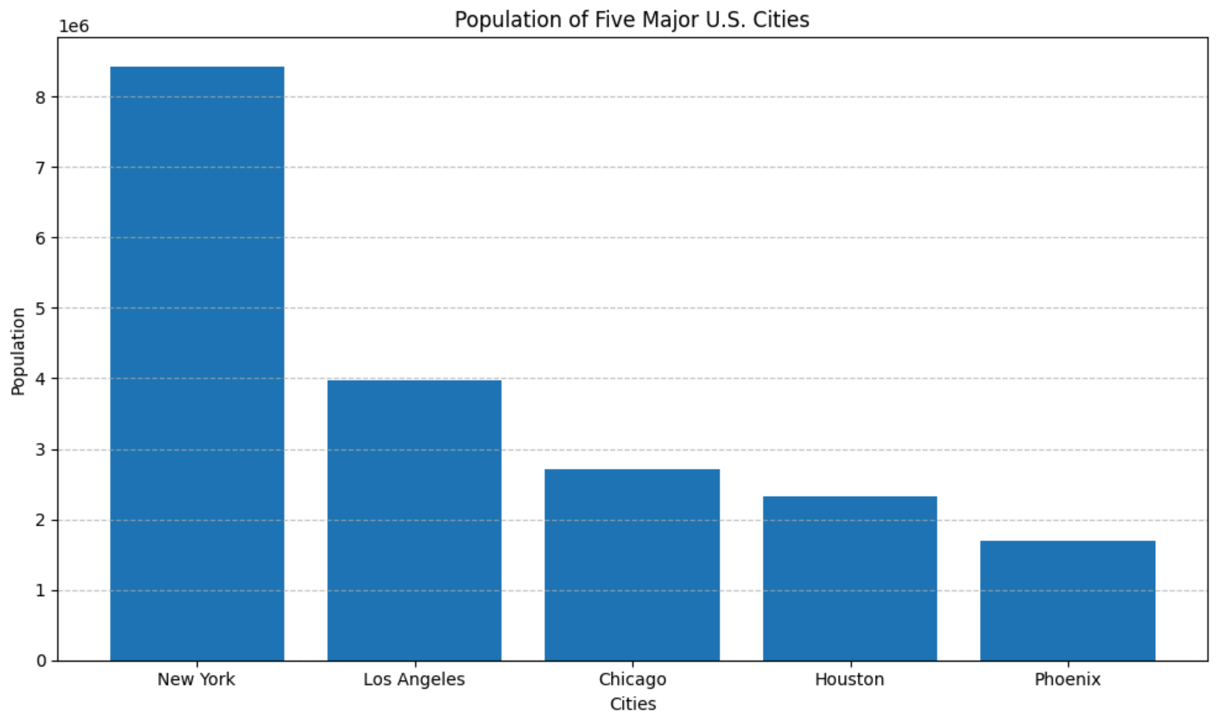
1. Write a Python program to create a bar graph that displays the population of five different cities. Customize the bar graph by adding a title, labels for the axes, and different colors for each bar.

In [7]:
```python
import matplotlib.pyplot as plt
# Data
cities = ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix']
populations = [8419000, 3980000, 2716000, 2328000, 1690000]
colors = ['red', 'blue', 'green', 'orange', 'purple']

#Code here::
plt.figure(figsize=(10, 6))
plt.bar(cities, populations)
plt.title('Population of Five Major U.S. Cities')
plt.xlabel('Cities')
plt.ylabel('Population')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

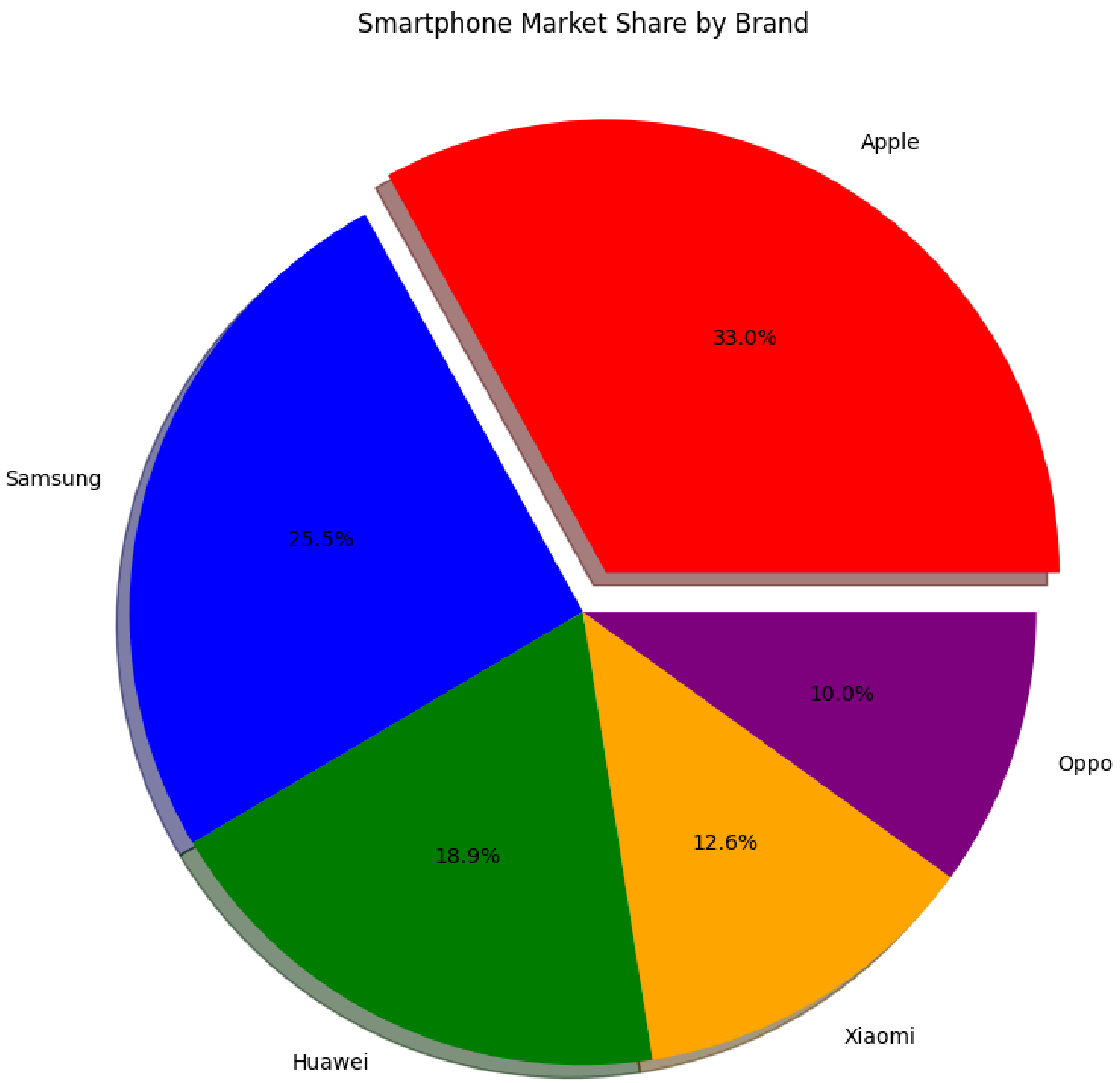


2. Write a Python program to create a pie chart that represents the market share of different smartphone brands. Label each wedge with the brand name and percentage, and highlight (explode) the wedge representing higher market share.

In [8]:
```
# Data
brands = ['Apple', 'Samsung', 'Huawei', 'Xiaomi', 'Oppo']
market_share = [27.5, 21.3, 15.8, 10.5, 8.3]

#Code here::

explode = [0.1 if share == max(market_share) else 0 for share in market_share]
plt.figure(figsize=(8, 8))
plt.pie(market_share, labels=brands, autopct='%1.1f%%', colors=colors, explode=expl
plt.title('Smartphone Market Share by Brand')
plt.tight_layout()
plt.show()
```



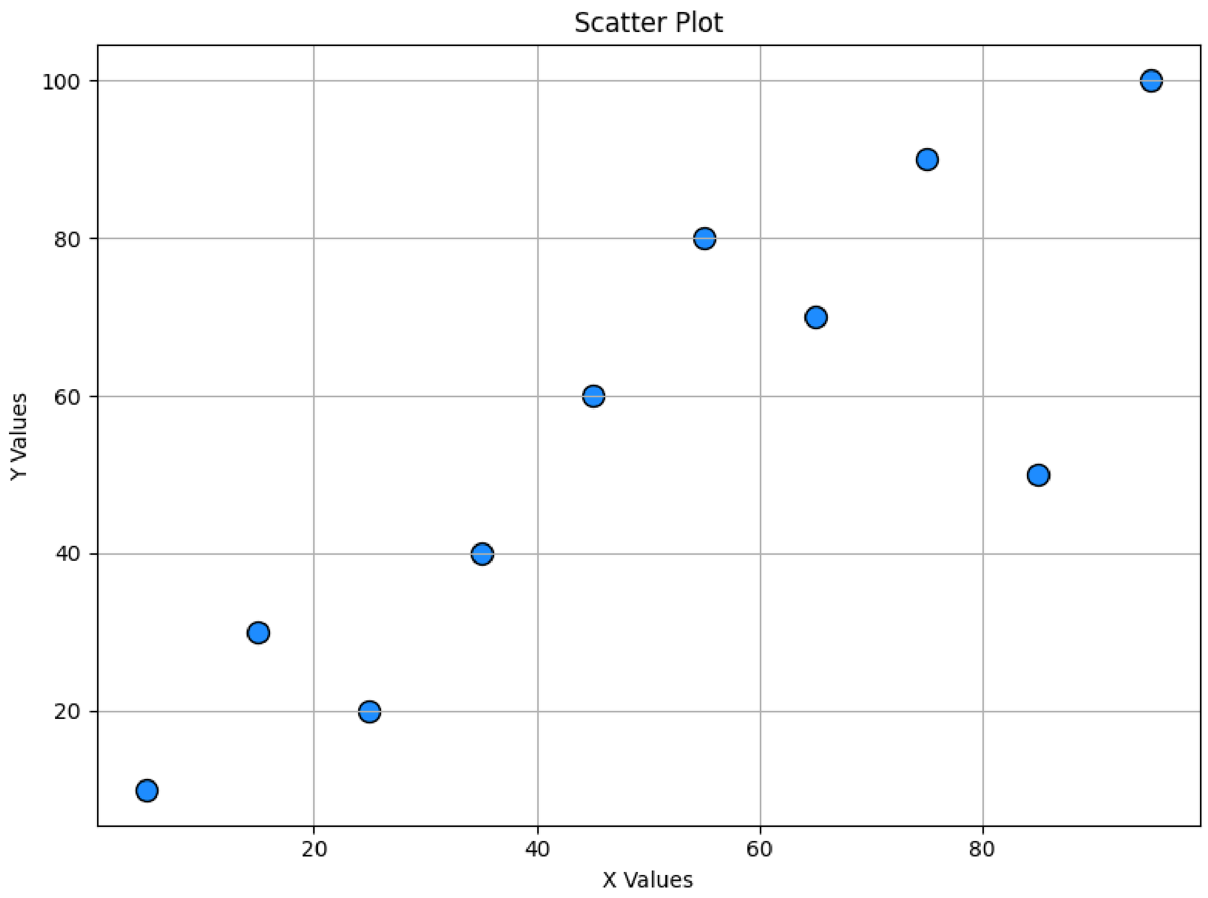
---

3. Create a Python program that generates a scatter plot using the provided data. Customize the plot by: Setting the size of each point to 100, adding a title "Scatter Plot", and labeling the x-axis as "X Values" and the y-axis as "Y Values".

In [11]:
```python
# Data
x = [5, 15, 25, 35, 45, 55, 65, 75, 85, 95]
y = [10, 30, 20, 40, 60, 80, 70, 90, 50, 100]

#Code here::

plt.figure(figsize=(8, 6))
plt.scatter(x, y, s=100, color='dodgerblue', edgecolor='black')
plt.title('Scatter Plot')
plt.xlabel('X Values')
plt.ylabel('Y Values')
plt.grid(True)
plt.tight_layout()
plt.show()
```

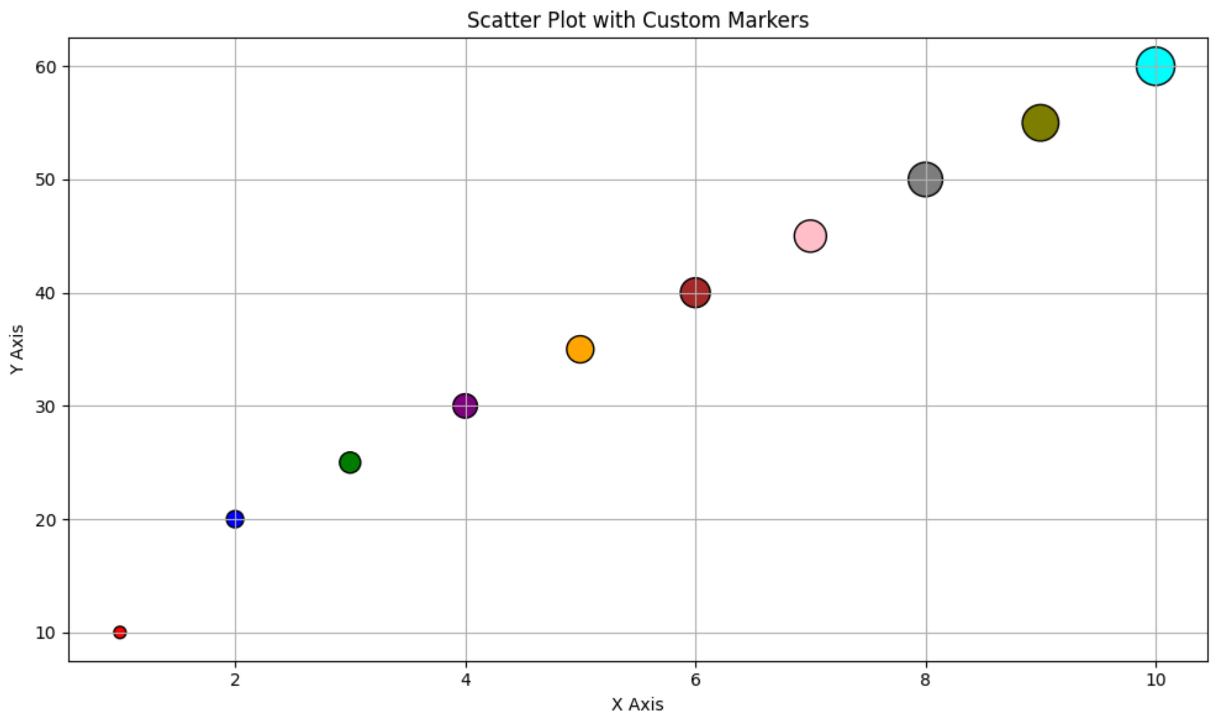


4. Write a Python program to create a scatter plot with custom markers Customize the markers to have different colors and sizes.

In [10]:
```python
# Data
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [10, 20, 25, 30, 35, 40, 45, 50, 55, 60]
colors = ['red', 'blue', 'green', 'purple', 'orange', 'brown', 'pink', 'gray', 'oli
sizes = [50, 100, 150, 200, 250, 300, 350, 400, 450, 500]
```

```python
#Code here::

plt.figure(figsize=(10, 6))
plt.scatter(x, y, c=colors, s=sizes, edgecolor='black')
plt.title('Scatter Plot with Custom Markers')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.grid(True)
plt.tight_layout()
plt.show()
```

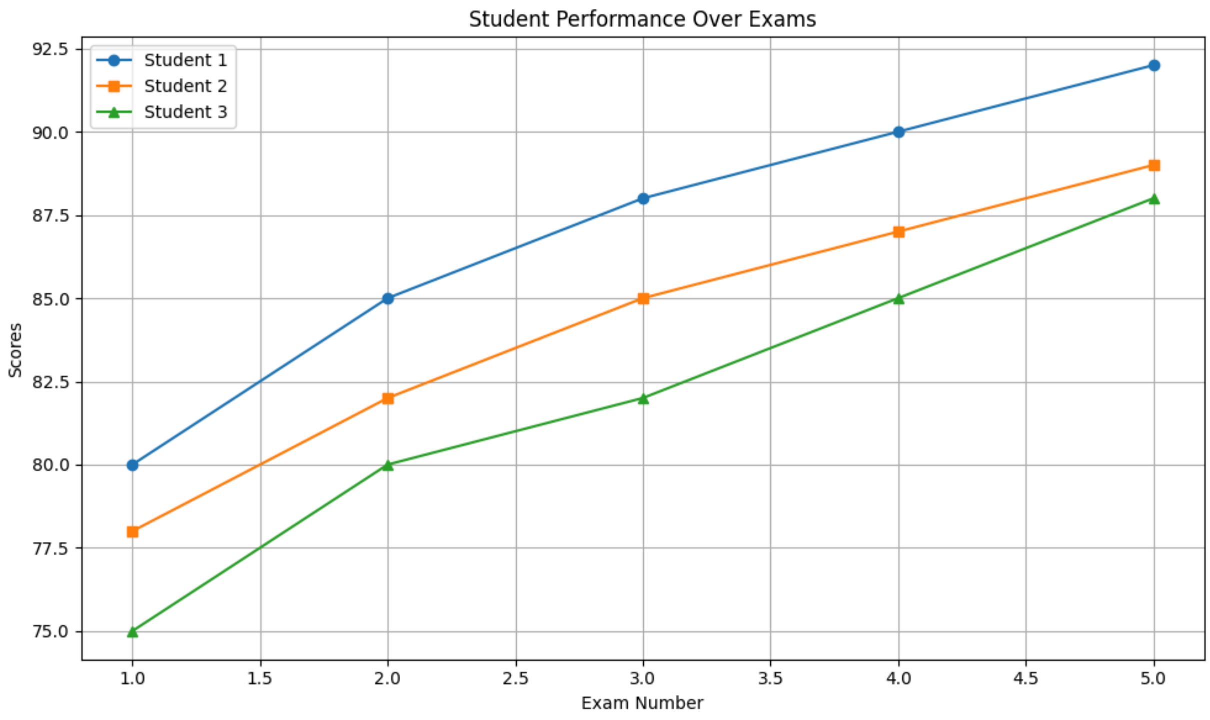


---

5. Write a Python program to create a line plot that shows the performance of three different students over five exams.Plot all three lines on the same graph, using different colors and markers for each line. Add a legend to identify each student.

In [12]:
```python
# Data
exams = [1, 2, 3, 4, 5]
student1_scores = [80, 85, 88, 90, 92]
student2_scores = [78, 82, 85, 87, 89]
student3_scores = [75, 80, 82, 85, 88]

#Code here::
plt.figure(figsize=(10, 6))
plt.plot(exams, student1_scores, marker='o', label='Student 1')
plt.plot(exams, student2_scores, marker='s', label='Student 2')
plt.plot(exams, student3_scores, marker='^', label='Student 3')
plt.title('Student Performance Over Exams')
plt.xlabel('Exam Number')
plt.ylabel('Scores')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

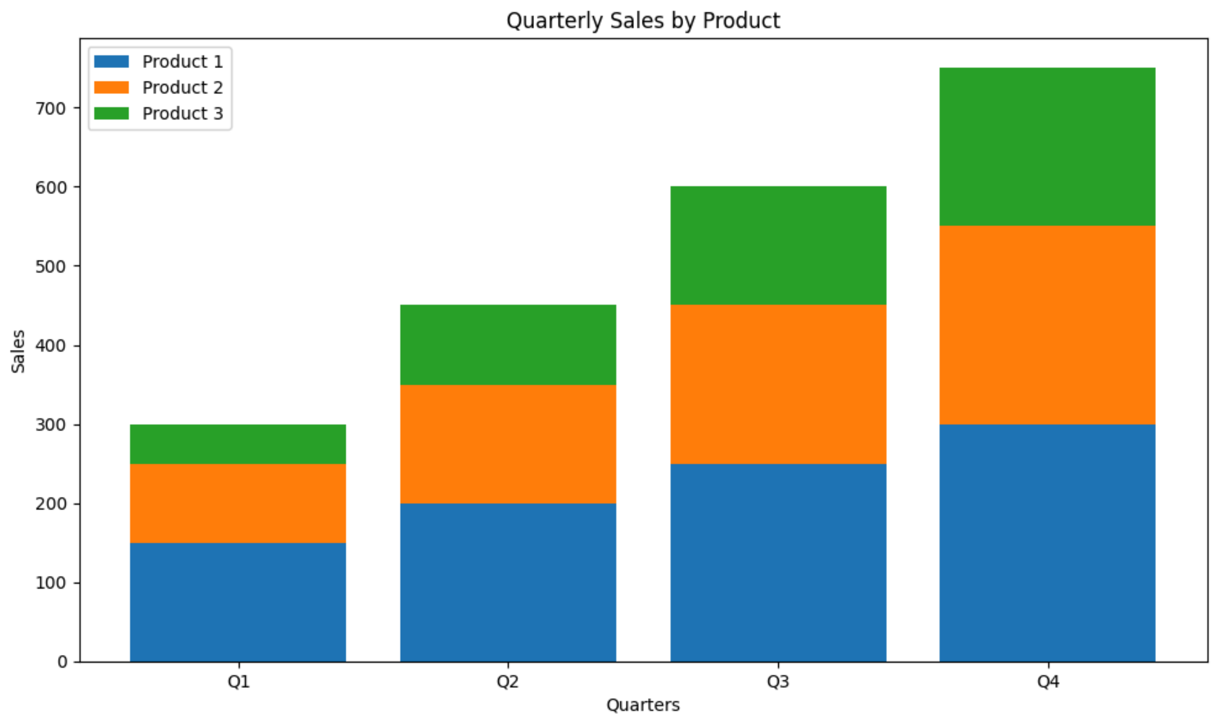


---

6. Write a Python program to create a stacked bar chart that represents the sales of three different products over four quarters. use different colors for each product, and add a legend.

In [13]:
```python
# Data
quarters = ['Q1', 'Q2', 'Q3', 'Q4']
product1_sales = [150, 200, 250, 300]
product2_sales = [100, 150, 200, 250]
product3_sales = [50, 100, 150, 200]

#Code here::
plt.figure(figsize=(10, 6))
plt.bar(quarters, product1_sales, label='Product 1')
plt.bar(quarters, product2_sales, bottom=product1_sales, label='Product 2')
bottom = [sum(x) for x in zip(product1_sales, product2_sales)]
plt.bar(quarters, product3_sales, bottom=bottom, label='Product 3')
plt.title('Quarterly Sales by Product')
plt.xlabel('Quarters')
plt.ylabel('Sales')
plt.legend()
plt.tight_layout()
plt.show()
```

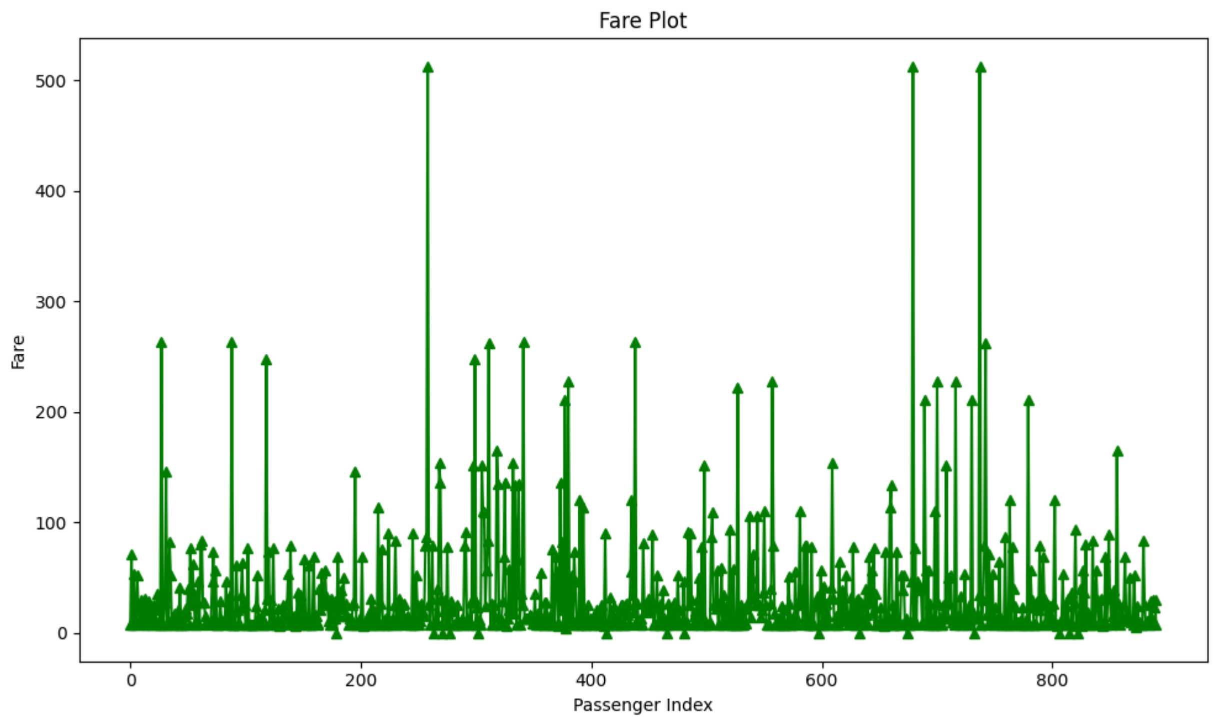


---

7. Load the titanic.csv dataset using pandas and plot the 'fare' column with a line plot. Customize the plot to have a green line with triangle markers, a title "Fare Plot", and labels for the x and y axes.?

In [15]:
```python
#Code here::
import pandas as pd
df = pd.read_csv('titanic.csv')

plt.figure(figsize=(10, 6))
plt.plot(df['fare'], marker='^', linestyle='-', color='green')
plt.title('Fare Plot')
plt.xlabel('Passenger Index')
plt.ylabel('Fare')
plt.tight_layout()
plt.show()
```

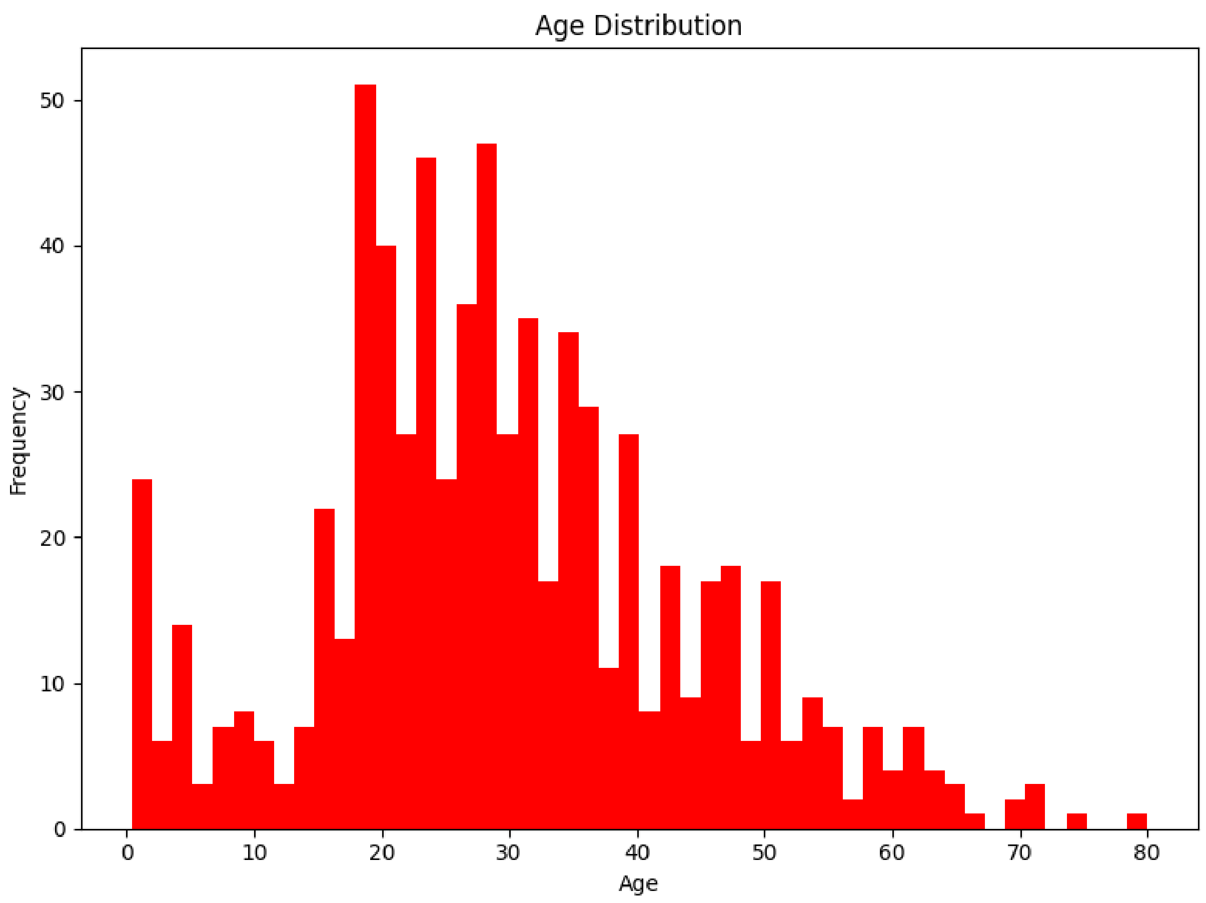


---

8. Plot a histogram of the 'age' column from the Titanic dataset. Customize the histogram to have 50 bins, a title "Age Distribution", and a red color for the bars.

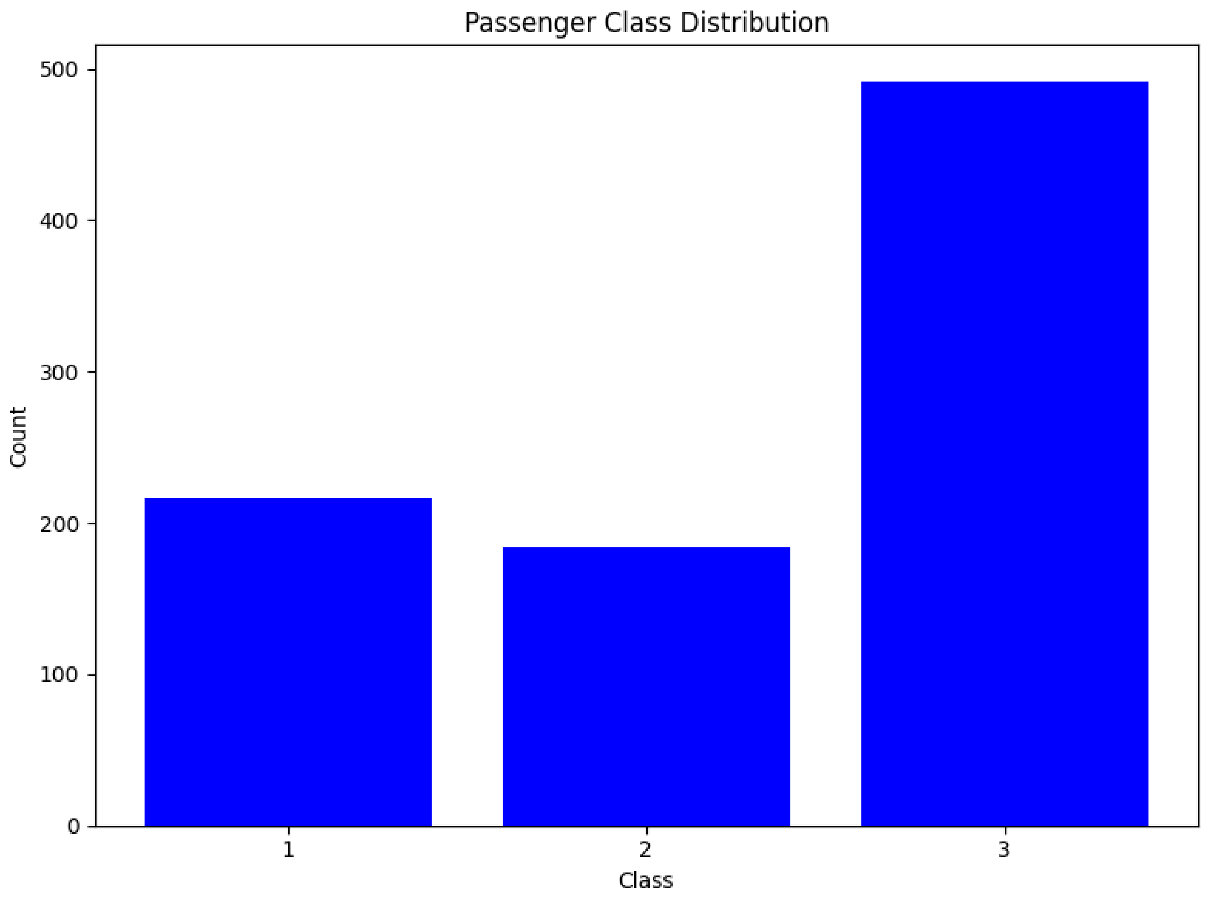
In [16]:

```
#Code here:
plt.figure(figsize=(8, 6))
plt.hist(df['age'].dropna(), bins=50, color='red')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



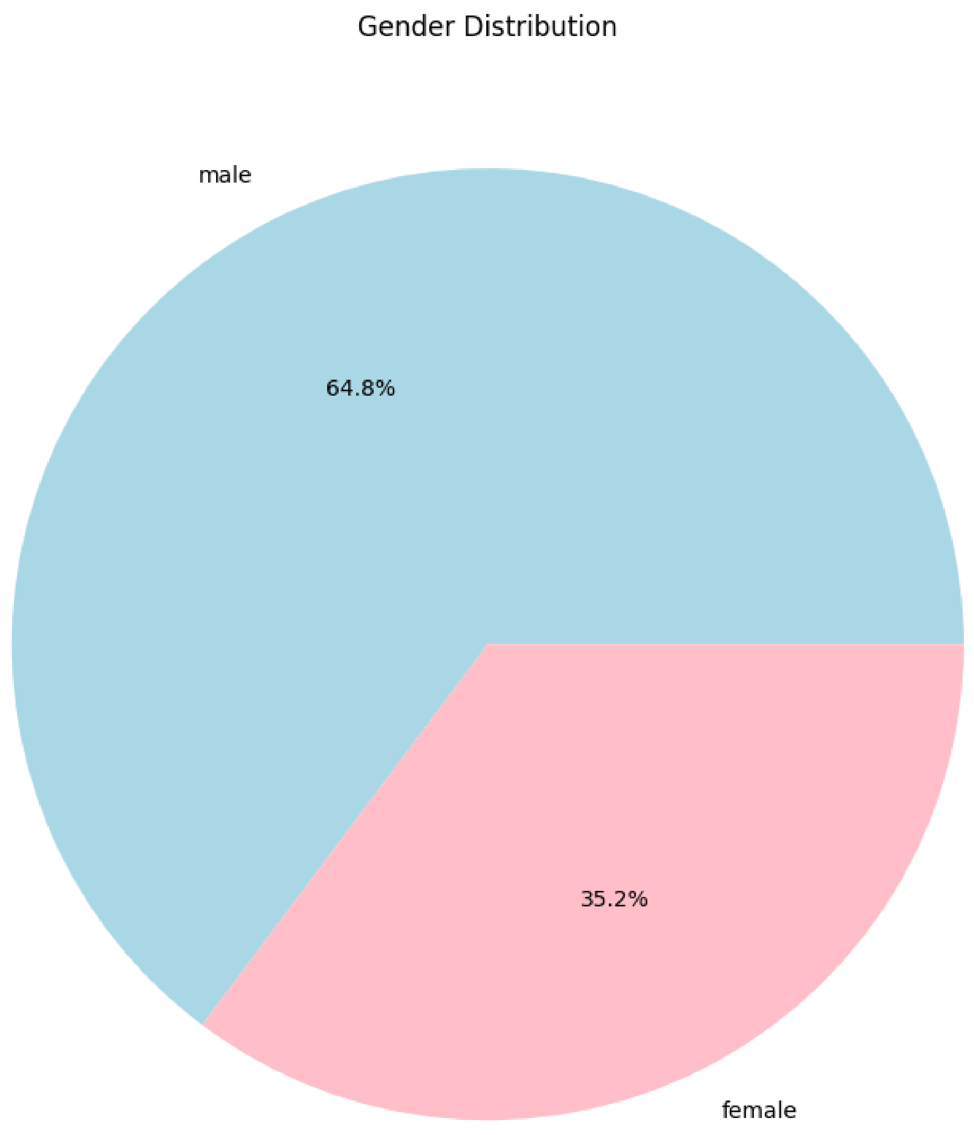
9.Create a bar chart of the 'pclass' column in Titanic dataset, showing the count of each class. Set the color of the bars to blue.

```
In [17]: #Code here:
pclass_counts = df['pclass'].value_counts().sort_index()
plt.figure(figsize=(8, 6))
plt.bar(pclass_counts.index.astype(str), pclass_counts.values, color='blue')
plt.title('Passenger Class Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



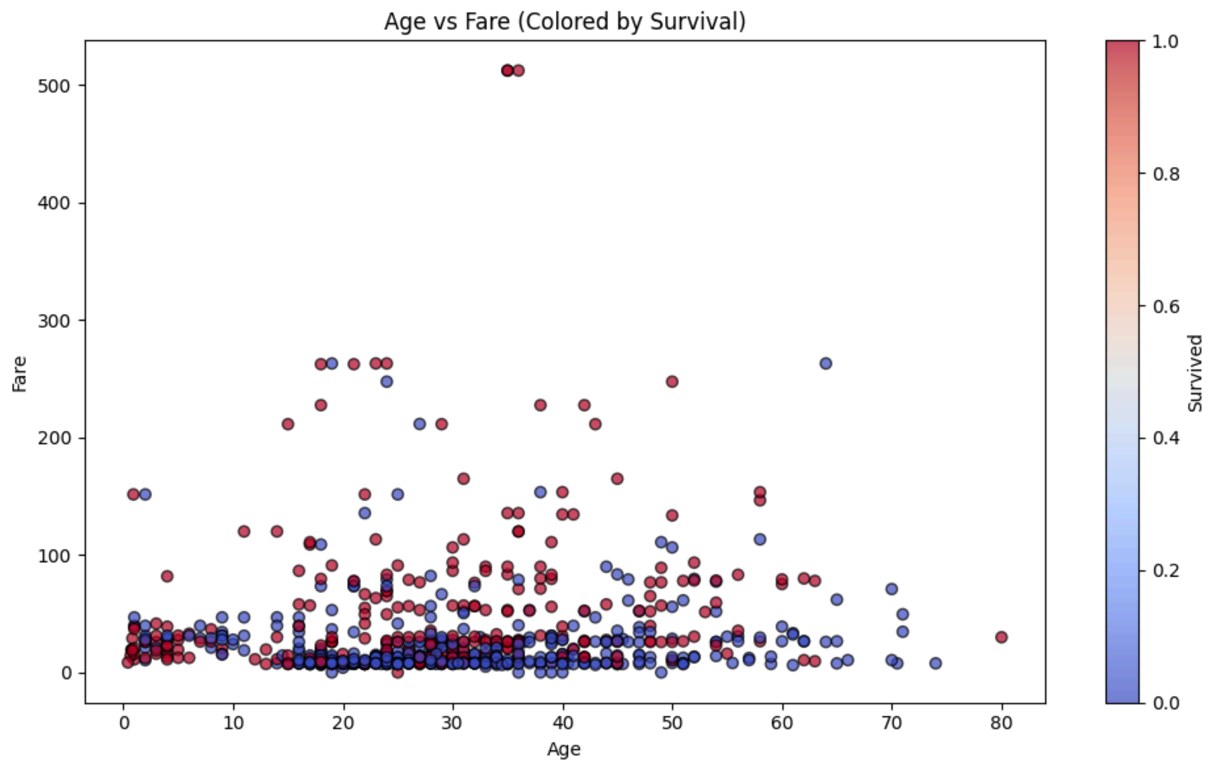
10. Create a pie chart for the 'sex' column in the Titanic dataset. Set the colors to ['lightblue', 'pink'] and add a title "Gender Distribution".

```
In [18]: #Code here:
gender_counts = df['sex'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(gender_counts, labels=gender_counts.index, colors=['lightblue', 'pink'], au
plt.title('Gender Distribution')
plt.tight_layout()
plt.show()
```



11. Using the Titanic dataset, plot a scatter plot of 'age' vs 'fare' and color the points by 'survived'. Set the colormap to 'coolwarm'.

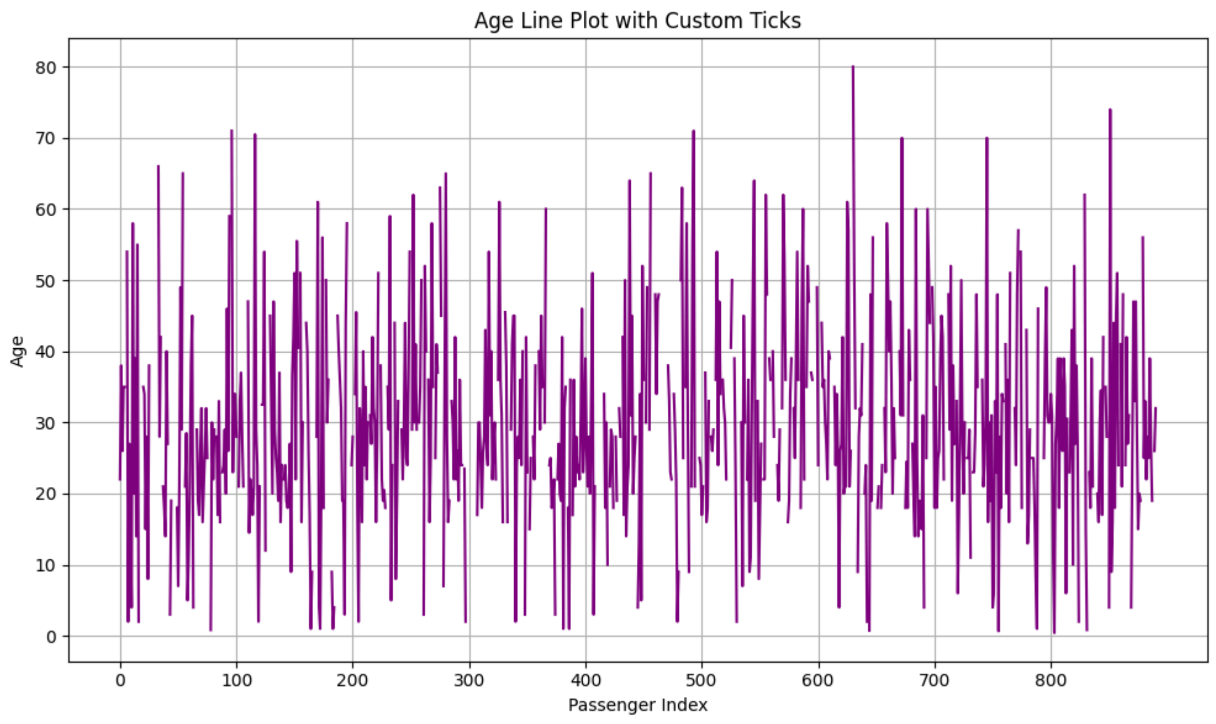
In [19]:
```
#Code here:
plt.figure(figsize=(10, 6))
plt.scatter(df['age'], df['fare'], c=df['survived'], cmap='coolwarm', edgecolor='k'
plt.title('Age vs Fare (Colored by Survival)')
plt.xlabel('Age')
plt.ylabel('Fare')
plt.colorbar(label='Survived')
plt.tight_layout()
plt.show()
```



---

12. Using the Titanic dataset, create a line plot of the 'age' column and customize the x-axis and y-axis ticks to show every 100 passengers and every 10 years of age, respectively.

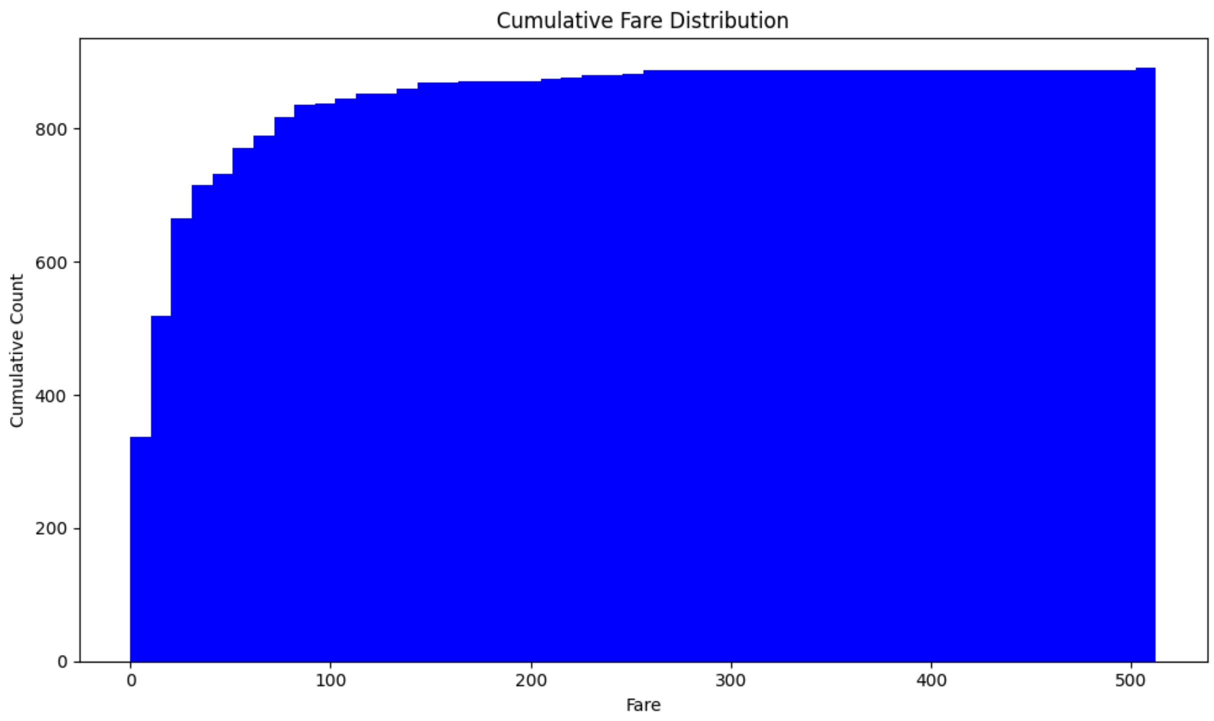
In [20]:
```
#Code here:
plt.figure(figsize=(10, 6))
plt.plot(df['age'], color='purple')
plt.xticks(ticks=range(0, len(df), 100))
plt.yticks(ticks=range(0, 90, 10))
plt.title('Age Line Plot with Custom Ticks')
plt.xlabel('Passenger Index')
plt.ylabel('Age')
plt.grid(True)
plt.tight_layout()
plt.show()
```



---

13. Create a cumulative histogram of the 'fare' column from the Titanic dataset with 50 bins and a blue color.

In [21]:
```
#Code here:
plt.figure(figsize=(10, 6))
plt.hist(df['fare'], bins=50, cumulative=True, color='blue')
plt.title('Cumulative Fare Distribution')
plt.xlabel('Fare')
plt.ylabel('Cumulative Count')
plt.tight_layout()
plt.show()
```

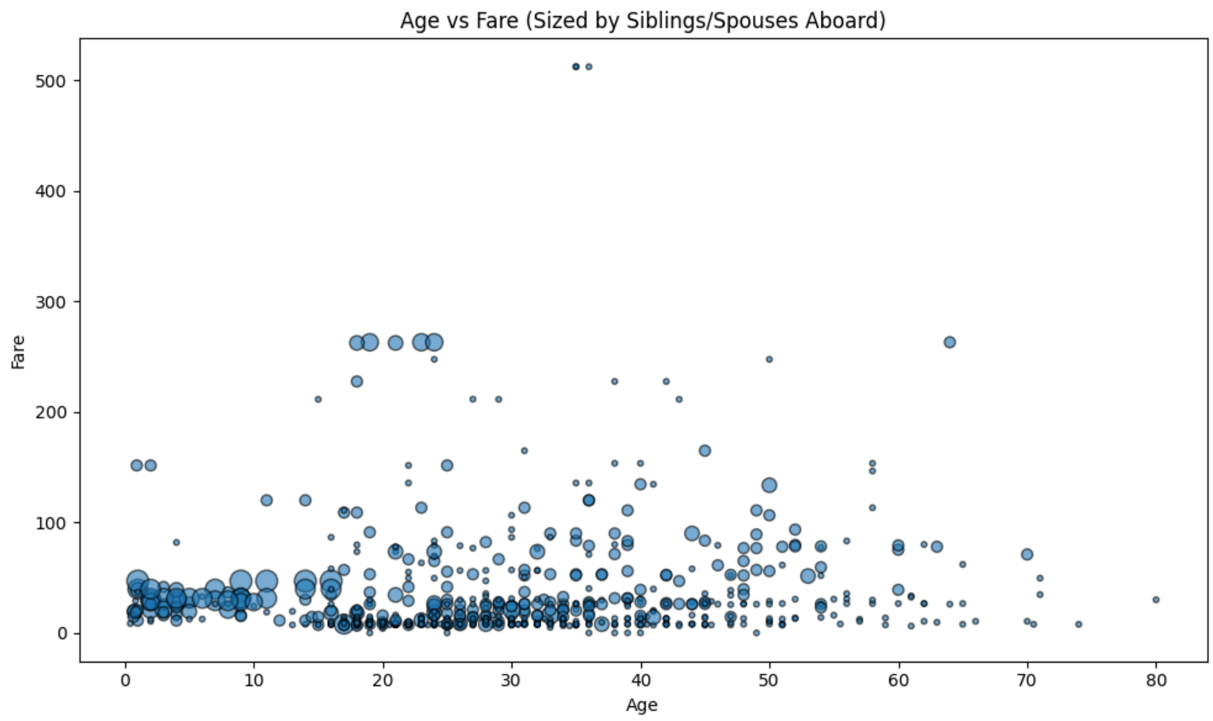


---

14. Using the Titanic dataset plot a customized scatter plot of 'age' vs 'fare' with the size of the points determined by 'sibsp' (number of siblings/spouses aboard).

In [22]:

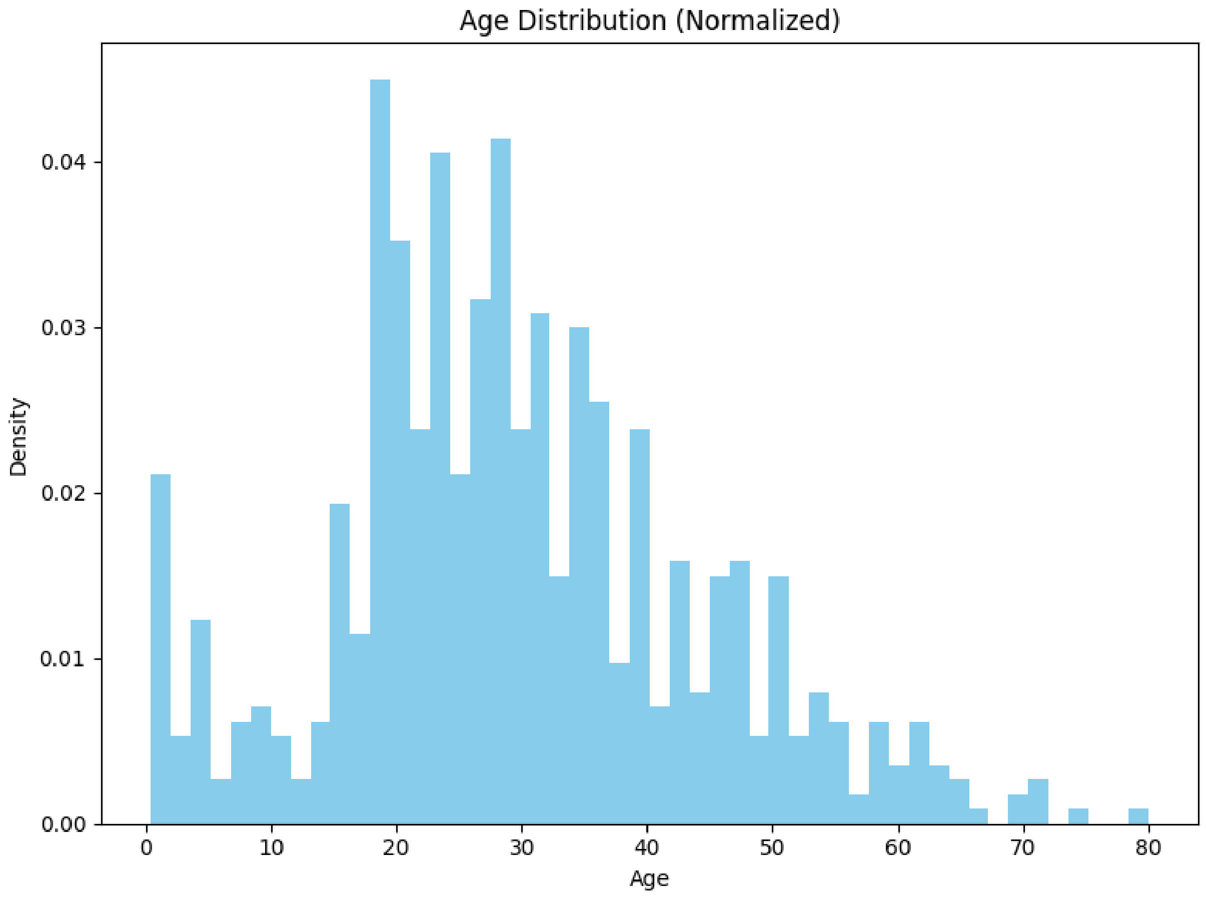
```
#Code here:
plt.figure(figsize=(10, 6))
plt.scatter(df['age'], df['fare'], s=df['sibsp']*30 + 10, alpha=0.6, edgecolor='k')
plt.title('Age vs Fare (Sized by Siblings/Spouses Aboard)')
plt.xlabel('Age')
plt.ylabel('Fare')
plt.tight_layout()
plt.show()
```



---

15. Create a histogram of the 'age' column from the Titanic dataset, setting the y-axis label to 'Density', and enabling density normalization.

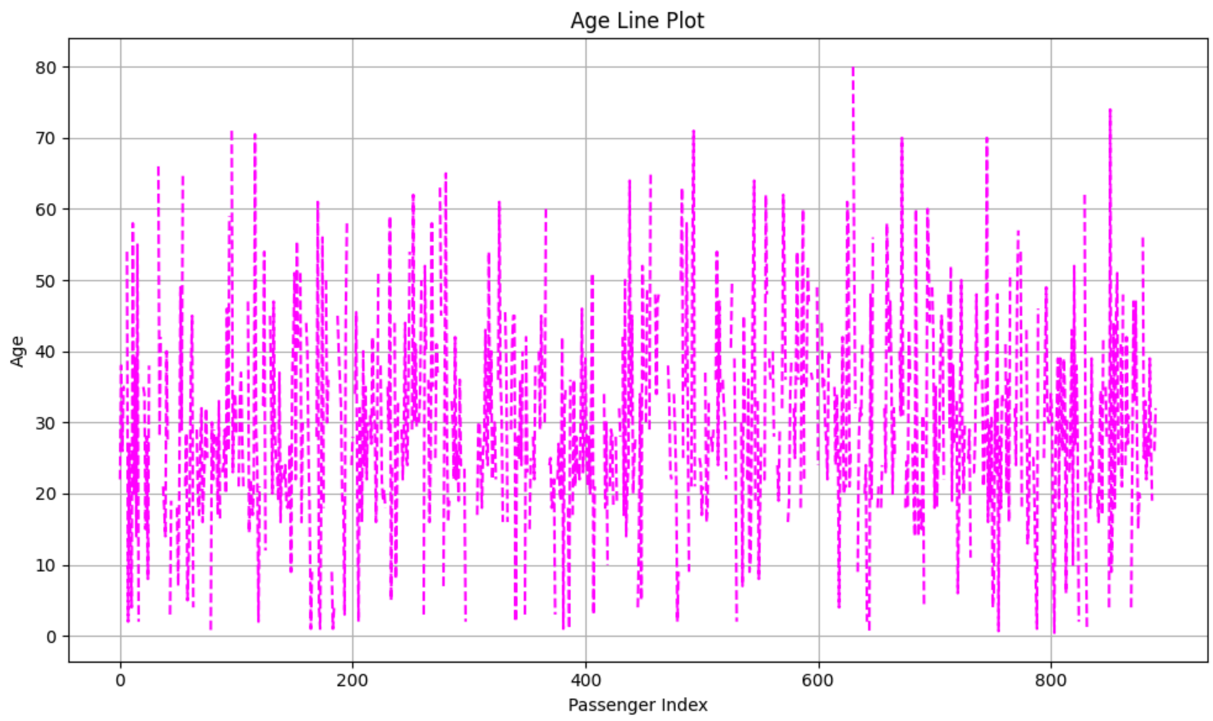
In [23]:
```
#Code here:
plt.figure(figsize=(8, 6))
plt.hist(df['age'].dropna(), bins=50, color='skyblue', density=True)
plt.title('Age Distribution (Normalized)')
plt.xlabel('Age')
plt.ylabel('Density')
plt.tight_layout()
plt.show()
```



---

16. Using the Titanic dataset and create a line plot of the 'age' column. Customize the plot with a dashed line style, magenta color, and add a grid.

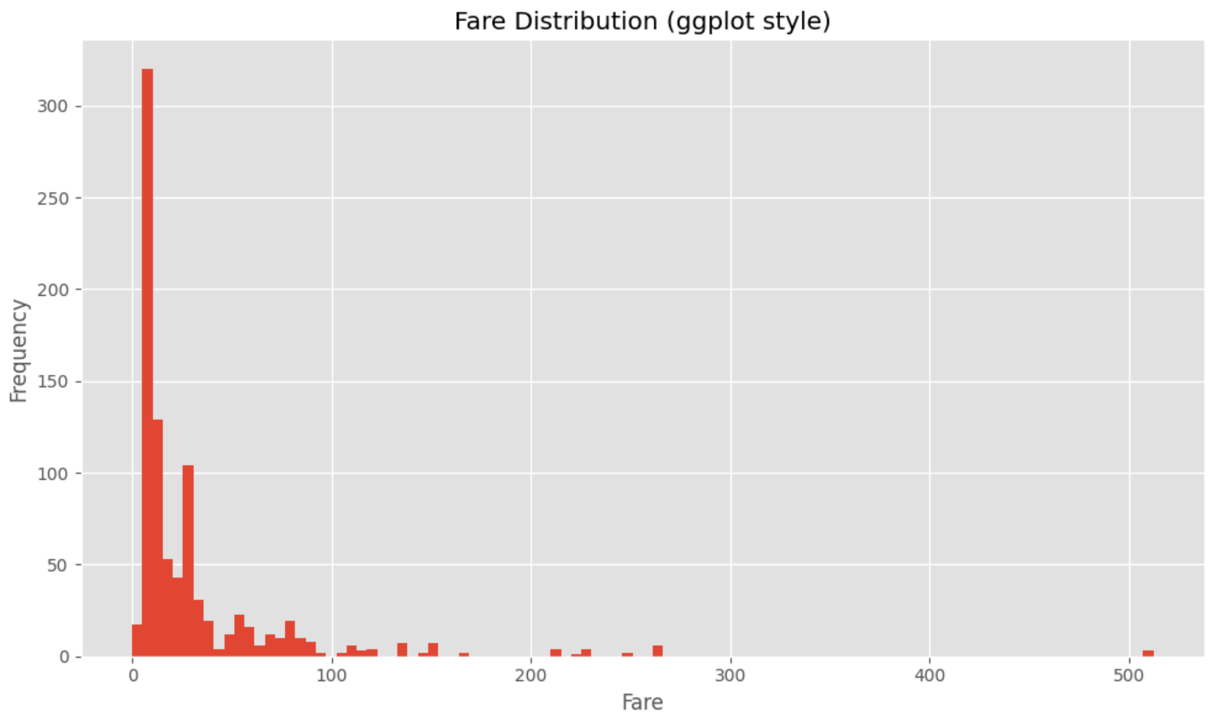
In [24]:
```
#Code here:
plt.figure(figsize=(10, 6))
plt.plot(df['age'], linestyle='--', color='magenta')
plt.title('Age Line Plot')
plt.xlabel('Passenger Index')
plt.ylabel('Age')
plt.grid(True)
plt.tight_layout()
plt.show()
```



---

17. Using the Titanic dataset, plot a histogram of the 'fare' column with 100 bins. Customize the plot to use the 'ggplot' style.

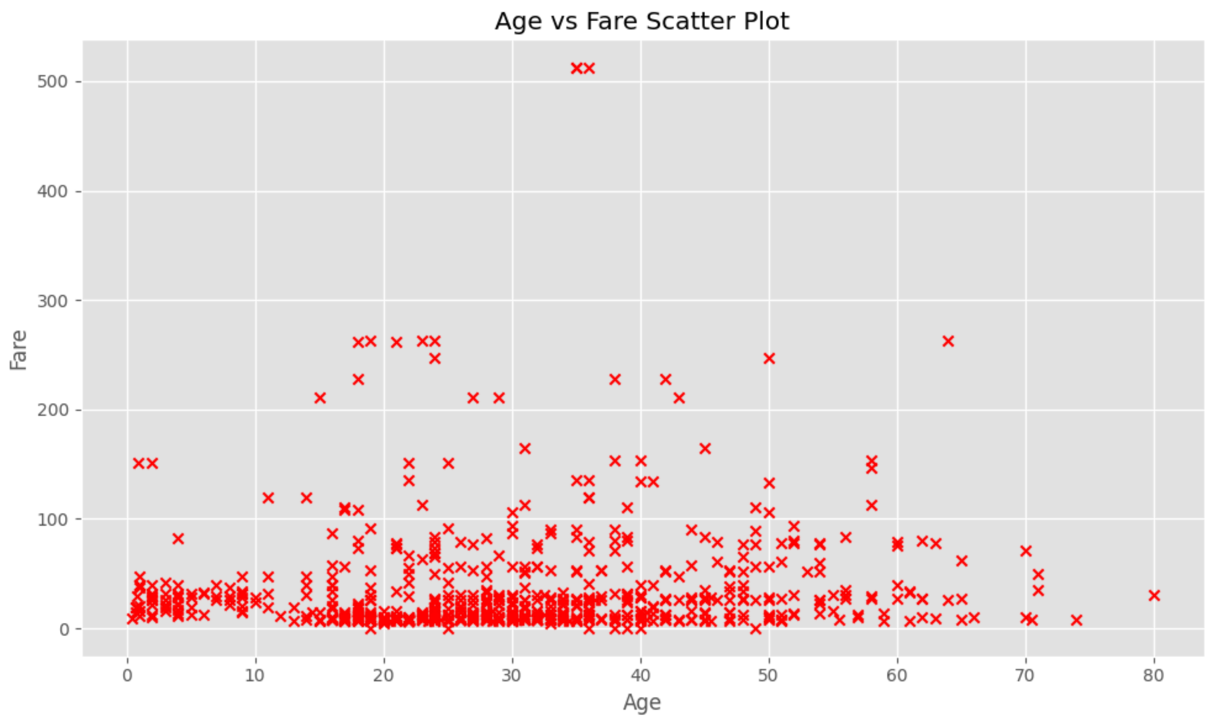
In [25]:
```
#Code here:
plt.style.use('ggplot')
plt.figure(figsize=(10, 6))
plt.hist(df['fare'], bins=100)
plt.title('Fare Distribution (ggplot style)')
plt.xlabel('Fare')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



18. Create a scatter plot from the Titanic dataset with 'age' on the x-axis and 'fare' on the y-axis. Customize the plot with a red cross ('x') marker and set the plot title to "Age vs Fare Scatter Plot".

In [26]:

```python
#Code here:
plt.figure(figsize=(10, 6))
plt.scatter(df['age'], df['fare'], marker='x', color='red')
plt.title('Age vs Fare Scatter Plot')
plt.xlabel('Age')
plt.ylabel('Fare')
plt.tight_layout()
plt.show()
```



---

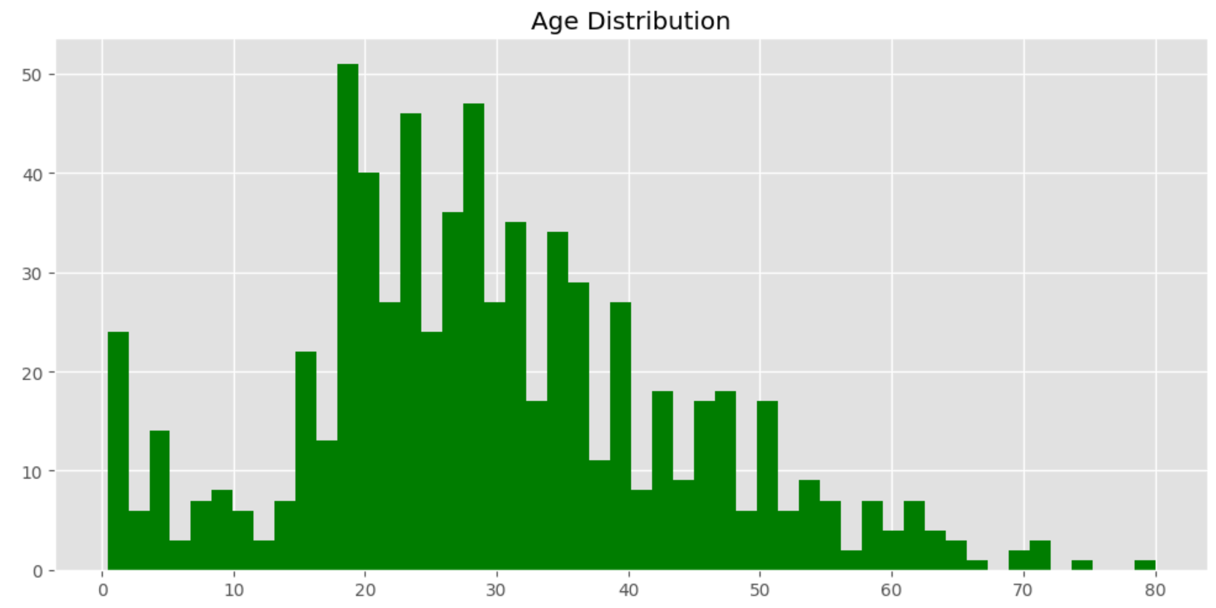
19. Using the Titanic dataset create a figure with two subplots: one subplot showing a histogram of the 'age' column with 50 bins and the second subplot showing a bar chart of the 'pclass' column. Ensure that both plots share the same x-axis.

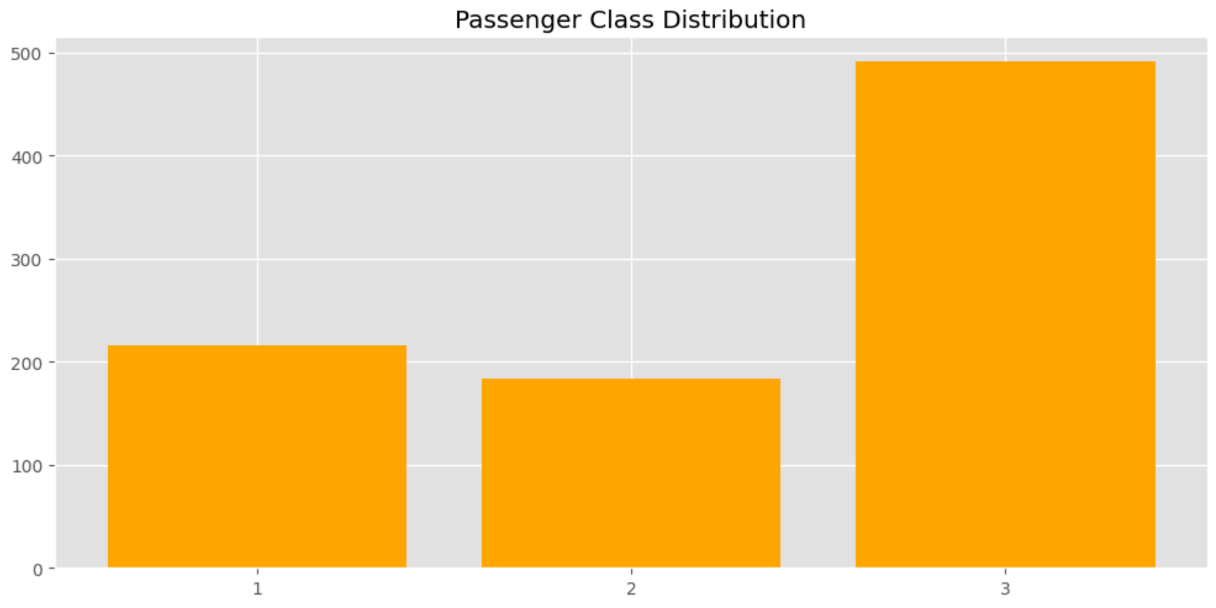
In [27]:
```
#Code here:
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10), sharex=False)

ax1.hist(df['age'].dropna(), bins=50, color='green')
ax1.set_title('Age Distribution')

pclass_counts = df['pclass'].value_counts().sort_index()
ax2.bar(pclass_counts.index.astype(str), pclass_counts.values, color='orange')
ax2.set_title('Passenger Class Distribution')

plt.tight_layout()
plt.show()
```





---

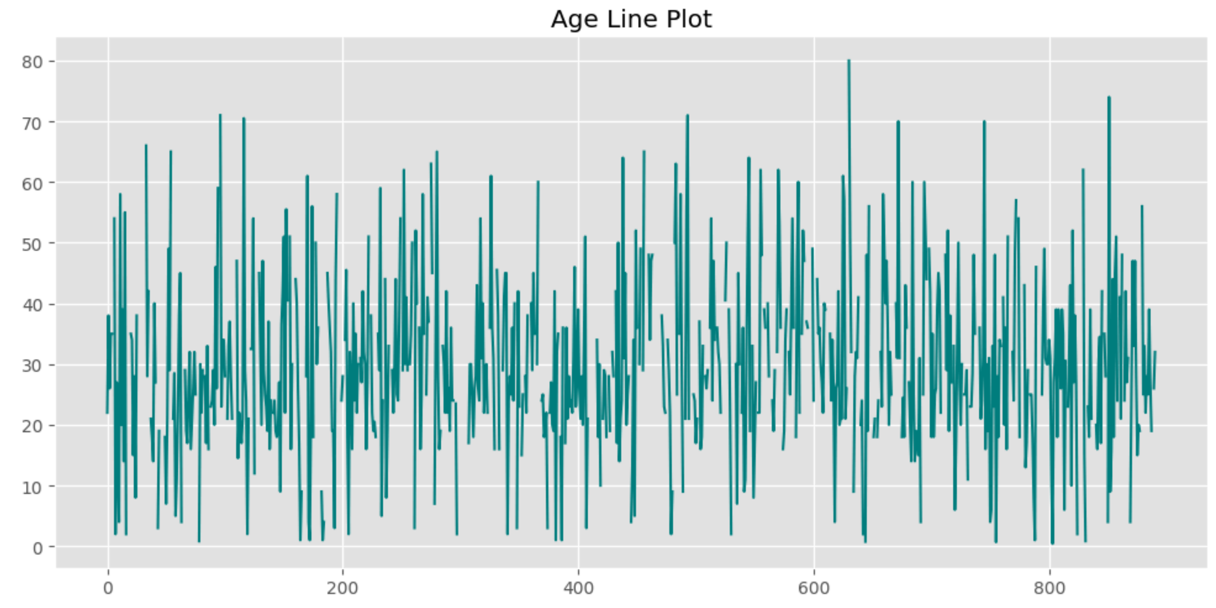
20. Using the Titanic dataset create a figure with two subplots: the first subplot should display a line plot of the 'age' column, and the second subplot should display a bar chart of the 'sex' column showing the count of each gender.

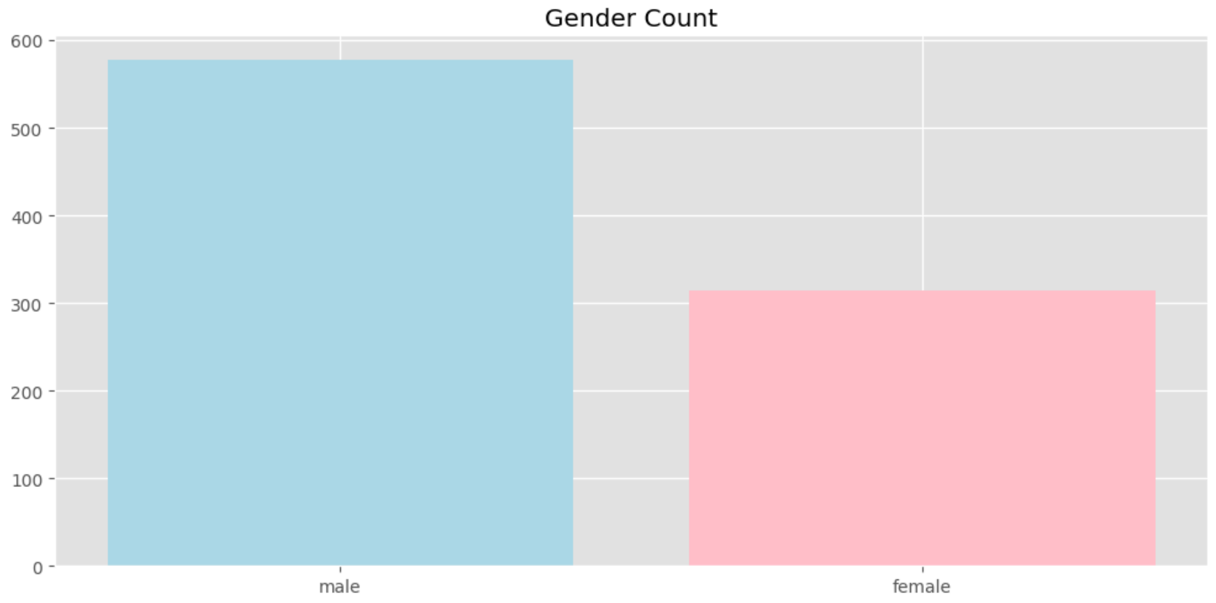
In [28]:
```
#Code here:
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10))

ax1.plot(df['age'], color='teal')
ax1.set_title('Age Line Plot')

gender_counts = df['sex'].value_counts()
ax2.bar(gender_counts.index, gender_counts.values, color=['lightblue', 'pink'])
ax2.set_title('Gender Count')
```

```
plt.tight_layout()
plt.show()
```





Good Luck!