Streamlit\RSD_app.py

```python
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'], 20),
        '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', 'std']).reset_index()
category_stats.columns = ['Category', 'Total Units Sold', 'Average Units Sold', 'Std Dev of Units Sold']
st.subheader("Category Statistics")
st.dataframe(category_stats)

# Inferential Statistics
confidence_level = 0.95
```

```python
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['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_error)

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 different from 20.")
else:
    st.write("Fail to reject the null hypothesis: The mean units sold is not significantly different from 20.")

# Visualizations
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)
```