OUTLIERS

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population.

A measurement error or data entry error, correct the error if possible. If you can’t fix it, remove that observation because you know it’s incorrect.

Not a part of the population you are studying (i.e., unusual properties or conditions), you can legitimately remove the outlier.

A natural part of the population you are studying, you should not remove it.

Another approach is to perform the analysis with and without these observations and discuss the differences. Comparing results in this manner is particularly useful when you’re unsure about removing an outlier.

Mean is the accurate measure to describe the data when we do not have any outliers present.
Median is used if there is an outlier in the dataset.
Mode is used if there is an outlier AND about ½ or more of the data is the same.

Mean is the accurate measure to describe the data when we do not have any outliers present.
Median is used if there is an outlier in the dataset.
Mode is used if there is an outlier AND about ½ or more of the data is the same.

Nonparametric tests (Nonparametric tests use the median as the measure of central tendency) don’t require that your data follow the normal distribution. They’re also known as distribution-free tests and can provide benefits in certain situations.

In [1]:
```
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
```

In [2]:
```
df=pd.read_csv("D:/Downloads/archive (21)/haberman.csv")
df
```

Out[2]:

	30	64	1	1.1
0	30	62	3	1
1	30	65	0	1
2	31	59	2	1
3	31	65	4	1
4	33	58	10	1
...	...	...	...	...
300	75	62	1	1
301	76	67	0	1
302	77	65	3	1
303	78	65	1	2
304	83	58	2	2

305 rows × 4 columns

In [3]:
```
df.columns=['patient_age', 'operation_year', 'positive_axillary_nodes', 'sur
```

In [4]:
```
df
```

Out[4]:

	patient_age	operation_year	positive_axillary_nodes	survival_status
0	30	62	3	1
1	30	65	0	1
2	31	59	2	1
3	31	65	4	1
4	33	58	10	1
...	...	...	...	...
300	75	62	1	1
301	76	67	0	1
302	77	65	3	1
303	78	65	1	2
304	83	58	2	2

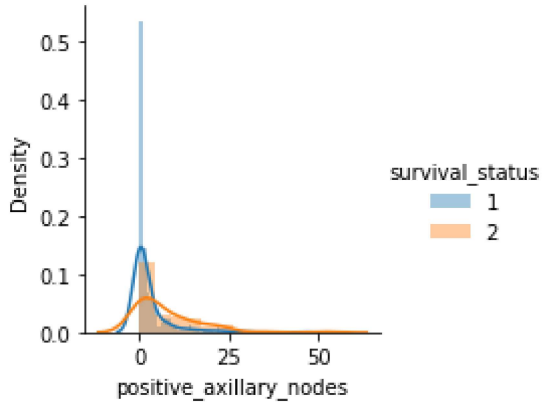
In [5]:
```
df.describe()
```

Out[5]:

	patient_age	operation_year	positive_axillary_nodes	survival_status
count	305.000000	305.000000	305.000000	305.000000
mean	52.531148	62.849180	4.036066	1.265574
std	10.744024	3.254078	7.199370	0.442364
min	30.000000	58.000000	0.000000	1.000000
25%	44.000000	60.000000	0.000000	1.000000
50%	52.000000	63.000000	1.000000	1.000000
75%	61.000000	66.000000	4.000000	2.000000
max	83.000000	69.000000	52.000000	2.000000

```
"there is a significant difference between the mean and the median
values(50%). This is because there are some outliers in our data and the
mean is influenced by the presence of outliers.It indicate potential
outliers, it's not conclusive proof.Or (mean>median) skewed"
(positive_axillary_nodes mean =4 and median=1 difference is high)
```

In [6]:
```
with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    g = sns.FacetGrid(df, hue="survival_status")
    g.map(sns.distplot, "positive_axillary_nodes", kde=True)
    g.add_legend()
    plt.figure(figsize=(12,6))
    plt.show()
```




```
<Figure size 864x432 with 0 Axes>
```

Detecting Outliers

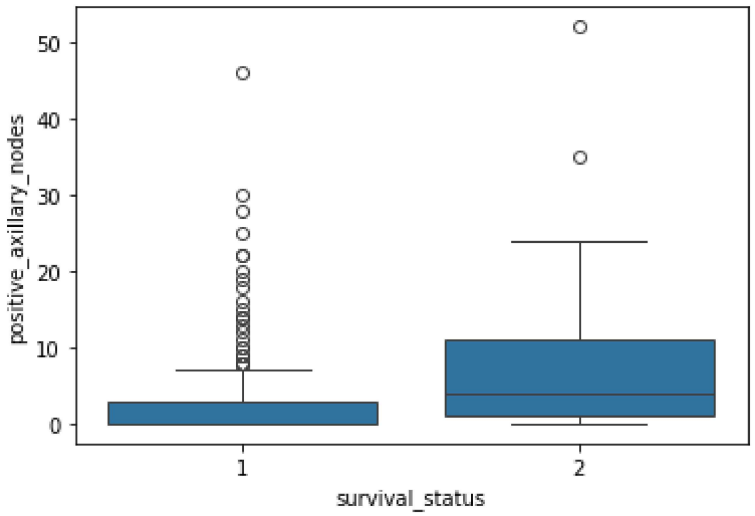
Zscore

In [7]:
```
def detect_outliers_zscore(data, threshold=3):
    z_scores = np.abs((data - np.mean(data)) / np.std(data))
    return z_scores > threshold
print(np.where(detect_outliers_zscore(df["positive_axillary_nodes"]))[0])
```

```
[  8  61 173 214 251]
```

Boxplot

In [8]:
```
sns.boxplot(x = 'survival_status', y = 'positive_axillary_nodes', data = df)
plt.show()
```



Interquartile Range (IQR) Method

```
    Calculates the range between the first quartile (Q1) and the third
quartile (Q3) of the data.
    Identifies outliers as data points that fall below Q1 - 1.5 * IQR or
above Q3 + 1.5 * IQR.
    Robust to outliers and resistant to skewness in the data.
    Particularly useful for skewed distributions or datasets with non-
normal distributions.
    Can be less sensitive to extreme values compared to the z-score
method.
```

In [9]:
```
def detect_outliers_iqr(data, threshold=1.5):
    Q1, Q3 = np.percentile(data, [25, 75])
    IQR = Q3 - Q1
    return (data < Q1 - threshold * IQR) | (data > Q3 + threshold * IQR)
print(np.where(detect_outliers_iqr(df["positive_axillary_nodes"]))[0])
```

```
[  8  13  21  23  30  42  58  61  65  74  78  84  91  95 105 106 107 123
 135 159 160 166 167 173 176 180 184 187 197 214 222 226 237 239 251 253
 259 260 268 286]
```

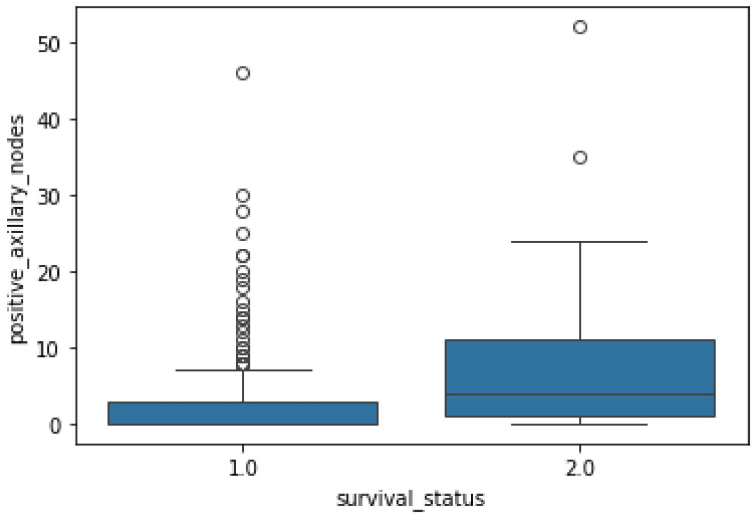
In this case outliers are usual one, Not need to remove.For learning purpose handling the outliers.

Remove Outliers

Imputation

Replace missing values or outliers with estimated values based on the remaining data, using techniques such as mean imputation, regression imputation, or multiple imputation.

In [10]:
```
imputer = SimpleImputer(strategy='mean')
imputed_data = imputer.fit_transform(df)
colum=['patient_age', 'operation_year', 'positive_axillary_nodes', 'survival
imputed_df = pd.DataFrame(imputed_data, columns=colum)
sns.boxplot(x = 'survival_status', y = 'positive_axillary_nodes', data = imp
plt.show()
```

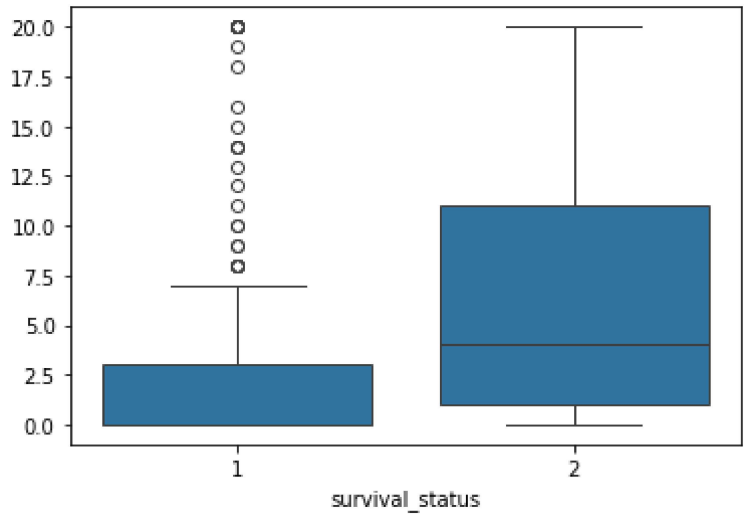


Winsorize

Replace extreme values with less extreme values, such as the nearest non-outlying value or a specified percentile value

In [11]:
```
import numpy as np
from scipy.stats.mstats import winsorize

# Winsorize extreme values
winsorized_data = winsorize(df['positive_axillary_nodes'], limits=[0.05, 0.0

print("Winsorized Data:")
print(winsorized_data)

sns.boxplot(x = 'survival_status', y =winsorized_data, data = df)
plt.show()

def detect_outliers_zscore(data, threshold=3):
    z_scores = np.abs((data - np.mean(data)) / np.std(data))
    return z_scores > threshold
print(np.where(detect_outliers_zscore(winsorized_data))[0])
```

```
Winsorized Data:
[ 3  0  2  4 10  0  0  9 20  1 10  7  0 13  0  1  0  0  0  0  6 15  0 20
  2  0  0  3  1  0 11  1  5  0  0  0  0  2  4  2  0  0 20  0  0  0  8  0
  0  8  0  0  1  0  0  1  2  4 20  0  1 20  2  0  0 14  2  3  0  2  0  4
  6  9 19  0  1  0 16  6  0  1  0  0 14  0  0  1  2  3  5 20  0  3  0 20
  0  0  0  6  0  0  3  4  4 12 11 11  7  8  2  0  0  0 10  1  0  0  1  1
  3  0  1 13  0  0  6  0  1  1  2  0  0  4  1 13  3  7  1  0  1  3  2  3
  4  0  4  0  4  5  0  1  0  0  0  4  1  3  9 20 12  1  1  2  1  0 11 20
  5  7  7  3  0 20  0  7 19  1  0  6 15  1  0  1 18  0  3 20  1  9  3  0
  2  1  0  0  5 14  1  9  0  0  0  0  0  0  0  0  3  1  0  0  3  2 20  0
  0  1  4  0  7  3 17  0  1  2 20  0  5  0  1  0  0  0  8  0  0 13  0 19
  6  0  0  0  1  0  0  0  0  0  9 20  0 20  0  0  0  0  2 20 15  0  0  0
  2  0  1  0 13  0  1  0  8  1  0  0  0  0  0  0  8  0  0  0  0  4 14  0
  0  8  0  2  0  0  0  3  0  0  3  0  1  0  3  1  2]
```




```
[]
```

In [12]:
```
trimmed_data = df[(df > np.percentile(df, 5)) & (df < np.percentile(df, 95))

print("Trimmed Data:")
print(trimmed_data)
sns.boxplot(x = 'survival_status', y ="positive_axillary_nodes", data = trim
plt.show()
```

```
Trimmed Data:
     patient_age  operation_year  positive_axillary_nodes  survival_statu
s
0           30.0            62.0                      3.0
1
1           30.0            65.0                      NaN
1
2           31.0            59.0                      2.0
1
3           31.0            65.0                      4.0
1
4           33.0            58.0                     10.0
1
..           ...             ...                      ...
...
300          NaN            62.0                      1.0
1
301          NaN             NaN                      NaN
1
```

SKEWNESS AND KURTOSIS

```
Skewness measures the degree of asymmetry of the distribution, while
Kurtosis measures the degree of peakedness and flatness of a distribution.

skewness is between -2 to +2 and kurtosis is between -7 to +7.
```

```
Close to 0: The distribution is approximately symmetric.
Greater than 0: The distribution is right-skewed (long tail on the right).
Less than 0: The distribution is left-skewed (long tail on the left).

Close to 0: The distribution has tails similar to a normal distribution
(mesokurtic).
Greater than 0: The distribution has heavier tails than a normal
distribution (leptokurtic).
Less than 0: The distribution has lighter tails than a normal distribution
(platykurtic).
```

```
Skewness
```

Measures the asymmetry of a graph, or how much a distribution deviates from a normal distribution. A distribution is asymmetrical if its left and right sides are not mirror images. Skewness can be positive, negative, or zero. A distribution is positively skewed when values are more concentrated on the right side, with a spread out left tail.

Kurtosis
Measures the distribution's tailedness, or how often outliers occur. It describes how peaked or flat a distribution is, relative to a normal distribution. A distribution with high kurtosis has heavy tails and more outliers, while a distribution with low kurtosis has light tails and fewer outliers. A normal distribution has a kurtosis of 3. A distribution with a kurtosis less than 3 is playkurtic, and has fewer and less extreme outliers than a normal distribution. A distribution with a kurtosis greater than 3 is leptokurtic, and has more outliers than a normal distribution.

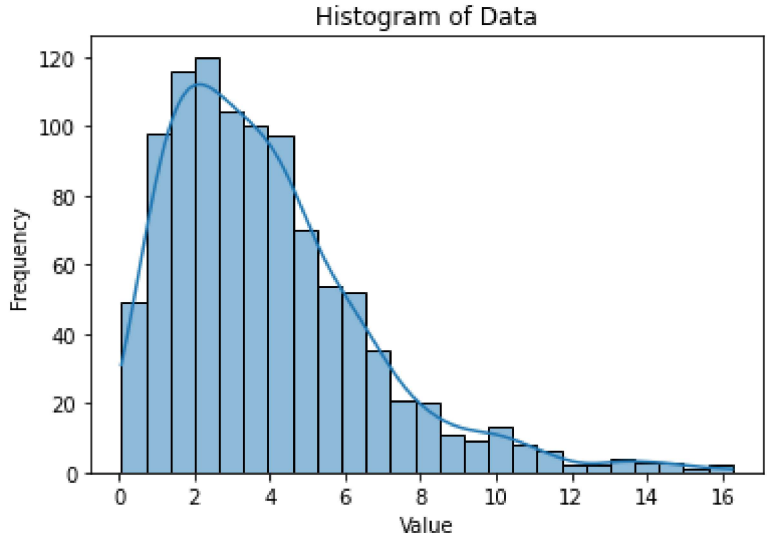
Data sets with high kurtosis have heavy tails and more outliers, while data sets with low kurtosis tend to have light tails and fewer outliers. Kurtosis determines whether the data exhibits a heavy-tailed or light-tailed distribution.
Excess kurtosis can be positive (Leptokurtic distribution), negative (Platykurtic distribution), or near zero (Mesokurtic distribution).

In [13]:
```
import numpy as np
import pandas as pd
from scipy.stats import skew, kurtosis
from scipy.stats import boxcox
import seaborn as sns
import matplotlib.pyplot as plt
```

In [14]:
```
np.random.seed(0)
data = np.random.gamma(2, scale=2, size=1000)
```

In [15]:
```
df = pd.DataFrame(data, columns=['Value'])
```

In [16]:
```
sns.histplot(df['Value'], kde=True)
plt.title('Histogram of Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



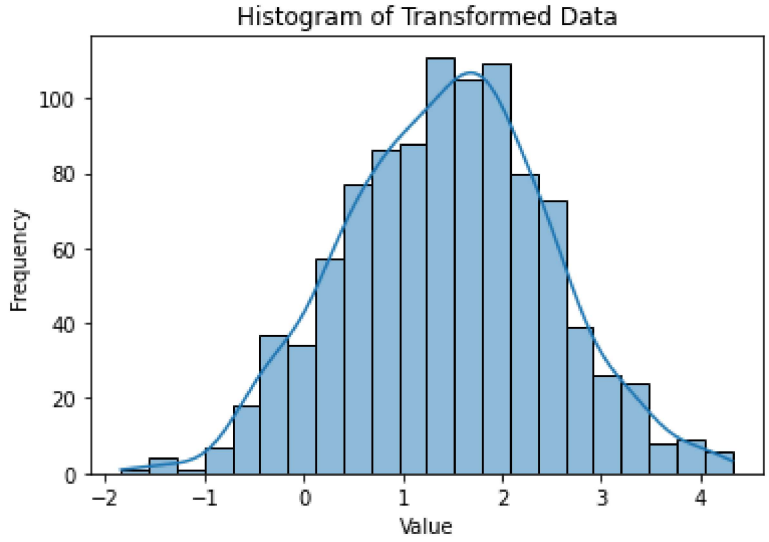
In [17]:
```
skewness = skew(df['Value'])
kurt = kurtosis(df['Value'])

print("Skewness:", skewness)
print("Kurtosis:", kurt)
```

```
Skewness: 1.358332294792333
Kurtosis: 2.4079180806130145
```

In [18]:
```
# Handle Skewness and Kurtosis
# Box-Cox Transformation
transformed_data, lambda_value = boxcox(df['Value'])
print("Lambda value for Box-Cox transformation:", lambda_value)
```

```
Lambda value for Box-Cox transformation: 0.29396531445691065
```

In [19]:
```
sns.histplot(transformed_data, kde=True)
plt.title('Histogram of Transformed Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()

# Re-calculate skewness and kurtosis after transformation
skewness_transformed = skew(transformed_data)
kurt_transformed = kurtosis(transformed_data)

print("Skewness after transformation:", skewness_transformed)
print("Kurtosis after transformation:", kurt_transformed)
```




```
Skewness after transformation: -0.011599377255897552
Kurtosis after transformation: -0.13618493270290521
```