In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

# Set the random seed for reproducibility
np.random.seed(42)

# Create a synthetic dataset
data = {
    'product_id': range(1, 21),
    'product_name': [f'Product {i}' for i in range(1, 21)],
    'category': np.random.choice(['Electronics', 'Clothing', 'Home', 'Sports'], 20)
    'units_sold': np.random.poisson(lam=20, size=20),  # Poisson distribution for s
    'sale_date': pd.date_range(start='2023-01-01', periods=20, freq='D')
}

sales_data = pd.DataFrame(data)

# Display the first few rows of the dataset
print("Sales Data:")
print(sales_data)


#save data set to local database into csv file

sales_data.to_csv('sales_data.csv', index=False)
```

```
Sales Data:
    product_id product_name     category  units_sold  sale_date
0            1    Product 1         Home          25 2023-01-01
1            2    Product 2       Sports          15 2023-01-02
2            3    Product 3  Electronics          17 2023-01-03
3            4    Product 4         Home          19 2023-01-04
4            5    Product 5         Home          21 2023-01-05
5            6    Product 6       Sports          17 2023-01-06
6            7    Product 7  Electronics          19 2023-01-07
7            8    Product 8  Electronics          16 2023-01-08
8            9    Product 9         Home          21 2023-01-09
9           10   Product 10     Clothing          21 2023-01-10
10          11   Product 11         Home          17 2023-01-11
11          12   Product 12         Home          22 2023-01-12
12          13   Product 13         Home          14 2023-01-13
13          14   Product 14         Home          17 2023-01-14
14          15   Product 15       Sports          17 2023-01-15
15          16   Product 16  Electronics          21 2023-01-16
16          17   Product 17       Sports          21 2023-01-17
17          18   Product 18       Sports          13 2023-01-18
18          19   Product 19       Sports          18 2023-01-19
19          20   Product 20         Home          25 2023-01-20
```

In [2]:

```python
# Descriptive stats

descriptive_stats = sales_data['units_sold'].describe()

print("\nDescriptive statistics for units sold:")
print(descriptive_stats)

mean_sales = sales_data['units_sold'].mean()
median_sales = sales_data['units_sold'].median()
mode_sales = sales_data['units_sold'].mode()
variance_sales = sales_data['units_sold'].var()
std_deviation_sales = sales_data['units_sold'].std()

category_stats = sales_data.groupby('category')['units_sold'].agg(['sum', 'mean', '
category_stats.columns = ['category', 'Total Units Sold', 'Average Units Sold', 'st

print('Statistical analysis')
print(f'mean units sold: {mean_sales}')
print(f'median units sold: {median_sales}')
print(f'mode units sold: {mode_sales}')
print(f'variance of units sold: {variance_sales}')
print(f'std deviation of units sold: {std_deviation_sales}')

print('Category stats')
print(category_stats)
```

```
Descriptive statistics for units sold:
count    20.000000
mean     18.800000
std       3.302312
min      13.000000
25%      17.000000
50%      18.500000
75%      21.000000
max      25.000000
Name: units_sold, dtype: float64
Statistical analysis
mean units sold: 18.8
median units sold: 18.5
mode units sold: 0    17
1    21
Name: units_sold, dtype: int32
variance of units sold: 10.90526315789474
std deviation of units sold: 3.3023117899275864
Category stats
```

	category	Total Units Sold	Average Units Sold	std Dev of Units Sold
0	Clothing	21	21.000000	NaN
1	Electronics	73	18.250000	2.217356
2	Home	181	20.111111	3.723051
3	Sports	101	16.833333	2.714160

In [3]:
```
# inferencial stats

confidence_level = 0.95
degrees_freedom = len(sales_data['units_sold']) - 1
sample_mean = mean_sales
std_deviation_error = std_deviation_sales / np.sqrt(len(sales_data['units_sold']))

t_score = stats.t.ppf((1 + confidence_level) / 2, degrees_freedom)
margin_of_error = t_score * std_deviation_error

confidence_intervel = (sample_mean - margin_of_error, sample_mean + margin_of_error
print('confidence for the mean of units sold:')
print(confidence_intervel)
```

