset=d("rst");c.ue_csm=c;c.ue=
a;c.ueLogError=d("err");c.ues=d("ues");c.uet=d("uet");c.uex=d("uex");c.uet("ue")})(window);(function(e,d){var a
=e.ue||{};function c(g){if(!g){return}var f=d.head||d.getElementsByTagName("head")[0]||d.documentElement,h=d.cr
eateElement("script");h.async="async";h.src=g;f.insertBefore(h,f.firstChild)}function b(){var k=e.ue_cdn||"m.me
dia-amazon.com",g=e.ue_cdns||"m.media-amazon.com",j="/images/G/01/csminstrumentation/",h=e.ue_file||"ue-full-11
e51f253e8ad9d145f4ed644b40f692._V1_.js",f,i;if(h.indexOf("NSTRUMENTATION_FIL")>=0){return}if("ue_https" in e){f
=e.ue_https}else{f=e.location&&e.location.protocol=="https:"?1:0}i=f?"https://":"http://";i+=f?g:k;i+=j;i+=h;c(
i)}if(!e.ue_inline){if(a.loadUEFull){a.loadUEFull()}else{b()}}a.uels=c;e.ue=a})(window,document);

    if (window.ue && window.ue.tag) { window.ue.tag('list:show:signed_out', ue.main_scope);window.ue.tag('list:
show:signed_out:desktop', ue.main_scope); }
  </script>
<!-- * Copied from https://info.analytics.a2z.com/#/docs/data_collection/csa/onboard */ -->
<script>
  //<![CDATA[
    !function(){function n(n,t){var r=i(n);return t&&(r=r("instance",t)),r}var r=[],c=0,i=function(t){return fu
nction(){var n=c++;return r.push([t,[].slice.call(arguments,0),n,{time:Date.now()}]),i(n)}};n._s=r,this.csa=n}(
);

    if (window.csa) {
      window.csa("Config", {
```

```
  //]]>
</script>
</body>
</html>
<!-- This is a random-length HTML comment: sfozbccoujgsizpejezdxfeargblngvafwtvmlhbfzzhcinrafvkxmckrthruexlyjvz
tshujzyybiahcmguuhhryvwillfdjmkdiwmnejpjjsftjyltcdkzuqnmhphcknsvixrepslcvbjhexrzeohhxbpiovietpsdtkiqezihjqvbnfr
rsqcoharbdhuhhpufbjrsvkybltwpvpxakfwzfedmlqgxtdekvkncovbdrzmmzxyvaiosvxvhpvdjafzhmscdtaqyqssrauhkkobxitlqekmgrw
rmvmpvtxnzlzhxxsrnptjfuwoivfugbdemsbeezzdrrrlszdvtxthlpyqqapmudiwfqolhbqjoirmvjqglraeslfwaryhqsjndtzdopgnlxvizb
quvrdfbkplxrlhvaejkpwazcfybwqywyidrefmjmbmjodagwdazcvzvajkgwgmgaavhuxynqxbffbzjmyuffbacssavpufwlxsmbpomtrhptlyi
wwjjehrmfepdlzobqjopjvyddmornytpbnbdfsqfwuqcjmrhdcyocmclywesrlazedxfbynwpmijzjntkjuclejtzjhqcggvmnjvyfjawfeighl
fpghfbprvqrpwnztvyiwmecqhtsqwtgcgxkujtxdmgrqycensiajbxhrctbjcslleuwlxqjizodbznvairdaaaedajbqnetgsasnzflvftvgdpf
nmqqscaturiojtehwakkqvkjwywchaculnplbdizgdgwuizivlqkptioudluqxuzkmvpyrrapygjzidnkqolruarzhyylvpshaexoxbeugdluwe
hrdtyjbjorugwmciojgbgwekzcnjcmzzznafdizevtecgovuoskuhclhgytnnxlwslbanfljaboewobezapguxcfbpuqcuqfutizenzbocnnsdl
draiqtywmruetedjyacxqidwjkynrgefetpgivxvcwuwwmsorxfygpcdkztrtkseqgklkzstfeltejcmgwpsiwzwbbbyicjnpxaehefkmfddieu
suluhxikrooatikwdeayybqvzvtkjlwdwzqfpfqeahlraurxfnxybzgskmglhrxwqvnwflbcifyyedhyztofzrdglahjrqqpdnphraequbgxzjt
pdtgdkrwtvovoxfvbmytpkjdqljxofmnjkwylwumclmefpxefymjdqmecxytcipfxshadjnkpujvhvsrjsygbbcvaubkfxysufwvmiozwgnfpcb
kbtoaouqjeicehkmmestomiwizytbknerzflzuwburbmlybjiovffvkrjjhakspklmivvvkwusqkieleruixgrse -->
```

In [12]:
```
rows = soup.find_all("tr", itemtype="http://schema.org/Book")

for row in rows[:10]:
    title = row.find("a", class_="bookTitle").get_text(strip=True)
    author = row.find("a", class_="authorName").get_text(strip=True)
    rating = row.find("span", class_="minirating").get_text(strip=True)
    score_tag = row.find("a", string=lambda text: text and "score:" in text)
    score = score_tag.get_text(strip=True).replace("score: ", "").replace(",", "") if score_tag else "0"
```

In [13]:
```
    print(title)
    print(author)
    print(rating)
    print(score_tag)
    print(score)
    print("-" * 40)
```

```
Know My Name
Chanel Miller
4.68 avg rating — 253,072 ratings
<a href="#" onclick="Lightbox.showBoxByID('score_explanation', 300); return false;">score: 4,845</a>
4845
----------------------------------------
```

In [14]:
```
titles = []
authors = []
ratings = []
scores = []
votes = []

for row in rows:
    title = row.find("a", class_="bookTitle").get_text(strip=True)
    author = row.find("a", class_="authorName").get_text(strip=True)
    rating = row.find("span", class_="minirating").get_text(strip=True)
    score_tag = row.find("a", string=lambda text: text and "score:" in text)
    if score_tag:
        score_val = score_tag.get_text(strip=True).replace("score: ", "").replace(",", "")
    else:
        score_val = "0"
    vote_tag = row.find("a", id=lambda x: x and x.startswith("loading_link_"))
    if vote_tag:

        vote_val = vote_tag.get_text(strip=True).split()[0].replace(",", "")
    else:
        vote_val = "0"

    titles.append(title)
    authors.append(author)
    ratings.append(rating)
    scores.append(score_val)
    votes.append(vote_val)

df = pd.DataFrame({
    "Title": titles,
    "Author": authors,
    "Rating": ratings,
    "Score": scores,
    "Votes": votes
})

print(df.head())
```

```
                                                Title             Author  \
0     Words of Radiance (The Stormlight Archive, #2)  Brandon Sanderson
1  Harry Potter and the Deathly Hallows (Harry Po...       J.K. Rowling
2                    Fourth Wing (The Empyrean, #1)     Rebecca Yarros
3                Crooked Kingdom (Six of Crows, #2)      Leigh Bardugo
4  A Court of Mist and Fury (A Court of Thorns an...      Sarah J. Maas

                              Rating  Score Votes
0   4.76 avg rating — 475,669 ratings  31537   316
1  4.62 avg rating — 4,071,971 ratings  21791   221
2  4.57 avg rating — 3,379,995 ratings  16120   163
3    4.57 avg rating — 758,912 ratings  15659   160
4  4.64 avg rating — 3,105,813 ratings  14114   144
```

In [15]:
```python
df = pd.DataFrame({
    "Title": titles,
    "Author": authors,
    "Rating": ratings,
    "Scores": scores,
    "Votes": votes
})

df
```

Out[15]:

	Title	Author	Rating	Scores	Votes
0	Words of Radiance (The Stormlight Archive, #2)	Brandon Sanderson	4.76 avg rating — 475,669 ratings	31537	316
1	Harry Potter and the Deathly Hallows (Harry Po...	J.K. Rowling	4.62 avg rating — 4,071,971 ratings	21791	221
2	Fourth Wing (The Empyrean, #1)	Rebecca Yarros	4.57 avg rating — 3,379,995 ratings	16120	163
3	Crooked Kingdom (Six of Crows, #2)	Leigh Bardugo	4.57 avg rating — 758,912 ratings	15659	160
4	A Court of Mist and Fury (A Court of Thorns an...	Sarah J. Maas	4.64 avg rating — 3,105,813 ratings	14114	144
...	...	...	...	...	...
95	Midnight Black (Gray Man, #14)	Mark Greaney	4.63 avg rating — 11,292 ratings	67	2
96	Demon in White (The Sun Eater, #3)	Christopher Ruocchio	4.63 avg rating — 17,582 ratings	64	2
97	Don Caselli (Caselli Family, #2)	Jahquel J.	4.76 avg rating — 10,060 ratings	64	1
98	Theo of Golden: A Novel	Allen Levi	4.66 avg rating — 33,551 ratings	61	1
99	The Proving Ground (The Lincoln Lawyer, #8)	Michael Connelly	4.60 avg rating — 14,267 ratings	60	1

100 rows × 5 columns

In [16]:
```python
# □□□□□ □□□□□ □□□□□□□□ □□□□□□□ □□□
# □□□□ avg rating
df['avg_rating'] = df['Rating'].str.extract(r'([0-9]+\.[0-9]+)')

# □□□□□□□ □□ □□□□□□ □□□□□□□□□□ □□□ □□□□
df['ratings'] = df['Rating'].str.extract(r'— ([0-9,]+)')[0].str.replace(',', '').astype(int)

# □□□□□□ □□□□□□ □□□
df = df.drop(columns=['Rating'])

print(df)
```

```
                                                Title               Author  \
0      Words of Radiance (The Stormlight Archive, #2)     Brandon Sanderson
1   Harry Potter and the Deathly Hallows (Harry Po...          J.K. Rowling
2                       Fourth Wing (The Empyrean, #1)        Rebecca Yarros
3                  Crooked Kingdom (Six of Crows, #2)         Leigh Bardugo
4   A Court of Mist and Fury (A Court of Thorns an...         Sarah J. Maas
..                                                ...                   ...
95                    Midnight Black (Gray Man, #14)          Mark Greaney
96                 Demon in White (The Sun Eater, #3)  Christopher Ruocchio
97                   Don Caselli (Caselli Family, #2)           Jahquel J.
98                            Theo of Golden: A Novel           Allen  Levi
99        The Proving Ground (The Lincoln Lawyer, #8)   Michael    Connelly

   Scores Votes avg_rating  ratings
0   31537   316       4.76   475669
1   21791   221       4.62  4071971
2   16120   163       4.57  3379995
3   15659   160       4.57   758912
4   14114   144       4.64  3105813
..    ...   ...        ...      ...
95     67     2       4.63    11292
96     64     2       4.63    17582
97     64     1       4.76    10060
98     61     1       4.66    33551
99     60     1       4.60    14267

[100 rows x 6 columns]
```

In [17]:
```
# □□□□□□□□□ □□□□□□ □□□□□□
df['avg_rating'] = pd.to_numeric(df['avg_rating'], errors='coerce')
df['Scores'] = pd.to_numeric(df['Scores'], errors='coerce')
df['Votes'] = pd.to_numeric(df['Votes'], errors='coerce')
df['ratings'] = pd.to_numeric(df['ratings'], errors='coerce')

print(df.dtypes)
```

```
Title          object
Author         object
Scores          int64
Votes           int64
avg_rating    float64
ratings         int64
dtype: object
```

In [18]:
```
df
```

Out[18]:

	Title	Author	Scores	Votes	avg_rating	ratings
0	Words of Radiance (The Stormlight Archive, #2)	Brandon Sanderson	31537	316	4.76	475669
1	Harry Potter and the Deathly Hallows (Harry Po...	J.K. Rowling	21791	221	4.62	4071971
2	Fourth Wing (The Empyrean, #1)	Rebecca Yarros	16120	163	4.57	3379995
3	Crooked Kingdom (Six of Crows, #2)	Leigh Bardugo	15659	160	4.57	758912
4	A Court of Mist and Fury (A Court of Thorns an...	Sarah J. Maas	14114	144	4.64	3105813
...	...	...	...	...	...	...
95	Midnight Black (Gray Man, #14)	Mark Greaney	67	2	4.63	11292
96	Demon in White (The Sun Eater, #3)	Christopher Ruocchio	64	2	4.63	17582
97	Don Caselli (Caselli Family, #2)	Jahquel J.	64	1	4.76	10060
98	Theo of Golden: A Novel	Allen Levi	61	1	4.66	33551
99	The Proving Ground (The Lincoln Lawyer, #8)	Michael Connelly	60	1	4.60	14267

100 rows × 6 columns

In [19]:
```
df.to_csv("goodreads_books.csv", index=False)
```

In [20]:
```
# 1. (□□□□ □□□□ □□□□□ □□□ □□□□□□□□)
print("Basic Statistics:")
print(df.describe())

print("-" * 30)

# 2. □□□□□□ □□□ □□□□ □□ □□□□□□ (Nulls)
print("\nMissing Values:")
print(df.isnull().sum())

print("-" * 30)
```

```
# 3. □□□□□□ □□ □□□□□□ □□□□□ □□ □□□□□□□
print("\nDuplicate Rows:")
print(df.duplicated().sum())
```

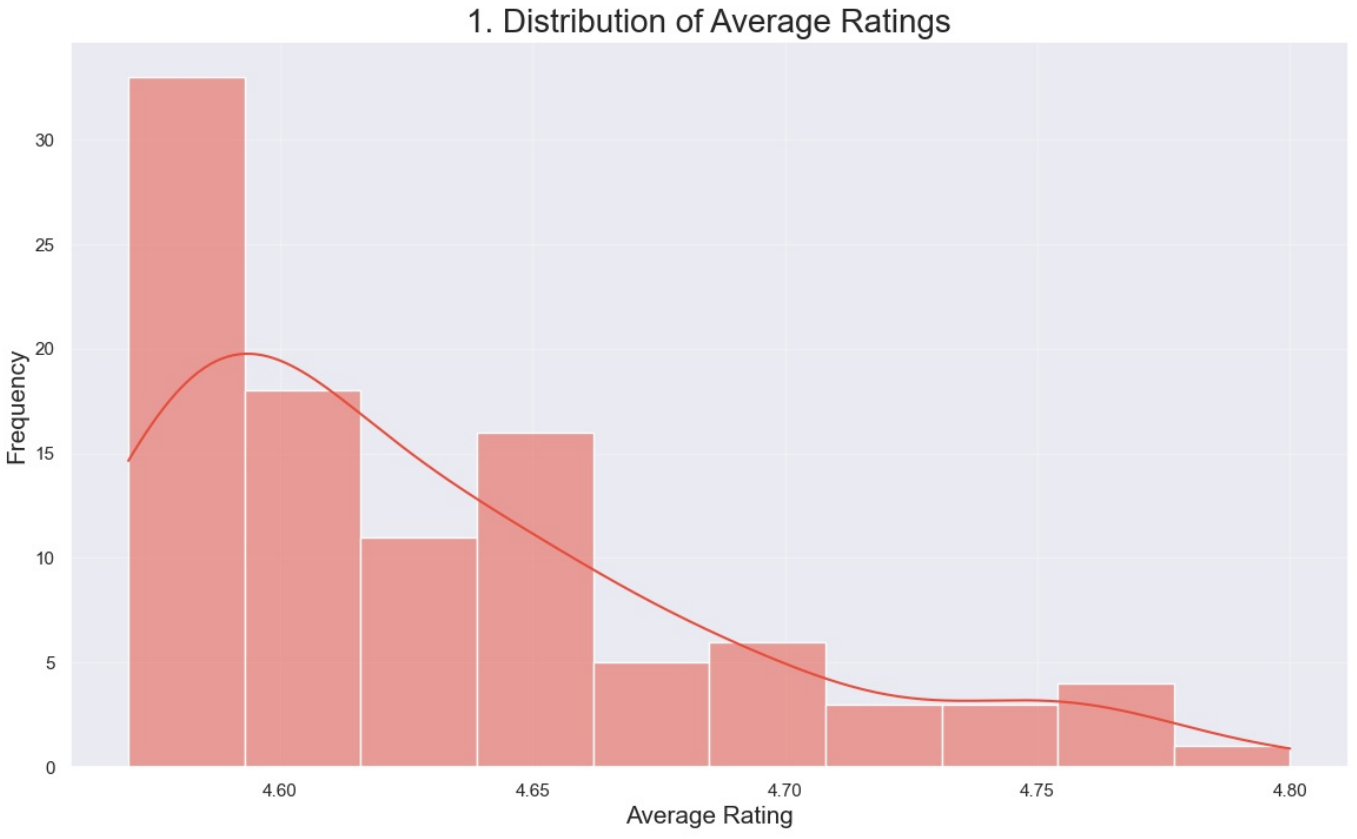
```
Basic Statistics:
             Scores      Votes  avg_rating       ratings
count    100.000000  100.00000  100.000000  1.000000e+02
mean    2055.660000   21.89000    4.629900  2.208662e+05
std     4780.141446   48.11432    0.055204  6.581560e+05
min       60.000000    1.00000    4.570000  1.006000e+04
25%      156.250000    3.00000    4.590000  1.369100e+04
50%      403.000000    5.00000    4.610000  2.144250e+04
75%     1153.000000   12.75000    4.660000  6.547950e+04
max    31537.000000  316.00000    4.800000  4.071971e+06
------------------------------

Missing Values:
Title         0
Author        0
Scores        0
Votes         0
avg_rating    0
ratings       0
dtype: int64
------------------------------

Duplicate Rows:
0
```

In [21]:

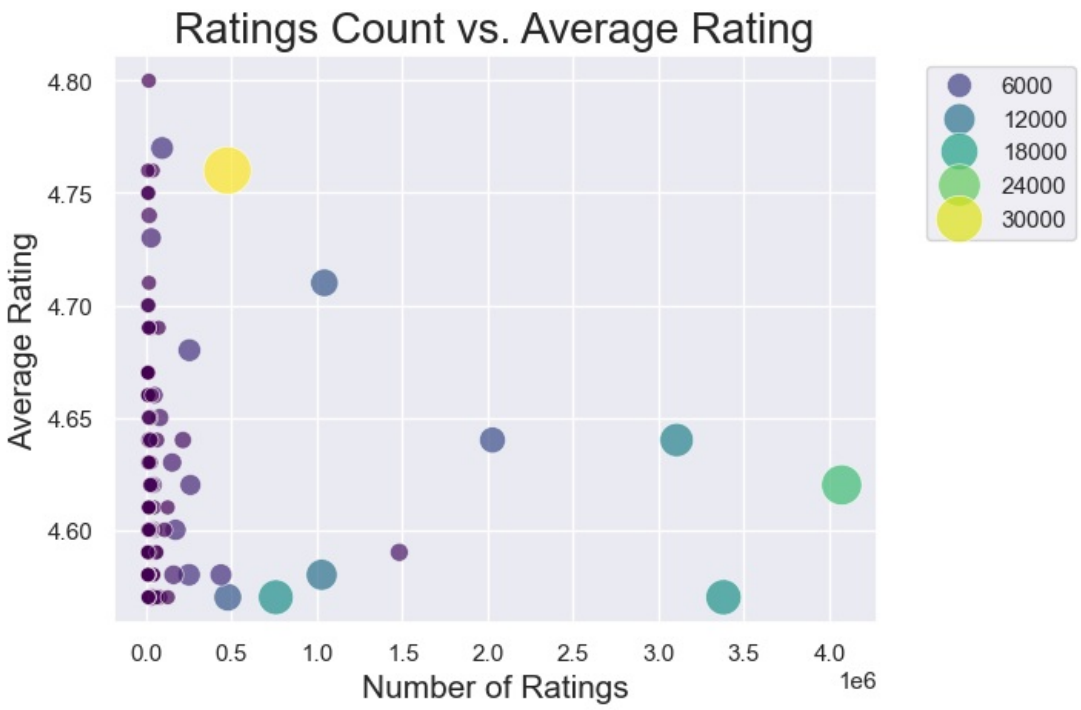
```
# 1. □□□□□□ □□□□□ □□□□ □□□□□□□ □□□□□□□
highest_rated = df.nlargest(1, 'avg_rating')
print("1. Highest Rated Book:\n", highest_rated[['Title', 'avg_rating']])

