1. Descriptive Statistics:

Descriptive statistics help summarize or describe a data set. These are the foundational tools for understanding the basic features of the data.

Key Measures:

- **Mean (Average):** The sum of all data points divided by the number of points.
- **Median:** The middle value when data points are sorted in order. It is useful when the data is skewed.
- **Mode:** The most frequently occurring value in the dataset.
- **Range:** The difference between the highest and lowest values.
- **Variance:** A measure of how spread out the values are. It shows the average of squared deviations from the mean.
- **Standard Deviation:** The square root of variance, showing how much data deviates from the mean on average.
- **Percentiles/Quartiles:** Divide the data into parts, with the 25th, 50th, and 75th percentiles (Q1, median, Q3) being the common quartiles.

Visualizations:

- **Histograms:** Represent the frequency distribution of a dataset.
- **Box Plots:** Show the spread and identify outliers (with quartiles).
- **Pie Charts:** Represent the percentage distribution of categorical data.
-

2. Inferential Statistics:

Inferential statistics use sample data to make generalizations about a population. It's about making predictions or inferences.

Key Concepts:

- **Population vs. Sample:** A population is the entire group you're interested in, while a sample is a subset of that population.
- **Sampling Distributions:** Distribution of sample statistics (like the sample mean) across multiple samples.
- **Central Limit Theorem (CLT):** States that the sampling distribution of the sample mean will tend to be normal if the sample size is large enough, no matter the distribution of the population.

Confidence Intervals:

- A range of values derived from the sample data that likely contains the true population parameter.
- Example: A 95% confidence interval means that 95% of samples drawn from the population would result in intervals that contain the true population mean.

Hypothesis Testing: Used to test an assumption or claim about a population using sample data. It's based on statistical significance.

3. Hypothesis Testing:

Hypothesis testing is a way to test the validity of a claim or hypothesis about a population using sample data. The goal is to determine whether there is enough evidence to reject a null hypothesis.

Steps in Hypothesis Testing:

1. **State the null hypothesis (H_0)** and the alternative hypothesis (H_1).
   - Null Hypothesis (H_0): Typically a statement of "no effect" or "no difference."
   - Alternative Hypothesis (H_1): The opposite of the null hypothesis, suggesting some effect or difference.
2. **Select the significance level (α):** Typically, $\alpha = 0.05$, meaning there's a 5% chance of rejecting a true null hypothesis (Type I error).
3. **Calculate the test statistic:** This depends on the type of test (e.g., z-test, t-test, chi-square test).
4. **Find the p-value:** The probability of obtaining the observed result, or more extreme, under the assumption that the null hypothesis is true. If the p-value is smaller than the significance level, reject H_0.
5. **Make a decision:** If the p-value is less than α, reject H_0. Otherwise, fail to reject H_0.

Types of Tests:

- **T-test:** Compares the means of two groups.
- **Chi-square test:** Tests relationships between categorical variables.
- **Z-test:** Used for hypothesis testing when the sample size is large, or population variance is known.

Types of Errors:

- **Type I Error:** Incorrectly rejecting the null hypothesis (false positive).
- **Type II Error:** Incorrectly failing to reject the null hypothesis (false negative).

4. Visualizations in Statistics:

Visualizations help in better understanding and interpretation of data.

Common Visualization Tools:

- **Bar Charts:** Display categorical data with rectangular bars. Height of bars shows value.
- **Histograms:** Show the distribution of continuous data by grouping data into bins.
- **Scatter Plots:** Show the relationship between two continuous variables.

- **Box Plots:** Represent the distribution of data and identify outliers.
- **Line Graphs:** Display data trends over time.
- **Heatmaps:** Show the intensity of values in two-dimensional data.

5. Key Statistical Tests and Methods:

- **ANOVA (Analysis of Variance):** Compares the means of three or more groups.
- **Linear Regression:** Examines the relationship between two continuous variables.
- **Correlation Coefficients:** Measures the strength and direction of a linear relationship between two variables (e.g., Pearson's r).
- **Chi-Square Test:** Tests relationships between categorical variables.

Pratical

Step 1: Import Libraries

In [3]:
```
import pandas as pd
import numpy  as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
```

Step 2: Create the Dataset

In [8]:
```
# 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'],
    'units_sold': np.random.poisson(lam=20, size=20),
    'sales_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)
```

```
Sales Data:
    Product_id  Product_name     category  units_sold  sales_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 [10]:
```python
# Save the DataFrame as a CSV file
sales_data.to_csv('sales_data.csv', index=False)
```

In [12]:
```python
# path location
import os
os.getcwd()
```

Out[12]: 'C:\\Users\\navee'

Step 3: Descriptive Statistics

In [43]:
```python
# Descriptive statistics
descriptive_stats = sales_data['units_sold'].describe()

# Display descriptive statistics
print("\nDescriptive Statistics for Units Sold:")
print(descriptive_stats)

# Additional statistics
mean_sales = sales_data['units_sold'].mean()
median_sales = sales_data['units_sold'].median()
mode_sales = sales_data['units_sold'].mode()[0]
variance_sales = sales_data['units_sold'].var()
std_deviation_sales = sales_data['units_sold'].std()

# Group by category and calculate total and average sales
category_stats = sales_data.groupby('category')['units_sold'].agg(['sum', 'mean'
category_stats.columns = ['Category', 'Total Units Sold', 'Average Units Sold',

# Display the results
print("\nStatistical 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"Standard Deviation of Units Sold: {std_deviation_sales}")
print("\nCategory Statistics:")
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: 17
Variance of Units Sold: 10.905263157894734
Standard Deviation of Units Sold: 3.3023117899275864

Category Statistics:
      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
```

Step 4: Inferential Statistics

In [28]:

```
# Confidence Interval for the mean of units sold
confidence_level = 0.95
degrees_freedom = len(sales_data['units_sold']) - 1
sample_mean = mean_sales
sample_standard_error = std_deviation_sales / np.sqrt(len(sales_data['units_sold
std_deviation_sales = np.std(sales_data['units_sold'], ddof=1)
# t-score for the confidence level
t_score = stats.t.ppf((1 + confidence_level) / 2, degrees_freedom)
margin_of_error = t_score * sample_standard_error

confidence_interval = (sample_mean - margin_of_error, sample_mean + margin_of_er
print("\nConfidence Interval for the Mean of Units Sold:")
print(confidence_interval)
```

```
Confidence Interval for the Mean of Units Sold:
(17.254470507823573, 20.34552949217643)
```

In [30]:

```
# Hypothesis Testing (t-test)
# Null hypothesis: Mean units sold is equal to 20
# Alternative hypothesis: Mean units sold is not equal to 20

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

print("\nHypothesis Testing (t-test):")
print(f"T-statistic: {t_statistic}, P-value: {p_value}")

if p_value < 0.05:
    print("Reject the null hypothesis: The mean units sold is significantly diff
```

```
else:
    print("Fail to reject the null hypothesis: The mean units sold is not signif
```

```
Hypothesis Testing (t-test):
T-statistic: -1.6250928099424466, P-value: 0.12061572226781002
Fail to reject the null hypothesis: The mean units sold is not significantly diff
erent from 20.
```

In [32]:
```
# Confidence Interval for the mean of units sold
confidence_level = 0.99
degrees_freedom = len(sales_data['units_sold']) - 1
sample_mean = mean_sales
sample_standard_error = std_deviation_sales / np.sqrt(len(sales_data['units_sold

# t-score for the confidence level
t_score = stats.t.ppf((1 + confidence_level) / 2, degrees_freedom)
margin_of_error = t_score * sample_standard_error

confidence_interval = (sample_mean - margin_of_error, sample_mean + margin_of_er
print("\nConfidence Interval for the Mean of Units Sold:")
print(confidence_interval)
```

```
Confidence Interval for the Mean of Units Sold:
(16.687430485978535, 20.912569514021467)
```

Hypothesis Testing

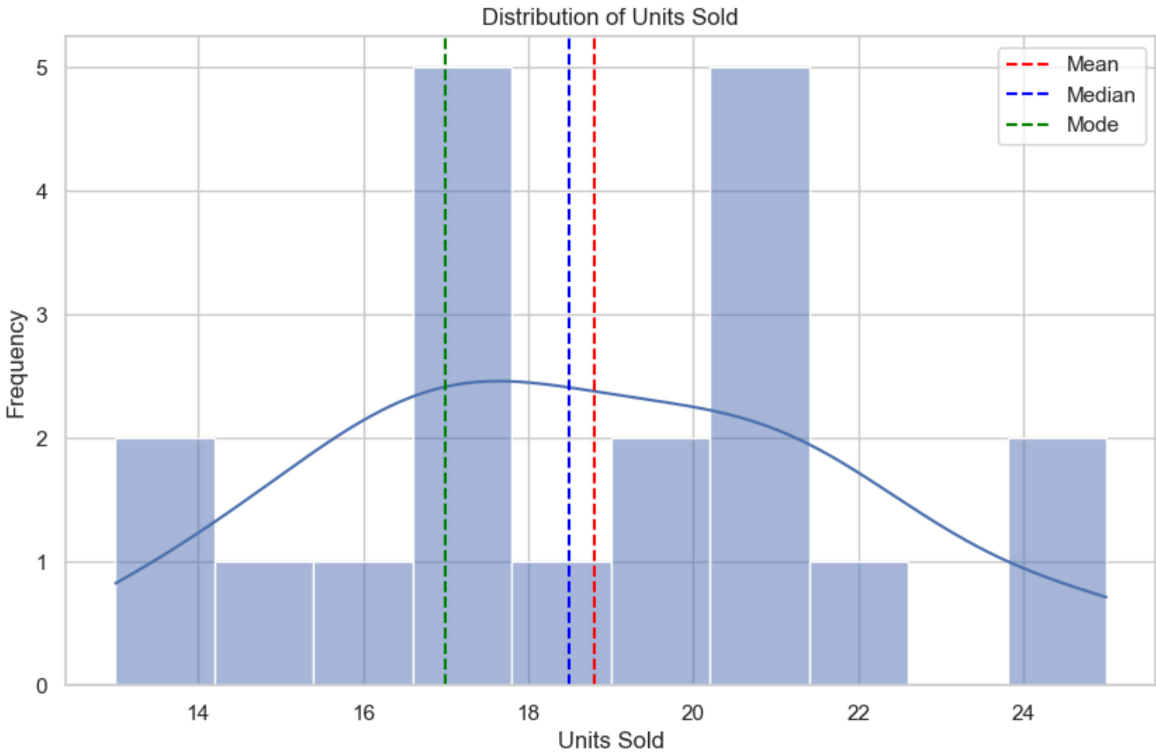
In [35]:
```
# Hypothesis Testing (t-test)
# Null hypothesis: Mean units sold is equal to 20
# Alternative hypothesis: Mean units sold is not equal to 20

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

print("\nHypothesis Testing (t-test):")
print(f"T-statistic: {t_statistic}, P-value: {p_value}")

if p_value < 0.05:
    print("Reject the null hypothesis: The mean units sold is significantly diff
else:
    print("Fail to reject the null hypothesis: The mean units sold is not signif
```

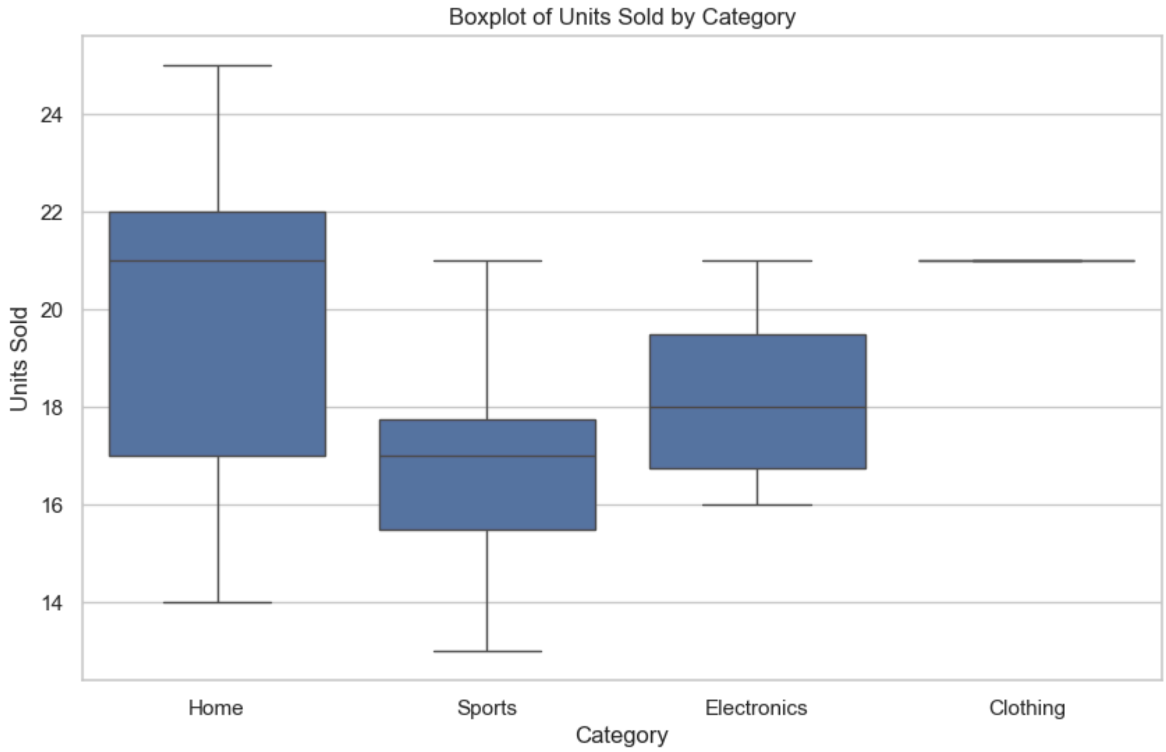
```
Hypothesis Testing (t-test):
T-statistic: -1.6250928099424466, P-value: 0.12061572226781002
Fail to reject the null hypothesis: The mean units sold is not significantly diff
erent from 20.
```

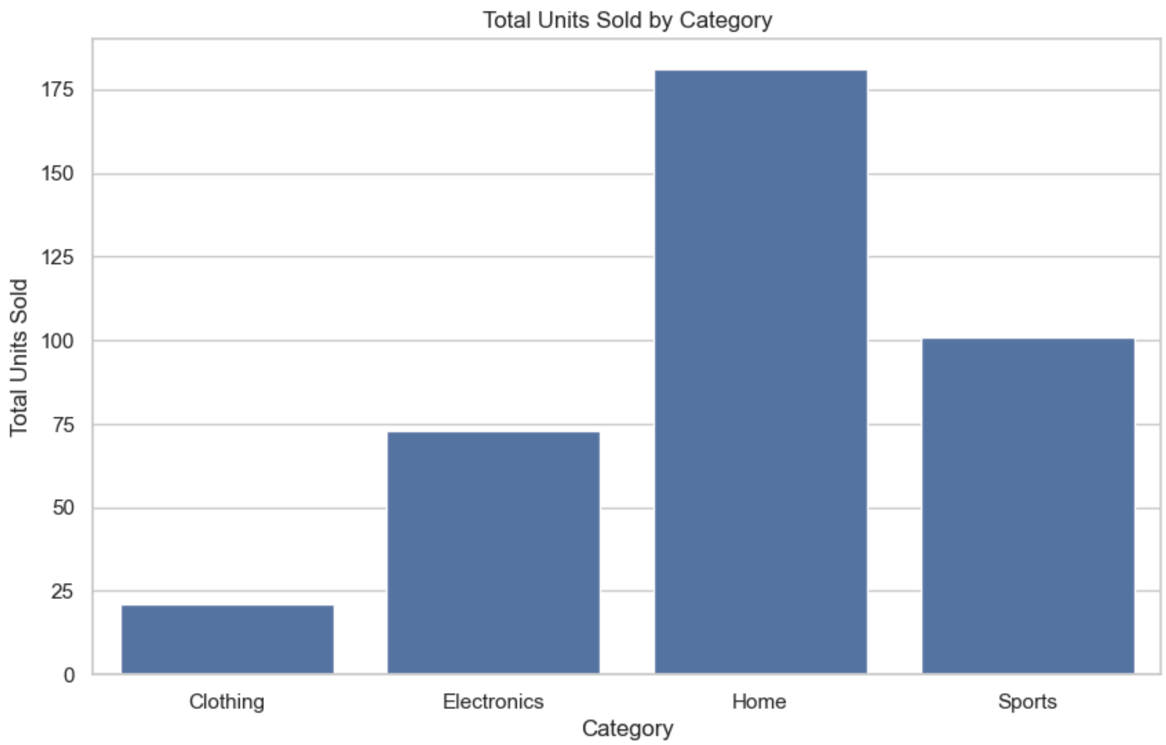
Step 5: Visualizations

In [51]:
```
# Visualizations
sns.set(style="whitegrid")

# Plot distribution of units sold
plt.figure(figsize=(10, 6))
sns.histplot(sales_data['units_sold'], bins=10, kde=True)
plt.title('Distribution of Units Sold')
plt.xlabel('Units Sold')
plt.ylabel('Frequency')
plt.axvline(mean_sales, color='red', linestyle='--', label='Mean')
```

```python
plt.axvline(median_sales, color='blue', linestyle='--', label='Median')
plt.axvline(mode_sales, color='green', linestyle='--', label='Mode')
plt.legend()
plt.show()

# 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 Units Sold by Category')
plt.xlabel('Category')
plt.ylabel('Units Sold')
plt.show()

# Bar plot for total units sold by category
plt.figure(figsize=(10, 6))
sns.barplot(x='Category', y='Total Units Sold', data=category_stats)
plt.title('Total Units Sold by Category')
plt.xlabel('Category')
plt.ylabel('Total Units Sold')
plt.show()
```







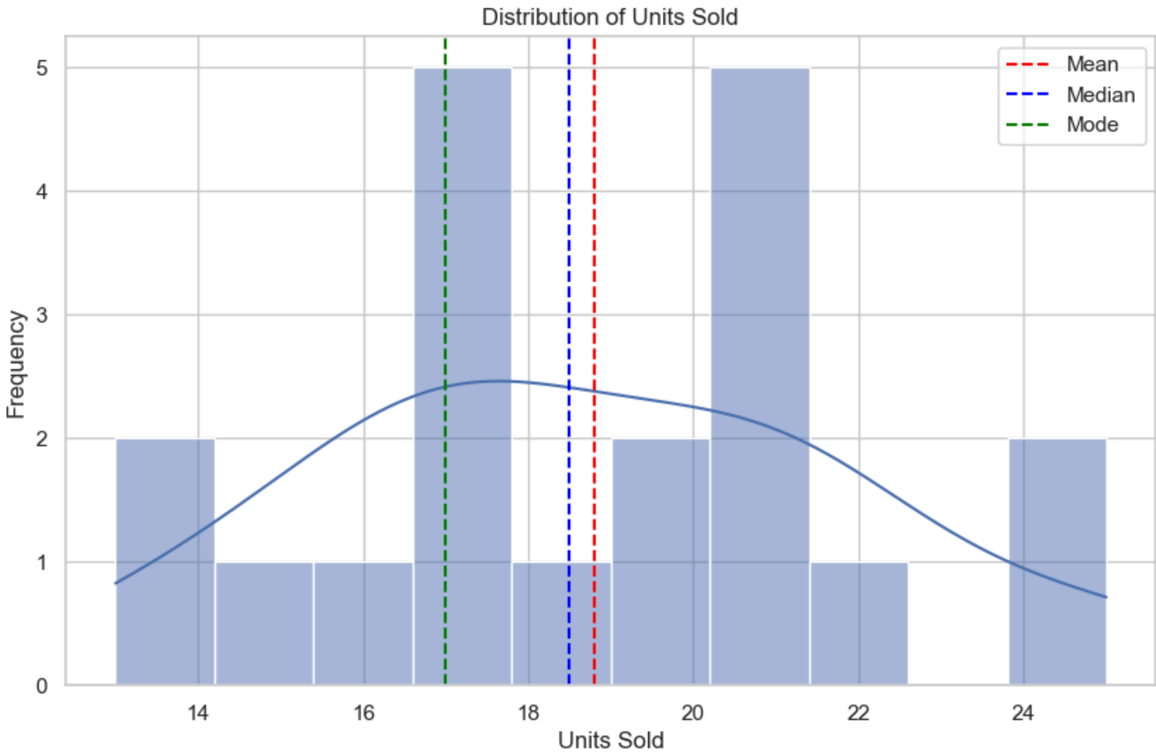
In [53]:
```
import streamlit as st
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns

# Set up the title and description of the app
st.title("Sales Data Analysis for Retail Store")
st.write("This application analyzes sales data for various product categories.")

# Generate synthetic sales data
def generate_data():
    np.random.seed(42)
```

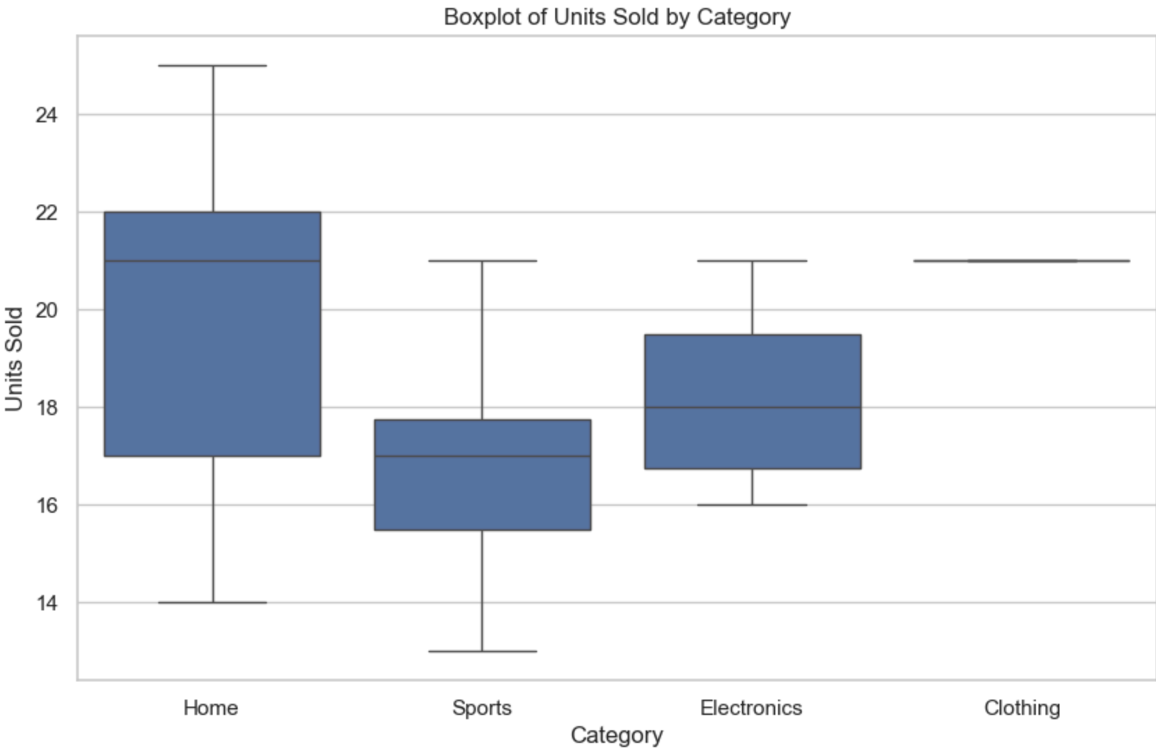
```
    data = {
        'product_id': range(1, 21),
        'product_name': [f'Product {i}' for i in range(1, 21)],
        'category': np.random.choice(['Electronics', 'Clothing', 'Home', 'Sports
        'units_sold': np.random.poisson(lam=20, size=20),
        'sale_date': pd.date_range(start='2023-01-01',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

sales_data = generate_data()

# Display the sales data
st.subheader("Sales Data")
st.dataframe(sales_data)

# Descriptive Statistics
st.subheader("Descriptive Statistics")
descriptive_stats = sales_data['units_sold'].describe()
st.write(descriptive_stats)

mean_sales = sales_data['units_sold'].mean()
median_sales = sales_data['units_sold'].median()
mode_sales = sales_data['units_sold'].mode()[0]

st.write(f"Mean Units Sold: {mean_sales}")
st.write(f"Median Units Sold: {median_sales}")
st.write(f"Mode Units Sold: {mode_sales}")

# Group statistics by category
category_stats = sales_data.groupby('category')['units_sold'].agg(['sum', 'mean'
category_stats.columns = ['Category', 'Total Units Sold', 'Average Units Sold',
st.subheader("Category Statistics")
st.dataframe(category_stats)

# Inferential Statistics
confidence_level = 0.95
degrees_freedom = len(sales_data['units_sold']) - 1
sample_mean = mean_sales
sample_standard_error = sales_data['units_sold'].std() / np.sqrt(len(sales_data[

# t-score for the confidence level
t_score = stats.t.ppf((1 + confidence_level) / 2, degrees_freedom)
margin_of_error = t_score * sample_standard_error
confidence_interval = (sample_mean - margin_of_error, sample_mean + margin_of_er

st.subheader("Confidence Interval for Mean Units Sold")
st.write(confidence_interval)

# Hypothesis Testing
t_statistic, p_value = stats.ttest_1samp(sales_data['units_sold'], 20)

st.subheader("Hypothesis Testing (t-test)")
st.write(f"T-statistic: {t_statistic}, P-value: {p_value}")

if p_value < 0.05:
    st.write("Reject the null hypothesis: The mean units sold is significantly d
else:
    st.write("Fail to reject the null hypothesis: The mean units sold is not sig

# Visualizations
```

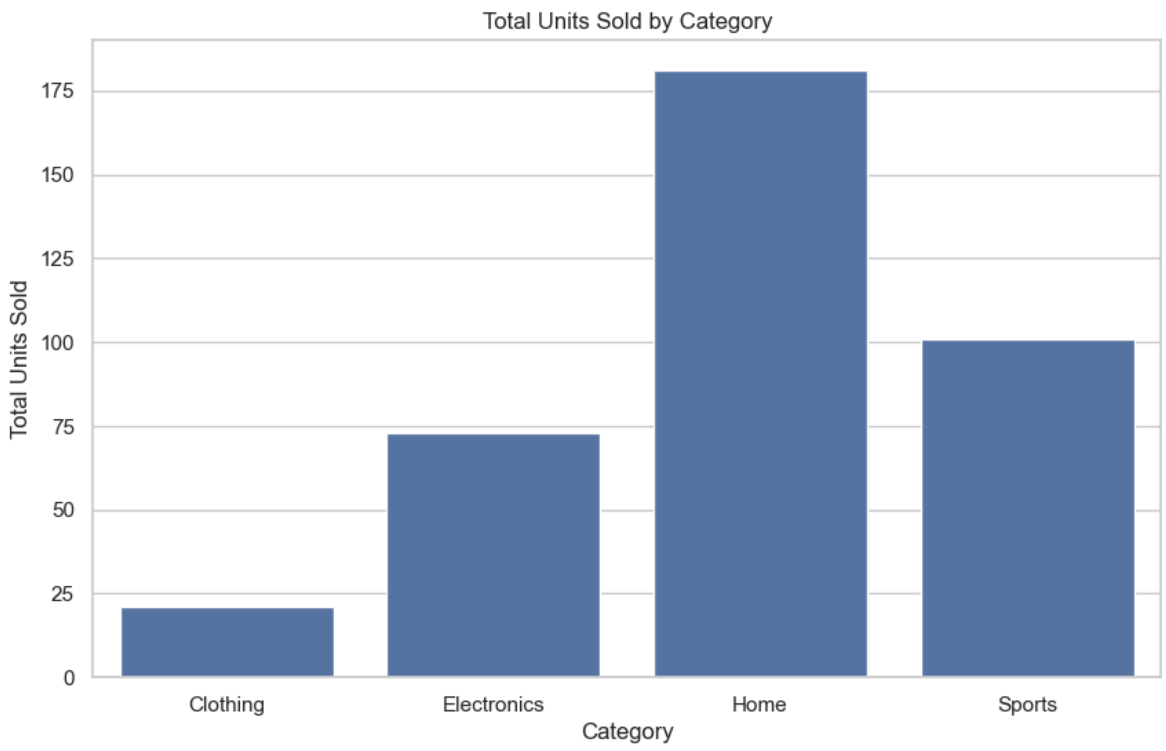
```python
st.subheader("Visualizations")

# Histogram of units sold
plt.figure(figsize=(10, 6))
sns.histplot(sales_data['units_sold'], bins=10, kde=True)
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.title('Distribution of Units Sold')
plt.xlabel('Units Sold')
plt.ylabel('Frequency')
plt.legend()
st.pyplot(plt)

# 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 Units Sold by Category')
plt.xlabel('Category')
plt.ylabel('Units Sold')
st.pyplot(plt)

# Bar plot for total units sold by category
plt.figure(figsize=(10, 6))
sns.barplot(x='Category', y='Total Units Sold', data=category_stats)
plt.title('Total Units Sold by Category')
plt.xlabel('Category')
plt.ylabel('Total Units Sold')
st.pyplot(plt)
```

```
2024-12-20 12:32:12.622
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run C:\Users\navee\anaconda3\Lib\site-packages\ipykernel_launcher.p
y [ARGUMENTS]
```

Out[53]: DeltaGenerator()


Distribution of Units Sold
Mean
Median
Mode
Frequency
0
1
2
3
4
5
14
16
18
20
22
24
Units Sold


Boxplot of Units Sold by Category
Units Sold
14
16
18
20
22
24
Home
Sports
Electronics
Clothing
Category



Example Application:

If you're testing the effectiveness of two different drugs, you could:

- Use a **t-test** to compare the means of the groups (i.e., the drug's effectiveness).
- Use **confidence intervals** to determine the range in which the true effect is likely to lie.
- **Visualize** the distribution of effectiveness scores using a **box plot** or **histogram**.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: