Practical 5 : Using Python and appropriate statistical libraries, perform the given tasks

Name : *Deepak Kumar Bharti*

Roll No. : 25056765013

Group : *A*

Problem Statement

Using Python and appropriate statistical libraries, perform the following tasks :

Use Python packages such as `numpy` , `scipy.stats` , `scikit-learn` , `seaborn` and `matplotlib` .

---

1. Generation of Random Samples

Generate random samples (size ≥ 500) from each of the following distributions :

- Normal distribution
- Exponential distribution
- Uniform distribution
- Poisson distribution

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
from scipy import stats
import statsmodels.api as sm
```

In [2]:
```
np.random.seed(42)
n = 1000
```

In [3]:
```
# Generating samples from Normal Distribution :
normal_data = np.random.normal(loc=0, scale=1, size=n)
```

In [4]:
```
print("First 10 random samples of Normal Variates :")
print(normal_data[:10])
print("\nSize of the random samples of Normal Variates : ",len(normal_data))
```

```
First 10 random samples of Normal Variates :
[ 0.49671415 -0.1382643   0.64768854  1.52302986 -0.23415337 -0.23413696
  1.57921282  0.76743473 -0.46947439  0.54256004]

Size of the random samples of Normal Variates :  1000
```

In [5]:
```
# Generating samples from Exponential Distribution :
exponential_data = np.random.exponential(scale=1, size=n)
```

In [6]:
```
print("First 10 random samples of Exponential Variates :")
print(exponential_data[:10])
print("\nSize of the random samples of Exponential Variates :
",len(exponential_data))
```

```
First 10 random samples of Exponential Variates :
[0.18330114 0.11044882 1.01178411 1.22579494 0.03209575 2.75219405
 0.05337049 0.77935089 1.23464    2.04770355]

Size of the random samples of Exponential Variates :  1000
```

In [7]:
```
# Generating samples from Uniform Distribution
uniform_data = np.random.uniform(low=0, high=1, size=n)
```

In [8]:
```
print("First 10 random samples of Uniform Variates :")
print(uniform_data[:10])
print("\nSize of the random samples of Uniform Variates : ",len(uniform_data))
```

```
First 10 random samples of Uniform Variates :
[0.21906881 0.03672136 0.10802575 0.33886065 0.80258568 0.57204864
 0.51266773 0.29348882 0.93175372 0.39701504]

Size of the random samples of Uniform Variates :  1000
```

In [9]:
```
# Generating samples from UPoisson Distribution with Lembda = 3
poisson_data = np.random.poisson(lam=3, size=n)
```

In [10]:
```
print("First 10 random samples of poisson Variates :")
print(poisson_data[:10])
print("\nSize of the random samples of Poisson Variates : ",len(poisson_data))
```

```
First 10 random samples of poisson Variates :
[1 5 2 1 4 3 3 6 4 9]

Size of the random samples of Poisson Variates :  1000
```

---

2. Descriptive Analysis

Perform descriptive analysis for each data.

In [11]:
```
# Defining a function for Descriptive Analysis
def descriptive_stats(data,name):
    print(f"\n{name} Distribution")
    print("_"*30,"\n")
    print(f"Mean : {np.mean(data):.4f}")
    print(f"Median : {np.median(data):.4f}")
    print(f"Variance : {np.var(data):.4f}")
    print(f"Standard Deviation : {np.std(data):.4f}")
    print(f"Skewness : {stats.skew(data):.4f}")
    print(f"Kurtosis : {stats.kurtosis(data):.4f}")
```

In [12]:
```
# Descriptive Analysis for Normal Distribution
descriptive_stats(normal_data,"Normal")
```

```
Normal Distribution
_____________________________

Mean : 0.0193
Median : 0.0253
Variance : 0.9579
Standard Deviation : 0.9787
Skewness : 0.1168
Kurtosis : 0.0662
```

In [13]:
```
# Descriptive Analysis for Exponential Distribution
descriptive_stats(exponential_data,"Exponential")
```

```
Exponential Distribution
_____________________________

Mean : 1.0080
Median : 0.7259
Variance : 1.0050
Standard Deviation : 1.0025
Skewness : 1.9808
Kurtosis : 5.3794
```

In [14]:
```
# Descriptive Analysis for Uniform Distribution
descriptive_stats(uniform_data,"Uniform")
```

```
Uniform Distribution
_____________________________

Mean : 0.4945
Median : 0.4917
Variance : 0.0834
Standard Deviation : 0.2888
Skewness : 0.0096
Kurtosis : -1.1804
```

In [15]:
```
# Descriptive Analysis for Poisson Distribution
descriptive_stats(poisson_data,"Poisson")
```

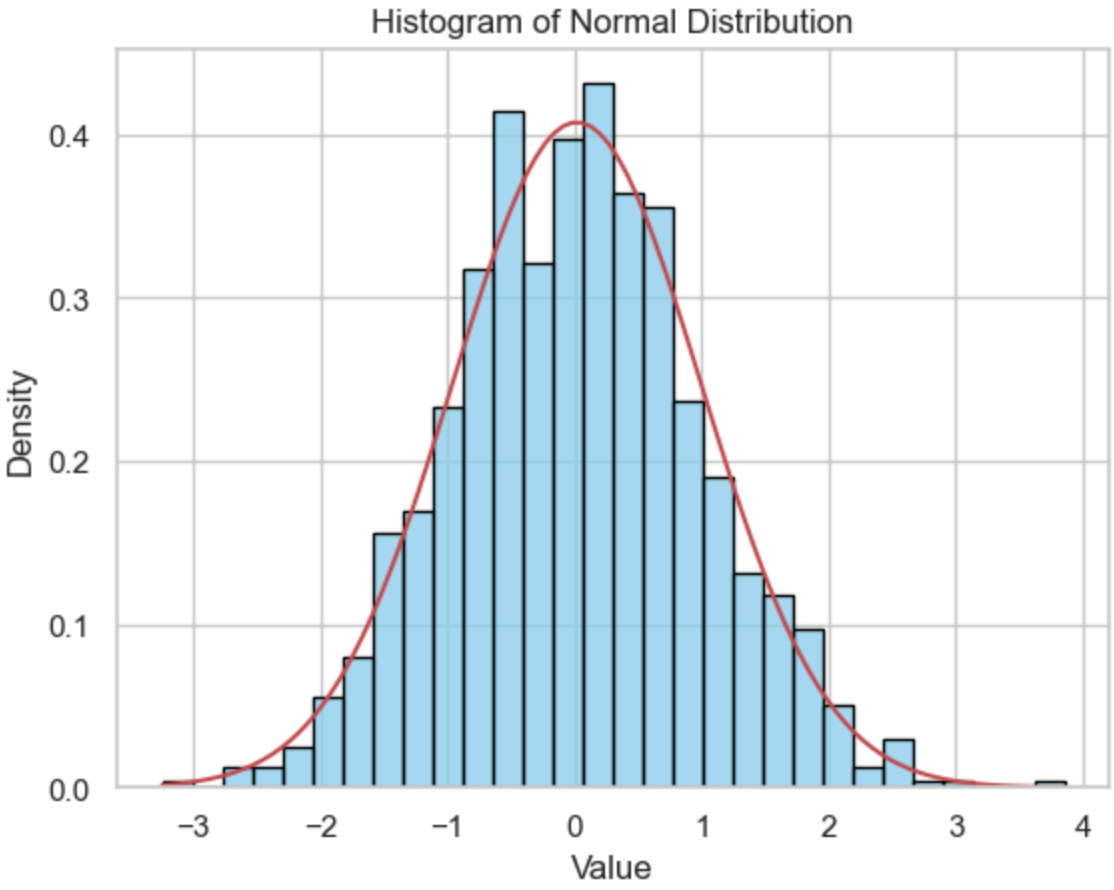
```
Poisson Distribution
_____________________________

Mean : 2.9820
Median : 3.0000
Variance : 2.6877
Standard Deviation : 1.6394
Skewness : 0.4428
Kurtosis : 0.1119
```

---

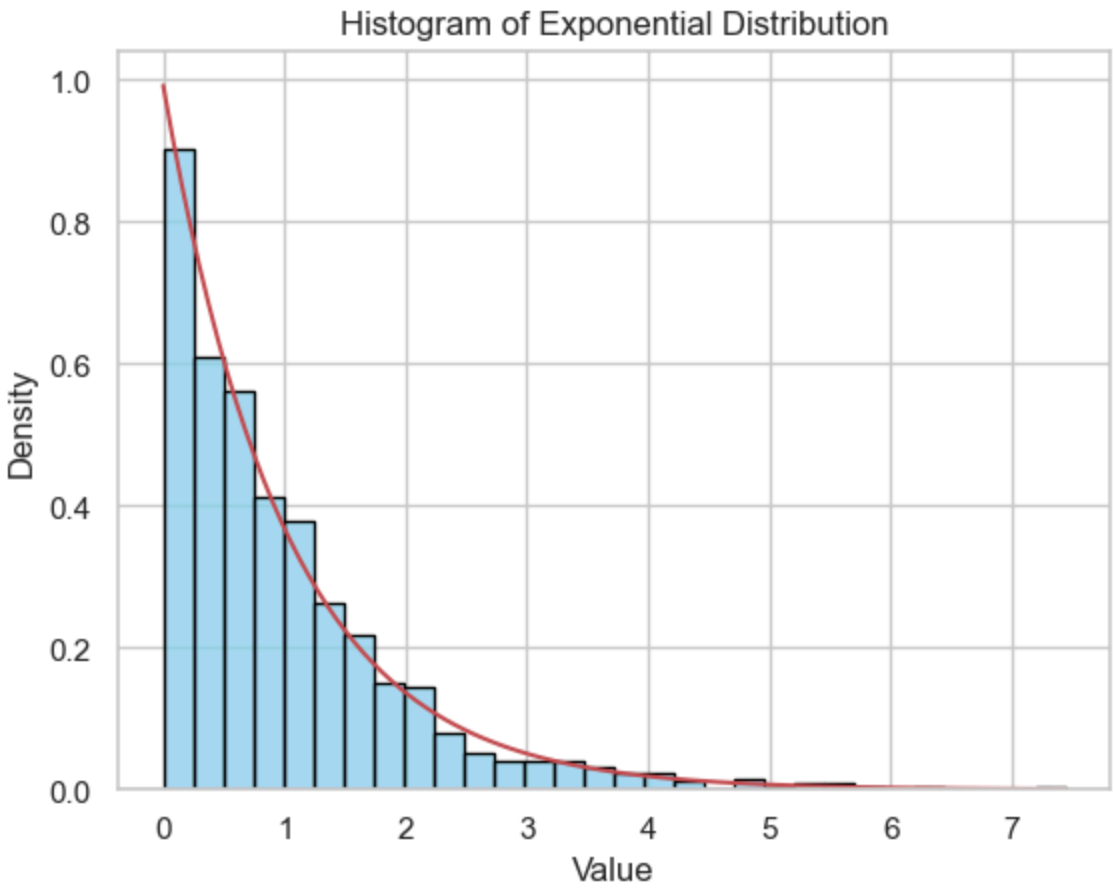
3. Histogram with Theoretical Density Curve

Plot **histograms with theoretical density curves** for each distribution.

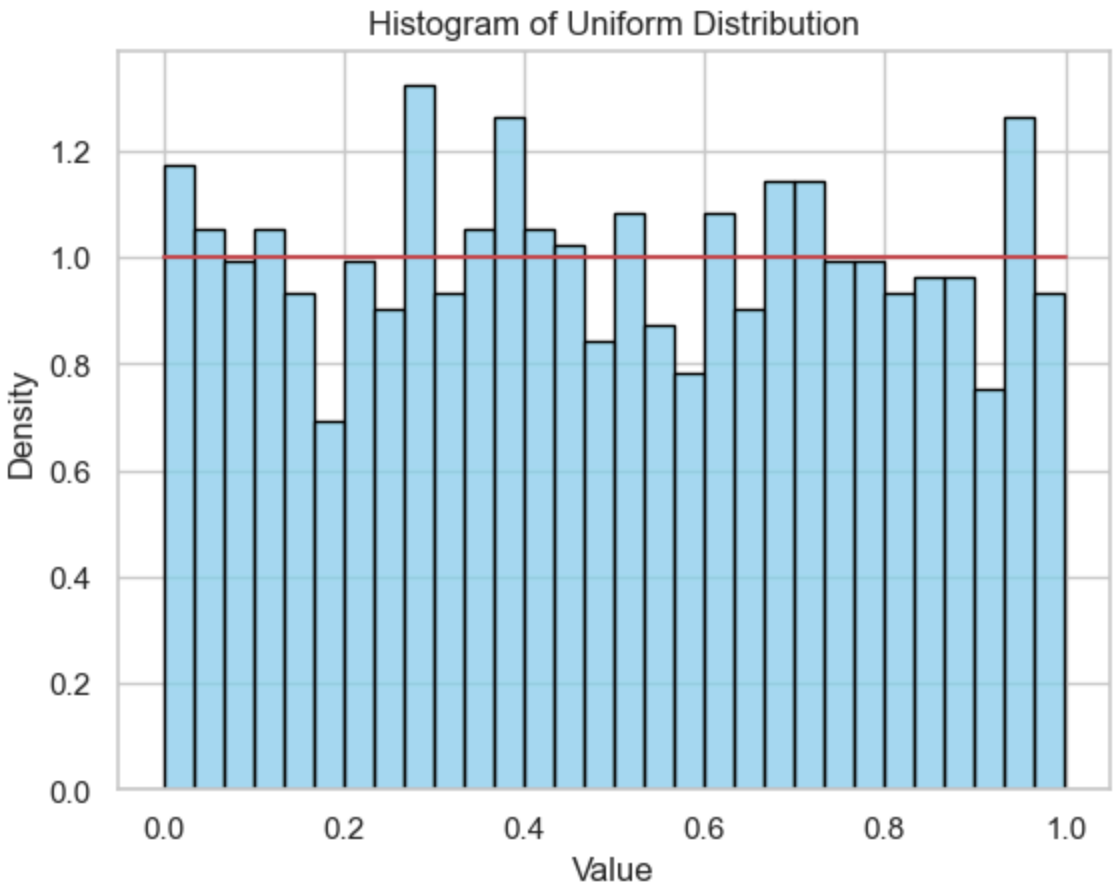
In [16]:
```
# Ploting histograms with theoretical density curves for Normal Distribution.
plt.figure()
sns.histplot(normal_data, bins=30, stat='density',
edgecolor="black",color='skyblue')

x = np.linspace(min(normal_data), max(normal_data), 100)
plt.plot(x, stats.norm.pdf(x, np.mean(normal_data), np.std(normal_data)), 'r')

plt.title("Histogram of Normal Distribution")
plt.xlabel("Value")
plt.ylabel("Density")
plt.show()
```



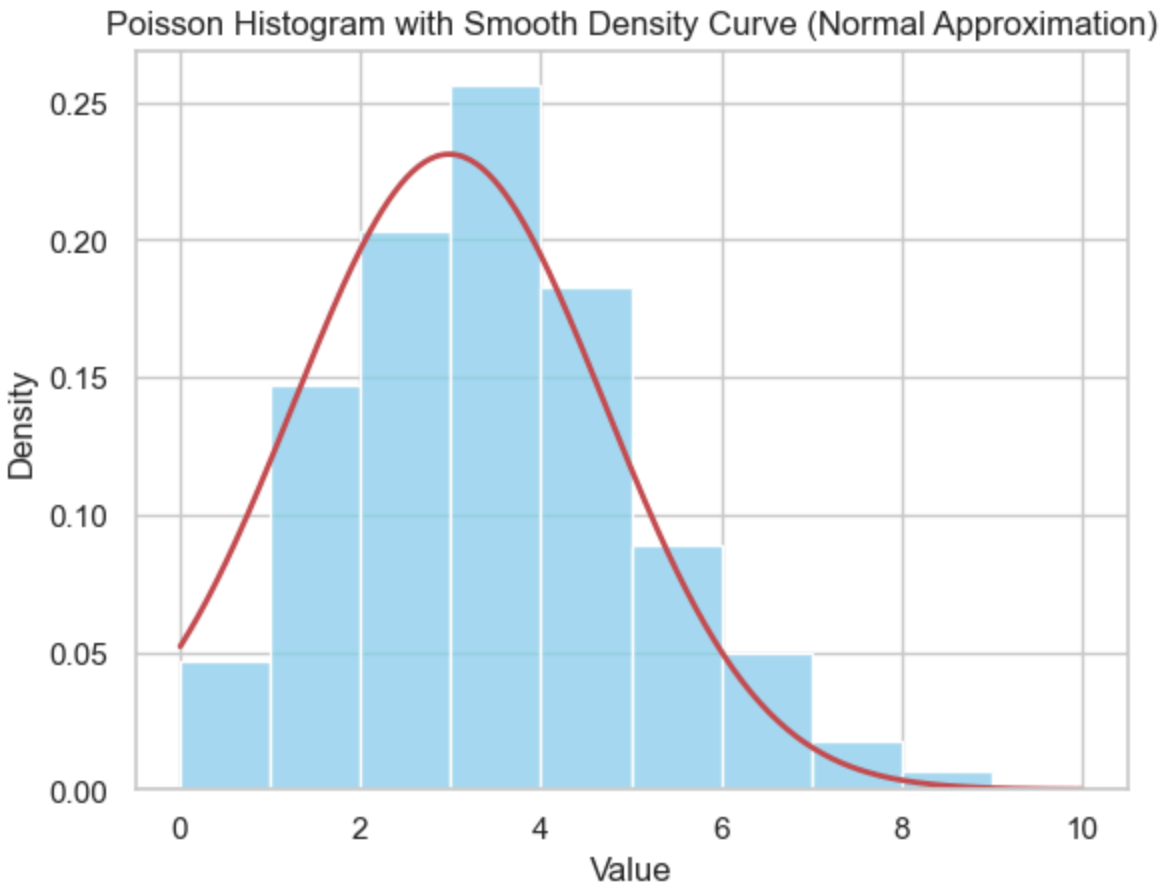
In [17]:
```
# Ploting histograms with theoretical density curves for Exponential Distribution.
plt.figure()
sns.histplot(exponential_data, bins=30,stat='density', edgecolor="black",
color='skyblue')

x = np.linspace(0, max(exponential_data), 100)
plt.plot(x, stats.expon.pdf(x, scale=np.mean(exponential_data)), 'r')

plt.title("Histogram of Exponential Distribution")
plt.xlabel("Value")
plt.ylabel("Density")
plt.show()
```



In [18]:
```
# Ploting histograms with theoretical density curves for Uniform Distribution.
plt.figure()
sns.histplot(uniform_data, bins=30, stat='density', edgecolor="black",
color='skyblue')

x = np.linspace(0, 1, 100)
plt.plot(x, stats.uniform.pdf(x, 0, 1), 'r')

plt.title("Histogram of Uniform Distribution")
plt.xlabel("Value")
plt.ylabel("Density")
plt.show()
```



In [19]:
```
# Ploting histograms with theoretical density curves for Poisson Distribution.
plt.figure()

# Histogram of Poisson data (density normalized)
sns.histplot(poisson_data, bins=range(poisson_data.max()+1), stat='density',
color='skyblue')

# Normal approximation curve
x = np.linspace(0, poisson_data.max()+1, 300)

mu = np.mean(poisson_data)
sigma = np.sqrt(mu)   # variance = mean for Poisson

plt.plot(x, stats.norm.pdf(x, mu, sigma), 'r', linewidth=2)

plt.title("Poisson Histogram with Smooth Density Curve (Normal Approximation)")
plt.xlabel("Value")
plt.ylabel("Density")
plt.show()
```



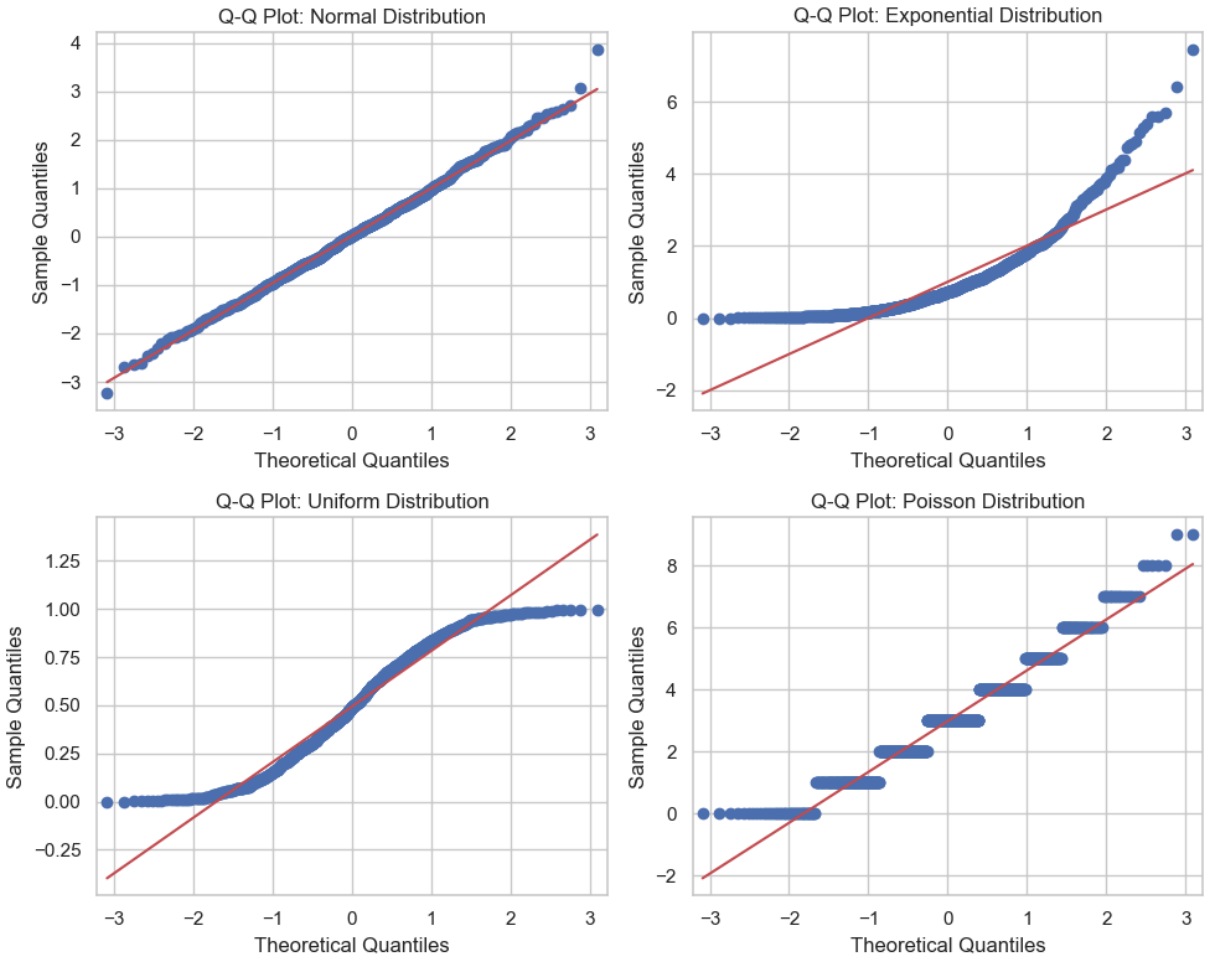
---

4. Q–Q Plots for Normality Assessment

Construct Q–Q plots to assess normality of each sample.

In [20]:
```
# Constructing Q-Q plots to assess normality of each sample.

fig, axes = plt.subplots(2, 2, figsize=(10, 8))

# Q-Q plot for Normal distribution
sm.qqplot(normal_data, line='s', ax=axes[0, 0])
axes[0, 0].set_title("Q-Q Plot: Normal Distribution")

# Q-Q plot for Exponential distribution
sm.qqplot(exponential_data, line='s', ax=axes[0, 1])
axes[0, 1].set_title("Q-Q Plot: Exponential Distribution")

# Q-Q plot for Uniform distribution
sm.qqplot(uniform_data, line='s', ax=axes[1, 0])
axes[1, 0].set_title("Q-Q Plot: Uniform Distribution")

# Q-Q plot for Poisson distribution
sm.qqplot(poisson_data, line='s', ax=axes[1, 1])
axes[1, 1].set_title("Q-Q Plot: Poisson Distribution")

# Adjust layout to avoid overlapping
plt.tight_layout()
```

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



---

5. Interpretation of Q–Q Plots

Comment on how Q–Q plots differ for:

- Symmetric vs Skewed distributions
- Light-tailed vs Heavy-tailed distributions

Answer :5 - Interpretation of Q–Q Plots

1. Symmetric vs Skewed Distributions (Q–Q Plot Behavior)

Symmetric Distribution (Normal)

(Top-left plot: Normal distribution)

- Points lie close to the straight reference line
- Deviations (if any) are balanced on both sides
- No systematic curvature

Interpretation :
The data is symmetric and closely follows a normal distribution.

Skewed Distribution (Exponential & Poisson)

(Top-right: Exponential, Bottom-right: Poisson)

- Points show systematic curvature
- For right-skewed distributions:
  - Left tail lies above the line
  - Right tail lies far above the line
- Points are not symmetric around the line

Interpretation :
Skewness causes the Q–Q plot to bend away from the reference line, indicating non-normality.

Uniform Distribution

(Bottom-left plot)

- Shows an S-shaped pattern
- Flat tails due to bounded support
- Middle portion near the line, ends deviate

Interpretation :
Uniform data is symmetric but not bell-shaped, hence does not follow normality.

---

2. Light-Tailed vs Heavy-Tailed Distributions

Light-Tailed Distributions (Uniform)

(Bottom-left plot)

- Points bend towards the line at the center
- Ends deviate inward
- Indicates fewer extreme values than normal

Interpretation :
Light-tailed distributions produce shorter extremes compared to a normal distribution.

Heavy-Tailed Distributions (Exponential, Poisson)

(Top-right & Bottom-right plots)

- Points deviate outward at the ends
- Extreme points lie far from the line

- Tails spread more than normal

Interpretation :

Heavy-tailed distributions indicate a higher probability of extreme values.

Summary Table

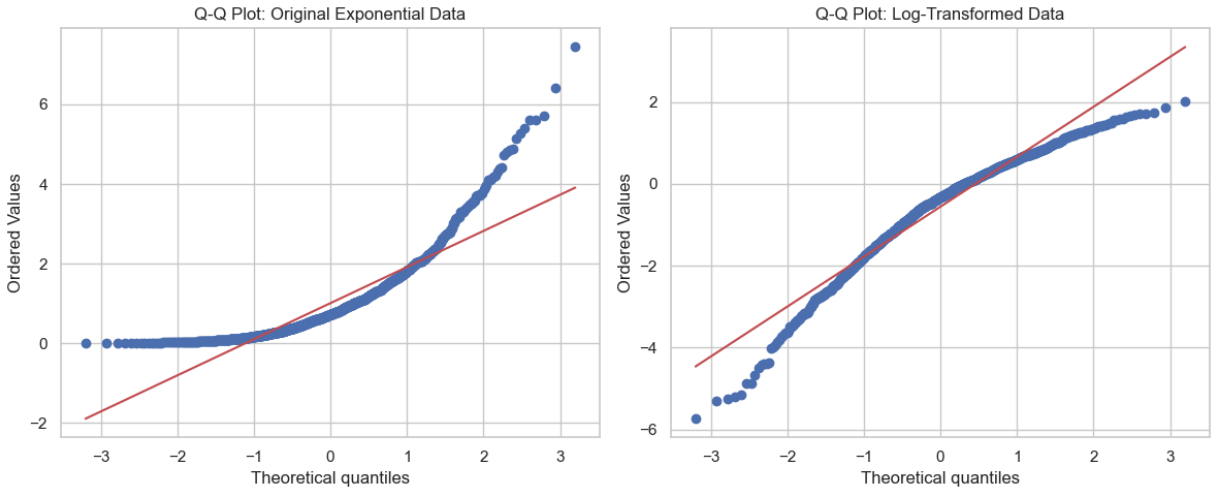
Property	Q–Q Plot Shape
Symmetric	Straight line
Right-skewed	Upward curvature
Left-skewed	Downward curvature
Light-tailed	Inward bending at ends
Heavy-tailed	Outward spread at ends

---

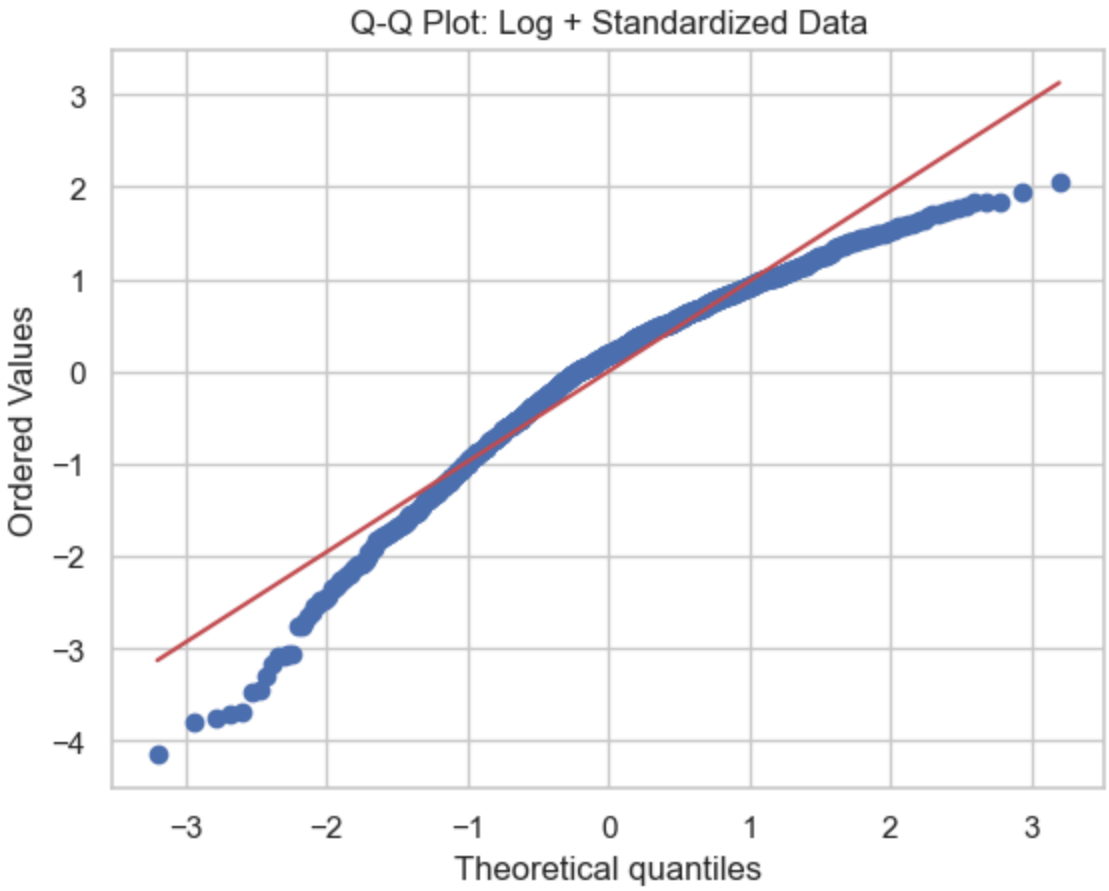
6. Data Transformation and Re-examination

Introduce standardization or transformation techniques (e.g., log or square-root) for a skewed distribution and re-examine normality using a Q–Q plot.

In [21]:
```
# Log transformation of Exponential Distribution
data_log = np.log(exponential_data)

# Create Q-Q plots
plt.figure(figsize=(12, 5))

# Original data
plt.subplot(1, 2, 1)
stats.probplot(exponential_data, dist="norm", plot=plt)
plt.title("Q-Q Plot: Original Exponential Data")

# Transformed data
plt.subplot(1, 2, 2)
stats.probplot(data_log, dist="norm", plot=plt)
plt.title("Q-Q Plot: Log-Transformed Data")

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



In [22]:
```
# Standardization after transformation
data_std = (data_log - np.mean(data_log)) / np.std(data_log)

# Q-Q plot after standardization
stats.probplot(data_std, dist="norm", plot=plt)
plt.title("Q-Q Plot: Log + Standardized Data")
plt.show()
```

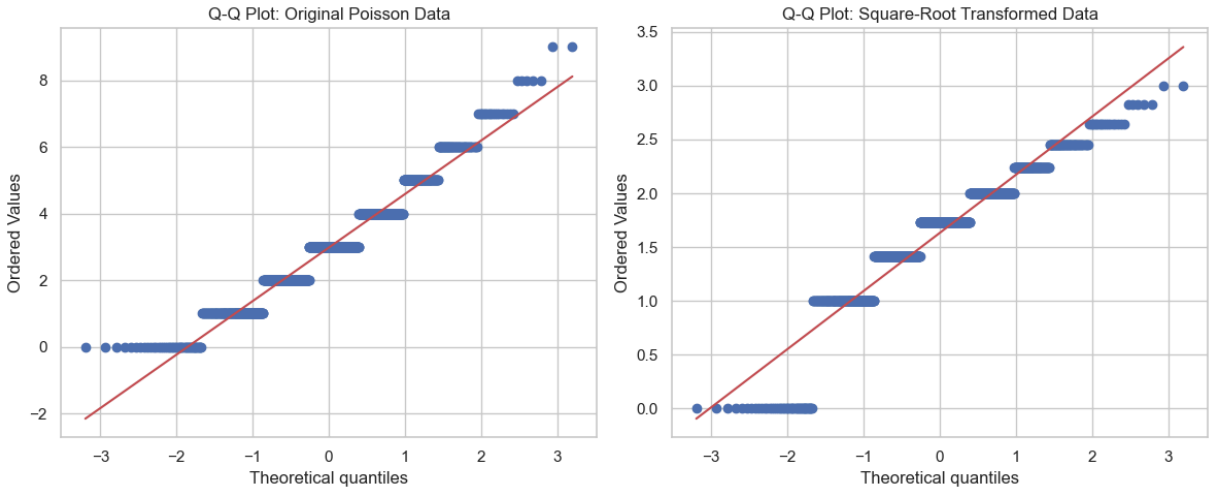


Interpretation

- **Before :** Strong upward curvature → non-normal
- **After log transform :** Points lie much closer to the straight line

- Normality significantly improved

In [23]:
```
# Square-root transformation of Poisson Distribution
data_sqrt = np.sqrt(poisson_data)

# Create Q-Q plots
plt.figure(figsize=(12, 5))

# Original data
plt.subplot(1, 2, 1)
stats.probplot(poisson_data, dist="norm", plot=plt)
plt.title("Q-Q Plot: Original Poisson Data")

# Transformed data
plt.subplot(1, 2, 2)
stats.probplot(data_sqrt, dist="norm", plot=plt)
plt.title("Q-Q Plot: Square-Root Transformed Data")

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



Interpretation

- **Before :** Step-like pattern + curvature (discrete + skewed)
- **After sqrt transform :** Much closer to linear trend
- Approximate normality achieved