```
confidence for the mean of units sold:
(17.254470507823573, 20.34552949217643)
```

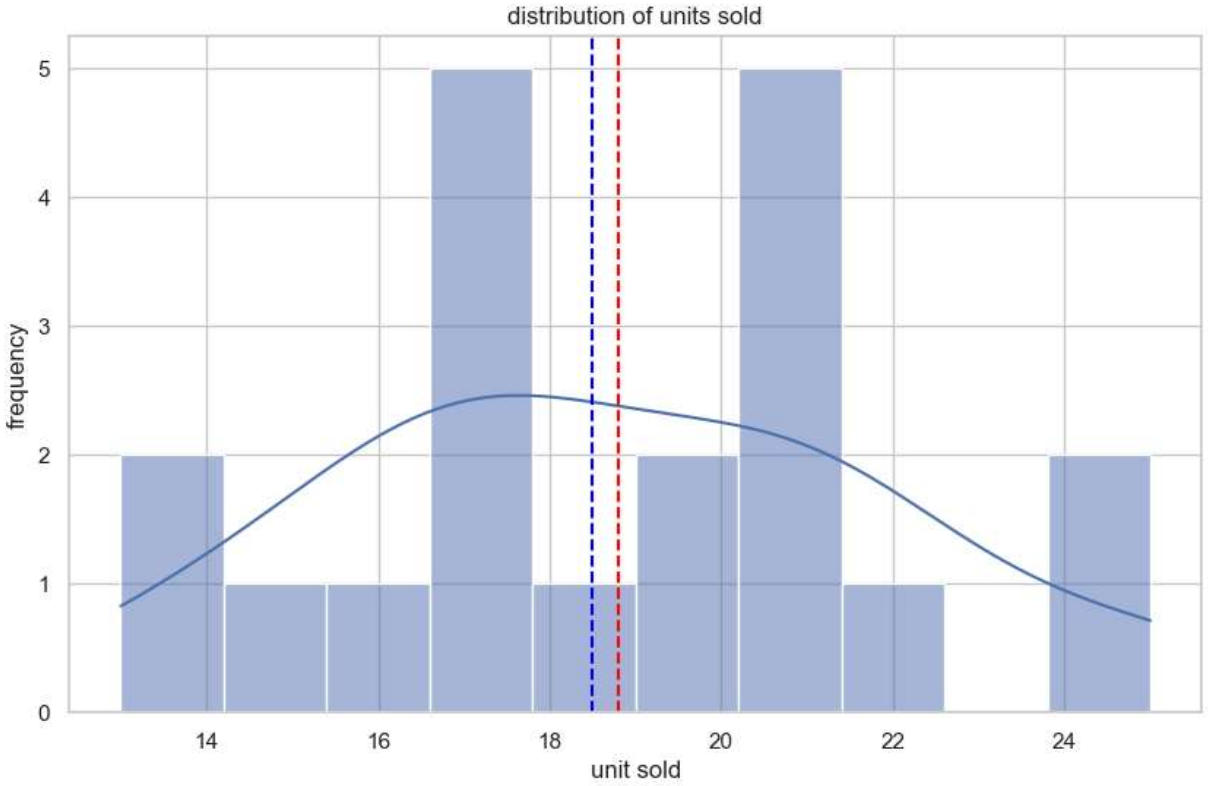
In [4]:
```

# Hypothesis testing

t_statistics, p_value = stats.ttest_1samp(sales_data['units_sold'],20)

print("\nHypothesis testing (t_testing):")
print(f"T-statistics: {t_statistics}, p-value: (p_value)")

if p_value < 0.05:
    print(f"reject the null hypothesis: the mean units is significantly different f
else:
    print("fail to reject the null hypothesis: the mean unit sold is not significas
```

```
Hypothesis testing (t_testing):
T-statistics: -1.6250928099424466, p-value: (p_value)
fail to reject the null hypothesis: the mean unit sold is not significasntly differe
nt from 20.
```

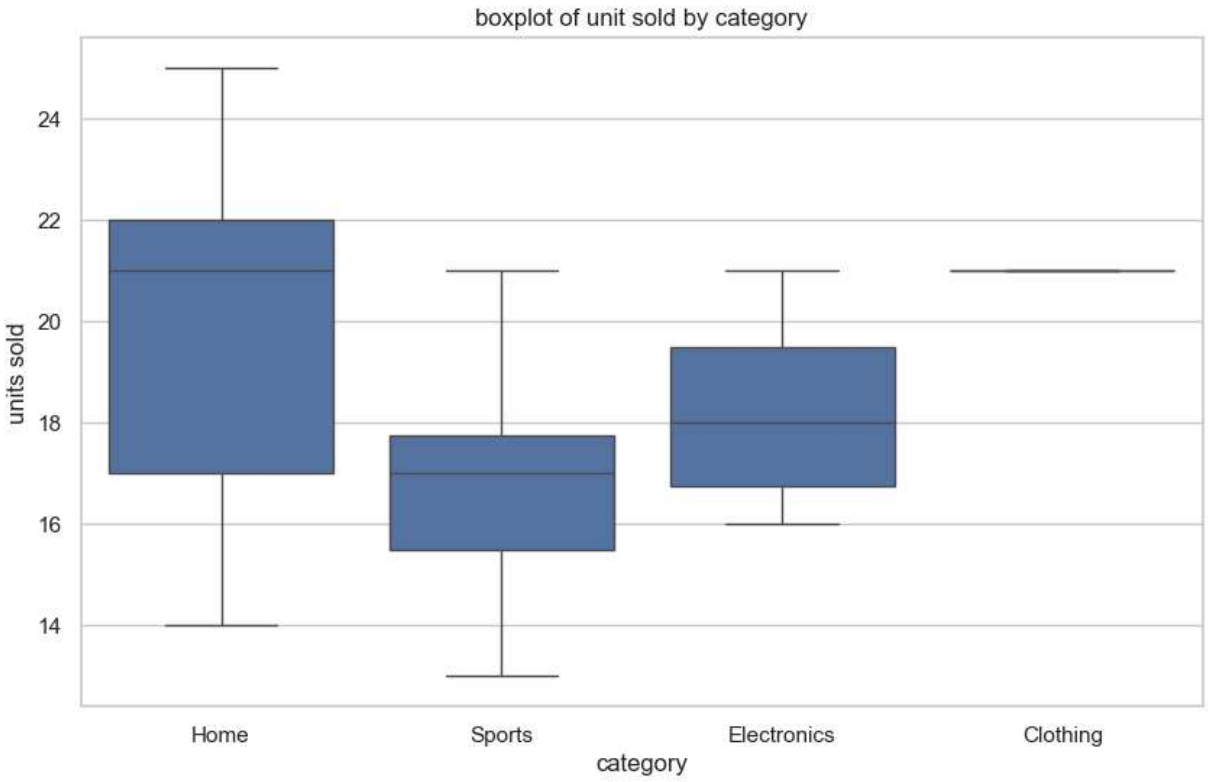
In [12]:

```
 #Visualization

sns.set(style='whitegrid')

plt.figure(figsize=(10,6))
sns.histplot(sales_data['units_sold'],bins=10, kde=True )
plt.title('distribution of units sold')
plt.xlabel('unit sold')
plt.ylabel('frequency')
plt.axvline(mean_sales, color='red', linestyle='--', label='Mean')
plt.axvline(median_sales, color='blue', linestyle='--', label='Median')
plt.axvline(mode_sales, color='green', linestyle='--', label='Mode')
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9148\2760288377.py in ?()
      8 plt.xlabel('unit sold')
      9 plt.ylabel('frequency')
     10 plt.axvline(mean_sales, color='red', linestyle='--', label='Mean')
     11 plt.axvline(median_sales, color='blue', linestyle='--', label='Median')
---> 12 plt.axvline(mode_sales, color='green', linestyle='--', label='Mode')
     13 plt.legend()
     14 plt.show()

~\anaconda3\Lib\site-packages\matplotlib\pyplot.py in ?(x, ymin, ymax, **kwargs)
   2729 @_copy_docstring_and_deprecators(Axes.axvline)
   2730 def axvline(x: float = 0, ymin: float = 0, ymax: float = 1, **kwargs) -> Lin
e2D:
-> 2731     return gca().axvline(x=x, ymin=ymin, ymax=ymax, **kwargs)

~\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py in ?(self, x, ymin, ymax, **k
wargs)
    848         xmin, xmax = self.get_xbound()
    849
    850         # Strip away the units for comparison with non-unitized bounds.
    851         xx, = self._process_unit_info([("x", x)], kwargs)
--> 852         scalex = (xx < xmin) or (xx > xmax)
    853
    854         trans = self.get_xaxis_transform(which='grid')
    855         l = mlines.Line2D([x, x], [ymin, ymax], transform=trans, **kwargs)

