In [1]:
```
#TASK1
import pandas as pd
data = pd.read_csv('titanic.csv')
mean_age = data['Age'].mean()
print("Mean Age of Passengers:", round(mean_age, 2))
```

```
Mean Age of Passengers: 29.7
```

In [2]:
```
import pandas as pd
data = pd.read_csv('titanic.csv')
median_fare = data['Fare'].median()
print("Median Fare Paid by Passengers:", round(median_fare, 2))
```

```
Median Fare Paid by Passengers: 14.45
```

In [3]:
```
import pandas as pd
data = pd.read_csv('titanic.csv')
common_embarkation = data['Embarked'].mode()[0]
print("Most Common Embarkation Point:", common_embarkation)
```

```
Most Common Embarkation Point: S
```

In [4]:
```
import pandas as pd
data = pd.read_csv('titanic.csv')
for pclass in sorted(data['Pclass'].unique()):
    fares = data[data['Pclass'] == pclass]['Fare']
    mean = fares.mean()
    median = fares.median()
    mode = fares.mode()[0]

    print(f"Pclass {pclass}:")
    print(f"  Mean Fare: {mean:.2f}")
    print(f"  Median Fare: {median:.2f}")
    print(f"  Mode Fare: {mode:.2f}\n")
```

```
Pclass 1:
  Mean Fare: 84.15
  Median Fare: 60.29
  Mode Fare: 26.55

Pclass 2:
  Mean Fare: 20.66
  Median Fare: 14.25
  Mode Fare: 13.00

Pclass 3:
  Mean Fare: 13.68
  Median Fare: 8.05
  Mode Fare: 8.05
```

In [5]:
```
import pandas as pd
data = pd.read_csv('titanic.csv')
mean_sibsp = data['SibSp'].mean()
median_sibsp = data['SibSp'].median()
print(f"Mean number of siblings/spouses aboard: {mean_sibsp:.2f}")
print(f"Median number of siblings/spouses aboard: {median_sibsp}")
```

```
Mean number of siblings/spouses aboard: 0.52
Median number of siblings/spouses aboard: 0.0
```

In [6]:
```
import pandas as pd
data = pd.read_csv('titanic.csv')
fare_skewness = data['Fare'].skew()
print(f"Skewness of Fare: {fare_skewness:.2f}")

if fare_skewness > 0:
    print("The Fare distribution is positively skewed (right-skewed).")
elif fare_skewness < 0:
    print("The Fare distribution is negatively skewed (left-skewed).")
else:
    print("The Fare distribution is symmetric.")
```

```
Skewness of Fare: 4.79
The Fare distribution is positively skewed (right-skewed).
```

In [7]:
```
import pandas as pd
data = pd.read_csv('titanic.csv')
age_kurtosis = data['Age'].kurt()
print(f"Kurtosis of Age: {age_kurtosis:.2f}")

if age_kurtosis > 0:
    print("The distribution is leptokurtic (peaked with heavy tails).")
elif age_kurtosis < 0:
    print("The distribution is platykurtic (flatter with light tails).")
else:
    print("The distribution is mesokurtic (normal-like).")
```

```
Kurtosis of Age: 0.18
The distribution is leptokurtic (peaked with heavy tails).
```

In [8]:
```
import pandas as pd
data = pd.read_csv('titanic.csv')
parch_skewness = data['Parch'].skew()
print(f"Skewness of Parch: {parch_skewness:.2f}")

if abs(parch_skewness) < 0.5:
    print("The Parch data is approximately symmetric.")
elif parch_skewness > 0:
    print("The Parch data is positively skewed (right-skewed).")
else:
    print("The Parch data is negatively skewed (left-skewed).")
```

```
Skewness of Parch: 2.75
The Parch data is positively skewed (right-skewed).
```

In [9]:
```
import pandas as pd
data = pd.read_csv('titanic.csv')
survived_skewness = data['Survived'].skew()
survived_kurtosis = data['Survived'].kurt()
print(f"Skewness of Survived: {survived_skewness:.2f}")
print(f"Kurtosis of Survived: {survived_kurtosis:.2f}")

if abs(survived_skewness) < 0.5:
    print("The 'Survived' data is approximately symmetric.")
elif survived_skewness > 0:
    print("The 'Survived' data is positively skewed.")
else:
    print("The 'Survived' data is negatively skewed.")

if survived_kurtosis > 0:
    print("The 'Survived' data is leptokurtic (sharper peak).")
```

```
elif survived_kurtosis < 0:
    print("The 'Survived' data is platykurtic (flatter).")
else:
    print("The 'Survived' data is mesokurtic (normal-like).")
```

```
Skewness of Survived: 0.48
Kurtosis of Survived: -1.78
The 'Survived' data is approximately symmetric.
The 'Survived' data is platykurtic (flatter).
```

In [10]:

```
import pandas as pd
data = pd.read_csv('titanic.csv')
fare_skew = data['Fare'].skew()
age_skew = data['Age'].skew()
fare_kurt = data['Fare'].kurt()
age_kurt = data['Age'].kurt()

print(f"Fare - Skewness: {fare_skew:.2f}, Kurtosis: {fare_kurt:.2f}")
print(f"Age  - Skewness: {age_skew:.2f}, Kurtosis: {age_kurt:.2f}")

if abs(fare_skew) > abs(age_skew):
    print("👉 'Fare' has more skewness (more asymmetry).")
else:
    print("👉 'Age' has more skewness (more asymmetry).")

if fare_kurt > age_kurt:
    print("👉 'Fare' has more kurtosis (more extreme outliers).")
else:
    print("👉 'Age' has more kurtosis (more extreme outliers).")
```

```
Fare - Skewness: 4.79, Kurtosis: 33.40
Age  - Skewness: 0.39, Kurtosis: 0.18
👉 'Fare' has more skewness (more asymmetry).
👉 'Fare' has more kurtosis (more extreme outliers).
```

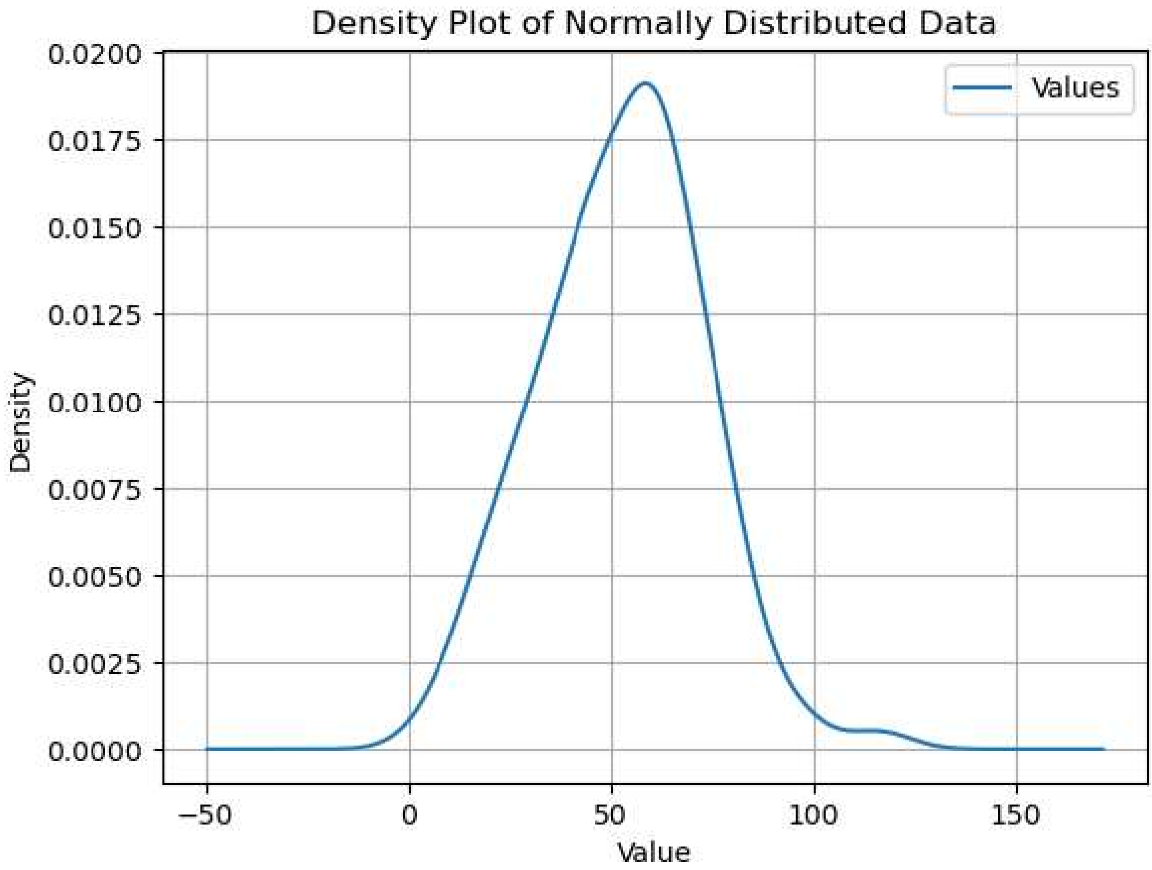
In [11]:

```
#TASK2

Exp = [1, 2, 3, 4, 5]
Salary = [1000, 2500, 4000, 5000, 7000]

import numpy as np
mean_exp = np.mean(Exp)
std_exp = np.std(Exp)
Standardized_exp = [(x - mean_exp) / std_exp for x in Exp]
mean_salary = np.mean(Salary)
std_salary = np.std(Salary)
Standardized_salary = [(x - mean_salary) / std_salary for x in Salary]

import pandas as pd
df = pd.DataFrame([Exp, Salary, Standardized_exp, Standardized_salary],
                  index=['Exp', 'Salary', 'Std_Exp', 'Std_Salary'])

print("Standardized DataFrame:\n")
print(df)
print("\nVerification of Standardization Properties:")
print(f"Mean of Standardized_exp: {np.mean(Standardized_exp):.2f}")
print(f"Standard Deviation of Standardized_exp: {np.std(Standardized_exp):.2f}")
print(f"Mean of Standardized_salary: {np.mean(Standardized_salary):.2f}")
print(f"Standard Deviation of Standardized_salary: {np.std(Standardized_salary):
```

```
Standardized DataFrame:

                    0            1            2            3            4
Exp          1.000000     2.000000     3.000000     4.000000     5.000000
Salary    1000.000000  2500.000000  4000.000000  5000.000000  7000.000000
Std_Exp     -1.414214    -0.707107     0.000000     0.707107     1.414214
Std_Salary  -1.408365    -0.679900     0.048564     0.534207     1.505493

Verification of Standardization Properties:
Mean of Standardized_exp: 0.00
Standard Deviation of Standardized_exp: 1.00
Mean of Standardized_salary: 0.00
Standard Deviation of Standardized_salary: 1.00
```

In [12]:

```python
#TASK3

import scipy.stats as stats
import pandas as pd
import matplotlib.pyplot as plt

data = stats.norm.rvs(loc=50, scale=20, size=100)

df = pd.DataFrame(data, columns=['Values'])

df.plot.density(title='Density Plot of Normally Distributed Data')
plt.xlabel('Value')
plt.grid(True)
plt.show()

mean_val = df['Values'].mean()
median_val = df['Values'].median()

print(f"Mean of the dataset: {mean_val:.2f}")
print(f"Median of the dataset: {median_val:.2f}")
```




```
Mean of the dataset: 52.63
Median of the dataset: 54.92
```

In [13]:
```
#TASK4
from scipy import stats
import numpy as np

mu = 168
sigma = 3.9
n = 36
sample_mean = 169.5
alpha = 0.05

z = (sample_mean - mu) / (sigma / np.sqrt(n))
z_critical = stats.norm.ppf(1 - alpha/2)

print(f"Z Score: {z:.2f}")
print(f"Z Critical (±): ±{z_critical:.2f}")

if abs(z) > z_critical:
    print("Reject the null hypothesis: There is enough evidence.")
else:
    print("Fail to reject the null hypothesis: Not enough evidence.")
```

```
Z Score: 2.31
Z Critical (±): ±1.96
Reject the null hypothesis: There is enough evidence.
```

In [14]:
```
 mean = 32
std_dev = 5.6
n = 40

conf_levels = [0.80, 0.90, 0.98]
```

```
print("\nConfidence Intervals:")
for conf in conf_levels:
    alpha = 1 - conf
    z = stats.norm.ppf(1 - alpha/2)
    margin_of_error = z * (std_dev / np.sqrt(n))
    lower = mean - margin_of_error
    upper = mean + margin_of_error
    print(f"{int(conf*100)}% CI: Z = {z:.2f}, Lower = {lower:.2f}, Upper = {uppe
```

```
Confidence Intervals:
80% CI: Z = 1.28, Lower = 30.87, Upper = 33.13
90% CI: Z = 1.64, Lower = 30.54, Upper = 33.46
98% CI: Z = 2.33, Lower = 29.94, Upper = 34.06
```

In [15]:

```
#TASK5
from scipy import stats
import numpy as np

mu = 100
sample_mean = 140
sample_std = 20
n = 30
alpha = 0.05
df = n - 1

t_stat = (sample_mean - mu) / (sample_std / np.sqrt(n))
t_critical = stats.t.ppf(1 - alpha/2, df)

print(f"T Statistic: {t_stat:.2f}")
print(f"T Critical (±): ±{t_critical:.2f}")

if abs(t_stat) > t_critical:
    print("✅ Reject the null hypothesis: Medication affects IQ.")
else:
    print("❌ Fail to reject the null hypothesis: No significant effect.")

t_result = stats.ttest_1samp(a=np.random.normal(loc=140, scale=20, size=30), pop
print(f"\nT-test (scipy function): T = {t_result.statistic:.2f}, p-value = {t_re
```

```
T Statistic: 10.95
T Critical (±): ±2.05
✅ Reject the null hypothesis: Medication affects IQ.

T-test (scipy function): T = 9.75, p-value = 0.00000
```

In [16]:

```
mean = 20
std_dev = 3.5
n = 15
df = n - 1
alpha = 0.05

t_critical = stats.t.ppf(1 - alpha/2, df)
margin_of_error = t_critical * (std_dev / np.sqrt(n))
lower = mean - margin_of_error
upper = mean + margin_of_error

print("\n95% Confidence Interval for Population Mean:")
print(f"T Critical: {t_critical:.3f}")
print(f"Lower Limit: {lower:.2f}")
print(f"Upper Limit: {upper:.2f}")
```

```
95% Confidence Interval for Population Mean:
T Critical: 2.145
Lower Limit: 18.06
Upper Limit: 21.94
```

In [ ]: