Question 1

The problem provides us to consider $L \sim N(2, 9)$, to compute the theoretical value at risk and expected shortfall for a given α of L, we use the following formulas:

$$\text{VaR}_\alpha(L) = \mu + \sigma\Phi^{-1}(\alpha) \tag{1}$$

$$\text{ES}_\alpha(L) = \mu + \sigma\frac{\Phi(\Phi^{-1}(\alpha))}{1-\alpha} \tag{2}$$

1)

The theoretical value of $\text{VaR}_{0.99(L)}$ using Equation 1 where we are given $\mu = 2$, $\sigma = 3$ and $\alpha = 0.99$:

$$\begin{aligned}\text{VaR}_{0.99}(L) &= 2 + 3\Phi^{-1}(0.99) \\ &= 8.98\end{aligned} \tag{3}$$

The theorical value of $\text{ES}_{0.99(L)}$ using Equation 2 with the mentioned given values is:

$$\begin{aligned}\text{ES}_{0.99}(L) &= 2 + 3\frac{\Phi(\Phi^{-1}(0.99))}{1-0.99} \\ &= 10.00\end{aligned} \tag{4}$$

2-3) (a-c)

All of these parts are done at the same time in our Python program, which will have comments to show where each part is completed. The following content is `q1.py`.

```python
from scipy.stats import norm
import numpy as np
from scipy.stats import shapiro, kstest
import matplotlib.pyplot as plt

# Given parameters
mu = 2
sigma = 3
alpha = 0.99
# we will loop through all sample sizes provided 2-3)
sample_sizes = [10**6, 10**7, 10**8]

# Theoretical values as calculated in 1)
VaR99_theo = mu + sigma * norm.ppf(alpha)
ES99_theo = mu + sigma * (norm.pdf(norm.ppf(alpha)) / (1 - alpha))

# Function to compute empirical VaR and ES
# As defined in Lecture 5, the empiricial Var
# is defined by the 99th percentile of the sample
# The empirical ES is defined by the average of the values
# greater than or equal to the empirical Var
def empirical_var_es(data, alpha=0.99):
    var = np.percentile(data, alpha * 100)
    es = np.mean(data[data >= var])
    return var, es

# Run simulations
results = {}
for N in sample_sizes:
    np.random.seed(42)  # For reproducibility
    sample = np.random.normal(mu, sigma, N)

    # a) Standardizing the data and Testing for normality
    # Standardize the data
    mean = np.mean(sample)
    std = np.std(sample)
    standardized_sample = (sample - mean) / std

    # Test for normality
    stat1, p_value1 = kstest(standardized_sample, 'norm')
    stat2, p_value2 = shapiro(standardized_sample[:5000])  # Using subset due to Shapiro limit

    is_normal = p_value1 > 0.05 and p_value2 > 0.05
```

```
    # Parametric estimates (if normal)
    if is_normal:
        VaR99_param = mean + std * norm.ppf(alpha)
        ES99_param = mean + std * (norm.pdf(norm.ppf(alpha)) / (1 - alpha))
    else:
        VaR99_param = None
        ES99_param = None

    # b) Empirical estimates
    # Empirical estimates
    VaR99_emp, ES99_emp = empirical_var_es(sample, alpha)

    # c) Compute the absolute errors and compare with true values in the results
    # Compute absolute errors
    abs_error_parametric = (abs(VaR99_param - VaR99_theo), abs(ES99_param - ES99_theo)) if is_normal
else (None, None)
    abs_error_param_perc = (abs_error_parametric[0] / VaR99_theo)*100, (abs_error_parametric[1] /
ES99_theo)*100 if is_normal else (None, None)
    abs_error_empirical = (abs(VaR99_emp - VaR99_theo), abs(ES99_emp - ES99_theo))
    abs_error_empirical_perc = (abs_error_empirical[0] / VaR99_theo)*100, (abs_error_empirical[1] /
ES99_theo)*100

    # Store results
    results[N] = {
        "data": sample,
        "VaR99_param": VaR99_param,
        "ES99_param": ES99_param,
        "VaR99_emp": VaR99_emp,
        "ES99_emp": ES99_emp,
        "abs_error_parametric": abs_error_parametric,
        "abs_error_empirical": abs_error_empirical,
        "abs_error_empirical_perc": abs_error_empirical_perc,
        "abs_error_param_perc": abs_error_param_perc,
        "is_normal": is_normal
    }

# Print results
for N, res in results.items():
    print(f"N = {N}")
    print(f"  Normality confirmed: {res['is_normal']}")
    print(f"  VaR_99 Parametric: {res['VaR99_param']:.4}, Abs Error: {res['abs_error_parametric']
[0]:.2e}")
    print(f"  ES_99 Parametric: {res['ES99_param']:.4}, Abs Error: {res['abs_error_parametric']
[1]:.2e}")
    print(f"  VaR_99 Empirical: {res['VaR99_emp']:.4}, Abs Error: {res['abs_error_empirical'][0]:.2e}")
    print(f"  ES_99 Empirical: {res['ES99_emp']:.4}, Abs Error: {res['abs_error_empirical'][1]:.2e}")
    print("-")
    print("Percentages:")
    print(f"  VaR_99 Parametric: {res['abs_error_param_perc'][0]:.2%}")
    print(f"  ES_99 Parametric: {res['abs_error_param_perc'][1]:.2%}")
    print(f"  VaR_99 Empirical: {res['abs_error_empirical_perc'][0]:.2%}")
    print(f"  ES_99 Empirical: {res['abs_error_empirical_perc'][1]:.2%}")

    # Normality check with histograms
    sample = results[N]['data']
    # Create the plot
    plt.figure(figsize=(8, 4))
    plt.hist(sample, bins=30, density=True, alpha=0.6, color='skyblue')
    plt.title(f'Normality Check for N={N}')
    plt.xlabel('Value')
    plt.ylabel('Density')
    plt.grid(True)
    plt.tight_layout()
    # Save the plot first
    plt.savefig(f'normality_check_{N}.png', dpi=300, bbox_inches='tight')
    # Show the plot
    plt.show()
    plt.close()
```

When we run the program (`$ python3 q1.py`), we get the following results which were reorganized in a table for a better reading experience.

Sample Size	Normality Confirmed	Parametric	Abs Error Parametric	Parametric % Error	Empirical	Abs Error Empirical	Empirical % Error
1000000	True	8.976	3.49e-03	3.88%	8.972	6.63e-03	7.38%
10000000	True	8.979	9.71e-05	0.11%	8.978	1.21e-03	1.34%
100000000	True	8.979	7.04e-05	0.08%	8.979	3.90e-04	0.43%

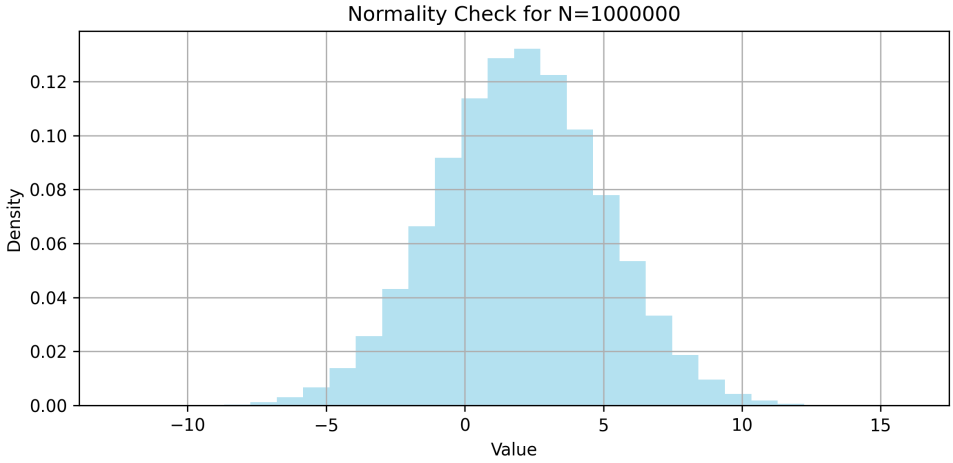
Table 1: VaR_{99} Results for all sample sizes

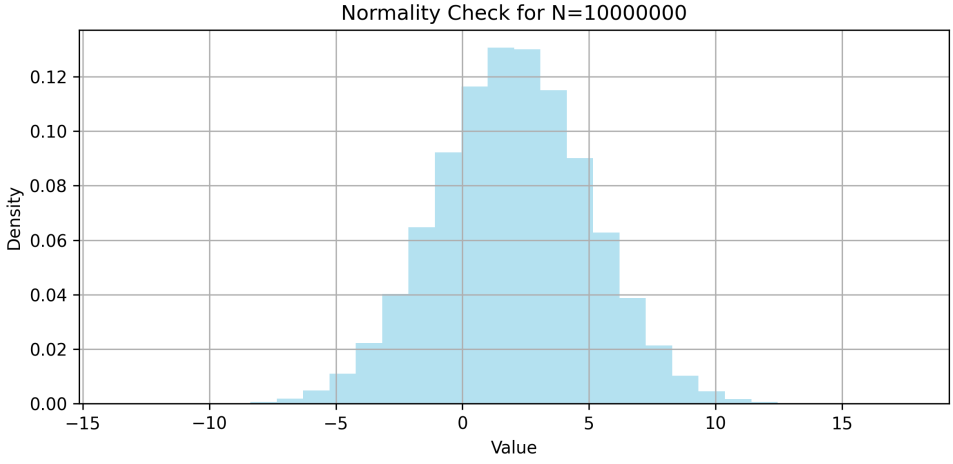
Sample Size	Normality Confirmed	ES_99 Parametric	Abs Error ES_99 Parametric	ES_99 Parametric % Error	ES_99 Empirical	Abs Error ES_99 Empirical	ES_99 Empirical % Error
1000000	True	9.992	3.30e-03	3.30%	9.982	1.36e-02	13.63%
10000000	True	9.996	8.33e-05	0.08%	9.994	1.68e-03	1.68%
100000000	True	9.996	1.44e-04	0.14%	9.996	1.39e-04	0.14%

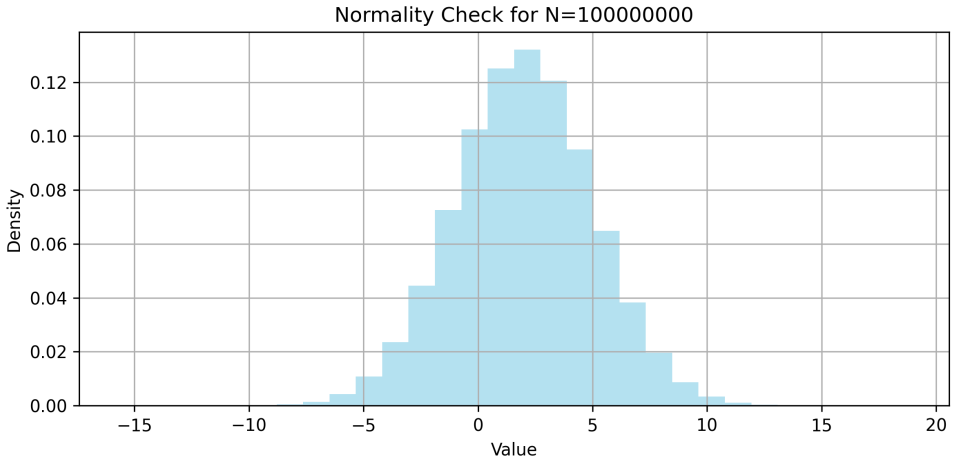
Table 2: ES_{99} Results for all sample sizes

From the results from both tables of VaR and ES respectively, we can notice that generally (apart for ES_{99} errors from second sample size to third for parametic) as the sample sizes increases the error decreases, thus showing better accuracy. As shown in the code we used both the Shapiro Wilk test and the kstest (which is better for large sample sizes like ours) both are used to determine if the distribution is normal from our hypothesis test, checking if the p-value for both test is greater than 0.05 for which both did.

We can also notice this in the following histograms:







Question 2

For question 2 we consider the distribution where $L \sim \text{Exp}(4)$.

1)

To compute the theoretical values for value at risk and expected shortfall for an exponential distrubution with α and λ, we have the following formulas:

$$\text{VaR}_\alpha(L) = -\frac{\ln(1-\alpha)}{\lambda} \tag{5}$$

$$\text{ES}_\alpha(L) = \frac{1-\ln(1-\alpha)}{\lambda} \tag{6}$$

From the question we are given $\alpha = 0.99$ and $\lambda = 4$ which will be plugged into Equation 5 and Equation 6 for value at risk and expected shortfall respectively:

$$\begin{aligned}\text{VaR}_{0.99}(L) &= -\frac{\ln(1-0.99)}{4} \\ &= 1.15\end{aligned} \tag{7}$$

$$\begin{aligned}\text{ES}_\alpha(L) &= \frac{1-\ln(1-0.99)}{4} \\ &= 1.40\end{aligned} \tag{8}$$

2-3) (a-c)

All of these parts are done in the program `q2.py` where each part is commented. Compared to the first question the code has been much more organized.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import norm

def theoretical_var_es_exponential(lambda_param, alpha):
    """Calculate theoretical VaR and ES for exponential distribution"""
    var = -np.log(1 - alpha) / lambda_param
    es = var + 1/lambda_param
    return var, es

# 2/3 a)
def parametric_normal_var_es(data, alpha):
    """Calculate VaR and ES assuming normal distribution"""
    mu = np.mean(data)
    sigma = np.std(data)
    var = norm.ppf(alpha, mu, sigma)
    es = mu + sigma * norm.pdf(norm.ppf(alpha)) / (1 - alpha)
    return var, es
# 2/3 b)
def empirical_var_es(data, alpha):
```

```
    """Calculate empirical VaR and ES"""
    var = np.percentile(data, alpha * 100)
    es = np.mean(data[data > var])
    return var, es

def run_analysis(N, lambda_param=4, alpha=0.99):
    """Run complete analysis for a given sample size"""
    # Generate exponential data
    np.random.seed(42)  # For reproducibility
    data = np.random.exponential(scale=1/lambda_param, size=N)

    # Get theoretical values
    theo_var, theo_es = theoretical_var_es_exponential(lambda_param, alpha)

    # Get parametric estimates (assuming normal)
    param_var, param_es = parametric_normal_var_es(data, alpha)

    # Get empirical estimates
    emp_var, emp_es = empirical_var_es(data, alpha)

    # 2/3 c)
    # Calculate absolute errors
    # We have also calculated the absolute percentage errors as well
    param_var_error = abs(param_var - theo_var)
    param_es_error = abs(param_es - theo_es)
    emp_var_error = abs(emp_var - theo_var)
    emp_es_error = abs(emp_es - theo_es)
    param_var_error_perc = (param_var_error / theo_var) * 100
    param_es_error_perc = (param_es_error / theo_es) * 100
    emp_var_error_perc = (emp_var_error / theo_var) * 100
    emp_es_error_perc = (emp_es_error / theo_es) * 100

    results = {
        'Sample Size': N,
        'Theoretical VaR': theo_var,
        'Theoretical ES': theo_es,
        'Parametric VaR': param_var,
        'Parametric ES': param_es,
        'Empirical VaR': emp_var,
        'Empirical ES': emp_es,
        'Parametric VaR Error': param_var_error,
        'Parametric ES Error': param_es_error,
        'Empirical VaR Error': emp_var_error,
        'Empirical ES Error': emp_es_error,
        'Parametric VaR Error (%)': param_var_error_perc,
        'Parametric ES Error (%)': param_es_error_perc,
        'Empirical VaR Error (%)': emp_var_error_perc,
        'Empirical ES Error (%)': emp_es_error_perc
    }

# Plotting the results
    plt.figure(figsize=(10, 6))
    plt.hist(data, bins=30, density=True, alpha=0.6, color='skyblue', label='Data Distribution')

    plt.title(f"Distribution with VaR and ES Estimates (N={N})")
    plt.xlabel('Value')
    plt.ylabel('Density')
    plt.legend()
    plt.tight_layout()
    plt.savefig(f'var_es_plots_N_{N}.png', dpi=300, bbox_inches='tight')
    #plt.show()
    plt.close()

    return pd.DataFrame([results])

# Completing 2/3 samples sizes
# Run analysis for the different sample sizes
sample_sizes = [10**6, 10**7, 10**8]
results = pd.DataFrame()
```

```
for N in sample_sizes:
    result = run_analysis(N)
    results = pd.concat([results, result], ignore_index=True)

print("\nResults for all sample sizes:")
print(results.to_string())
```

When we run the program (`$ python3 q2.py`), we get the following results which were reorganized in a table for a better reading experience.

Sample Size	Theoretical VaR	Parametric VaR	Empirical VaR	Parametric VaR Error	Empirical VaR Error	Parametric VaR Error (%)	Empirical VaR Error (%)
1000000	1.151293	0.832468	1.152163	0.318824	0.000871	27.692730	0.075632
10000000	1.151293	0.831392	1.150999	0.319901	0.000293	27.786243	0.025478
100000000	1.151293	0.831571	1.151576	0.319722	0.000283	27.770701	0.024617

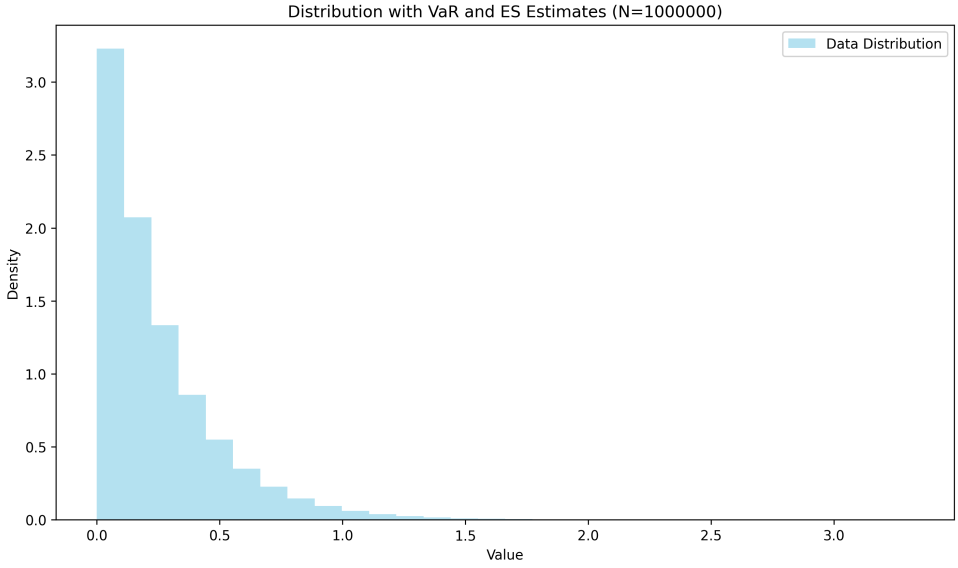
Table 3: VaR_{99} Results for all sample sizes

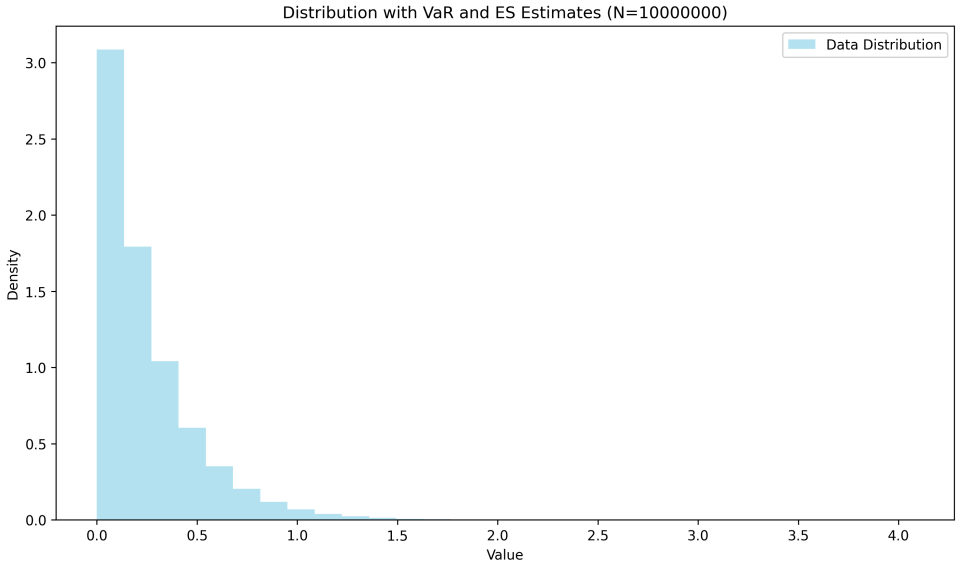
Sample Size	Theoretical ES	Parametric ES	Empirical ES	Parametric ES Error	Empirical ES Error	Parametric ES Error (%)	Empirical ES Error (%)
1000000	1.401293	0.917273	1.402837	0.484020	0.001544	34.540963	0.110205
10000000	1.401293	0.916084	1.400674	0.485209	0.000619	34.625813	0.044141
100000000	1.401293	0.916288	1.401702	0.485005	0.000409	34.611226	0.029218

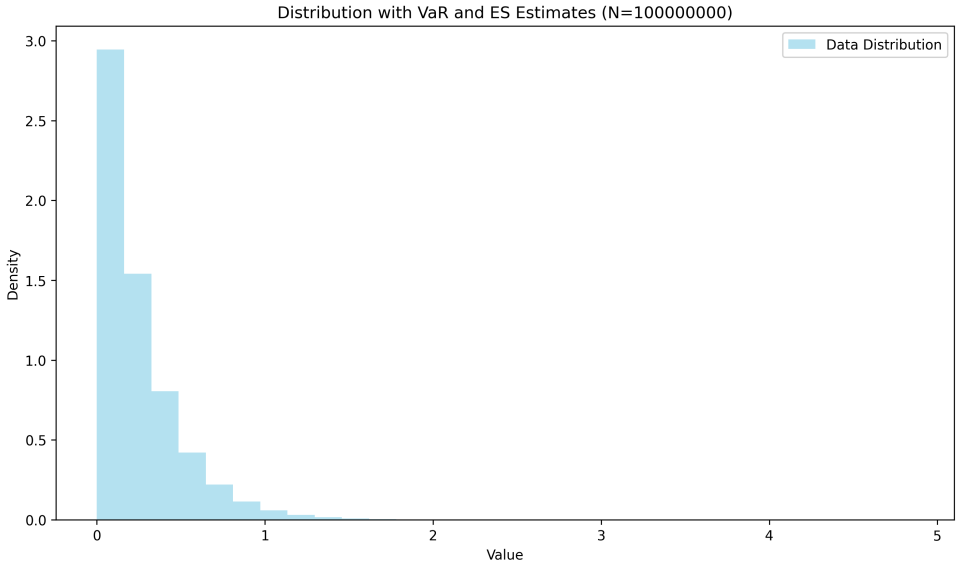
Table 4: ES_{99} Results for all sample sizes

From our result we are able to see in this case the empirical results do much better compared to the parametric, the opposite from question 1. From there we are able to see that the empirical results' errors go down as each sample size increases thus increasing the accuracy while the parametrics' errors doesn't move much as sample size increases. The same can also be said for Expected Shortfall as well.

To be able to view how the distribution looks like, consider the following histograms, which show that the distributions are exponential.




Distribution with VaR and ES Estimates (N=10000000)
Data Distribution
Density
Value
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Distribution with VaR and ES Estimates (N=100000000)
Data Distribution
Density
Value
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
1
2
3
4
5