~\anaconda3\Lib\site-packages\pandas\core\generic.py in ?(self)
   1575     @final
   1576     def __nonzero__(self) -> NoReturn:
-> 1577         raise ValueError(
   1578             f"The truth value of a {type(self).__name__} is ambiguous. "
   1579             "Use a.empty, a.bool(), a.item(), a.any() or a.all()."
   1580         )

ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item
(), a.any() or a.all().
```

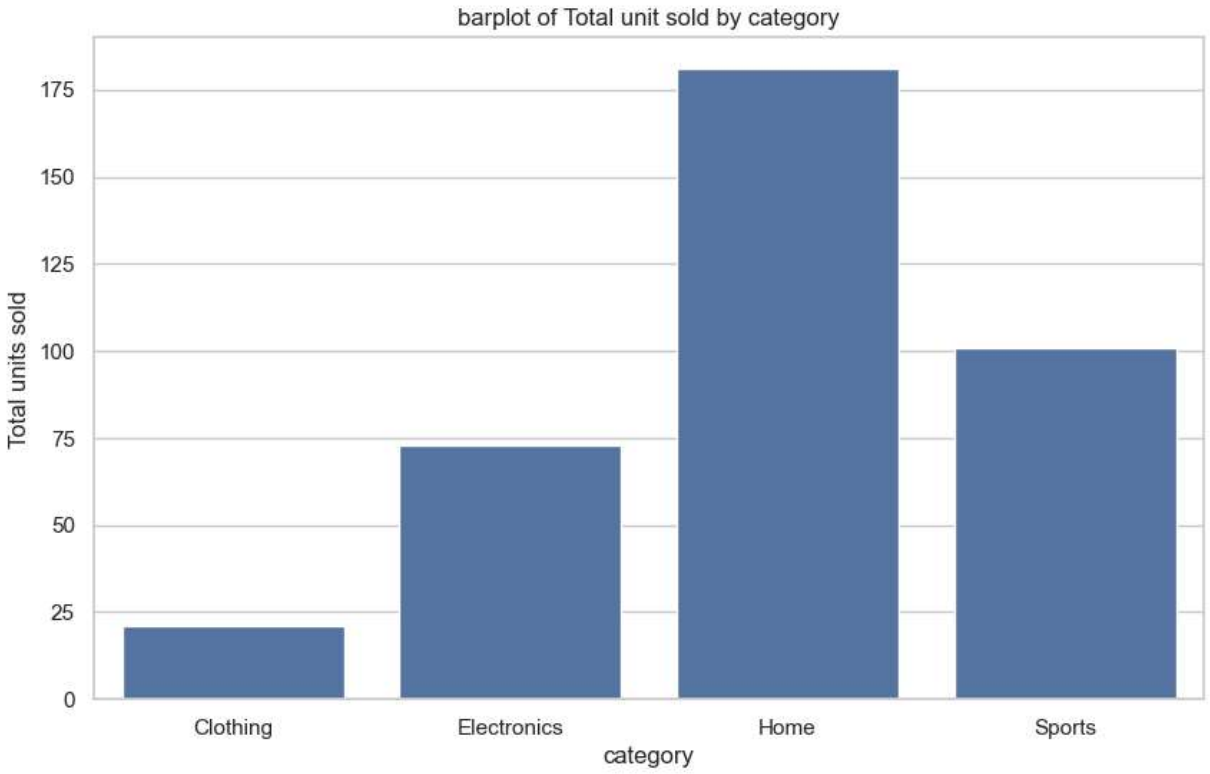


In [6]:
```
#boxplot for units sold by category
plt.figure(figsize=(10,6))
sns.boxplot(x='category', y='units_sold', data=sales_data)
plt.title('boxplot of unit sold by category')
plt.xlabel('category')
plt.ylabel('units sold')
plt.show()
```



In [ ]:

In [8]:
```
 #boxplot for total units sold by category
plt.figure(figsize=(10,6))
sns.barplot(x='category', y='Total Units Sold', data= category_stats)
plt.title('barplot of Total unit sold by category')
plt.xlabel('category')
plt.ylabel('Total units sold')
plt.show()
```



In [ ]: