```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

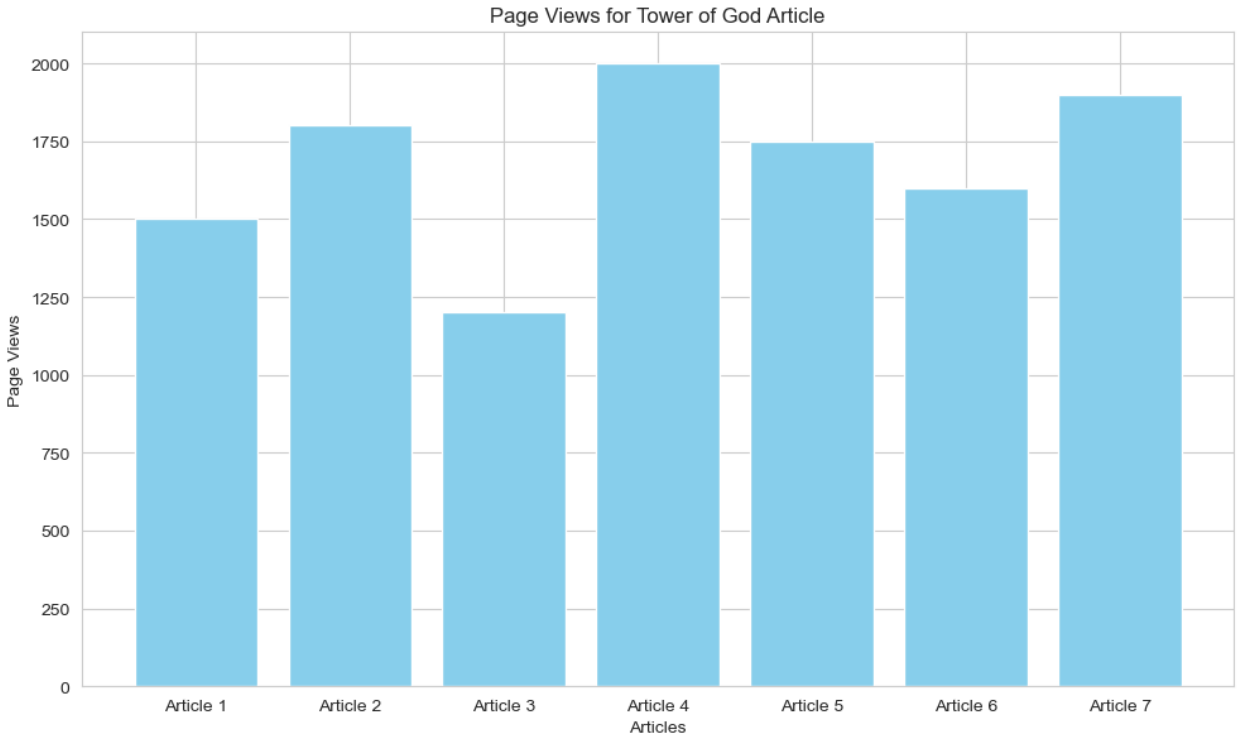
Create a dataset with 'Page View', 'Average Time Spent (seconds)', and 'Bounce Rate (%)'

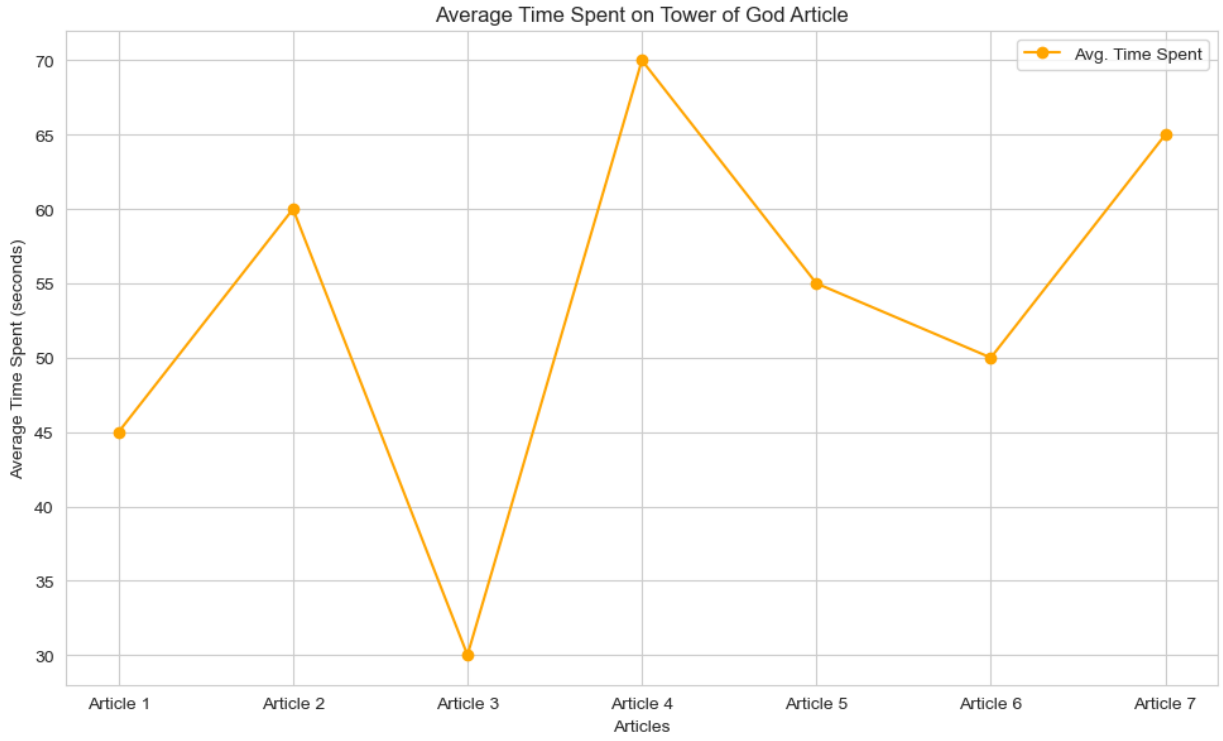
```
data = {
    'Page View': [1500, 1800, 1200, 2000, 1750, 1600, 1900],
    'Average Time Spent (seconds)': [45, 60, 30, 70, 55, 50, 65],
    'Bounce Rate (%)': [50, 45, 55, 40, 47, 49, 43]
}

# Convert the data into a DataFrame
df = pd.DataFrame(data)

df
```

	Page View	Average Time Spent (seconds)	Bounce Rate (%)
0	1500	45	50
1	1800	60	45
2	1200	30	55
3	2000	70	40
4	1750	55	47
5	1600	50	49
6	1900	65	43

```
# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Create a figure with subplots
fig, ax = plt.subplots(2, 1, figsize=(10, 12))

# First subplot: Bar graph for Page Views
ax[0].bar(df.index, df['Page View'], color='skyblue')
ax[0].set_title('Page Views for Tower of God Article')
ax[0].set_xlabel('Articles')
ax[0].set_ylabel('Page Views')
ax[0].set_xticks(df.index)
ax[0].set_xticklabels([f'Article {i+1}' for i in df.index])

# Second subplot: Line graph for Average Time Spent
ax[1].plot(df.index, df['Average Time Spent (seconds)'], marker='o',
color='orange', label='Avg. Time Spent')
ax[1].set_title('Average Time Spent on Tower of God Article')
ax[1].set_xlabel('Articles')
```

```
ax[1].set_ylabel('Average Time Spent (seconds)')
ax[1].set_xticks(df.index)
ax[1].set_xticklabels([f'Article {i+1}' for i in df.index])
ax[1].legend()

# Adjust layout
plt.tight_layout()
plt.show()
fig.savefig("output.png")
```


Page Views for Tower of God Article
Page Views
2000
1750
1500
1250
1000
750
500
250
0
Article 1
Article 2
Article 3
Article 4
Article 5
Article 6
Article 7
Articles


Average Time Spent on Tower of God Article
Avg. Time Spent
Average Time Spent (seconds)
70
65
60
55
50
45
40
35
30
Article 1
Article 2
Article 3
Article 4
Article 5
Article 6
Article 7
Articles

Strategies to Increase Time Spent:

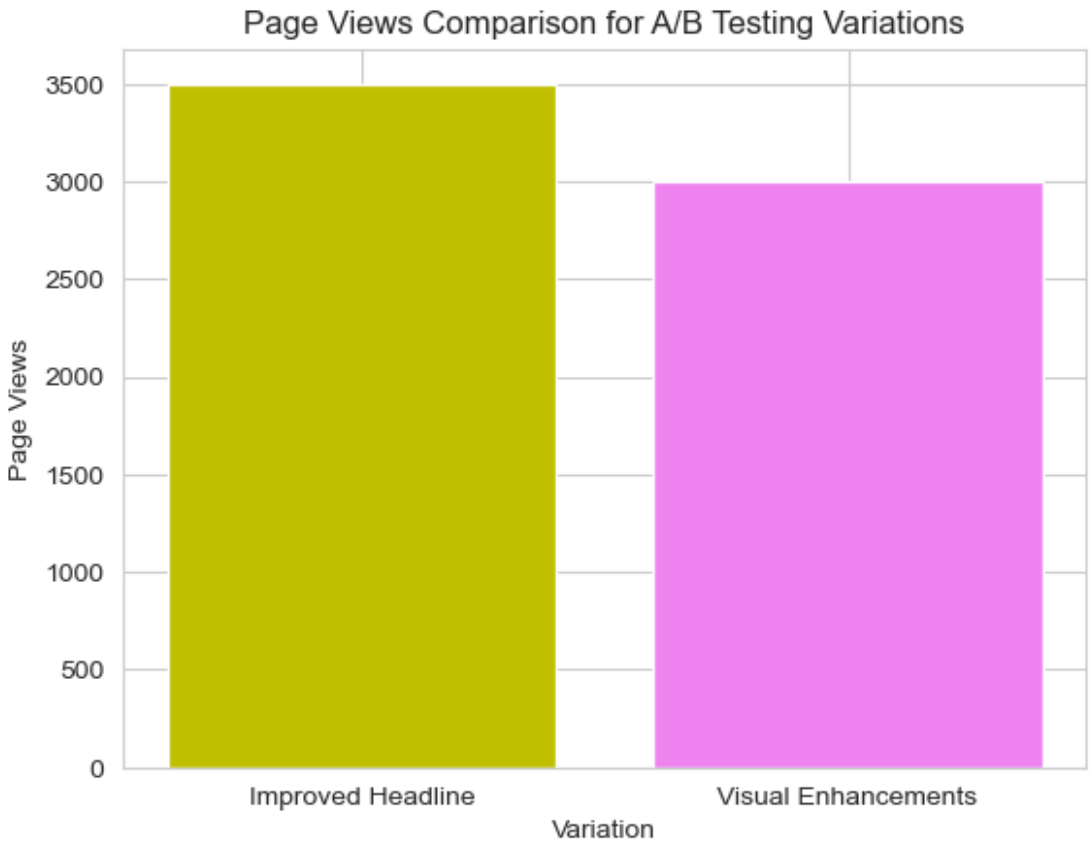
Interactive Elements: Add interactive content, such as quizzes or polls, to engage users and encourage them to stay longer. Enhanced Visuals and Related Content: Incorporate more visuals like infographics or related webtoon recommendations embedded in the article, which might capture attention and extend the user session.

Task 2

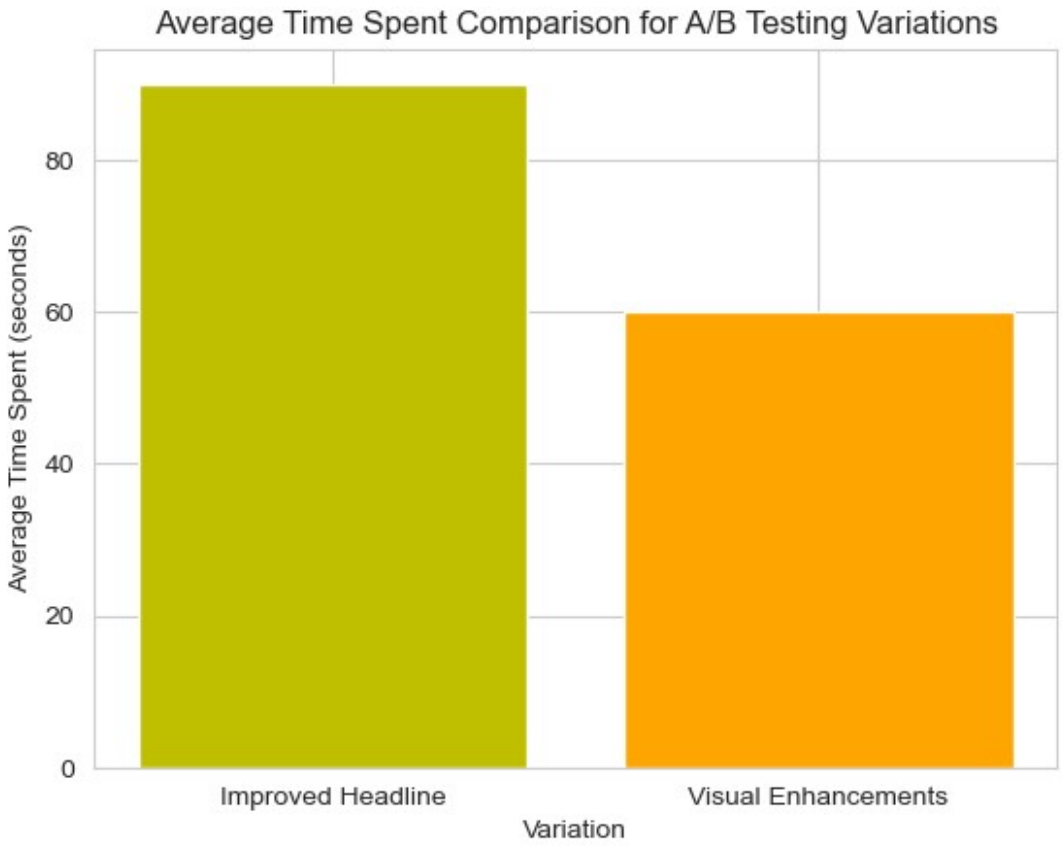
Refund High School Arc Analysis:

A/B Testing Strategy:

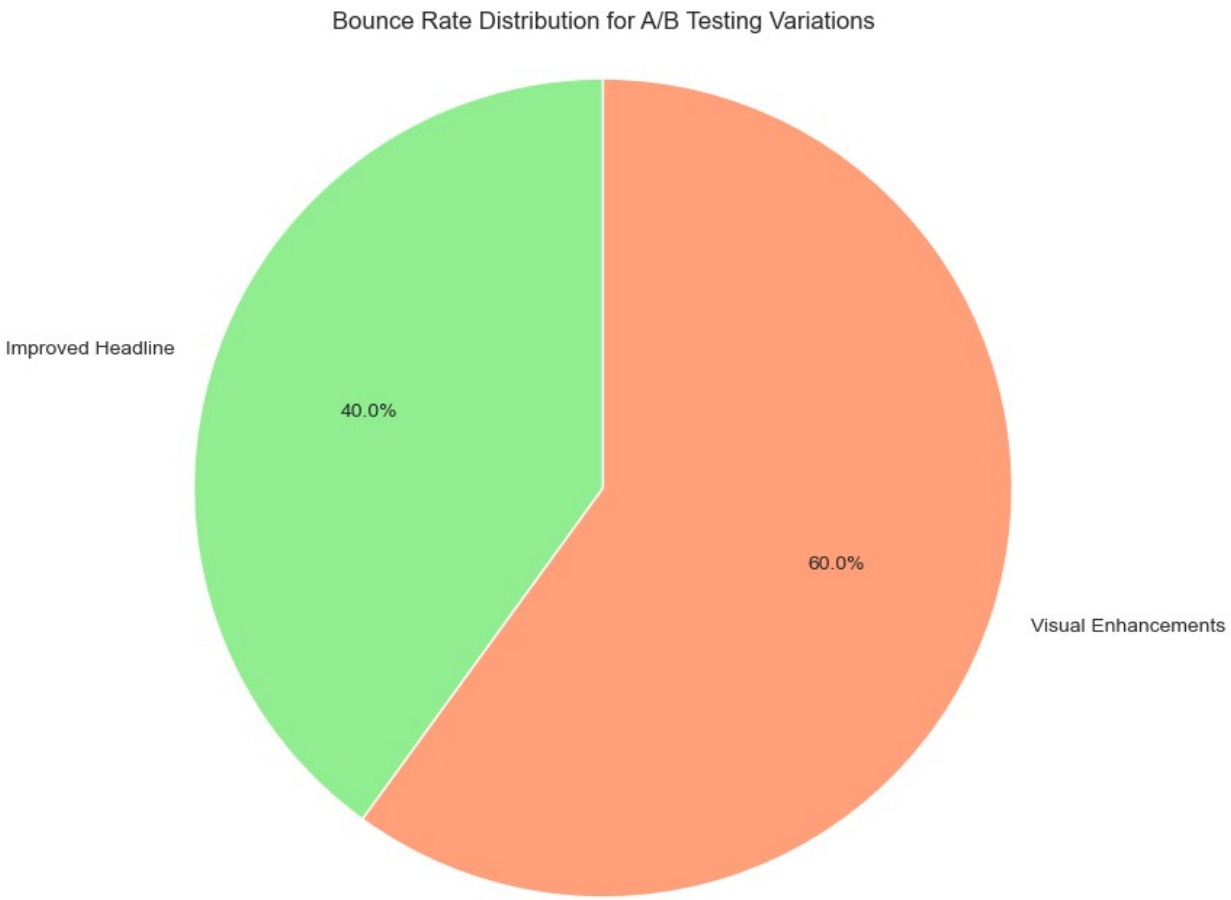
```
# Create a dataset for A/B testing of two variations
data = {
    'Variation': ['Dynamic Headline', 'Visual Enhancements'],
    'Page Views': [3200, 2800],
    'Average Time Spent (seconds)': [85, 60],
    'Bounce Rate (%)': [30, 45]
}
# Convert the data into a DataFrame
df = pd.DataFrame(data)

# Bar Graph for Page Views
plt.bar(x = df['Variation'], height = df['Page Views'] ,color = ["y" ,
"violet"])
plt.title('Page Views Comparison for A/B Testing Variations'  )
plt.xlabel('Variation')
plt.ylabel('Page Views')
# Save the figure with an appropriate filename and extension
plt.savefig("page_views_comparison.png", format='png')
# Show the plot
plt.show()
```




```
# Bar Graph for Page Views
plt.bar(x = df['Variation'], height = df['Average Time Spent
(seconds)'] ,color = ["y" , "orange"])
plt.title('Average Time Spent Comparison for A/B Testing Variations')
plt.xlabel('Variation')
plt.ylabel('Average Time Spent (seconds)')
plt.savefig("Average_Time_Spent_Comparison1.png", format='png')
plt.show()
```



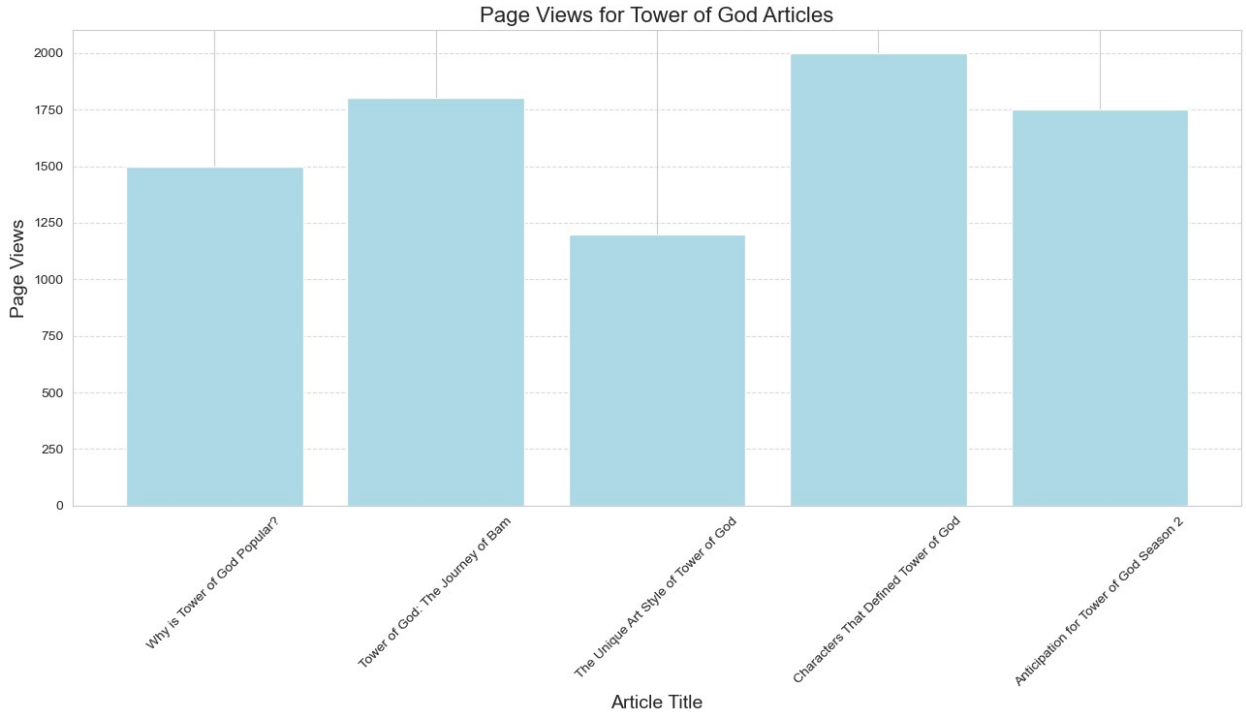

```
# Visualization: Pie Chart for Bounce Rate
plt.figure(figsize=(8, 8))
plt.pie(df['Bounce Rate (%)'], labels=df['Variation'], autopct='%1.1f%
%', startangle=90, colors=['lightgreen', 'lightsalmon'])
plt.title('Bounce Rate Distribution for A/B Testing Variations')
plt.axis('equal')  # Equal aspect ratio ensures pie chart is circular.
plt.savefig("Bounce Rate Distribution.png", format='png')
plt.show()
```

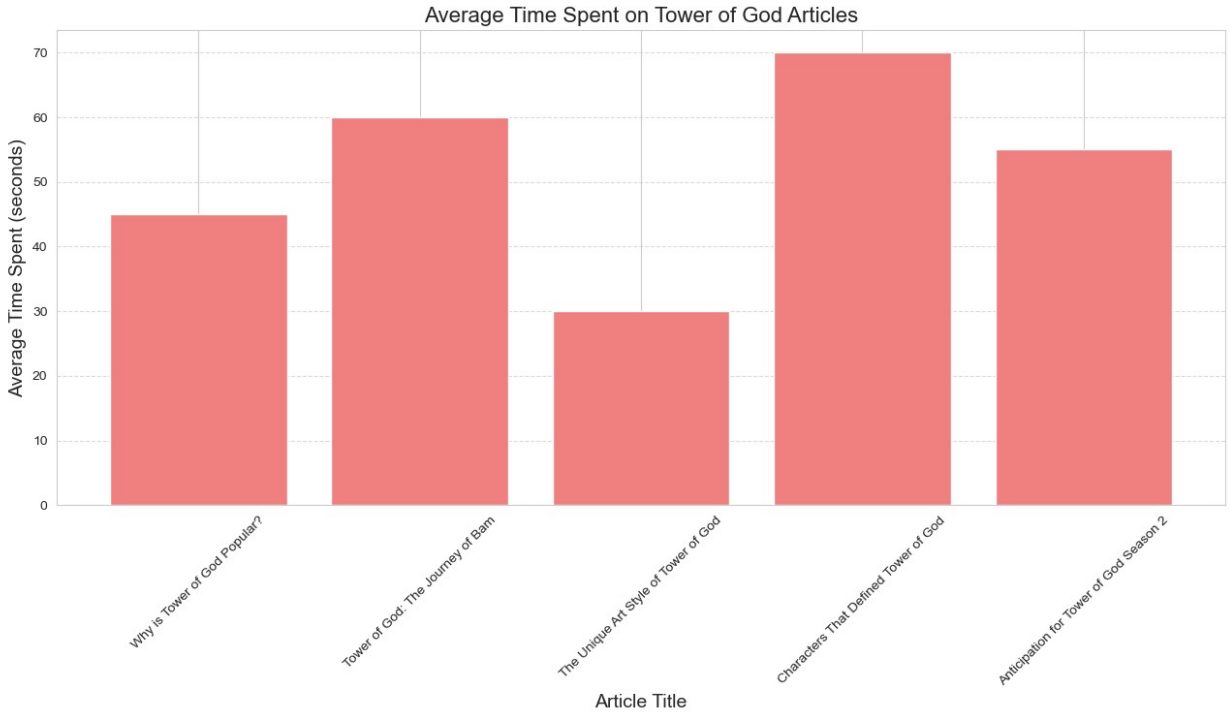


Create a dataset for Tower of God articles

```
data = {
    'Article Title': [
        'Why is Tower of God Popular?',
        'Tower of God: The Journey of Bam',
        'The Unique Art Style of Tower of God',
        'Characters That Defined Tower of God',
        'Anticipation for Tower of God Season 2'
    ],
    'Page Views': [1500, 1800, 1200, 2000, 1750],
    'Average Time Spent (seconds)': [45, 60, 30, 70, 55],
    'Bounce Rate (%)': [50, 45, 55, 40, 47]
}

# Convert the data into a DataFrame
df = pd.DataFrame(data)

# Visualization: Bar Graph for Page Views and Average Time Spent
```

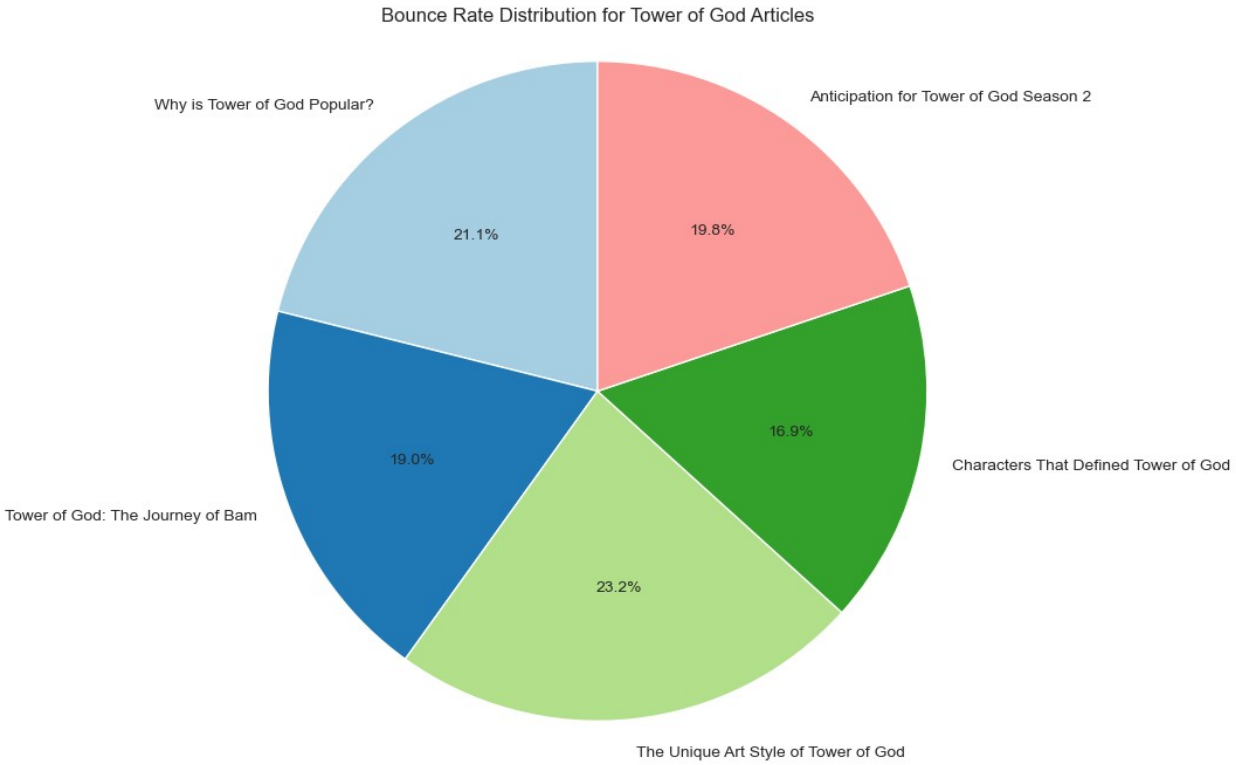
```
fig, ax = plt.subplots(2, 1, figsize=(13, 15))

# Bar Graph for Page Views
ax[0].bar(df['Article Title'], df['Page Views'], color='lightblue')
ax[0].set_title('Page Views for Tower of God Articles', fontsize=16)
ax[0].set_xlabel('Article Title', fontsize=14)
ax[0].set_ylabel('Page Views', fontsize=14)
ax[0].tick_params(axis='x', rotation=45)
ax[0].grid(axis='y', linestyle='--', alpha=0.7)

# Bar Graph for Average Time Spent
ax[1].bar(df['Article Title'], df['Average Time Spent (seconds)'],
color='lightcoral')
ax[1].set_title('Average Time Spent on Tower of God Articles',
fontsize=16)
ax[1].set_xlabel('Article Title', fontsize=14)
ax[1].set_ylabel('Average Time Spent (seconds)', fontsize=14)
ax[1].tick_params(axis='x', rotation=45)
ax[1].grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()  # Adjust layout to prevent overlap
plt.savefig("tower_of_god.png", format='png')
plt.show()
```






```
# Visualization: Pie Chart for Bounce Rate
plt.figure(figsize=(8, 8))
plt.pie(df['Bounce Rate (%)'], labels=df['Article Title'],
autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)
plt.title('Bounce Rate Distribution for Tower of God Articles')
plt.axis('equal')  # Equal aspect ratio ensures pie chart is circular.
```

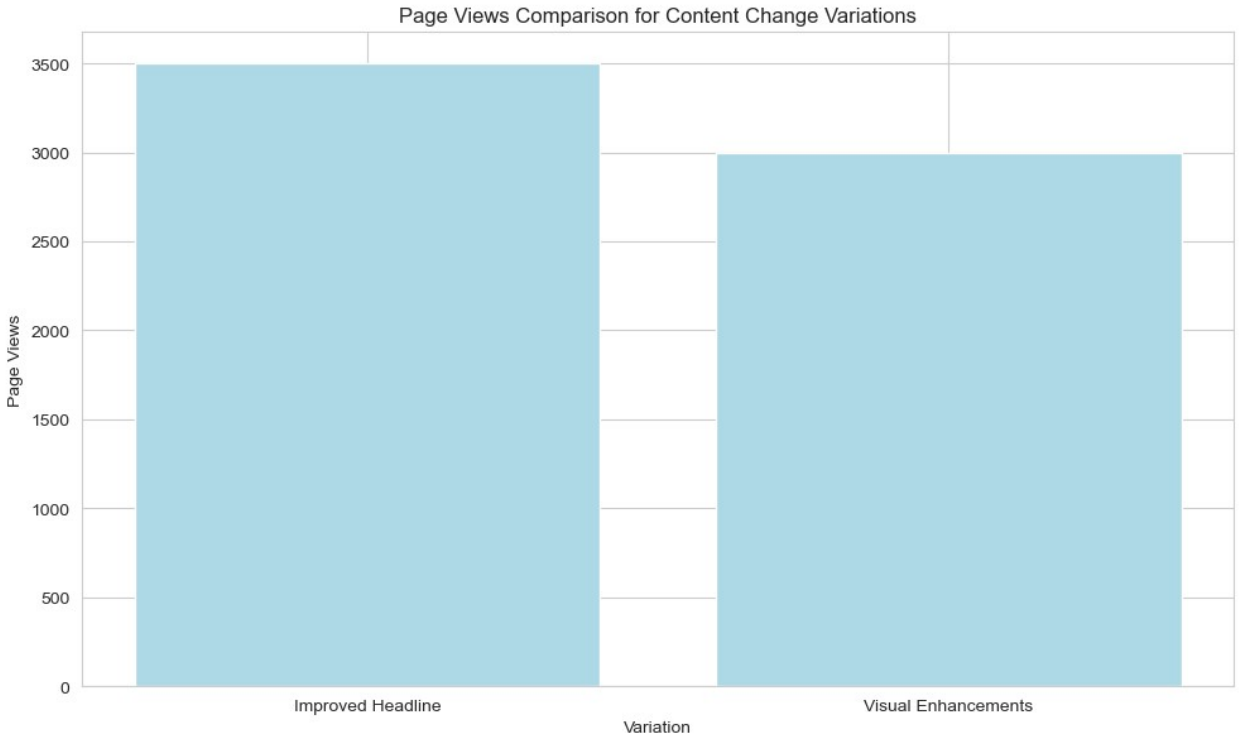
```
plt.savefig("Bounce_Rate_Distribution.png", format='png')
plt.show()
```

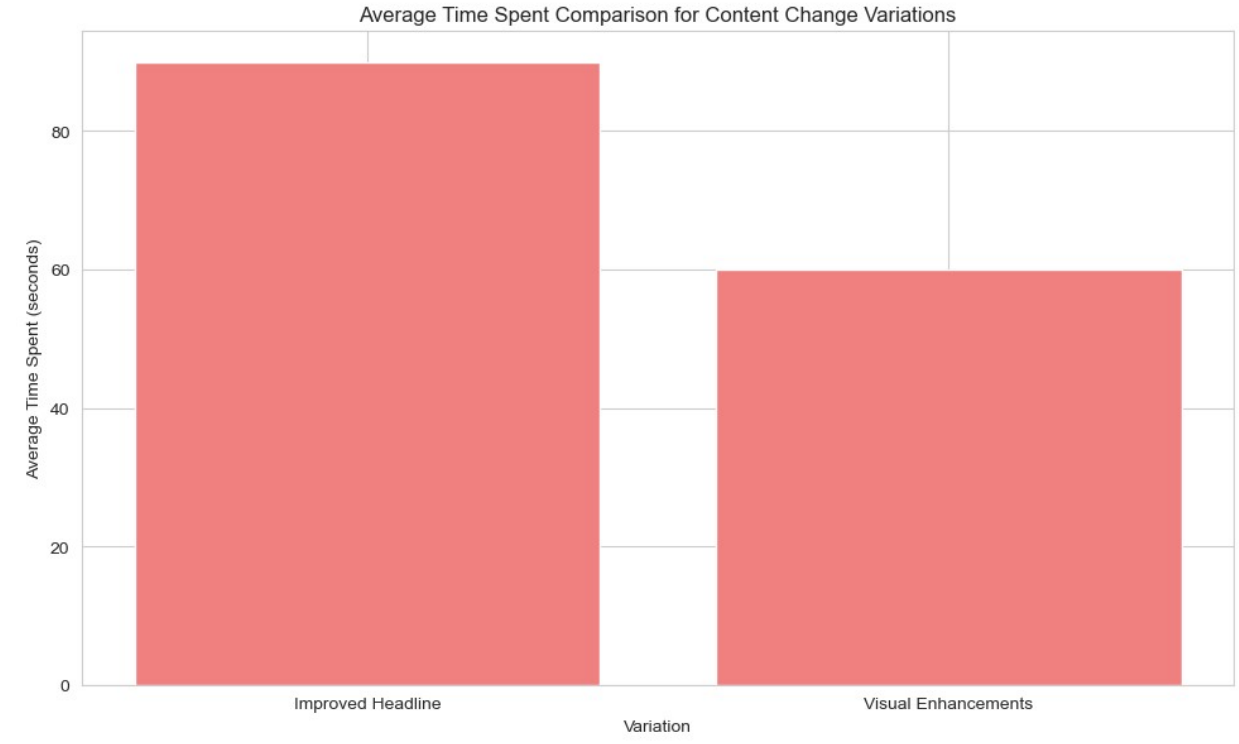


b> Content Change Suggestions

```
# Create a dataset for content change suggestions
data = {
    'Variation': ['Improved Headline', 'Visual Enhancements'],
    'Page Views': [3500, 3000],
    'Average Time Spent (seconds)': [90, 60],
    'Bounce Rate (%)': [30, 45]
}

# Convert the data into a DataFrame
df = pd.DataFrame(data)

# Visualization: Bar Graph for Page Views and Average Time Spent
fig, ax = plt.subplots(2, 1, figsize=(10, 12))

# Bar Graph for Page Views
ax[0].bar(df['Variation'], df['Page Views'], color='lightblue')
ax[0].set_title('Page Views Comparison for Content Change Variations')
ax[0].set_xlabel('Variation')
ax[0].set_ylabel('Page Views')
```

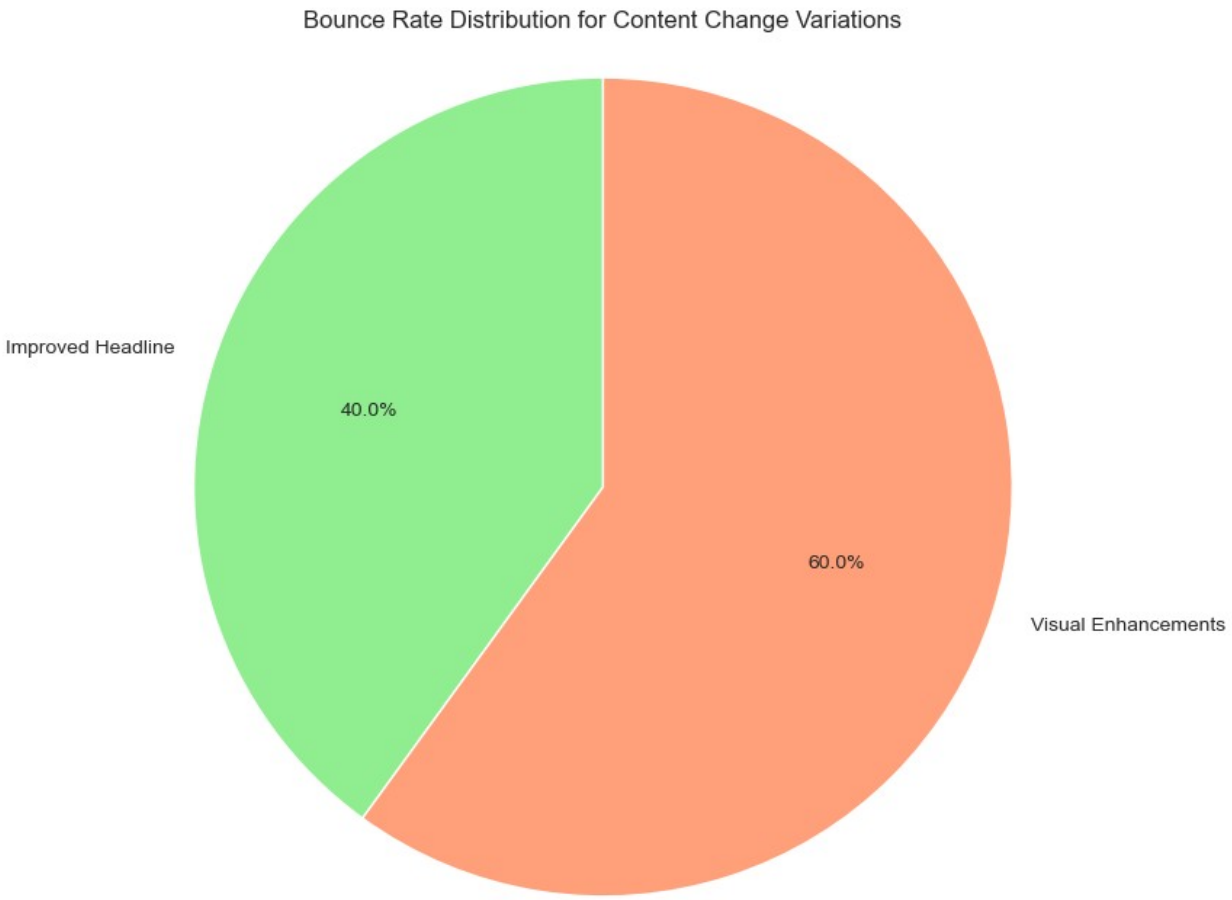
```
# Bar Graph for Average Time Spent
ax[1].bar(df['Variation'], df['Average Time Spent (seconds)'],
color='lightcoral')
ax[1].set_title('Average Time Spent Comparison for Content Change
Variations')
ax[1].set_xlabel('Variation')
ax[1].set_ylabel('Average Time Spent (seconds)')

# Adjust layout
plt.tight_layout()
plt.savefig("Content_Change_Suggestions.png", format='png')

plt.show()
```






```
# Visualization: Pie Chart for Bounce Rate
plt.figure(figsize=(8, 8))
plt.pie(df['Bounce Rate (%)'], labels=df['Variation'], autopct='%1.1f%
%', startangle=90, colors=['lightgreen', 'lightsalmon'])
plt.title('Bounce Rate Distribution for Content Change Variations')
```

```
plt.axis('equal')  # Equal aspect ratio ensures pie chart is circular.
plt.savefig("Bounce Rate Distribution.png" , format = "png" )
plt.show()
```



TASK -> 3

Solo Leveling Arcs - User Segmentation:

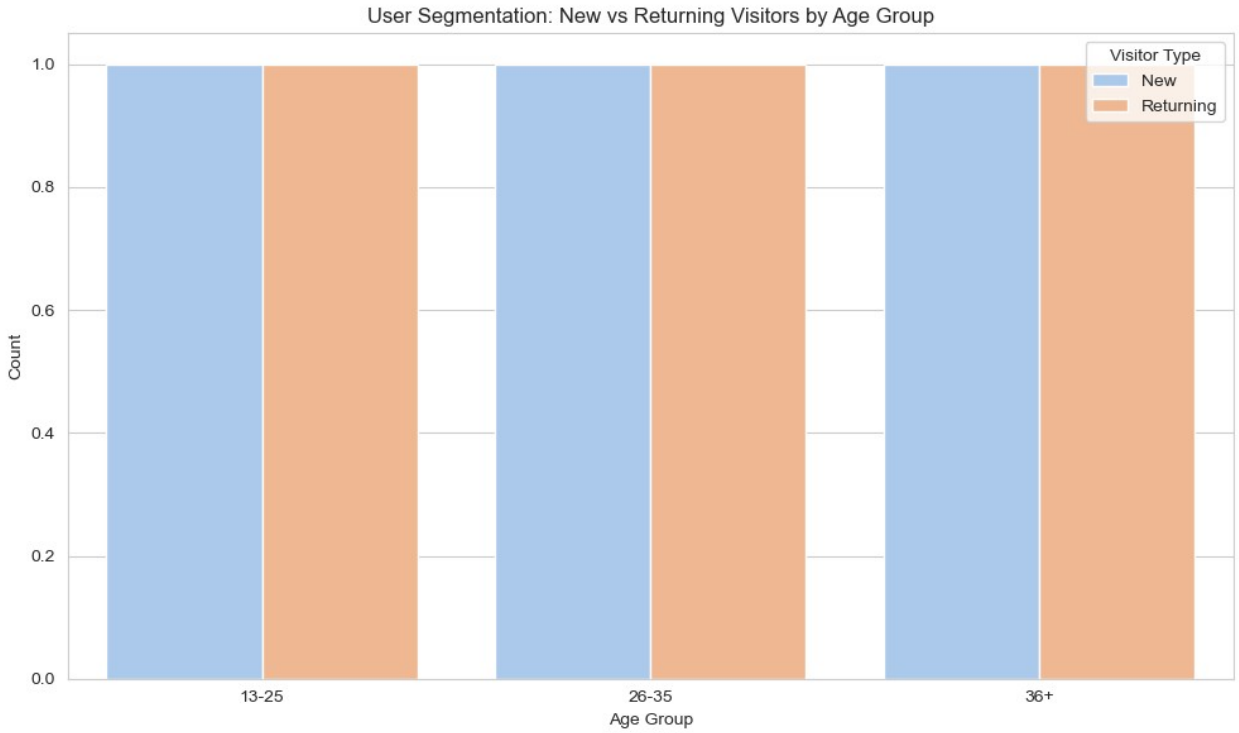
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Create the dataset for user segmentation
data = {
    'User Segment': [
        'Younger Audience (New)', 'Younger Audience (Returning)',
        'Adult Audience (New)', 'Adult Audience (Returning)',
```

```
        'Mature Audience (New)', 'Mature Audience (Returning)'
    ],
    'Age Group': ['13-25', '13-25', '26-35', '26-35', '36+', '36+'],
    'New/Returning': ['New', 'Returning', 'New', 'Returning', 'New',
'Returning'],
    'Recommended Content': [
        'Introductory guide to arcs',
        'Easter eggs and advanced theories',
        'Detailed analysis of key arcs',
        'Character deep dives and plot connections',
        'Overview of themes and character journeys',
        'In-depth critiques and plot analysis'
    ]
}
df1 = pd.DataFrame(data)

df1
```

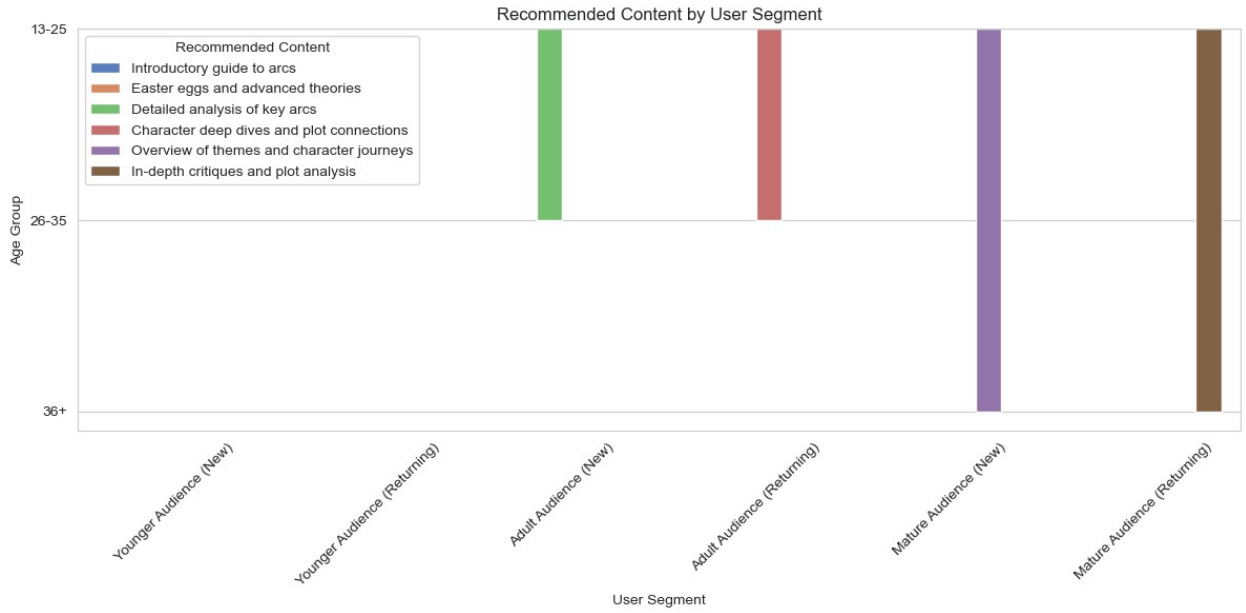
	User Segment	Age Group	New/Returning
0	Younger Audience (New)	13-25	New
1	Younger Audience (Returning)	13-25	Returning
2	Adult Audience (New)	26-35	New
3	Adult Audience (Returning)	26-35	Returning
4	Mature Audience (New)	36+	New
5	Mature Audience (Returning)	36+	Returning

	Recommended Content
0	Introductory guide to arcs
1	Easter eggs and advanced theories
2	Detailed analysis of key arcs
3	Character deep dives and plot connections
4	Overview of themes and character journeys
5	In-depth critiques and plot analysis

```
# Visualization: Count plot for New vs Returning Users by Age Group
plt.figure(figsize=(10, 6))
sns.countplot(data=df1, x='Age Group', hue='New/Returning',
palette='pastel')
plt.title('User Segmentation: New vs Returning Visitors by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.legend(title='Visitor Type')
plt.tight_layout()
plt.savefig("New vs Returning Users by Age Group.png" , format = "png"
)
plt.show()
```




```
# Visualization: Sample Recommendations
plt.figure(figsize=(12, 6))
sns.barplot(data=df1, x='User Segment', y='Age Group',
hue='Recommended Content', dodge=True, palette='muted')
plt.title('Recommended Content by User Segment')
plt.xlabel('User Segment')
plt.ylabel('Age Group')
plt.xticks(rotation=45)
plt.legend(title='Recommended Content')
plt.tight_layout()
plt.savefig("Recommended Content by user.png" , format = "png" )
plt.show()
```



Solo Leveling Arcs - User Segmentation:

User Segmentation: Age Group Segmentation: Create content targeted at different age groups. Younger audiences might appreciate more visual content, while older readers might prefer in-depth analysis. Returning vs. New Visitors: Tailor content to provide background information for new visitors, while offering unique insights or Easter eggs for returning users.

Tailored Content: For new users, create an introductory guide to the arcs or storyline. For returning users, offer more intricate details, spoiler discussions, or advanced theories.