# 2. □□□□□□□□ □□□□ □□□□□) □□□□□□ □□□□□□□ □□□□□□□)
most_popular = df.nlargest(1, 'ratings')
print("\n2. Most Popular Book (Most Ratings):\n", most_popular[['Title', 'ratings']])

# 3. □□□□□□□□ □□ □□□□□□□□ □□□□□□□ □□□□□□□
top_author = df['Author'].value_counts().idxmax()
author_count = df['Author'].value_counts().max()
print(f"\n3. Top Author: {top_author} with {author_count} books.")

# 4. □□□□ □□□□ □□□□□□ □□□□□□□ □□□□□□□ □□□□□□ Score
top_5_score = df.nlargest(5, 'Scores')
print("\n4. Top 5 Books by Score:\n", top_5_score[['Title', 'Scores']])

# 5. □□□□ □□□□□□□□ Votes □ Scores
correlation = df['Votes'].corr(df['Scores'])
print(f"\n5. Correlation between Votes and Scores: {correlation:.2f}")

# 6. □□□□□□ □□□□ □□□□□□□□ □□□□□□
mean_rating = df['avg_rating'].mean()
print(f"\n6. Average Rating for all books: {mean_rating:.2f}")

# 7. □□□□□□□□□□ □□□□ □□ □□□□□ □□□□
least_rated = df.nsmallest(1, 'ratings')
print("\n7. Least Rated Book:\n", least_rated[['Title', 'ratings']])

# 8. 4.6 □□ □□□□□ □□□□□□□ □□□□□□ □□□□
high_rated_count = df[df['avg_rating'] > 4.6].shape[0]
print(f"\n8. Number of books with rating > 4.6: {high_rated_count}")
```

```
1. Highest Rated Book:
                                                 Title  avg_rating
70  Heaven Official's Blessing: Tian Guan Ci Fu (N...         4.8

2. Most Popular Book (Most Ratings):
                                                 Title  ratings
1  Harry Potter and the Deathly Hallows (Harry Po...  4071971

3. Top Author: Sarah J. Maas with 2 books.

4. Top 5 Books by Score:
                                                 Title  Scores
0     Words of Radiance (The Stormlight Archive, #2)   31537
1  Harry Potter and the Deathly Hallows (Harry Po...   21791
2                    Fourth Wing (The Empyrean, #1)   16120
3                Crooked Kingdom (Six of Crows, #2)   15659
4  A Court of Mist and Fury (A Court of Thorns an...   14114

5. Correlation between Votes and Scores: 1.00

6. Average Rating for all books: 4.63

7. Least Rated Book:
                               Title  ratings
97  Don Caselli (Caselli Family, #2)    10060

8. Number of books with rating > 4.6: 55
```

In [22]:
```
# □4.8 □□ 4.5 □□□□□ □□□□ □□) □□□□□□□□□ □□□□□ □□□ □□□□ □□□□□ □□□
plt.figure(figsize=(14, 8))
sns.histplot(df['avg_rating'], bins=10, kde=True, color='#e74c3c')
plt.title('1. Distribution of Average Ratings', fontsize=20)
plt.xlabel('Average Rating', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
plt.grid(True, alpha=0.3)
plt.show()
```

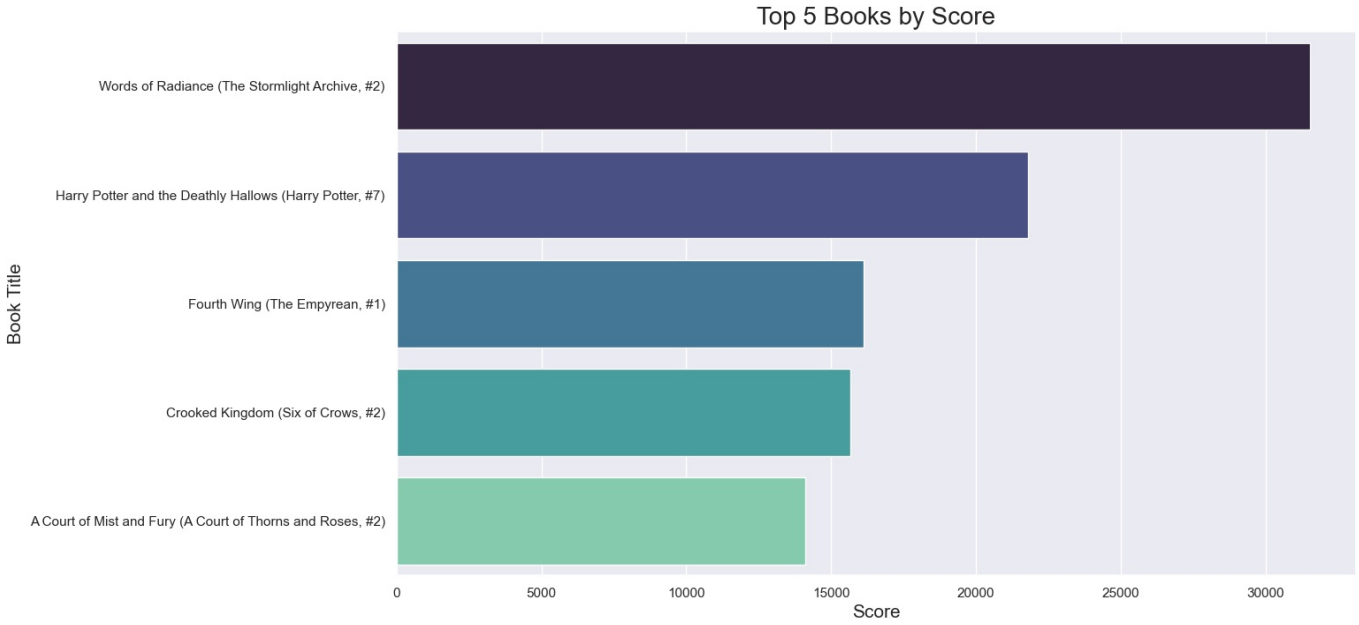


In [23]:
```
# 2. □□□□□□□□□ □□□□□□□□□ □□□ □□□ □□□□□□□□ □□□ (Scatter Plot)
sns.scatterplot(data=df, x='ratings', y='avg_rating', hue='Scores', size='Scores', sizes=(50, 500), palette='vi
plt.title('Ratings Count vs. Average Rating', fontsize=20)
plt.xlabel('Number of Ratings', fontsize=15)
plt.ylabel('Average Rating', fontsize=15)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

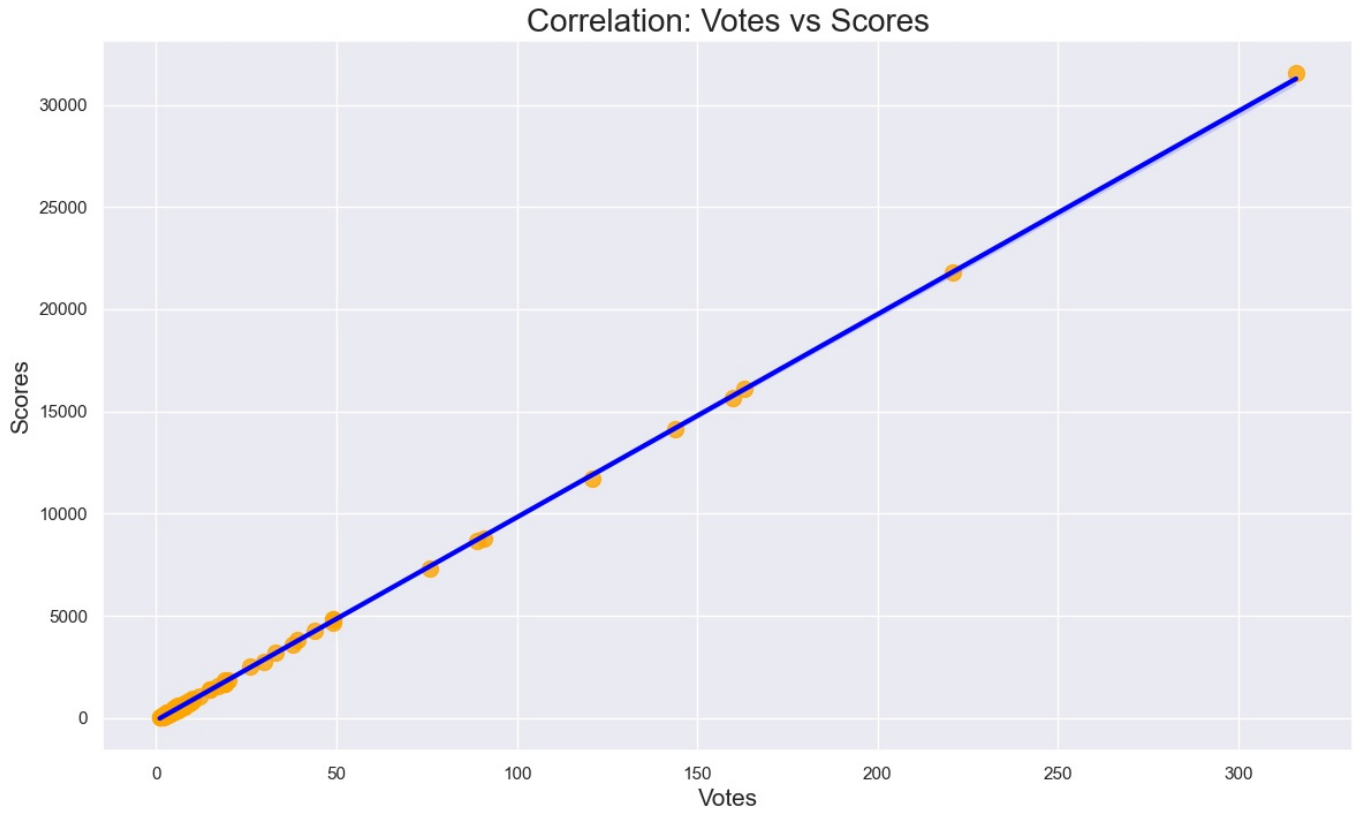


In [24]:
```
# 3. □5□□□Score□□(Bar Chart)
plt.figure(figsize=(14, 8))
top_5 = df.nlargest(5, 'Scores')
sns.barplot(data=top_5, x='Scores', y='Title', palette='mako')
plt.title('Top 5 Books by Score', fontsize=20)
plt.xlabel('Score', fontsize=15)
plt.ylabel('Book Title', fontsize=15)
plt.show()
```

```
C:\Users\Ahmed ELmarasy\AppData\Local\Temp\ipykernel_30448\2285873423.py:4: 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(data=top_5, x='Scores', y='Title', palette='mako')
```



In [25]:
```
# 4. □□□ □□□□□□ Votes □ Scores (□□□□□ □□□□□□□□ □□□□□)
plt.figure(figsize=(14, 8))
sns.regplot(data=df, x='Votes', y='Scores', scatter_kws={'color': 'orange', 's': 100}, line_kws={'color': 'blue
plt.title('Correlation: Votes vs Scores', fontsize=20)
plt.xlabel('Votes', fontsize=15)
plt.ylabel('Scores', fontsize=15)
plt.show()
```



In [26]:
```
#5- Books with Highest Reader Engagement (□□□□□□□ □□ □□□□□ □□□□ □□□ □□□□□).
plt.figure(figsize=(12, 8))
top_popular_books = df.nlargest(15, 'ratings')
sns.barplot(data=top_popular_books, x='ratings', y='Title', palette='magma')
plt.title('Top 15 Most Popular Books (by Ratings Count)', fontsize=18)
plt.xlabel('Number of Ratings', fontsize=14)
plt.ylabel('Book Title', fontsize=14)
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```

```
C:\Users\Ahmed ELmarasy\AppData\Local\Temp\ipykernel_30448\1717420256.py:4: 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(data=top_popular_books, x='ratings', y='Title', palette='magma')
```