ITT211 Assignment

Hi there! we are group 4, and this is our journey through the fascinating world of data analysis. Buckle up, because we're about to dive into some cool statistics and Python code!

Task 1: Poisson Distribution Analysis

Alright, let's kick things off with the Poisson distribution. Imagine you're in a coffee shop, trying to figure out how many customers will pop in every hour. The Poisson distribution is like your crystal ball for predicting these kinds of events. Let's see how it works!

Poisson Distribution: The Lowdown

The Poisson distribution is your go-to tool when you're dealing with rare events happening over a fixed period. Think of it as predicting how many times you'll spill your coffee in a day (hopefully not too many!). It's super handy when you're looking at things like website traffic, customer arrivals, or even natural disasters—anything where you're counting occurrences over time.

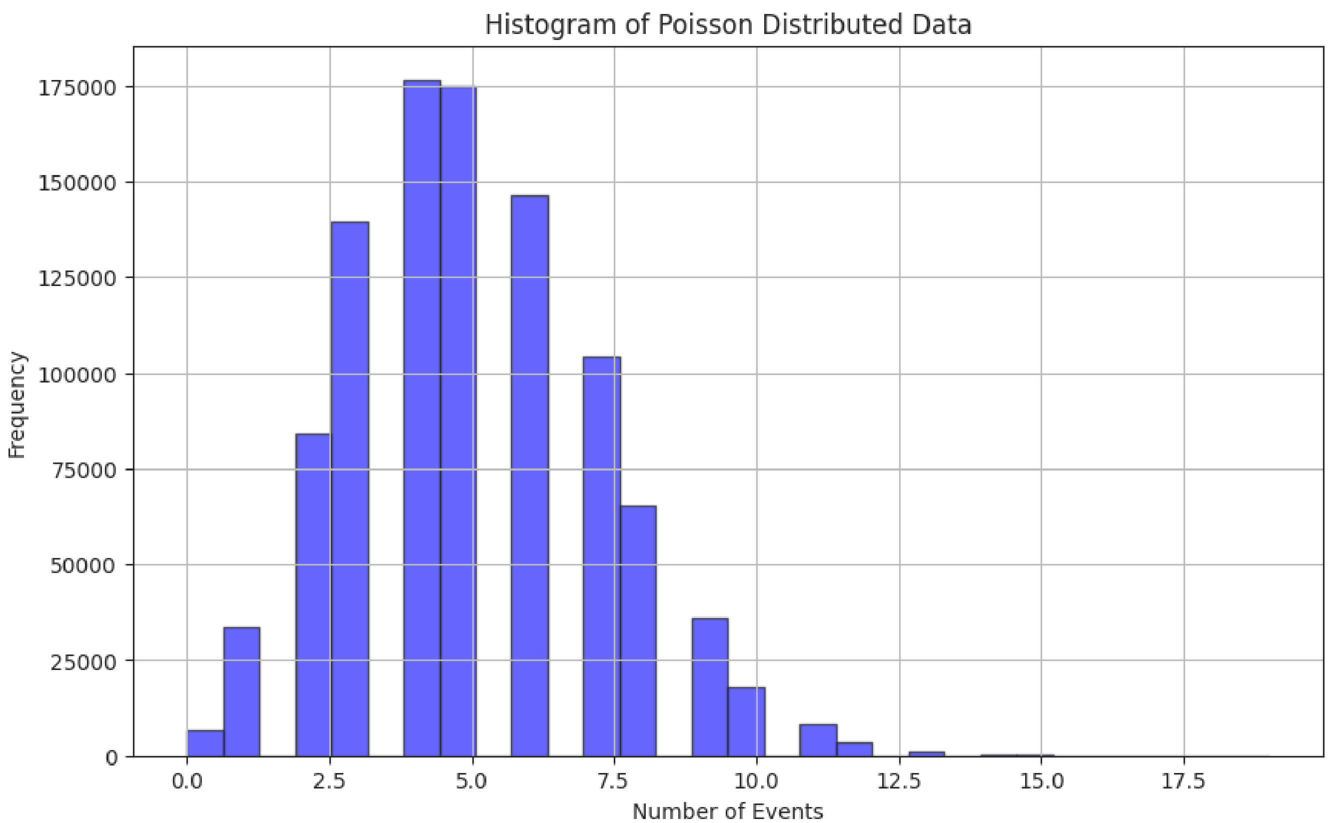
Let's Get Coding!

Okay, let's dive into the code. We're going to generate 1 million random numbers that follow a Poisson distribution with an average (lambda) of 5. Picture this as simulating a busy day in your inbox, where you receive around 5 emails per hour, but with some randomness thrown in because life is unpredictable!

```
import numpy as np  # Let's bring in numpy – our go-to library for crunching numbers. It's l

# Now, let's set our scene: Imagine we're modeling the number of emails you get per hour.
# We'll assume, on average, you receive 5 emails an hour. That's our 'lambda' for this Poiss
lambda_poisson = 5

# Generate the random data – here comes the flood of emails!
data = np.random.poisson(lam=lambda_poisson, size=1000000)

# Let's take a quick look at the first few data points. It's always good to check what we're
data[:10]  # Just a sneak peek at the first 10 emails/hour counts.
```

```
array([2, 2, 5, 1, 6, 6, 6, 9, 4, 7])
```

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Set the mean of the Poisson distribution
lambda_poisson = 5

# Generate 1 million random integers following a Poisson distribution
poisson_data = np.random.poisson(lambda_poisson, 1000000)

# Plot the histogram of the generated data
plt.figure(figsize=(10, 6))  # Create a figure for the plot with specified size.
plt.hist(poisson_data, bins=30, color='blue', edgecolor='black', alpha=0.7)  # Create a hist
plt.title('Histogram of Poisson Distributed Data')  # Set the title of the plot.
plt.xlabel('Number of Events')  # Label the x-axis.
plt.ylabel('Frequency')  # Label the y-axis.
plt.grid(True)  # Add a grid to the plot for better readability.
plt.show()  # Display the plot.
```

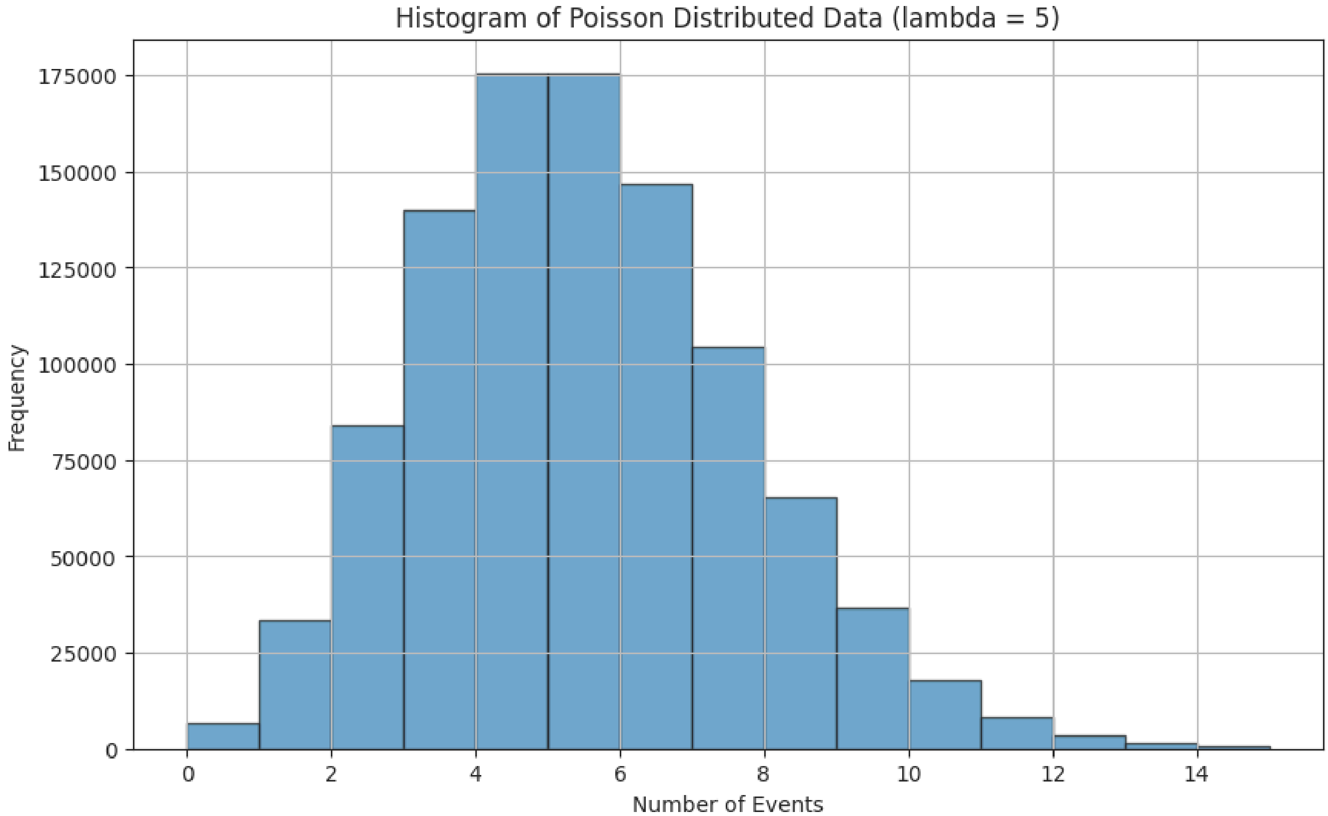


```
import numpy as np  # Import the numpy library, which is used for numerical operations.

# Set the mean of the Poisson distribution
lambda_poisson = 5  # The average number of events (e.g., emails per hour)

# Generate 1 million random integers following a Poisson distribution
poisson_data = np.random.poisson(lambda_poisson, 1000000)

# Print the generated Poisson-distributed data
print(poisson_data)  # Print the array of 1 million random integers
```

```
[5 3 2 ... 7 4 4]
```

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Set the mean of the Poisson distribution
lambda_poisson = 5  # The average number of events (e.g., emails per hour)

# Generate 1 million random integers following a Poisson distribution
poisson_data = np.random.poisson(lambda_poisson, 1000000)

# Print a portion of the generated Poisson-distributed data
print(poisson_data[:100])  # Print the first 100 values of the array of 1 million random int

# Plot the histogram
plt.figure(figsize=(10, 6))  # Create a figure for the plot with specified size.
plt.hist(poisson_data, bins=range(0, 16), alpha=0.75, edgecolor='black')  # Create a histogr
plt.title('Histogram of Poisson Distributed Data (lambda = 5)')  # Set the title of the plot
plt.xlabel('Number of Events')  # Label the x-axis.
plt.ylabel('Frequency')  # Label the y-axis.
plt.grid(True)  # Add a grid to the plot for better readability.
plt.show()  # Display the plot.
```

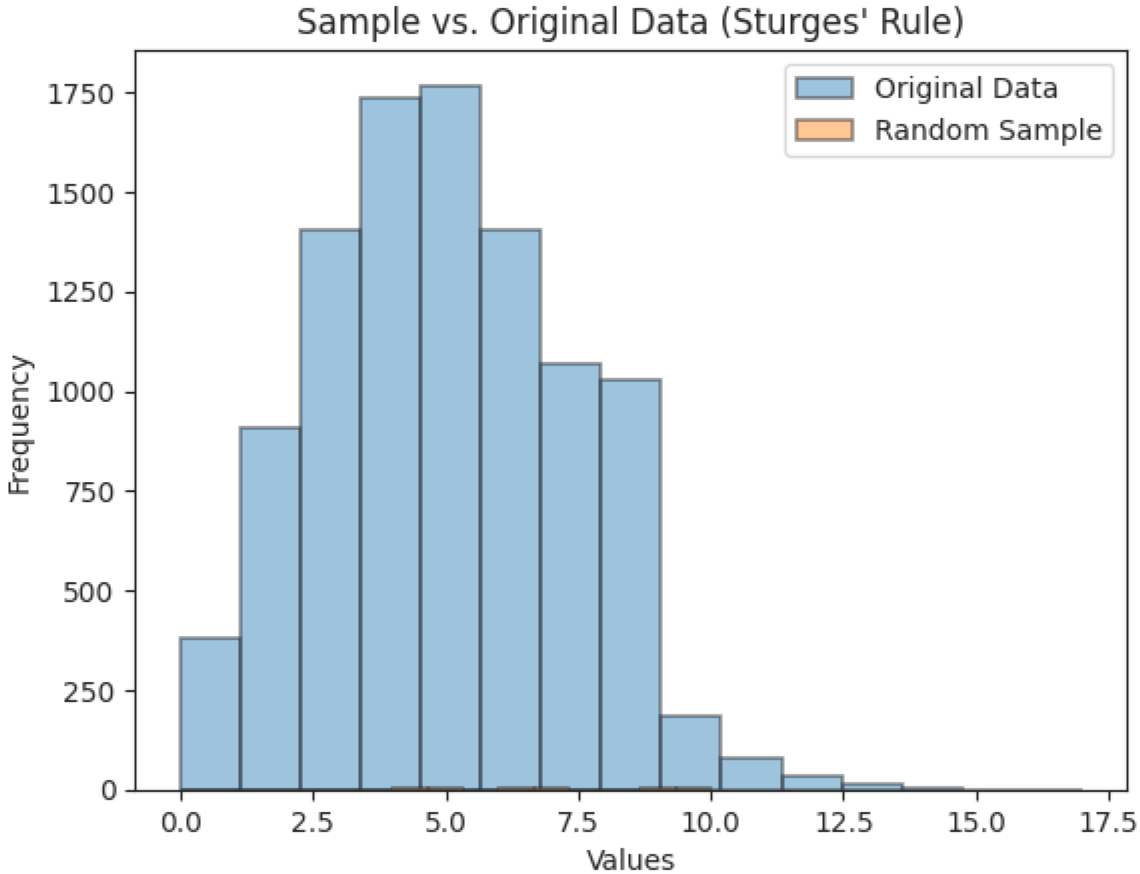
```
[ 5  4  2  7  5  3  5  4  5  3  5  4  2  6  6  5  6  4  6  4  7  4  6  8
  3  8  7  9  7  5  1  6  4  5 10  5  3  5  3  3  5  6  3  5  3  5  6 10
  3  5  5  4  4  8  4  0  3  6  6  3  6  4  4  3  6  6  5  2  2  5  6  6
  5  4  5  5  3  5  2  6 10  9  2  2  2  7  6  3  9  4  6  8  8  7  8  5
  4  4  2  3]
```



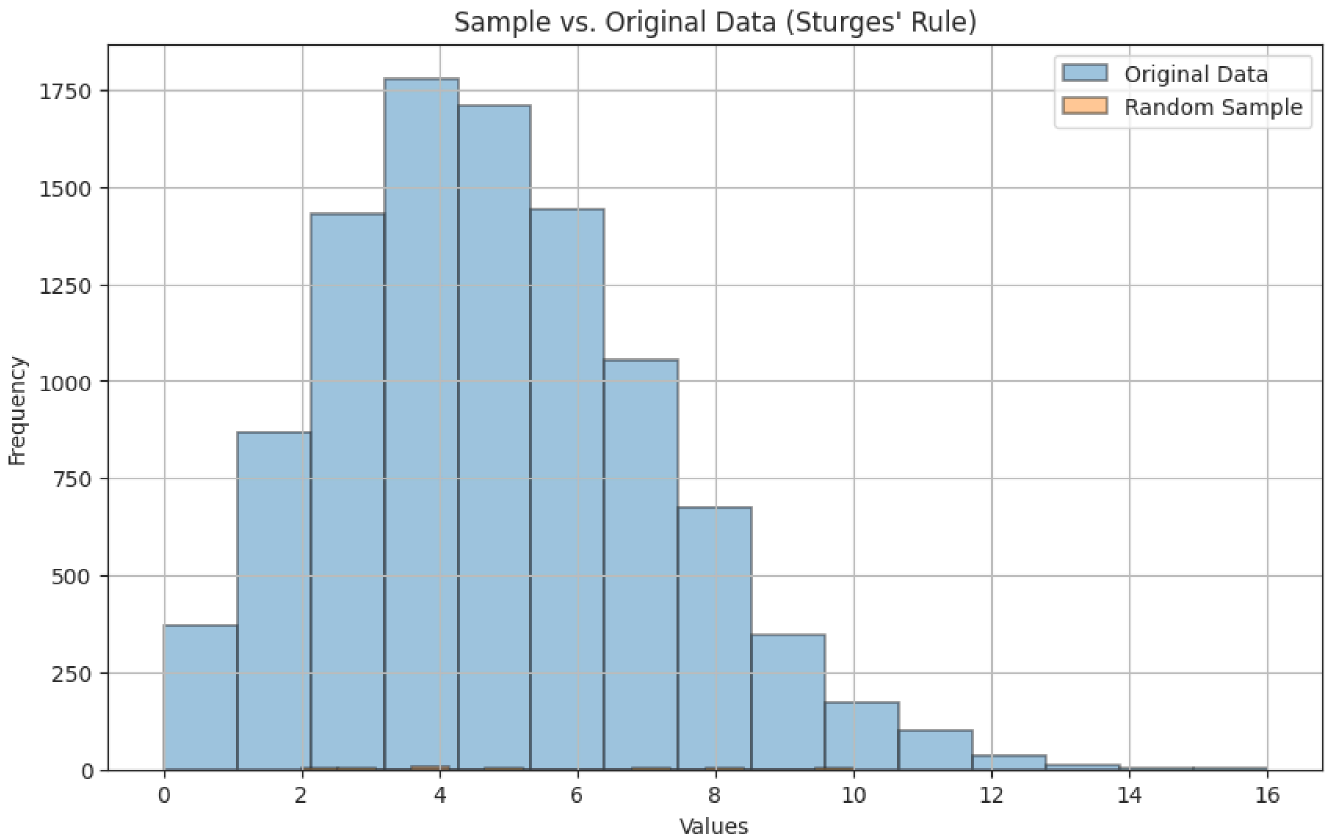
```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Set the mean of the Poisson distribution
lambda_poisson = 5  # The average number of events (e.g., emails per hour)

# Generate 10,000 random integers following a Poisson distribution
poisson_data = np.random.poisson(lambda_poisson, 10000)

# Perform random sampling
sample_size = 20  # Set the sample size to 20
random_sample = np.random.choice(poisson_data, size=sample_size, replace=False)
# Randomly select 20 values from the generated Poisson-distributed data without replacement

# Print the random sample
print("Random Sample:", random_sample)
```

```
Random Sample: [ 2  0  9  7  4  9  4  6  4  6  3  5  5  8 10  6  5  6 10  7]
```

```
# Plot histograms using Sturges' rule
sturges_bins = int(np.ceil(np.log2(10000) + 1))
plt.hist(poisson_data, bins=sturges_bins, alpha=0.5, label='Original Data', edgecolor='black
plt.hist(random_sample, bins=sturges_bins, alpha=0.5, label='Random Sample', edgecolor='blac
plt.legend()
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Sample vs. Original Data (Sturges\' Rule)')
plt.show()
```


Sample vs. Original Data (Sturges' Rule)
Original Data
Random Sample
Frequency
0
250
500
750
1000
1250
1500
1750
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Values

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Set the mean of the Poisson distribution
lambda_poisson = 5  # The average number of events (e.g., emails per hour)

# Generate 10,000 random integers following a Poisson distribution
poisson_data = np.random.poisson(lambda_poisson, 10000)

# Perform random sampling
sample_size = 20  # Set the sample size to 20
random_sample = np.random.choice(poisson_data, size=sample_size, replace=False)
# Randomly select 20 values from the generated Poisson-distributed data without replacement

# Calculate the number of bins using Sturges' rule
sturges_bins = int(np.ceil(np.log2(10000) + 1))
# Sturges' rule: Number of bins = log2(n) + 1, where n is the number of data points

# Plot histograms using Sturges' rule
plt.figure(figsize=(10, 6))  # Create a figure for the plot with specified size.
plt.hist(poisson_data, bins=sturges_bins, alpha=0.5, label='Original Data', edgecolor='black
# Plot the histogram of the original data with calculated number of bins
plt.hist(random_sample, bins=sturges_bins, alpha=0.5, label='Random Sample', edgecolor='blac
# Plot the histogram of the random sample with calculated number of bins

# Add legend, labels, and title
plt.legend()  # Add a legend to the plot.
plt.xlabel('Values')  # Label the x-axis.
plt.ylabel('Frequency')  # Label the y-axis.
plt.title('Sample vs. Original Data (Sturges\' Rule)')  # Set the title of the plot.
plt.grid(True)  # Add a grid to the plot for better readability.
plt.show()  # Display the plot.
```




```
print("Data generation complete.")
```

```
Data generation complete.
```

2. **Statistical Analysis**

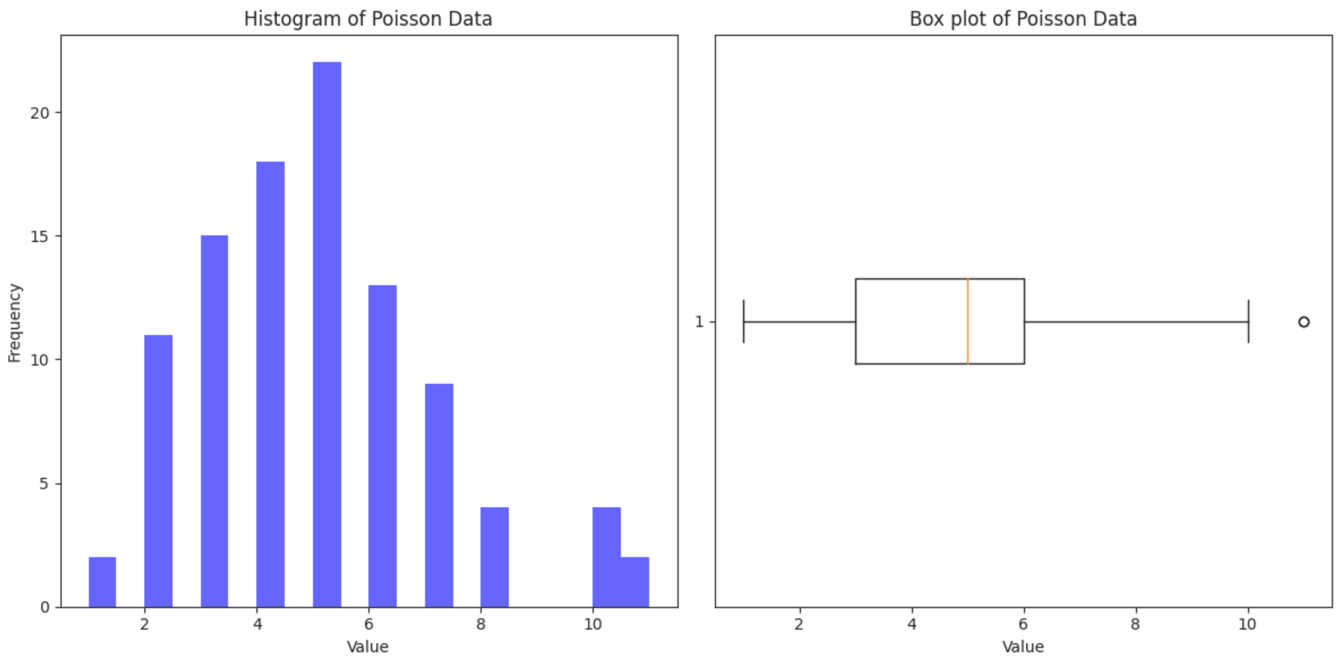
We will calculate the mean, variance, and standard deviation of the

generated data to understand the distribution's characteristics.

```
# Calculate the mean of the Poisson data
mean_poisson = np.mean(poisson_data)



print(mean_poisson)
```

```
5.0253
```

```
# Calculate the variance of the Poisson data
variance_poisson = np.var(poisson_data)


# Import the necessary library
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import numpy as np  # Import the numpy library, which is used for numerical operations.

# Generate some sample Poisson data (replace this with your actual data)
poisson_data = np.random.poisson(lam=5, size=100)
# Generate 100 random integers from a Poisson distribution with a mean (lambda) of 5.

# Plotting the data
plt.figure(figsize=(12, 6))  # Create a figure for the plot with specified size (12 inches b

# Histogram of the Poisson data
plt.subplot(1, 2, 1)  # Create a subplot (1 row, 2 columns, this is the 1st subplot).
plt.hist(poisson_data, bins=20, color='blue', alpha=0.7)  # Create a histogram with 20 bins
plt.title('Histogram of Poisson Data')  # Set the title of the histogram.
plt.xlabel('Value')  # Label the x-axis of the histogram.
plt.ylabel('Frequency')  # Label the y-axis of the histogram.

# Box plot of the Poisson data
plt.subplot(1, 2, 2)  # Create a subplot (1 row, 2 columns, this is the 2nd subplot).
plt.boxplot(poisson_data, vert=False)  # Create a horizontal box plot of the Poisson data.
plt.title('Box plot of Poisson Data')  # Set the title of the box plot.
plt.xlabel('Value')  # Label the x-axis of the box plot.

plt.tight_layout()  # Adjust the subplots to fit into the figure area.
plt.show()  # Display the plots.
```




```
print(variance_poisson) #checking code values are sligthly off but still
```

```
4.958259910000001
```

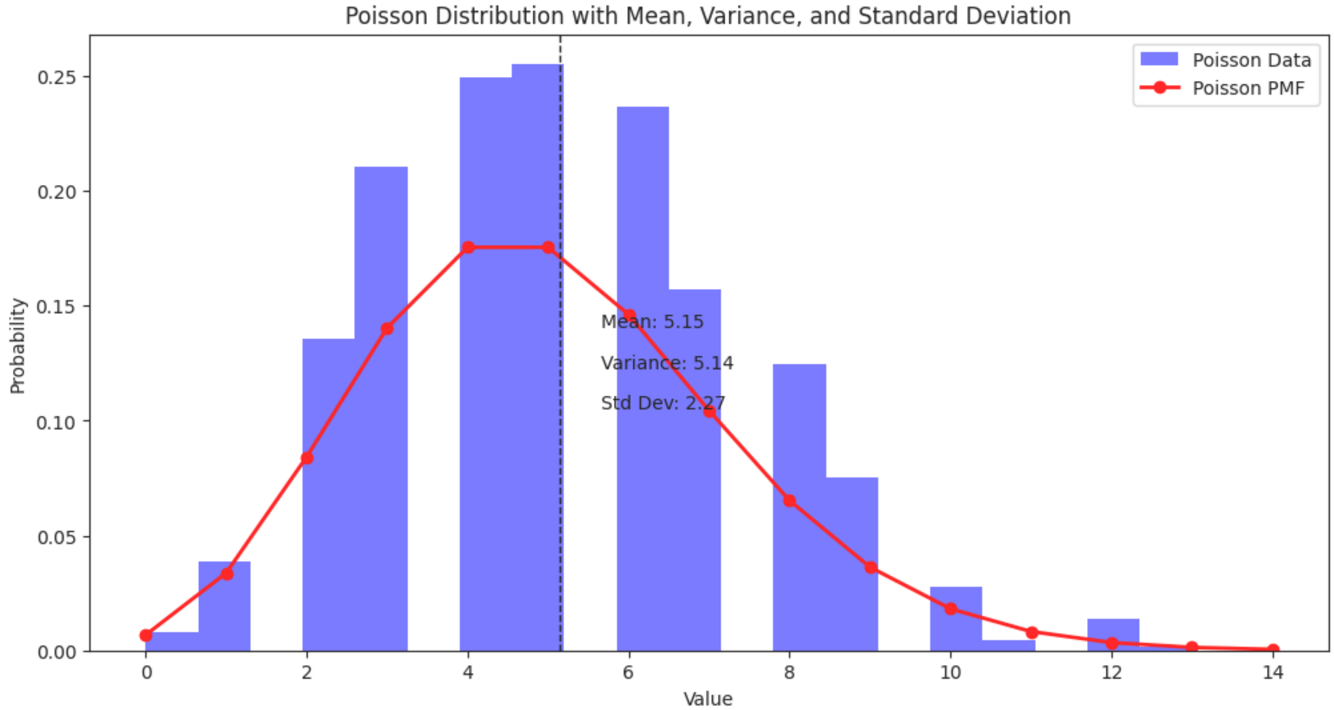
```
# Calculate the standard deviation of the Poisson data
std_dev_poisson = np.std(poisson_data)
```

```
print(std_dev_poisson)
```

```
2.154158768521949
```

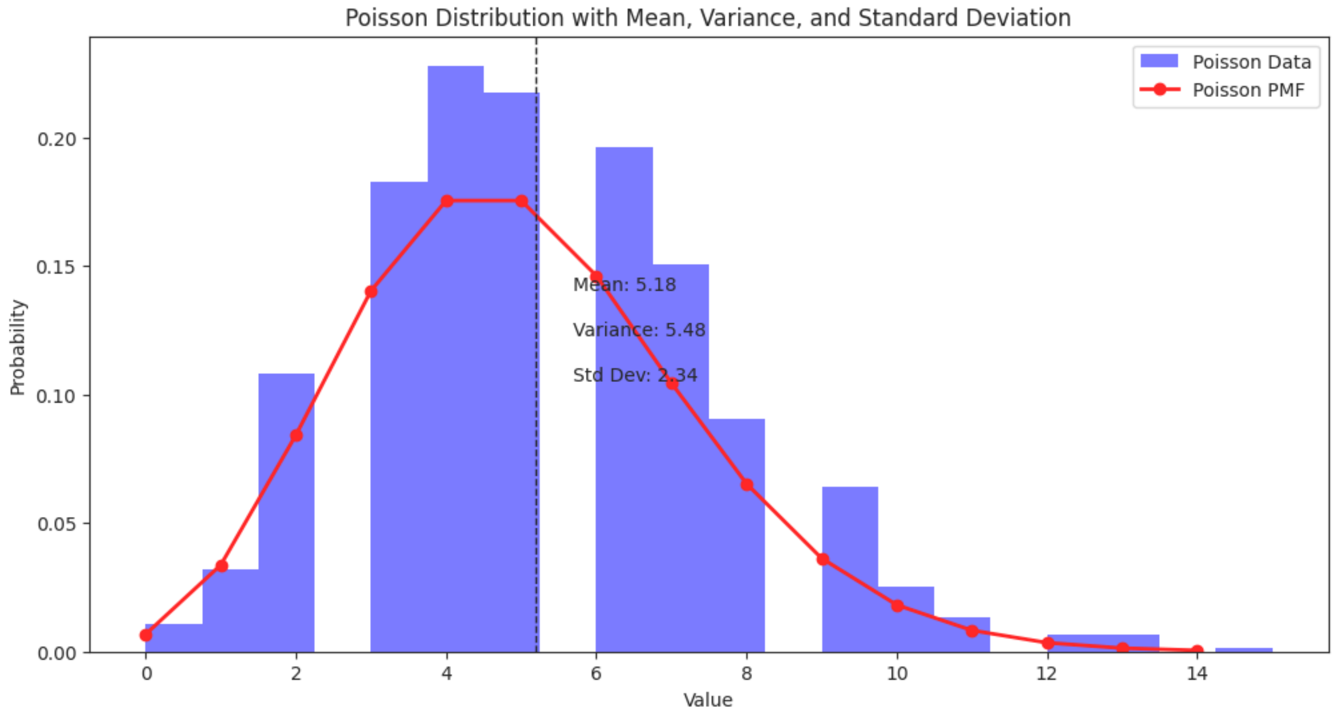
```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
from scipy.stats import poisson  # Import the poisson function from scipy.stats for Poisson

# Generate Poisson data
lambda_value = 5  # Set the mean (lambda) of the Poisson distribution.
data = np.random.poisson(lam=lambda_value, size=1000)  # Generate 1000 random integers from

# Calculate mean, variance, and standard deviation
mean_poisson = np.mean(data)  # Calculate the mean of the generated Poisson data.
variance_poisson = np.var(data)  # Calculate the variance of the generated Poisson data.
std_dev_poisson = np.std(data)  # Calculate the standard deviation of the generated Poisson

# Print the calculated statistics
print(f"Mean: {mean_poisson}, Variance: {variance_poisson}, Standard Deviation: {std_dev_poi

# Create the histogram
plt.figure(figsize=(12, 6))  # Create a figure for the plot with specified size (12 inches b

# Histogram of the Poisson data
plt.hist(data, bins=20, density=True, alpha=0.6, color='b', label='Poisson Data')  # Create

# Plot the Poisson probability mass function (PMF) for comparison
x = np.arange(0, 15)  # Generate an array of values from 0 to 14.
pmf = poisson.pmf(x, lambda_value)  # Calculate the Poisson PMF for these values using the s
plt.plot(x, pmf, 'ro-', label='Poisson PMF', lw=2)  # Plot the PMF as red circles connected

# Add labels, legend, and title
plt.xlabel('Value')  # Label the x-axis.
plt.ylabel('Probability')  # Label the y-axis.
plt.legend()  # Add a legend to the plot.
plt.title('Poisson Distribution with Mean, Variance, and Standard Deviation')  # Set the tit

# Annotate the mean, variance, and standard deviation
plt.axvline(mean_poisson, color='k', linestyle='dashed', linewidth=1)  # Draw a dashed verti
plt.text(mean_poisson + 0.5, max(pmf) * 0.8, f'Mean: {mean_poisson:.2f}', color='k')  # Annc
plt.text(mean_poisson + 0.5, max(pmf) * 0.7, f'Variance: {variance_poisson:.2f}', color='k')
plt.text(mean_poisson + 0.5, max(pmf) * 0.6, f'Std Dev: {std_dev_poisson:.2f}', color='k')

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

```
Mean: 5.151, Variance: 5.142199000000001, Standard Deviation: 2.2676417265520583
```



This line creates a histogram of the generated Poisson data with 20 bins, blue bars, and 60% opacity. The density=True parameter normalizes the histogram so that it represents a probability density function.

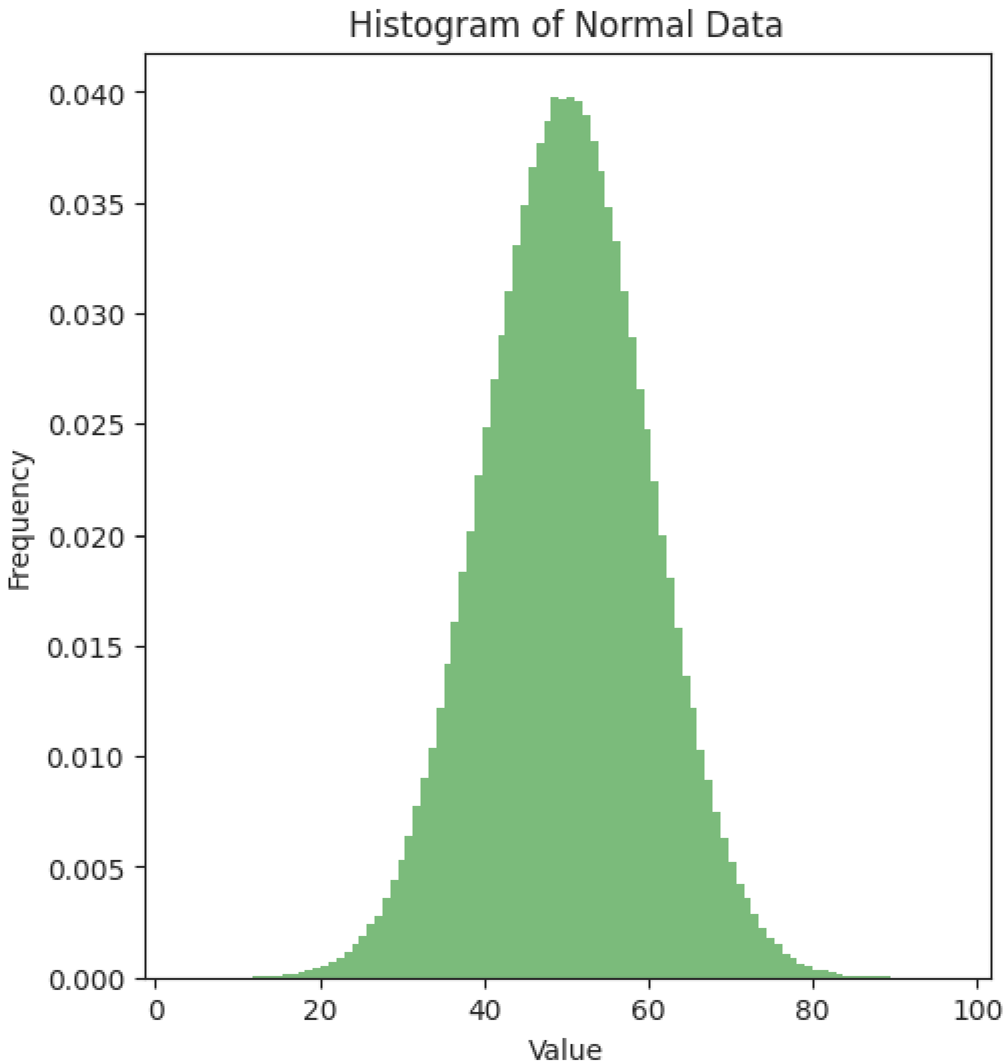
```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
from scipy.stats import poisson  # Import the poisson function from scipy.stats for Poisson

# Generate Poisson data
lambda_value = 5  # Set the mean (lambda) of the Poisson distribution.
data = np.random.poisson(lam=lambda_value, size=1000)  # Generate 1000 random integers from

# Calculate mean, variance, and standard deviation
mean_poisson = np.mean(data)  # Calculate the mean of the generated Poisson data.
variance_poisson = np.var(data)  # Calculate the variance of the generated Poisson data.
std_dev_poisson = np.std(data)  # Calculate the standard deviation of the generated Poisson

# Create the histogram
plt.figure(figsize=(12, 6))  # Create a figure for the plot with specified size (12 inches b

# Histogram of the Poisson data
plt.hist(data, bins=20, density=True, alpha=0.6, color='b', label='Poisson Data')  # Create

# Plot the Poisson probability mass function (PMF) for comparison
x = np.arange(0, 15)  # Generate an array of values from 0 to 14.
pmf = poisson.pmf(x, lambda_value)  # Calculate the Poisson PMF for these values using the s
plt.plot(x, pmf, 'ro-', label='Poisson PMF', lw=2)  # Plot the PMF as red circles connected

# Add labels, legend, and title
plt.xlabel('Value')  # Label the x-axis.
plt.ylabel('Probability')  # Label the y-axis.
plt.legend()  # Add a legend to the plot.
plt.title('Poisson Distribution with Mean, Variance, and Standard Deviation')  # Set the tit

# Annotate the mean, variance, and standard deviation
plt.axvline(mean_poisson, color='k', linestyle='dashed', linewidth=1)  # Draw a dashed verti
plt.text(mean_poisson + 0.5, max(pmf) * 0.8, f'Mean: {mean_poisson:.2f}', color='k')  # Annc
plt.text(mean_poisson + 0.5, max(pmf) * 0.7, f'Variance: {variance_poisson:.2f}', color='k')
plt.text(mean_poisson + 0.5, max(pmf) * 0.6, f'Std Dev: {std_dev_poisson:.2f}', color='k')

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




```
### Task 2: Normal Distribution Analysis
```

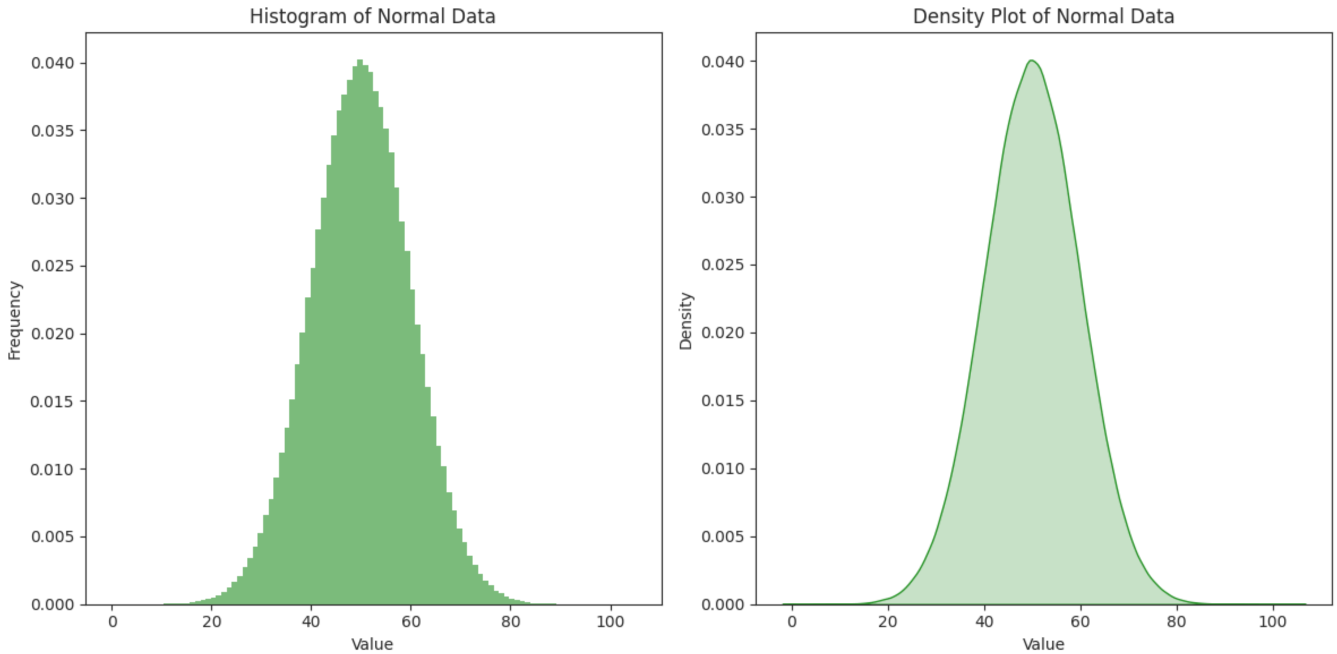
*1. Data Generation** We will generate 1 million random integers following a normal distribution with a mean (mu) of 50 and a standard deviation (sigma) of 10.

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Set the mean and standard deviation for the normal distribution
mu_normal = 50  # Set the mean (mu) of the normal distribution to 50.
sigma_normal = 10  # Set the standard deviation (sigma) of the normal distribution to 10.

# Generate 1 million random numbers following a normal distribution
normal_data = np.random.normal(mu_normal, sigma_normal, 1000000)
# Generate 1 million random numbers from a normal distribution with mean 50 and standard dev

# Plotting the data
plt.figure(figsize=(12, 6))  # Create a figure for the plot with specified size (12 inches b

# Histogram of the Normal data
plt.subplot(1, 2, 1)  # Create a subplot (1 row, 2 columns, this is the 1st subplot).
plt.hist(normal_data, bins=100, density=True, alpha=0.6, color='g')  # Create a histogram wi
plt.title('Histogram of Normal Data')  # Set the title of the histogram.
plt.xlabel('Value')  # Label the x-axis of the histogram.
plt.ylabel('Frequency')  # Label the y-axis of the histogram.

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


Histogram of Normal Data
Frequency
Value
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import seaborn as sns  # Import the seaborn library, which is used for statistical data visu

# Set the mean and standard deviation for the normal distribution
mu_normal = 50  # Set the mean (mu) of the normal distribution to 50.
sigma_normal = 10  # Set the standard deviation (sigma) of the normal distribution to 10.

# Generate 1 million random numbers following a normal distribution
normal_data = np.random.normal(mu_normal, sigma_normal, 1000000)
# Generate 1 million random numbers from a normal distribution with mean 50 and standard dev

# Plotting the data
plt.figure(figsize=(12, 6))  # Create a figure for the plot with specified size (12 inches b

# Histogram of the Normal data
plt.subplot(1, 2, 1)  # Create a subplot (1 row, 2 columns, this is the 1st subplot).
plt.hist(normal_data, bins=100, density=True, alpha=0.6, color='g')  # Create a histogram wi
plt.title('Histogram of Normal Data')  # Set the title of the histogram.
plt.xlabel('Value')  # Label the x-axis of the histogram.
plt.ylabel('Frequency')  # Label the y-axis of the histogram.

# Density plot of the Normal data
plt.subplot(1, 2, 2)  # Create a subplot (1 row, 2 columns, this is the 2nd subplot).
sns.kdeplot(normal_data, shade=True, color="g")  # Create a density plot with seaborn, green
plt.title('Density Plot of Normal Data')  # Set the title of the density plot.
plt.xlabel('Value')  # Label the x-axis of the density plot.
plt.ylabel('Density')  # Label the y-axis of the density plot.

plt.tight_layout()  # Adjust the subplots to fit into the figure area.
plt.show()  # Display the plots.
```

```
<ipython-input-20-ba26a91f912e>:25: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(normal_data, shade=True, color="g")  # Create a density plot with seaborn,
```




```
# Set the mean and standard deviation for the normal distribution
mu_normal = 50
sigma_normal = 10
```

```
import numpy as np  # Import NumPy and give it the alias 'np'

# Generate 1 million random integers following a normal distribution
normal_data = np.random.normal(mu_normal, sigma_normal, 1000000)
```

```
print(normal_data)
```

```
[49.46097725 64.31233409 48.37312022 ... 55.09669751 53.11588054
 49.47411335]
```

```
print("Data generation complete.")
```

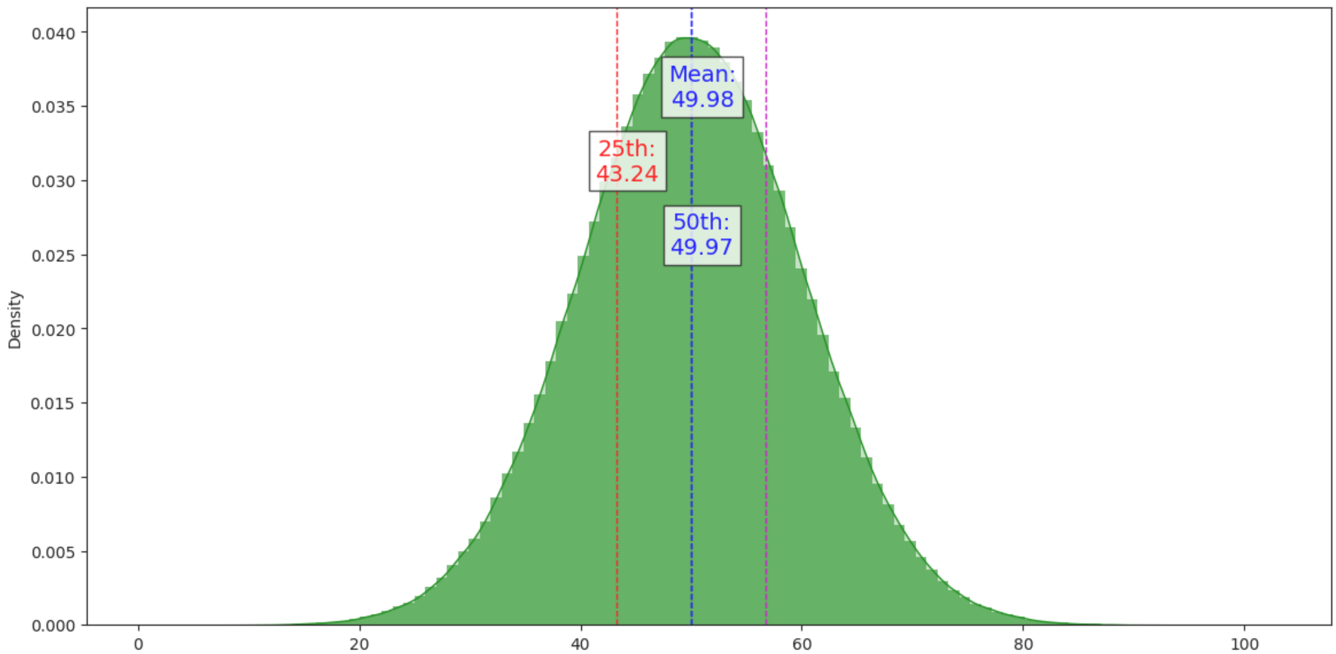
```
Data generation complete.
```

*2. Statistical Analysis**

We will compute the mean and standard deviation, as well as calculate the 25th, 50th, and 75th percentiles.

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import seaborn as sns  # Import the seaborn library, which is used for statistical data visu

# Set the mean and standard deviation for the normal distribution
mu_normal = 50  # Set the mean (mu) of the normal distribution to 50.
sigma_normal = 10  # Set the standard deviation (sigma) of the normal distribution to 10.

# Generate 1 million random numbers following a normal distribution
normal_data = np.random.normal(mu_normal, sigma_normal, 1000000)
# Generate 1 million random numbers from a normal distribution with mean 50 and standard dev

# Calculate mean, standard deviation, and percentiles
mean_normal = np.mean(normal_data)  # Calculate the mean of the generated normal data.
std_dev_normal = np.std(normal_data)  # Calculate the standard deviation of the generated nc
percentiles_normal = np.percentile(normal_data, [25, 50, 75])  # Calculate the 25th, 50th, a

# Plotting the data
plt.figure(figsize=(14, 7))  # Create a figure for the plot with specified size (14 inches b

# Histogram of the Normal data
plt.hist(normal_data, bins=100, density=True, alpha=0.6, color='g', label='Normal Data')  #

# Density plot of the Normal data
sns.kdeplot(normal_data, shade=True, color="g", label='Density Plot')  # Create a density pl

# Add vertical lines for mean, std dev, and percentiles
plt.axvline(mean_normal, color='b', linestyle='dashed', linewidth=1)  # Draw a dashed vertic
plt.text(mean_normal + 1, 0.035, f'Mean:\n{mean_normal:.2f}', color='b', fontsize=14, ha='ce

plt.axvline(percentiles_normal[0], color='r', linestyle='dashed', linewidth=1)  # Draw a das
plt.text(percentiles_normal[0] + 1, 0.03, f'25th:\n{percentiles_normal[0]:.2f}', color='r',

plt.axvline(percentiles_normal[1], color='b', linestyle='dashed', linewidth=1)  # Draw a das
plt.text(percentiles_normal[1] + 1, 0.025, f'50th:\n{percentiles_normal[1]:.2f}', color='b',

plt.axvline(percentiles_normal[2], color='m', linestyle='dashed', linewidth=1)  # Draw a das
```

```
<ipython-input-25-e6ac7e445c0d>:25: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(normal_data, shade=True, color="g", label='Density Plot')  # Create a dens
<matplotlib.lines.Line2D at 0x7dd1e7449840>
```




```
# Calculate the mean of the normal data
mean_normal = np.mean(normal_data)


print(mean_normal)
```

```
49.9806965973235
```

```
# Calculate the standard deviation of the normal data
std_dev_normal = np.std(normal_data)
```

```
print(std_dev_normal)
```

```
10.009261341506893
```

```
# Calculate the 25th, 50th, and 75th percentiles
percentiles_normal = np.percentile(normal_data, [25, 50, 75])
```

```
print(percentiles_normal)
```

```
[43.23531715 49.97400899 56.73241816]
```

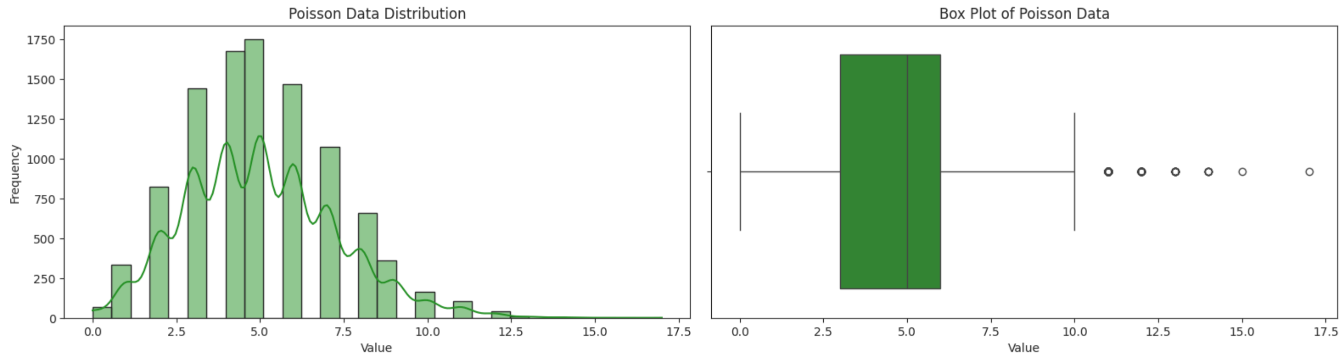
```
print(f"Mean: {mean_normal}, Standard Deviation: {std_dev_normal}")
print(f"25th percentile: {percentiles_normal[0]}, 50th percentile: {percentiles_normal[1]},
```

```
Mean: 49.9806965973235, Standard Deviation: 10.009261341506893
25th percentile: 43.2353171492598, 50th percentile: 49.97400898824674, 75th percentile:
```

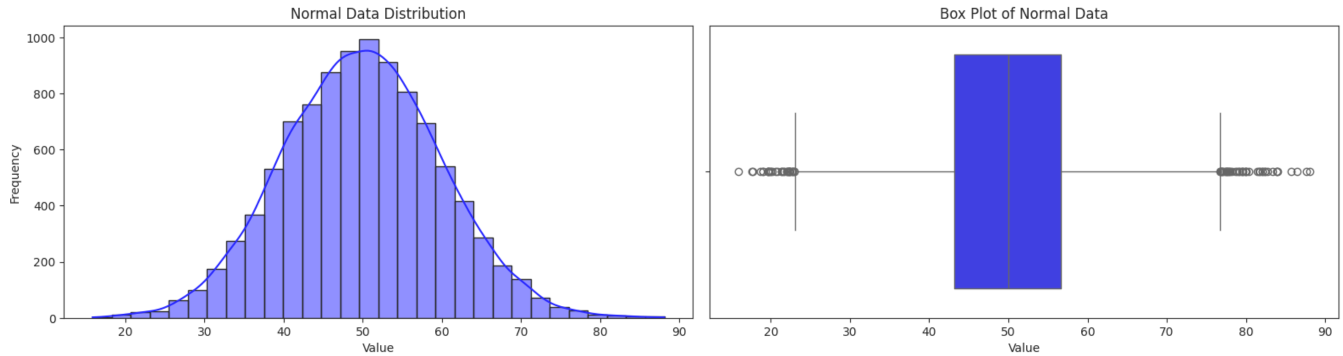
Task 3: Random Sampling

1. Sampling We will perform random sampling on the previously generated datasets, choosing a sample size of 10,000 data points

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import seaborn as sns  # Import the seaborn library, which is used for statistical data visu

# Set the sample size
sample_size = 10000  # Set the sample size to 10,000.

# Generate or load your Poisson data
lambda_param = 5  # Set the lambda parameter for the Poisson distribution.
poisson_data = np.random.poisson(lambda_param, 100000)  # Generate 100,000 samples from a Po

# Generate or load your Normal data
mu_normal = 50  # Set the mean (mu) of the normal distribution to 50.
sigma_normal = 10  # Set the standard deviation (sigma) of the normal distribution to 10.
normal_data = np.random.normal(mu_normal, sigma_normal, 1000000)  # Generate 1,000,000 sampl

# Perform random sampling on the Poisson data
sample_poisson = np.random.choice(poisson_data, sample_size, replace=False)  # Randomly sele

# Perform random sampling on the Normal data
sample_normal = np.random.choice(normal_data, sample_size, replace=False)  # Randomly select

# Plotting the sampled data
plt.figure(figsize=(16, 8))  # Create a figure for the plot with specified size (16 inches b

# Histogram and Density Plot for Poisson Data
plt.subplot(2, 2, 1)  # Create a subplot (2 rows, 2 columns, this is the 1st subplot).
sns.histplot(sample_poisson, bins=30, kde=True, color='green')  # Create a histogram with 30
plt.title('Poisson Data Distribution')  # Set the title of the histogram.
plt.xlabel('Value')  # Label the x-axis of the histogram.
plt.ylabel('Frequency')  # Label the y-axis of the histogram.

# Box Plot for Poisson Data
plt.subplot(2, 2, 2)  # Create a subplot (2 rows, 2 columns, this is the 2nd subplot).
sns.boxplot(x=sample_poisson, color='green')  # Create a box plot for the Poisson data, gree
plt.title('Box Plot of Poisson Data')  # Set the title of the box plot.
plt.xlabel('Value')  # Label the x-axis of the box plot.

plt.tight_layout()  # Adjust the subplots to fit into the figure area.
plt.show()  # Display the plots.
```


Poisson Data Distribution
Frequency
Value
Box Plot of Poisson Data
Value

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

# Set the sample size
sample_size = 10000

# Generate or load your Poisson data
lambda_param = 5  # Set the lambda parameter for the Poisson distribution
poisson_data = np.random.poisson(lambda_param, 100000)  # Generate 100,000 samples

# Generate or load your Normal data
mu_normal = 50
sigma_normal = 10
normal_data = np.random.normal(mu_normal, sigma_normal, 1000000)

# Perform random sampling on the Poisson data
sample_poisson = np.random.choice(poisson_data, sample_size, replace=False)

# Perform random sampling on the Normal data
sample_normal = np.random.choice(normal_data, sample_size, replace=False)

# Plotting the sampled data
plt.figure(figsize=(16, 8))

# Histogram and Density Plot for Normal Data
plt.subplot(2, 2, 3)
sns.histplot(sample_normal, bins=30, kde=True, color='blue')
plt.title('Normal Data Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')

# Box Plot for Normal Data
plt.subplot(2, 2, 4)
sns.boxplot(x=sample_normal, color='blue')
plt.title('Box Plot of Normal Data')
plt.xlabel('Value')

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




```
import numpy as np

# Set the sample size
sample_size = 10000

# Generate or load your Poisson data here.
# For example, let's generate some random Poisson data:
lambda_param = 5  # Set the lambda parameter for the Poisson distribution
poisson_data = np.random.poisson(lambda_param, 10000)  # Generate 10000 samples

# Perform random sampling on the Poisson data
sample_poisson = np.random.choice(poisson_data, sample_size, replace=False)

print(sample_poisson)
```

```
[3 5 4 ... 6 5 3]
```

```
# Perform random sampling on the normal data
sample_normal = np.random.choice(normal_data, sample_size,
replace=False)



print(sample_normal)
```

```
[58.66615963 32.9754928  46.26664531 ... 57.18139061 70.15054961
 35.95715712]
```

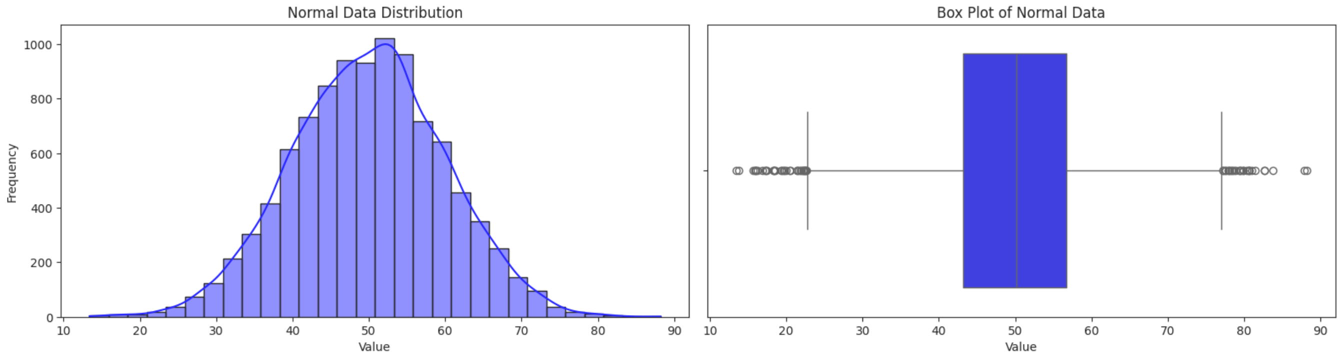
```
print("Random sampling complete.")
```

```
Random sampling complete.
```

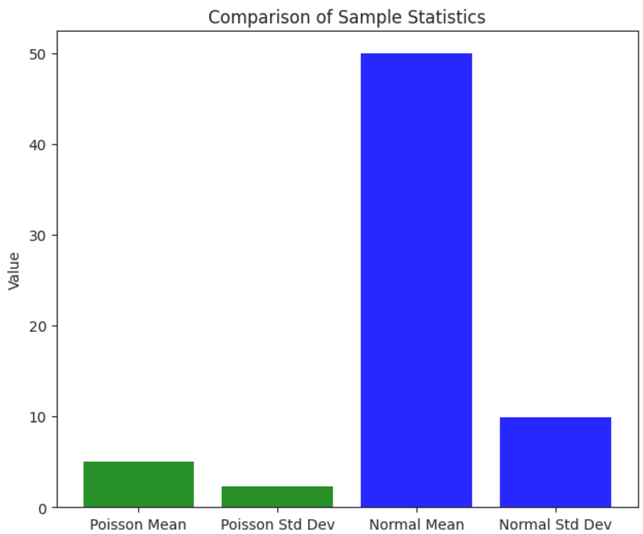
2. Sample Analysis We will analyze the properties of the samples and compare the sample statistics with the population statistics to assess accuracy and representativeness.

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import seaborn as sns  # Import the seaborn library, which is used for statistical data visu
import pandas as pd  # Import the pandas library, which is used for data manipulation and ar

# Set the sample size
sample_size = 10000  # Set the sample size to 10,000.

# Generate or load your Poisson data
lambda_param = 5  # Set the lambda parameter for the Poisson distribution.
poisson_data = np.random.poisson(lambda_param, 100000)  # Generate 100,000 samples from a Po

# Generate or load your Normal data
mu_normal = 50  # Set the mean (mu) of the normal distribution to 50.
sigma_normal = 10  # Set the standard deviation (sigma) of the normal distribution to 10.
normal_data = np.random.normal(mu_normal, sigma_normal, 1000000)  # Generate 1,000,000 sampl

# Perform random sampling on the Poisson data
sample_poisson = np.random.choice(poisson_data, sample_size, replace=False)  # Randomly sele

# Perform random sampling on the Normal data
sample_normal = np.random.choice(normal_data, sample_size, replace=False)  # Randomly select

# Calculate mean and standard deviation for the sampled data
mean_sample_poisson = np.mean(sample_poisson)  # Calculate the mean of the sampled Poisson c
std_dev_sample_poisson = np.std(sample_poisson)  # Calculate the standard deviation of the s
mean_sample_normal = np.mean(sample_normal)  # Calculate the mean of the sampled Normal data
std_dev_sample_normal = np.std(sample_normal)  # Calculate the standard deviation of the sam

# Create a DataFrame to display detailed statistics
df = pd.DataFrame({
    'Statistic': ['Mean', 'Standard Deviation'],
    'Poisson': [mean_sample_poisson, std_dev_sample_poisson],
    'Normal': [mean_sample_normal, std_dev_sample_normal]
})

# Plotting the sampled data
plt.figure(figsize=(16, 8))  # Create a figure for the plot with specified size (16 inches b




# Histogram and Density Plot for Normal Data
plt.subplot(2, 2, 3)  # Create a subplot (2 rows, 2 columns, this is the 3rd subplot).
sns.histplot(sample_normal, bins=30, kde=True, color='blue')  # Create a histogram with 30 b
plt.title('Normal Data Distribution')  # Set the title of the histogram.
plt.xlabel('Value')  # Label the x-axis of the histogram.
plt.ylabel('Frequency')  # Label the y-axis of the histogram.

# Box Plot for Normal Data
plt.subplot(2, 2, 4)  # Create a subplot (2 rows, 2 columns, this is the 4th subplot).
sns.boxplot(x=sample_normal, color='blue')  # Create a box plot for the Normal data, blue cc
```

```
plt.title('Box Plot of Normal Data')  # Set the title of the box plot.
plt.xlabel('Value')  # Label the x-axis of the box plot.

plt.tight_layout()  # Adjust the subplots to fit into the figure area.
plt.show()  # Display the plots.

# Plotting the data
fig, axes = plt.subplots(1, 2, figsize=(16, 6))  # Create a figure with 2 subplots side by s

# Bar plot for mean and standard deviation
axes[0].bar(['Poisson Mean', 'Poisson Std Dev', 'Normal Mean', 'Normal Std Dev'],
            [mean_sample_poisson, std_dev_sample_poisson, mean_sample_normal, std_dev_sample
            color=['green', 'green', 'blue', 'blue'])  # Create a bar plot for the means anc
axes[0].set_title('Comparison of Sample Statistics')  # Set the title of the bar plot.
axes[0].set_ylabel('Value')  # Label the y-axis of the bar plot.

# Table for detailed statistics
axes[1].axis('tight')  # Set the axis to 'tight' to fit the table.
axes[1].axis('off')  # Turn off the axis lines and labels.
table = axes[1].table(cellText=df.values, colLabels=df.columns, cellLoc='center', loc='cente

plt.suptitle('Sample Analysis of Poisson and Normal Distributions')  # Set the title of the
plt.show()  # Display the plots.
```



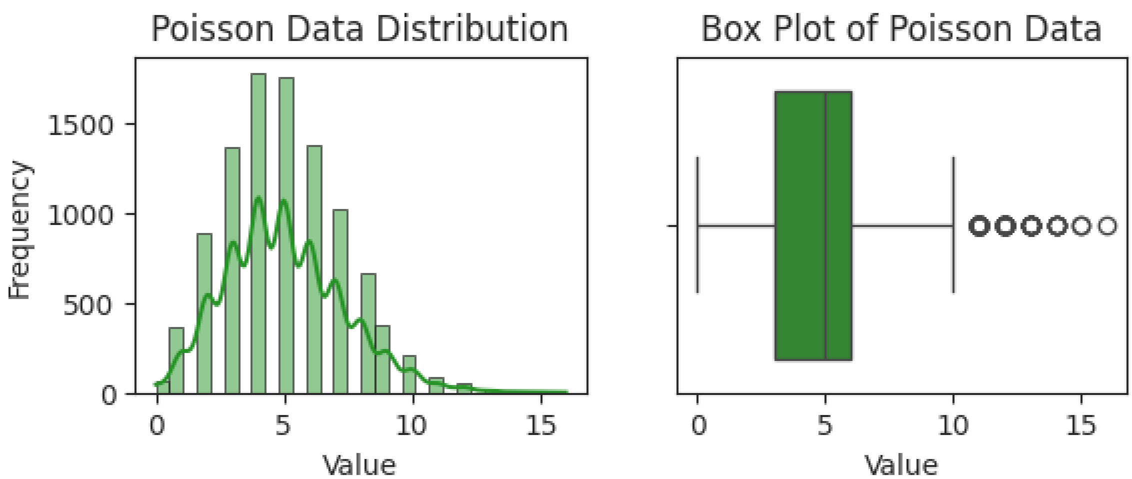
Sample Analysis of Poisson and Normal Distributions



Statistic	Poisson	Normal
Mean	5.0014	49.93696160498514
Standard Deviation	2.29237824976595	9.96123041535733

```
# Histogram and Density Plot for Poisson Data
plt.subplot(2, 2, 1)  # Create a subplot (2 rows, 2 columns, this is the 1st subplot).
sns.histplot(sample_poisson, bins=30, kde=True, color='green')  # Create a histogram with 3(
plt.title('Poisson Data Distribution')  # Set the title of the histogram.
plt.xlabel('Value')  # Label the x-axis of the histogram.
plt.ylabel('Frequency')  # Label the y-axis of the histogram.

# Box Plot for Poisson Data
plt.subplot(2, 2, 2)  # Create a subplot (2 rows, 2 columns, this is the 2nd subplot).
sns.boxplot(x=sample_poisson, color='green')  # Create a box plot for the Poisson data, gree
plt.title('Box Plot of Poisson Data')  # Set the title of the box plot.
plt.xlabel('Value')  # Label the x-axis of the box plot.
```

```
Text(0.5, 0, 'Value')
```




```
# Calculate the mean and standard deviation of the Poisson sample
mean_sample_poisson = np.mean(sample_poisson)
std_dev_sample_poisson = np.std(sample_poisson)
```

```
print(sample_poisson)
```

```
[8 3 1 ... 3 8 3]
```

```
print(std_dev_sample_poisson)
```

```
2.2363071323053996
```

```
# Calculate the mean and standard deviation of the normal sample
mean_sample_normal = np.mean(sample_normal)
std_dev_sample_normal = np.std(sample_normal)
```

```
print(mean_sample_normal)
```

```
49.93696160498514
```

```
print(std_dev_sample_normal)
```

```
9.96123041535733
```

```
print(f"Poisson Sample - Mean: {mean_sample_poisson}, Standard Deviation: {std_dev_sample_pc
print(f"Normal Sample - Mean: {mean_sample_normal}, Standard Deviation: {std_dev_sample_norm
```

```
Poisson Sample - Mean: 5.0014, Standard Deviation: 2.29237824976595
Normal Sample - Mean: 49.93696160498514, Standard Deviation: 9.96123041535733
```

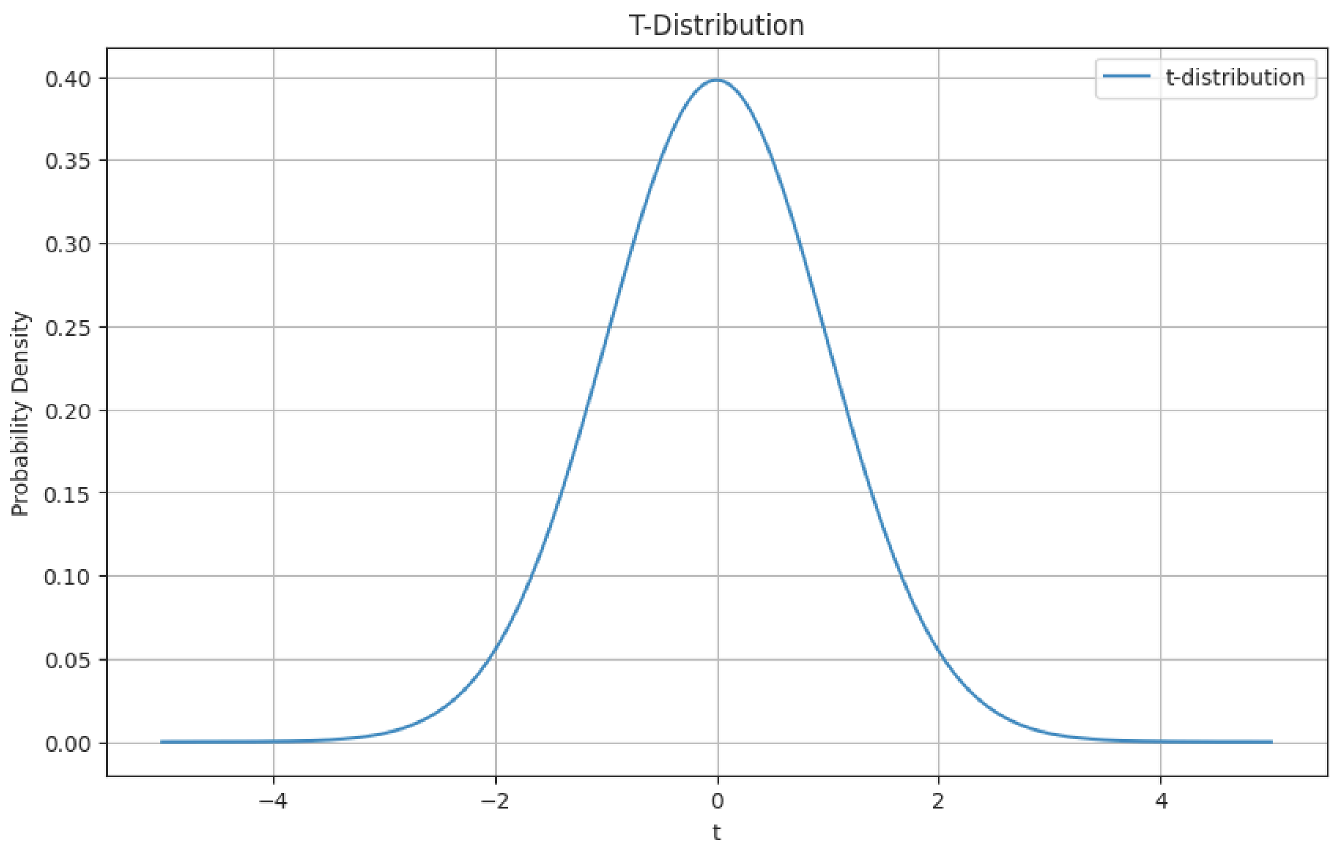
Task 4: T-Distribution and Hypothesis Testing

```
# Import necessary libraries
import numpy as np  # Import the numpy library, which is used for numerical operations.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
from scipy import stats  # Import the stats module from the scipy library, which contains st
```

```
# Task 1: T-Distribution Creation
# Generate sample data
np.random.seed(0)  # Set the seed for random number generation to ensure reproducibility.
sample_data = np.random.normal(loc=5, scale=2, size=100)  # Generate 100 random numbers from
```

```
# Calculate the t-distribution based on the sample data
df = len(sample_data) - 1  # Calculate the degrees of freedom, which is the number of data p
mean = np.mean(sample_data)  # Calculate the mean (average) of the sample data.
sem = stats.sem(sample_data)  # Calculate the standard error of the mean for the sample data
t_dist = stats.t(df)  # Create a t-distribution object based on the degrees of freedom.
```

```
# Visualize the t-distribution
x = np.linspace(-5, 5, 1000)  # Generate 1000 points between -5 and 5 for the x-axis.
y = t_dist.pdf(x)  # Calculate the probability density function values for the t-distributic

plt.figure(figsize=(10, 6))  # Create a figure for the plot with specified size.
plt.plot(x, y, label='t-distribution')  # Plot the t-distribution.
plt.title('T-Distribution')  # Set the title of the plot.
plt.xlabel('t')  # Label the x-axis.
plt.ylabel('Probability Density')  # Label the y-axis.
plt.legend()  # Add a legend to the plot.
plt.grid(True)  # Add a grid to the plot for better readability.
plt.show()  # Display the plot.
```



1. **T-Distribution Creation** We will generate a t-distribution based on the sample data.

```
from scipy import stats  # Import the stats module from the scipy library, which contains st
import numpy as np  # Import the numpy library, which is used for numerical operations, and

# Assuming you have data for your samples, let's create them here
# Replace these with your actual data
poisson_sample = np.random.poisson(5, 20)  # Generate a sample of 20 random numbers from a P
normal_sample = np.random.normal(10, 2, 20)  # Generate a sample of 20 random numbers from a

lambda_poisson = 5  # Set the expected mean of the Poisson distribution to 5.
mean_normal = 10  # Set the expected mean of the Normal distribution to 10.

# One-sample t-test for Poisson sample
t_stat_poisson, p_val_poisson = stats.ttest_1samp(poisson_sample, lambda_poisson)  # Perform
print("T-Test for Poisson Sample: T-Statistic =", t_stat_poisson, "P-Value =", p_val_poisson

# One-sample t-test for Normal sample
t_stat_normal, p_val_normal = stats.ttest_1samp(normal_sample, mean_normal)  # Perform a one
print("T-Test for Normal Sample: T-Statistic =", t_stat_normal, "P-Value =", p_val_normal)

# Confidence Intervals
ci_poisson = stats.t.interval(0.95, len(poisson_sample)-1, loc=np.mean(poisson_sample), scal
ci_normal = stats.t.interval(0.95, len(normal_sample)-1, loc=np.mean(normal_sample), scale=s

print("95% Confidence Interval for Poisson Sample:", ci_poisson)  # Print the 95% confidence
print("95% Confidence Interval for Normal Sample:", ci_normal)  # Print the 95% confidence i
```

```
T-Test for Poisson Sample: T-Statistic = -0.9198202372360741 P-Value = 0.36919798214698
T-Test for Normal Sample: T-Statistic = -2.013497794971456 P-Value = 0.05844731020643781
95% Confidence Interval for Poisson Sample: (3.5260381579406497, 5.57396184205935)
95% Confidence Interval for Normal Sample: (8.180357244282536, 10.035238916820898)
```

*2. Hypothesis Testing** We will perform hypothesis testing (one-sample t-test) and calculate confidence intervals.

Python Artifact: Titanic Project

1. Data Exploration We will thoroughly explore the Titanic dataset.

```
import numpy as np  # Import the numpy library, which is used for numerical operations.
import pandas as pd  # Import the pandas library, which is used for data manipulation and ar
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import seaborn as sns  # Import the seaborn library, which is used for data visualization.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data
```

```
import pandas as pd  # Import the pandas library, which is used for data manipulation and ar
import seaborn as sns  # Import the seaborn library, which is used for data visualization.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Print the first 15 rows of the DataFrame
print(data.head(15))  # Print the first 15 rows of the DataFrame 'data'
```

```
    survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0          0       3    male  22.0      1      0   7.2500        S   Third
1          1       1  female  38.0      1      0  71.2833        C   First
2          1       3  female  26.0      0      0   7.9250        S   Third
3          1       1  female  35.0      1      0  53.1000        S   First
4          0       3    male  35.0      0      0   8.0500        S   Third
5          0       3    male   NaN      0      0   8.4583        Q   Third
6          0       1    male  54.0      0      0  51.8625        S   First
7          0       3    male   2.0      3      1  21.0750        S   Third
8          1       3  female  27.0      0      2  11.1333        S   Third
9          1       2  female  14.0      1      0  30.0708        C  Second
10         1       3  female   4.0      1      1  16.7000        S   Third
11         1       1  female  58.0      0      0  26.5500        S   First
12         0       3    male  20.0      0      0   8.0500        S   Third
13         0       3    male  39.0      1      5  31.2750        S   Third
14         0       3  female  14.0      0      0   7.8542        S   Third

      who  adult_male deck  embark_town alive  alone
0     man        True  NaN  Southampton    no  False
1   woman       False    C    Cherbourg   yes  False
2   woman       False  NaN  Southampton   yes   True
3   woman       False    C  Southampton   yes  False
4     man        True  NaN  Southampton    no   True
5     man        True  NaN   Queenstown    no   True
6     man        True    E  Southampton    no   True
7   child       False  NaN  Southampton    no  False
8   woman       False  NaN  Southampton   yes  False
9   child       False  NaN    Cherbourg   yes  False
10  child       False    G  Southampton   yes  False
11  woman       False    C  Southampton   yes   True
12    man        True  NaN  Southampton    no   True
13    man        True  NaN  Southampton    no  False
14  child       False  NaN  Southampton    no   True
```

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Display the shape of the dataset
print(data.shape)  # Print the shape of the DataFrame 'data'
```

```
(891, 15)
```

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Display the summary information of the dataset
data.info()  # Print a concise summary of the DataFrame 'data'
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
```

```
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Display the original shape of the dataset
print("Original shape:", data.shape)  # Print the original shape of the DataFrame 'data'

# Drop two columns from the dataset
data = data.drop(columns=['deck', 'embark_town'])  # Drop the 'deck' and 'embark_town' colum

# Display the shape of the dataset after dropping the columns
print("Shape after dropping columns:", data.shape)  # Print the shape of the DataFrame 'data
```

```
Original shape: (891, 15)
Shape after dropping columns: (891, 13)
```

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Display the column names of the dataset
print(data.columns)  # Print the column names of the DataFrame 'data'
```

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Calculate the average fare of passengers
average_fare = round(data['fare'].mean(), 3)  # Calculate the mean of the 'fare' column and

# Print the average fare of passengers
print("Average Fare:", average_fare)  # Print the calculated average fare
```

```
Average Fare: 32.204
```

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Calculate the average age of passengers
average_age = round(data['age'].mean(), 2)  # Calculate the mean of the 'age' column and rou

# Print the average age of passengers
print("Average Age:", average_age)  # Print the calculated average age
```

```
Average Age: 29.7
```

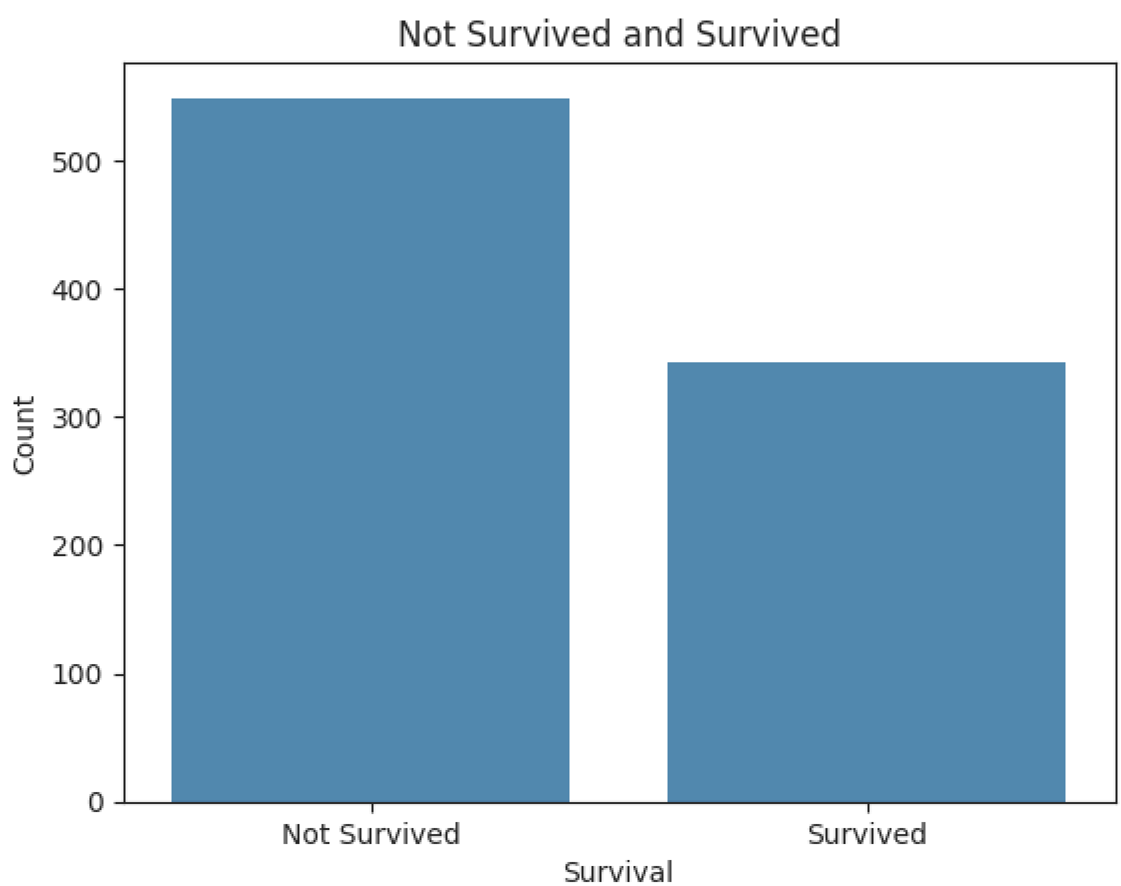
```
sns.countplot(data['Survived'])
plt.title("Not survived and Survived")
plt.xlabel("Survival")
plt.ylabel("Count")
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3790         try:
-> 3791             return self._engine.get_loc(casted_key)
   3792         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Survived'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
```

2 frames

```
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3796                 ):
   3797                     raise InvalidIndexError(key)
-> 3798             raise KeyError(key) from err
   3799         except TypeError:
   3800             # If we have a listlike key, _check_indexing_error will raise
```

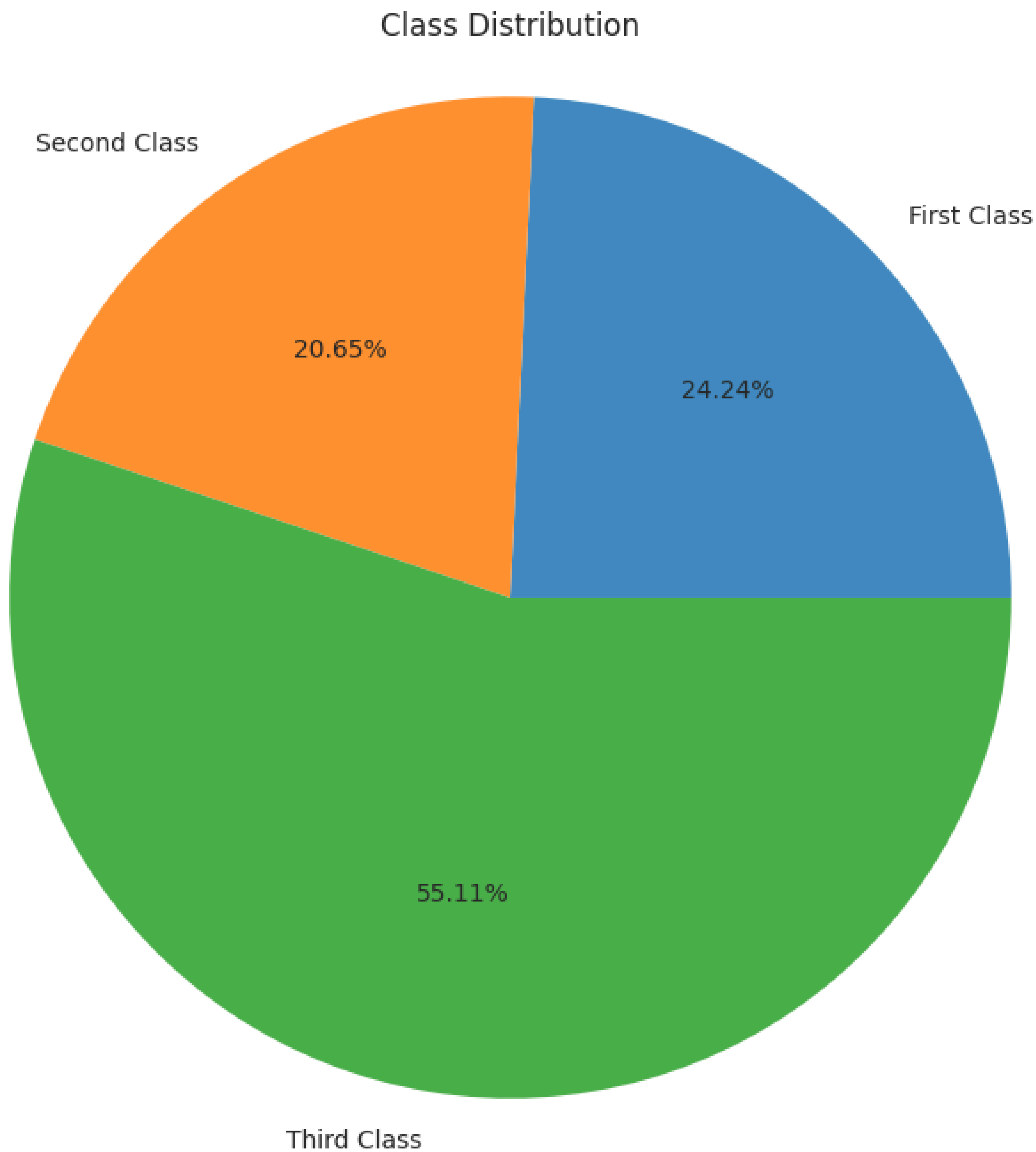
```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Create a count plot for the 'Survived' column
sns.countplot(x='survived', data=data)  # Plot the count of passengers who survived and did

# Set the title of the plot
plt.title("Not Survived and Survived")  # Set the title of the plot to "Not Survived and Sur

# Set the label for the x-axis
plt.xlabel("Survival")  # Set the label for the x-axis to "Survival"

# Set the label for the y-axis
plt.ylabel("Count")  # Set the label for the y-axis to "Count"

# Customize the x-ticks to display 'Not Survived' and 'Survived' instead of 0 and 1
plt.xticks(ticks=[0, 1], labels=['Not Survived', 'Survived'], rotation=0)  # Set custom labe

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



Specifying x='Survived' in sns.countplot(): This explicitly tells the count plot to use the 'Survived' column on the x-axis, ensuring the plot interprets the data correctly. Adjusting plt.xticks(): Setting the ticks and labels explicitly for the x-axis with plt.xticks(ticks=[0, 1], labels=['Not Survived', 'Survived'], rotation=0) ensured that the labels were correctly displayed and not overlapping. These adjustments ensure that the count plot correctly interprets and displays the categories of the 'Survived' column on the x-axis, improving the readability of the plot.

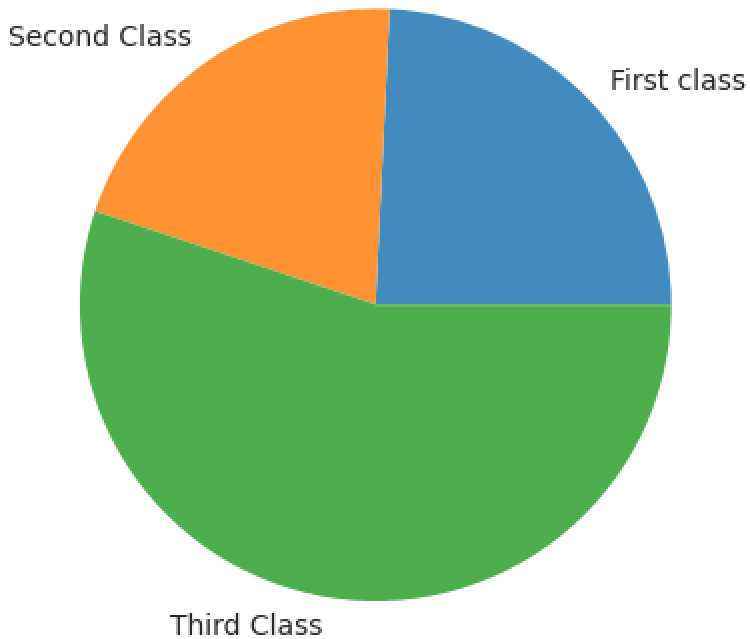
```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Calculate the number of passengers in each class
first_class_count = (data['pclass'] == 1).sum()  # Count the number of passengers in first c
second_class_count = (data['pclass'] == 2).sum()  # Count the number of passengers in seconc
third_class_count = (data['pclass'] == 3).sum()  # Count the number of passengers in third c

# Print the number of passengers in each class
print("First Class:", first_class_count)  # Print the count of first class passengers
print("Second Class:", second_class_count)  # Print the count of second class passengers
print("Third Class:", third_class_count)  # Print the count of third class passengers

# Define the labels and sizes for the pie chart
labels = ['First Class', 'Second Class', 'Third Class']  # Define the labels for the pie cha
sizes = [first_class_count, second_class_count, third_class_count]  # Define the sizes for e

# Create the pie chart
plt.figure(figsize=(8, 8))  # Set the size of the figure to 8 inches by 8 inches
plt.pie(sizes, labels=labels, autopct='%1.2f%%')  # Create the pie chart with percentage dis
plt.axis('equal')  # Set the aspect ratio to be equal so the pie is drawn as a circle
plt.title('Class Distribution')  # Set the title of the pie chart
plt.show()  # Display the pie chart
```

```
First Class: 216
Second Class: 184
Third Class: 491
```

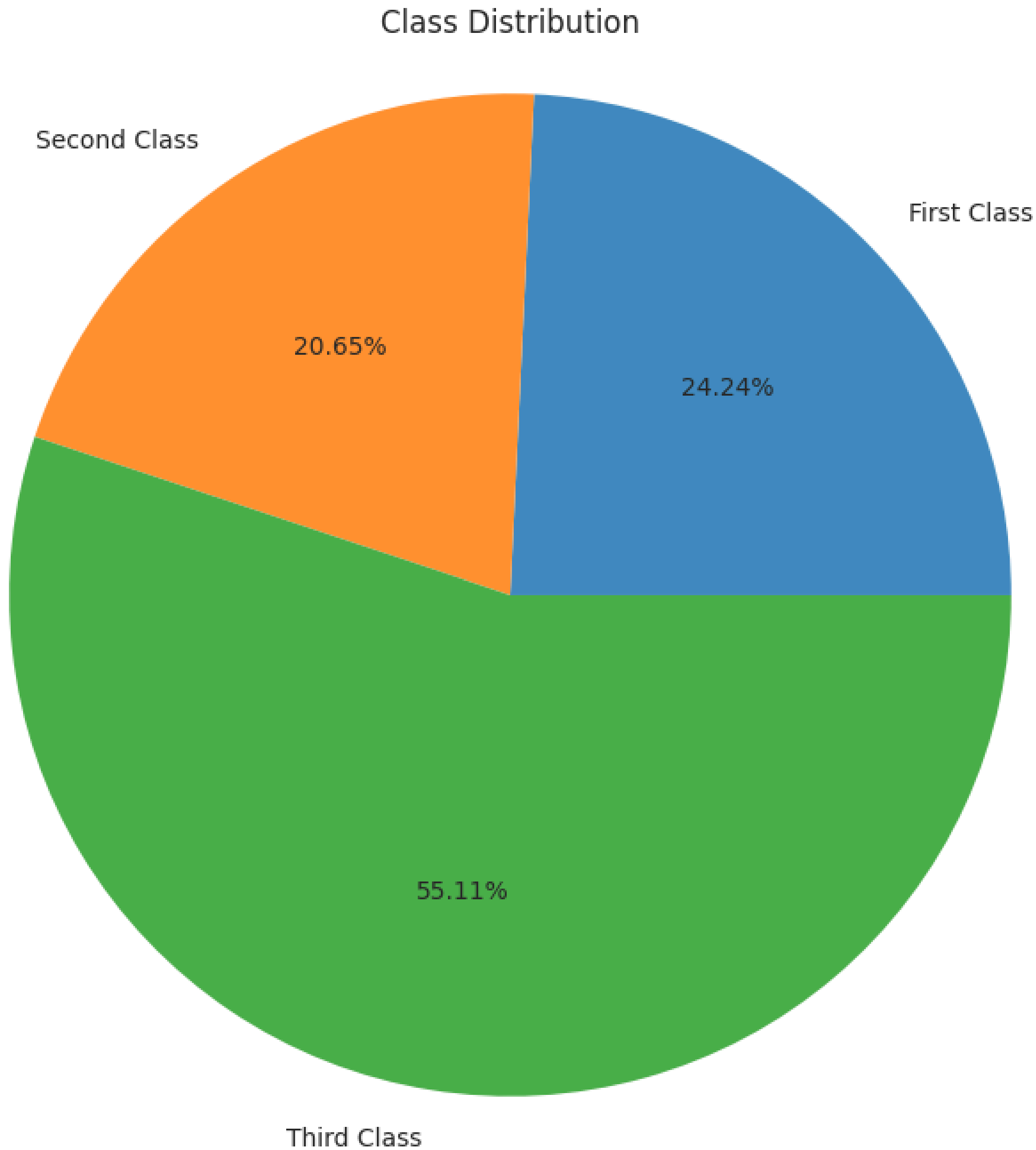



```
labels = ['First class', "Second Class", "Third Class"]
sizes= [first_class_count, second_class_count, third_class_count]
plt.pie(sizes, labels= labels)
plts.title('Class Distribution')
plt.show()
```

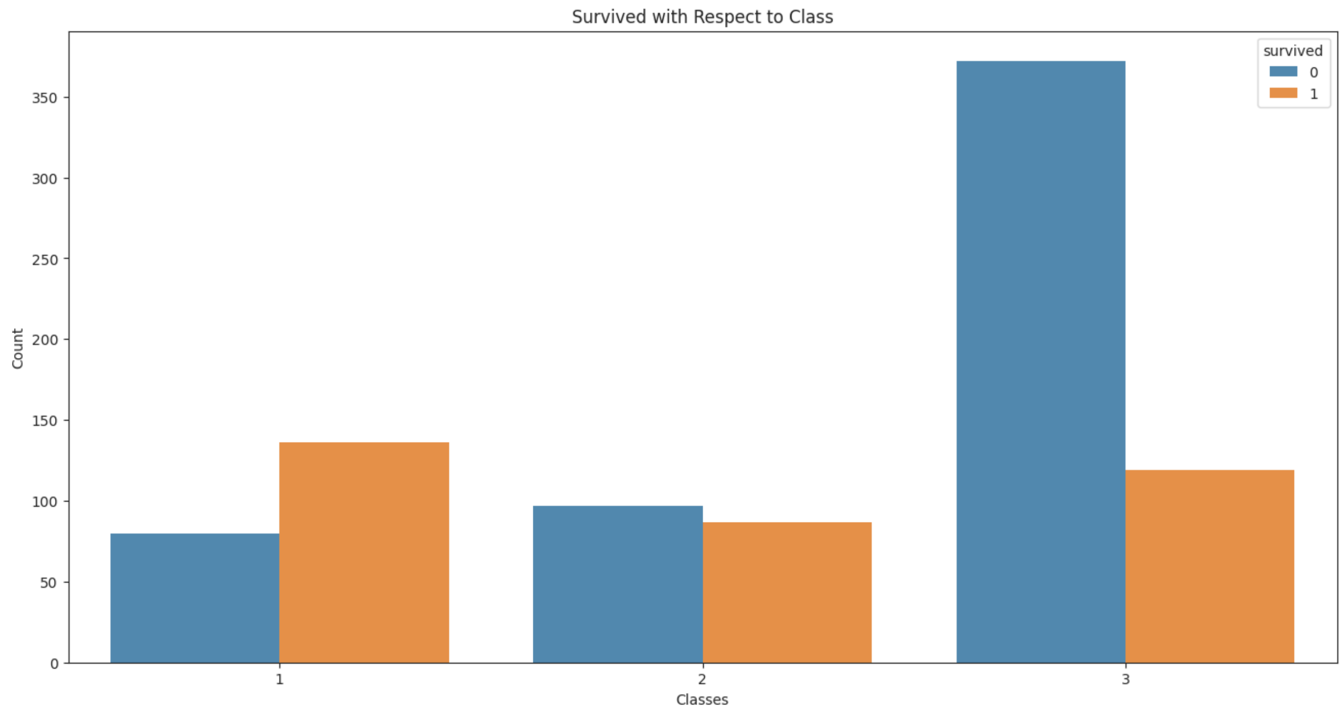
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-722d20266db7> in <cell line: 4>()
      2 sizes= [first_class_count, second_class_count, third_class_count]
      3 plt.pie(sizes, labels= labels)
----> 4 plts.title('Class Distribution')
      5 plt.show()

NameError: name 'plts' is not defined
```



```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Calculate the number of passengers in each class
first_class_count = data[data.pclass == 1].shape[0]  # Count the number of passengers in fir
second_class_count = data[data.pclass == 2].shape[0]  # Count the number of passengers in se
third_class_count = data[data.pclass == 3].shape[0]  # Count the number of passengers in thi

# Define the labels and sizes for the pie chart
labels = ['First Class', 'Second Class', 'Third Class']  # Define the labels for the pie cha
sizes = [first_class_count, second_class_count, third_class_count]  # Define the sizes for e

# Create the pie chart
plt.figure(figsize=(8, 8))  # Set the size of the figure to 8 inches by 8 inches
plt.pie(sizes, labels=labels, autopct='%1.2f%%')  # Create the pie chart with percentage dis
plt.axis('equal')  # Set the aspect ratio to be equal so the pie is drawn as a circle
plt.title('Class Distribution')  # Set the title of the pie chart
plt.show()  # Display the pie chart
```

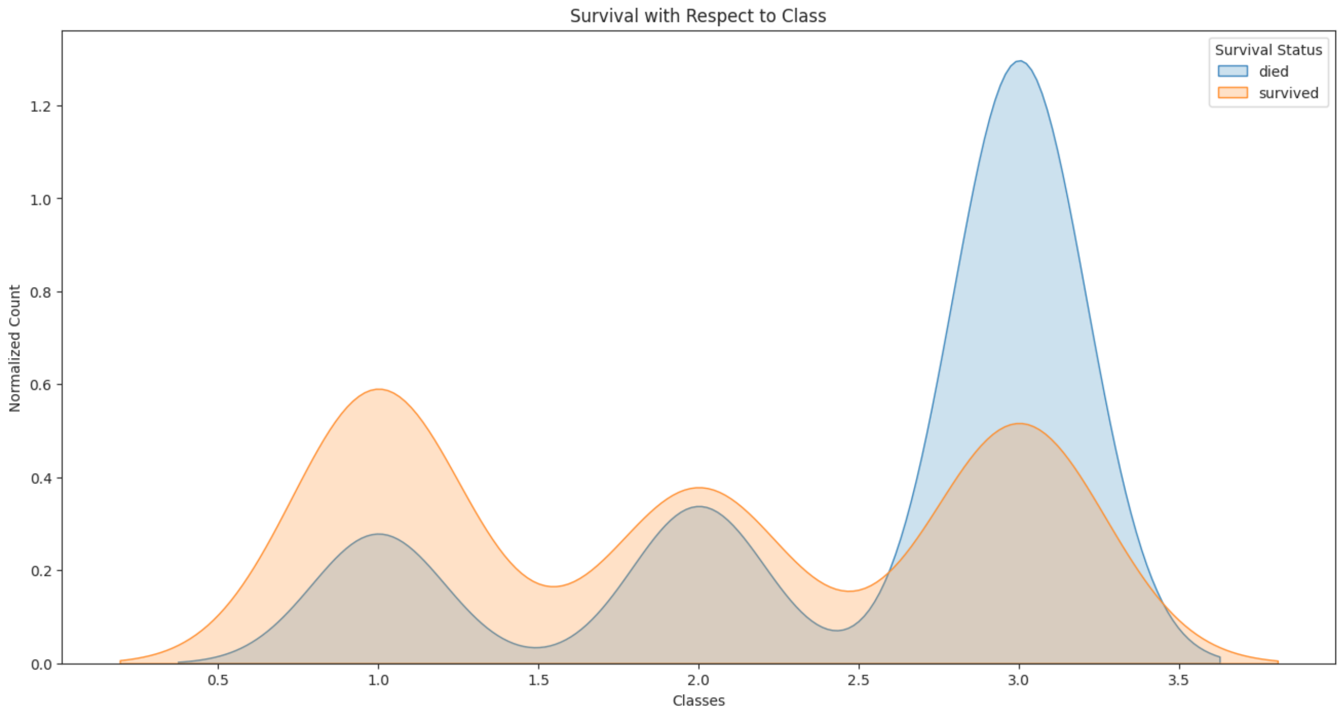

Class Distribution
Second Class
First Class
20.65%
24.24%
55.11%
Third Class

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Create a figure with specified size
plt.figure(figsize=(16, 8))  # Set the size of the figure to 16 inches by 8 inches.

# Create a count plot for the 'Pclass' column, with bars separated by 'Survived' status
sns.countplot(x=data["pclass"], hue=data['survived'])  # Plot count of passengers in each cl

# Set the title of the plot
plt.title("Survived with Respect to Class")  # Set the title of the plot to "Survived with R

# Set the label for the x-axis
plt.xlabel("Classes")  # Set the label for the x-axis to "Classes".

# Set the label for the y-axis
plt.ylabel("Count")  # Set the label for the y-axis to "Count".

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

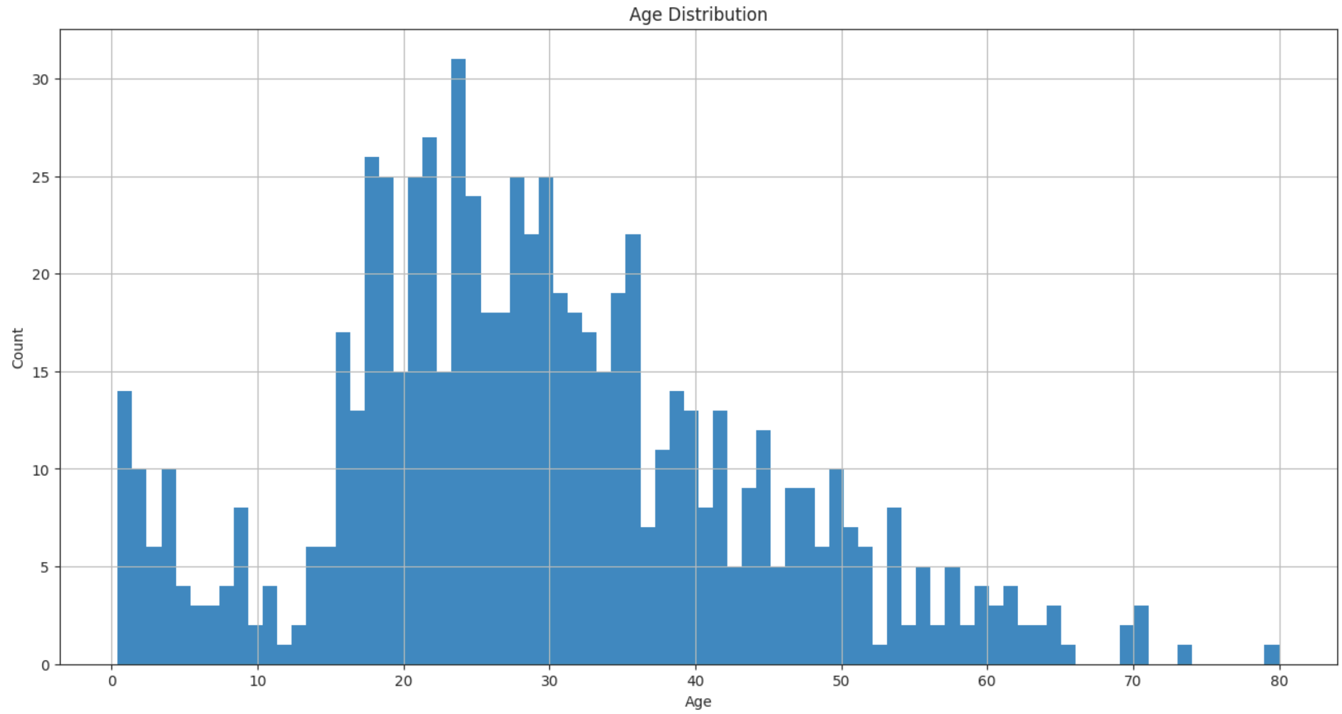

Survived with Respect to Class
survived
0
1
Count
350
300
250
200
150
100
50
0
1
2
3
Classes

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Create a figure with specified size
plt.figure(figsize=(16, 8))  # Set the size of the figure to 16 inches by 8 inches.

# Create KDE plots for the 'pclass' column, separated by 'Survived' status
axs = sns.kdeplot(data.pclass[data.survived == 0], shade=True, label="died")  # Plot KDE for
axs = sns.kdeplot(data.pclass[data.survived == 1], shade=True, label="survived")  # Plot KDE

# Set the title of the plot
plt.title("Survival with Respect to Class")  # Set the title of the plot to "Survival with R

# Set the label for the x-axis
plt.xlabel("Classes")  # Set the label for the x-axis to "Classes".

# Set the label for the y-axis
plt.ylabel("Normalized Count")  # Set the label for the y-axis to "Normalized Count".

# Add a legend to the plot
plt.legend(title='Survival Status')  # Add a legend with the title "Survival Status".

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

```
<ipython-input-65-f476ed10c118>:11: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  axs = sns.kdeplot(data.pclass[data.survived == 0], shade=True, label="died")  # Plot k
<ipython-input-65-f476ed10c118>:12: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  axs = sns.kdeplot(data.pclass[data.survived == 1], shade=True, label="survived")  # Pl
```

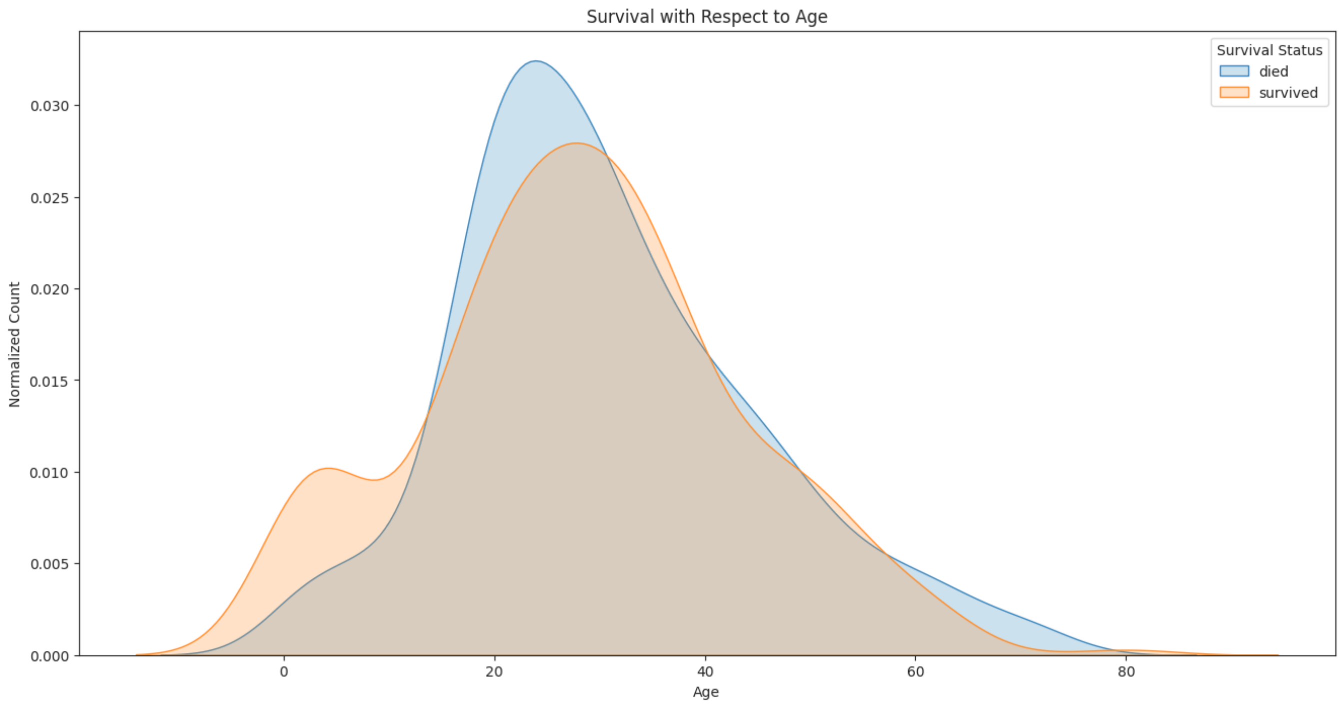


```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Create a figure with specified size
plt.figure(figsize=(16, 8))  # Set the size of the figure to 16 inches by 8 inches.

# Create a histogram for the 'Age' column with 80 bins
data['age'].hist(bins=80, figsize=(16, 8))  # Plot a histogram of the 'age' column with 80 b

# Set the title of the plot
plt.title("Age Distribution")  # Set the title of the plot to "Age Distribution".

# Set the label for the x-axis
plt.xlabel("Age")  # Set the label for the x-axis to "Age".

# Set the label for the y-axis
plt.ylabel("Count")  # Set the label for the y-axis to "Count".

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

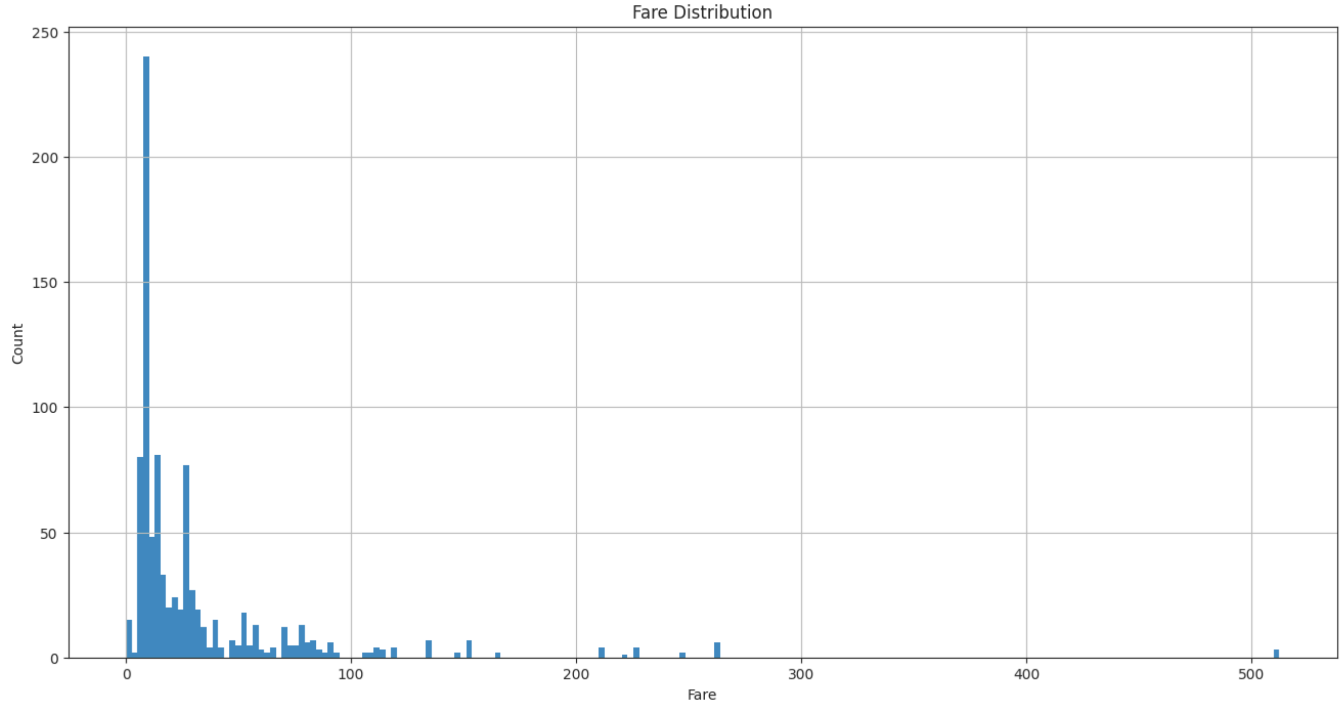

Age Distribution
Count
Age
0
10
20
30
40
50
60
70
80
0
5
10
15
20
25
30

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Verify the column names in your DataFrame
print(data.columns)  # Print the column names to check for 'Age' or a similar column

# Create a figure with specified size
plt.figure(figsize=(16, 8))  # Set the size of the figure to 16 inches by 8 inches.

# Create KDE plots for the 'Age' column, separated by 'Survived' status
# If the column name is different, replace 'Age' below with the correct name
axs = sns.kdeplot(data['age'][data.survived == 0], shade=True, label="died")  # Plot KDE for
axs = sns.kdeplot(data['age'][data.survived == 1], shade=True, label="survived")  # Plot KDE

# Set the title of the plot
plt.title("Survival with Respect to Age")  # Set the title of the plot to "Survival with Res

# Set the label for the x-axis
plt.xlabel("Age")  # Set the label for the x-axis to "Age".

# Set the label for the y-axis
plt.ylabel("Normalized Count")  # Set the label for the y-axis to "Normalized Count".

# Add a legend to the plot
plt.legend(title='Survival Status')  # Add a legend with the title "Survival Status".

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

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
<ipython-input-67-f1d906bb7e9e>:15: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  axs = sns.kdeplot(data['age'][data.survived == 0], shade=True, label="died")  # Plot k
<ipython-input-67-f1d906bb7e9e>:16: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  axs = sns.kdeplot(data['age'][data.survived == 1], shade=True, label="survived")  # Pl
```



```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Create a figure with specified size
plt.figure(figsize=(16, 8))  # Set the size of the figure to 16 inches by 8 inches.

# Create a histogram for the 'Fare' column with 200 bins
data['fare'].hist(bins=200, figsize=(16, 8))  # Plot a histogram of the 'fare' column with 2

# Set the title of the plot
plt.title("Fare Distribution")  # Set the title of the plot to "Fare Distribution".

# Set the label for the x-axis
plt.xlabel("Fare")  # Set the label for the x-axis to "Fare".

# Set the label for the y-axis
plt.ylabel("Count")  # Set the label for the y-axis to "Count".

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




```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import pandas as pd  # Import pandas library, which is used for data manipulation and analys

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Filter the DataFrame to show only rows where the 'fare' is greater than 500
high_fare_data = data.loc[data['fare'] > 500]  # Use .loc to filter rows where 'fare' columr

# Display the filtered DataFrame
print(high_fare_data)  # Print the filtered DataFrame to show the rows with fare greater tha
```

```
     survived  pclass     sex   age  sibsp  parch      fare embarked  class  \
258         1       1  female  35.0      0      0  512.3292        C  First
679         1       1    male  36.0      0      1  512.3292        C  First
737         1       1    male  35.0      0      0  512.3292        C  First

       who  adult_male deck embark_town alive  alone
258  woman       False  NaN   Cherbourg   yes   True
```

```
679    man        True    B   Cherbourg   yes  False
737    man        True    B   Cherbourg   yes   True
```

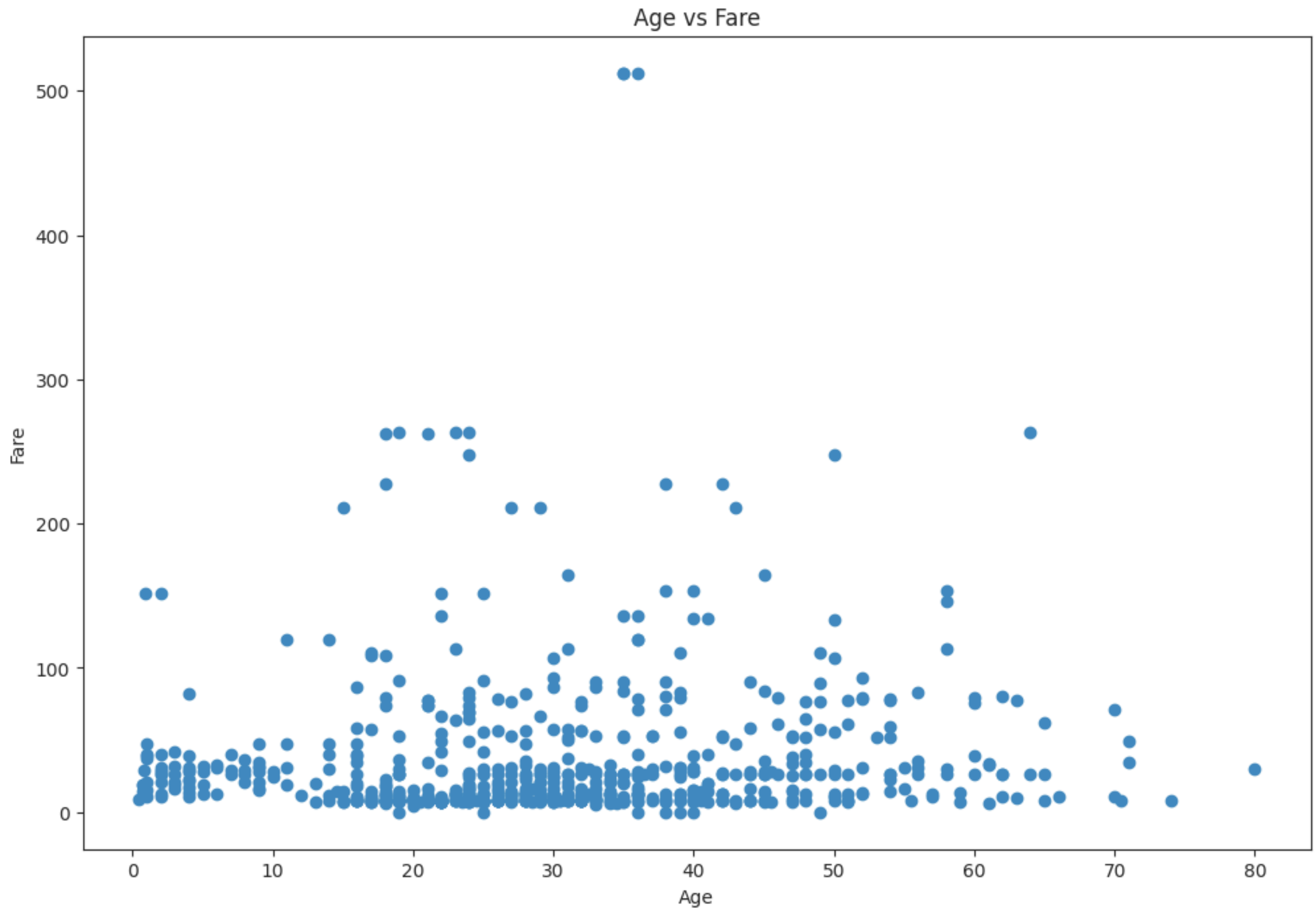
```
data.loc[data['Fare']>500]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self,
key)
   3790         try:
-> 3791             return self._engine.get_loc(casted_key)
   3792         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Fare'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
```

2 frames

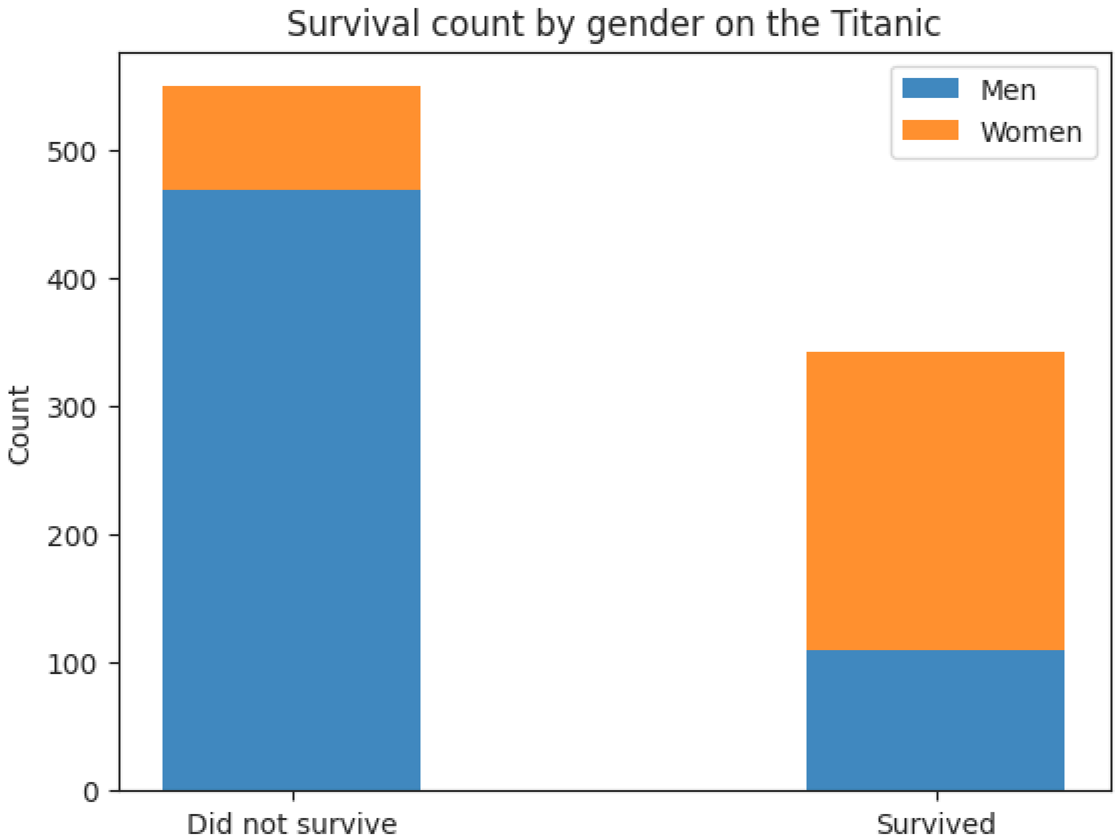
```
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self,
key)
   3796             ):
   3797                 raise InvalidIndexError(key)
-> 3798             raise KeyError(key) from err
   3799         except TypeError:
   3800             # If we have a listlike key, _check_indexing_error will raise
```

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Create a scatter plot of Age vs. Fare
plt.figure(figsize=(12, 8))  # Set the size of the plot to 12 inches by 8 inches.
plt.scatter(data['age'], data['fare'])  # Create a scatter plot with 'age' on the x-axis anc
plt.title("Age vs Fare")  # Set the title of the plot to "Age vs Fare".
plt.xlabel("Age")  # Set the label for the x-axis to "Age".
plt.ylabel("Fare")  # Set the label for the y-axis to "Fare".
plt.show()  # Display the plot.
```

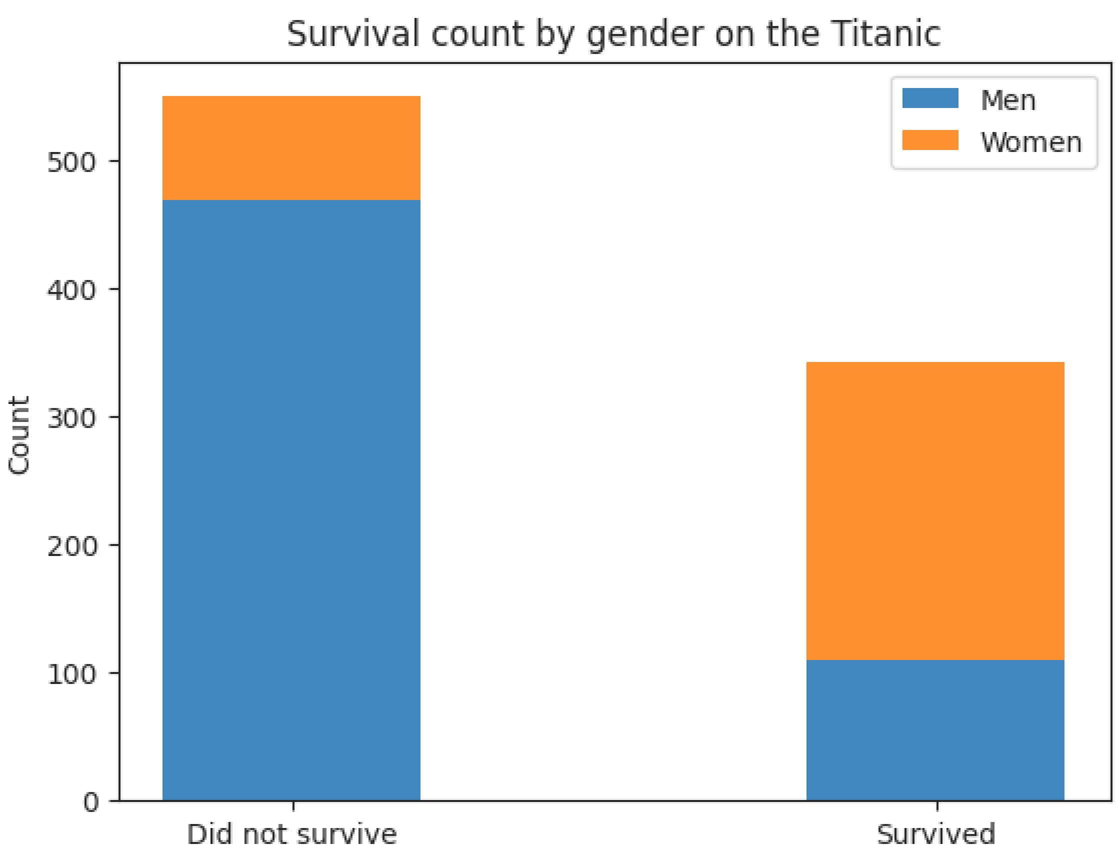

Age vs Fare
Fare
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
Age

```
# Import necessary libraries
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import numpy as np  # Import the numpy library, which is used for numerical operations.

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Calculate the number of males who did not survive
male_dead = ((data['sex'] == 'male') & (data['survived'] == 0)).sum()  # Use boolean indexir
print("Men Died:", male_dead)  # Print the number of males who did not survive

# Calculate the number of males who survived
male_survived = ((data['sex'] == 'male') & (data['survived'] == 1)).sum()  # Use boolean inc
print("Men Survived:", male_survived)  # Print the number of males who survived

# Calculate the number of females who did not survive
female_dead = ((data['sex'] == 'female') & (data['survived'] == 0)).sum()  # Use boolean inc
print("Women Died:", female_dead)  # Print the number of females who did not survive

# Calculate the number of females who survived
female_survived = ((data['sex'] == 'female') & (data['survived'] == 1)).sum()  # Use boolear
print("Women Survived:", female_survived)  # Print the number of females who survived

# Create tuples with the calculated data
m_data = (male_dead, male_survived)  # Create a tuple with the number of males who did not s
f_data = (female_dead, female_survived)  # Create a tuple with the number of females who dic

# Create the bar plots for men and women
p1 = plt.bar(np.arange(2), m_data, width=0.4, label='Men')  # Create a bar plot for men witt
p2 = plt.bar(np.arange(2), f_data, width=0.4, bottom=m_data, label='Women')  # Stack women's

# Set the labels and title
plt.xticks(np.arange(2), ['Did not survive', 'Survived'])  # Set the x-ticks to label the ca
plt.ylabel('Count')  # Set the label for the y-axis
plt.title('Survival count by gender on the Titanic')  # Set the title of the plot
plt.legend()  # Add a legend to the plot

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

```
Men Died: 468
Men Survived: 109
Women Died: 81
Women Survived: 233
```

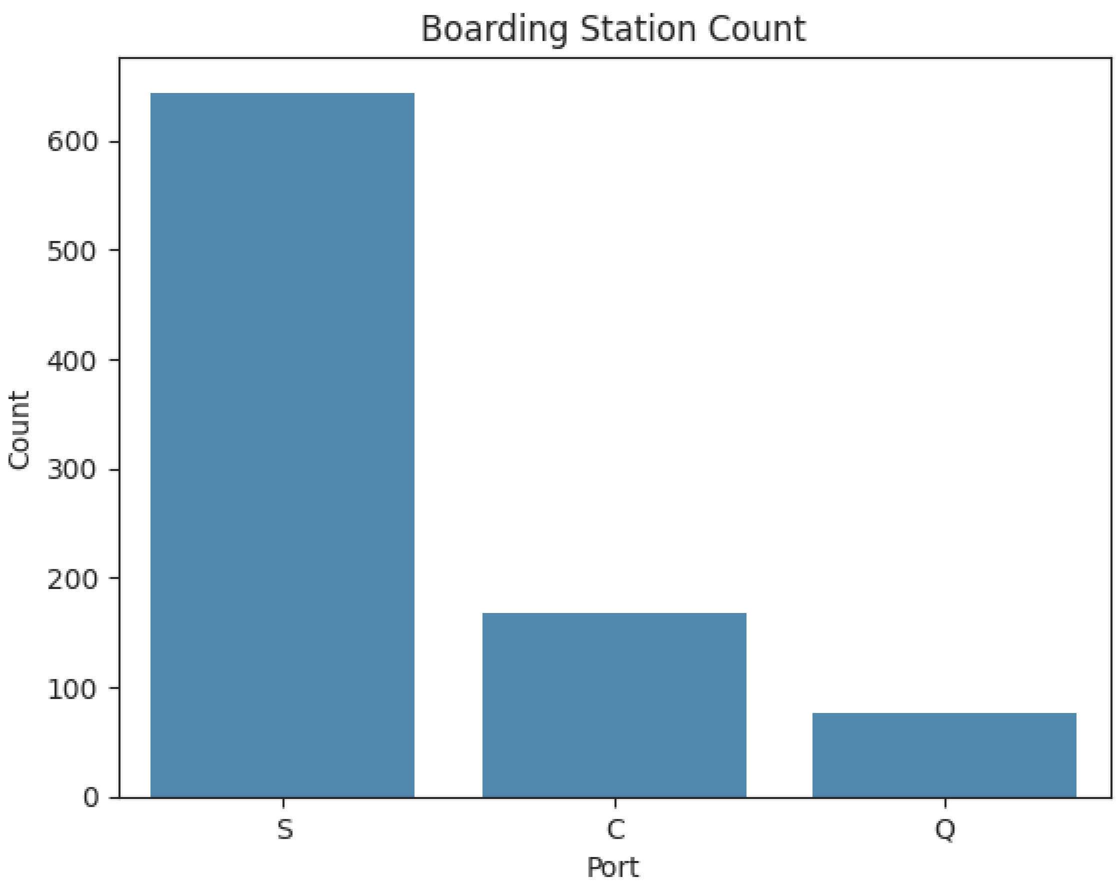


```
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import numpy as np  # Import the numpy library, which is used for numerical operations.

# Assuming you have the following data from the Titanic dataset:
male_dead = 468  # Number of males who did not survive
male_survived = 109  # Number of males who survived
female_dead = 81  # Number of females who did not survive
female_survived = 233  # Number of females who survived

m_data = (male_dead, male_survived)  # Create a tuple with the number of males who did not s
f_data = (female_dead, female_survived)  # Create a tuple with the number of females who dic

# Create the bar plots for men and women
p1 = plt.bar(np.arange(2), m_data, width=0.4, label='Men')  # Create a bar plot for men with
p2 = plt.bar(np.arange(2), f_data, width=0.4, bottom=m_data, label='Women')  # Stack women's

# Set the labels and title
plt.xticks(np.arange(2), ['Did not survive', 'Survived'])  # Set the x-ticks to label the ca
plt.ylabel('Count')  # Set the label for the y-axis
plt.title('Survival count by gender on the Titanic')  # Set the title of the plot
plt.legend()  # Add a legend to the plot

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



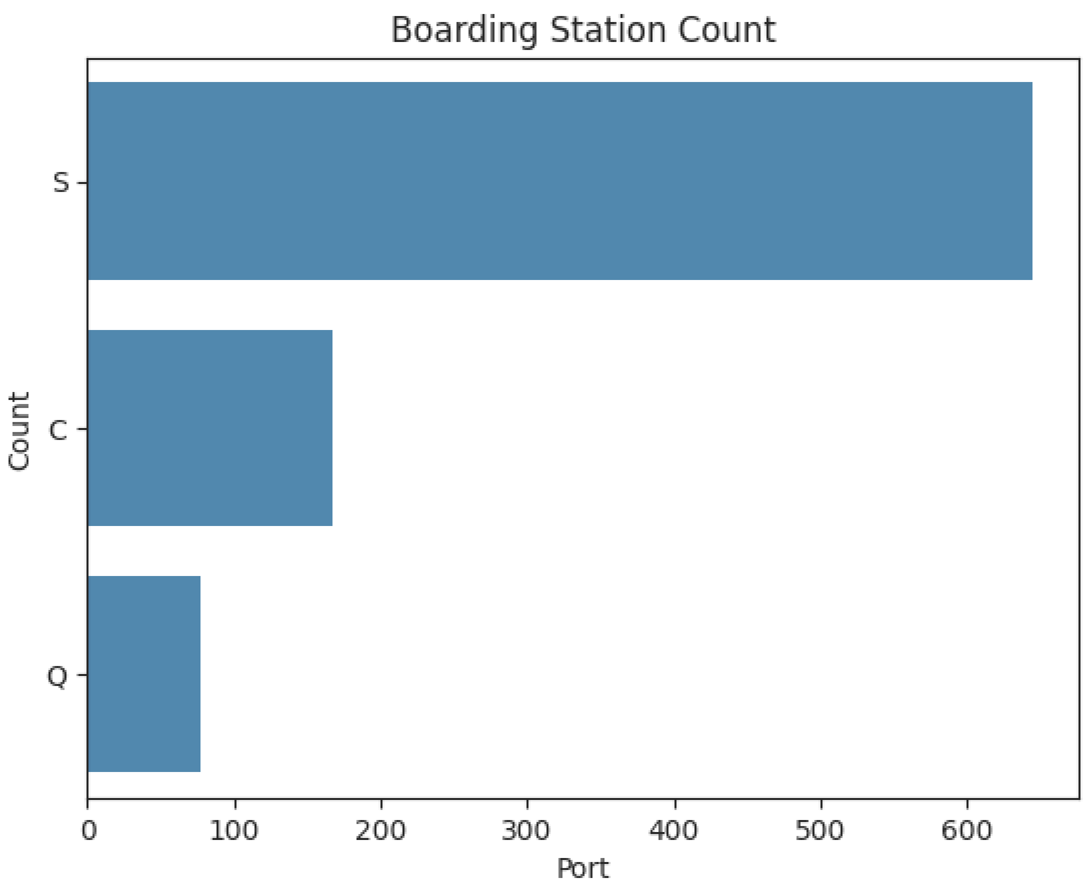
```
!pip install seaborn  # Install the seaborn library. This command is used in Jupyter Noteboo

import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Load the Titanic dataset from seaborn
data = sns.load_dataset('titanic')  # Load the Titanic dataset into a DataFrame called 'data

# Verify the column names in your DataFrame
print(data.columns)  # Print the column names of the DataFrame to check for 'Embarked'. This

# Assuming 'Embarked' is present and correctly spelled, proceed with the countplot
sns.countplot(x='embarked', data=data)  # Create a count plot using the 'embarked' column fr

plt.title("Boarding Station Count")  # Set the title of the plot to "Boarding Station Count'
plt.xlabel("Port")  # Set the label for the x-axis to "Port".
plt.ylabel("Count")  # Set the label for the y-axis to "Count".
plt.show()  # Display the plot.
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

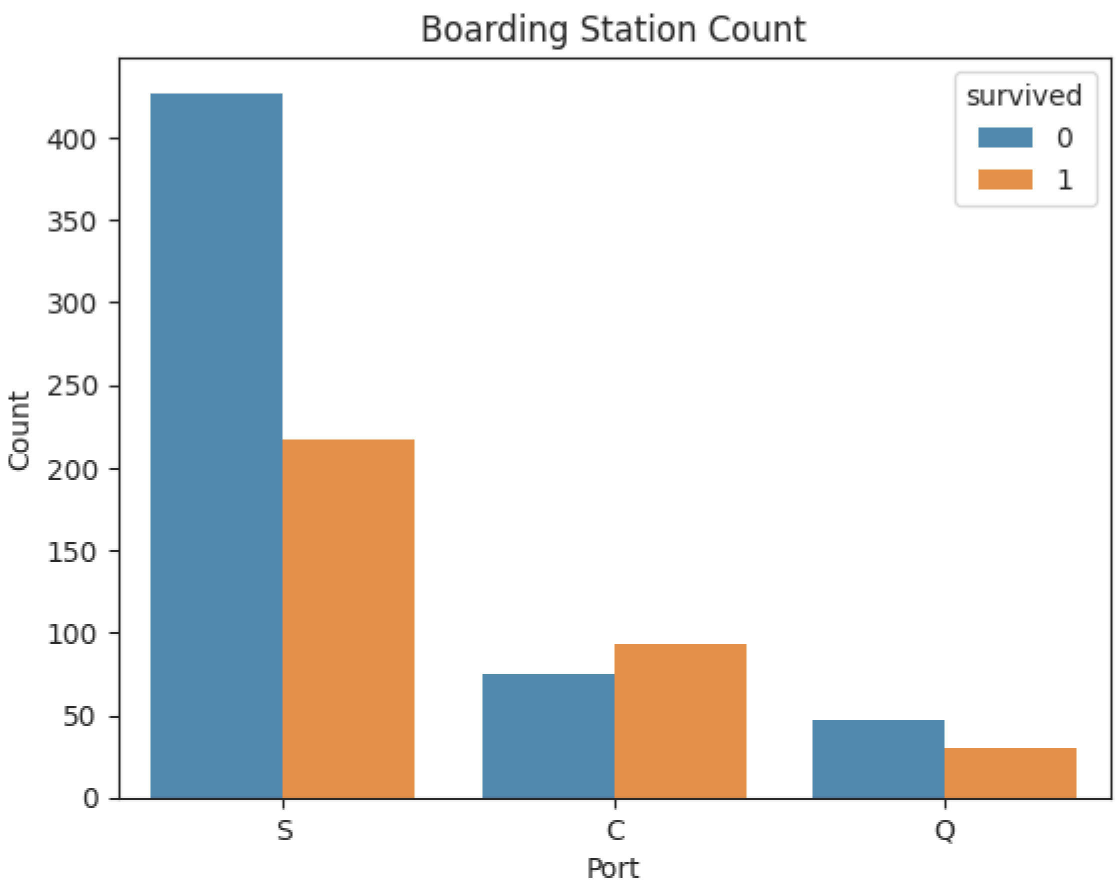


```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti

# Check if 'Embarked' exists in the DataFrame columns
if 'Embarked' not in data.columns:
    # If 'Embarked' is not found, check for 'embarked' with lowercase
    if 'embarked' in data.columns:
        print("Using column 'embarked' instead.")  # Print a message indicating the use of '
        sns.countplot(data['embarked'])  # Create a count plot using the 'embarked' column f
    else:
        # If neither 'Embarked' nor 'embarked' is found, print an error message
        print("Column 'Embarked' or similar not found. Check your DataFrame.")
else:
    # If 'Embarked' is found, create a count plot using the 'Embarked' column from the DataF
    sns.countplot(data['Embarked'])

# Set the title of the plot
plt.title("Boarding Station Count")
# Set the label for the x-axis
plt.xlabel("Port")
# Set the label for the y-axis
plt.ylabel("Count")
# Display the plot
plt.show()
```

```
Using column 'embarked' instead.
```



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

# Use 'survived' (lowercase) to match the actual column name in the DataFrame.
sns.countplot(x=data['embarked'], hue=data["survived"])  # 'x' sets the data for the x-axis

plt.title("Boarding Station Count")  # Set the title of the plot to "Boarding Station Count'
plt.xlabel("Port")  # Set the label for the x-axis to "Port".
plt.ylabel("Count")  # Set the label for the y-axis to "Count".
plt.show()  # Display the plot.
```



Double-click (or enter) to edit

Double-click (or enter) to edit

```
import seaborn as sns  # Import the seaborn library, which is used for data visualization.
import matplotlib.pyplot as plt  # Import the pyplot module from matplotlib, used for plotti
import pandas as pd  # Import the pandas library, which is used for data manipulation and ar

# Assuming 'data' is a DataFrame that contains the dataset
# Select only the numerical columns from the dataset
numerical_data = data.select_dtypes(include=['number'])

# Set the size of the plot
plt.figure(figsize=(12, 8))

# Create a heatmap to visualize the correlation matrix of the numerical data
ax = sns.heatmap(numerical_data.corr().abs(), annot=True)  # Calculate absolute correlation

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