```python
import pandas as pd
from faker import Faker
import random

# Initialize Faker
fake = Faker()
# Load the data
df = pd.read_csv('fake_data.csv')

def generate_random_data(num_records):
    random_data = []
    for _ in range(num_records):
        order_id = "LK" + ''.join(random.choices('0123456789', k=7))
        email = fake.email()
        sales = random.choice([0, random.randint(100, 1000)])
        date = fake.date_time_this_year().strftime("%Y-%m-%d %H:%M:%S %z")
        product_quantity = 1
        product_name = fake.sentence(nb_words=6)  # Random product name
        product_sku = ''.join(random.choices('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', k=10))
        customer_name = fake.name()
        customer_city = fake.city()
        customer_zip = fake.zipcode()
        customer_phone = ''.join(random.choices('0123456789', k=10))

        random_data.append({
            "Order id": order_id,
            "Email": email,
            "Sales": sales,
            "Date": date,
            "Product quantity": product_quantity,
            "Product name": product_name,
            "Product sku": product_sku,
            "Customer Name": customer_name,
            "Customer City": customer_city,
            "Customer Zip": customer_zip,
            "Customer Phone": customer_phone
        })

    return pd.DataFrame(random_data)

# Generate 10 random records
random_df = generate_random_data(100000)

# Combine the sample data with the random data
combined_df = pd.concat([df, random_df], ignore_index=True)

# Save to CSV
combined_df.to_csv("combined_data.csv", index=False)

print(combined_df)

```

```
         Order id                         Email  Sales                 Date  \
0       LK5331309          ashley78@example.com    587  2024-07-04 12:38:16   
1       LK5742967      andrewgraham@example.org      0  2024-04-14 13:57:15   
2       LK1892755        ruizrobert@example.com      0  2024-05-01 15:27:08   
3       LK8705549            hfrey@example.org       0  2024-01-30 17:07:39   
4       LK0157039             apena@example.net    150  2024-05-13 13:35:12   
...           ...                           ...    ...                  ...   
199995  LK4325732         xjimenez@example.org     760  2024-01-07 07:23:24   
199996  LK4923586          zgaines@example.com       0  2024-05-16 20:33:34   
199997  LK9781515  kennethhernandez@example.net      0  2024-07-02 21:28:06   
199998  LK1225201     anthonyperkins@example.org   867  2024-05-29 06:35:33   
199999  LK0268894         steven13@example.org       0  2024-04-04 10:00:43   

        Product quantity                                 Product name  \
0                      1                          Fall explain cover.   
1                      1        Article education factor yeah project poor.   
2                      1                   Policy identify sit age.   
3                      1                   Drive recent system seat.   
4                      1                       Test herself five job.   
...                  ...                                          ...   
199995                 1  Represent chair staff campaign something anima...   
199996                 1                   Beat up door office state Mr.   
199997                 1                 Guess good sort unit marriage.   
199998                 1        Sea two finally wonder mention music about.   
199999                 1                 Space pressure race hand perhaps.   

       Product sku      Customer Name    Customer City Customer Zip  \
0       QLZA1MEZRA         Jacob King    Jonathanside        17251   
1       WD5QA9S0FX  Stephanie Shaffer       New Renee        97013   
2       447Z4O8GXU        Joseph Rice     Lake Krista        75772   
3       ERNY4YVY0M     Matthew Wright       Shawhaven        49529   
4       9UPB6QZS16        Cory Romero    Alvaradohaven        41927   
...            ...                ...              ...          ...   
199995  6KCHQYDY1B     Andres Stewart   West Susanfurt        40488   
199996  2AZNS8FYS6     Heather Bradley        Karinaton        89531   
199997  DAQYGFAVL6       Patricia Cain New Raymondfort        91569   
199998  8AO7F93LN4      Lisa Phillips       Mosleyview        08531   
```

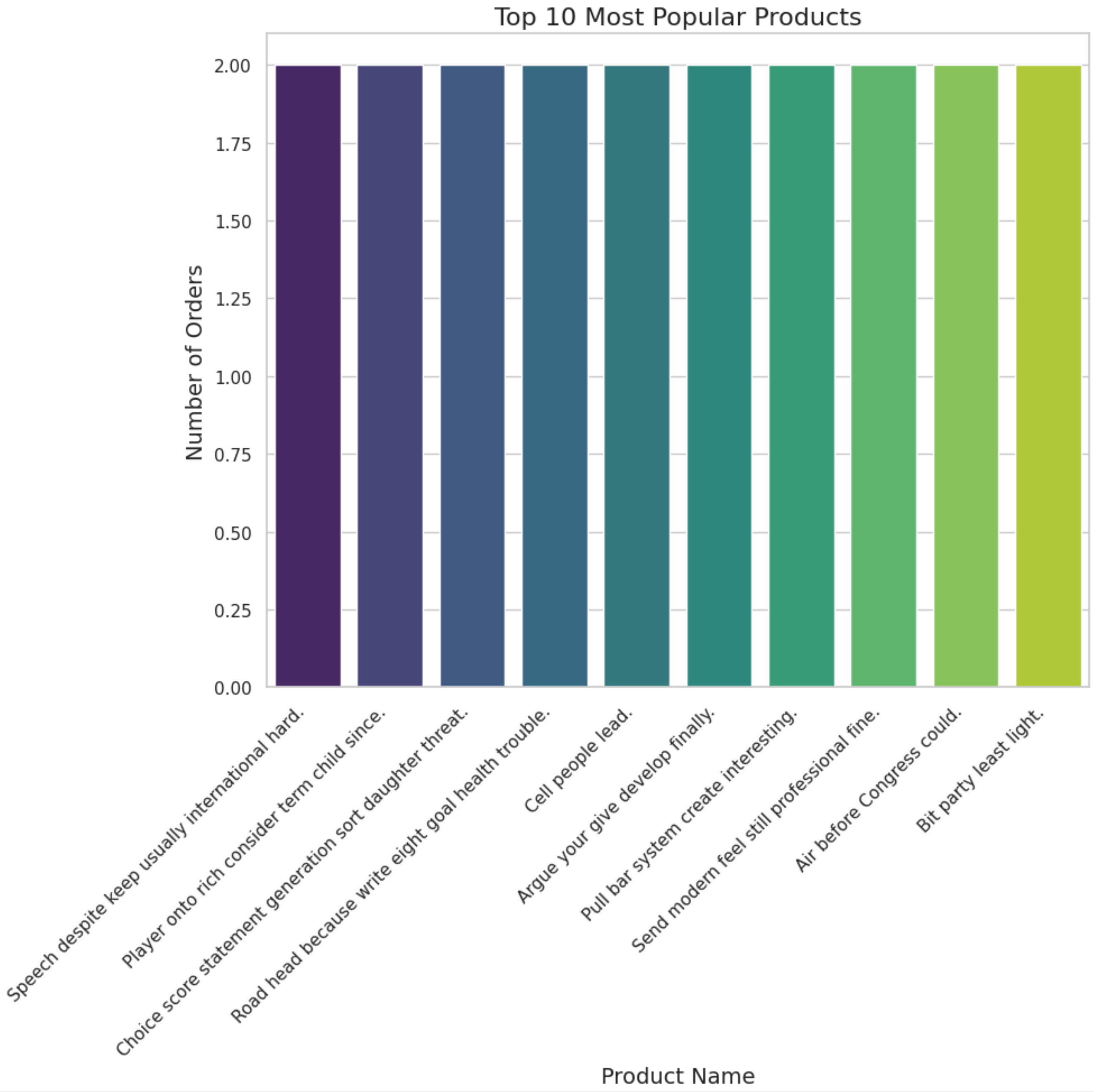
```
199999  4IUGIAOH6R      Colleen Perez     Michelleberg        05399

       Customer Phone
0          1943237940
1          9457143601
2          6520579073
3          3483957334
4          1956670057
...               ...
199995     8435625242
199996     5061574841
199997     8950008847
199998     1509151731
199999     3473473982

[200000 rows x 11 columns]
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Load the data
df = pd.read_csv('fake_data.csv')

# Prepare the data
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
df['Month'] = df.index.month
df['Year'] = df.index.year

# Feature and target variable
X = df[['Month', 'Year']]
y = df['Sales']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Function to evaluate models
def evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return mse, r2, y_pred

# Initialize models
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=0),
    'Random Forest': RandomForestRegressor(random_state=0),
    'XGBoost': xgb.XGBRegressor(objective='reg:squarederror', random_state=0)
}

# Evaluate each model
results = {}
for name, model in models.items():
    mse, r2, y_pred = evaluate_model(model, X_train, X_test, y_train, y_test)
    results[name] = {'MSE': mse, 'R-squared': r2, 'Predictions': y_pred}

# Print results
for name, result in results.items():
    print(f"{name} - MSE: {result['MSE']:.4f}, R-squared: {result['R-squared']:.4f}")

# Choose the best model based on R-squared
best_model_name = max(results, key=lambda k: results[k]['R-squared'])
best_model = models[best_model_name]

# Compare actual and predicted sales
y_pred_best = results[best_model_name]['Predictions']
comparison_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_best})
print(comparison_df)

# Ensure indices match
predicted_df = pd.DataFrame(y_pred_best, index=X_test.index, columns=['Predicted_Sales'])
df = df.join(predicted_df)

# Predict next month's sales based on the last billing date for each customer
last_billing_dates = df.reset_index().groupby(['Email', 'Customer Name'])['Date'].max()
predictions = []

for (email, customer_name), last_date in last_billing_dates.items():
    # Get the sales for the last billing date
    last_sales = df.loc[df.index == last_date, 'Sales'].values[0]

    next_month = (last_date.month % 12) + 1
    next_year = last_date.year if next_month > last_date.month else last_date.year + 1
    next_month_features = np.array([[next_month, next_year]])
    predicted_sales = best_model.predict(next_month_features)

    predictions.append((email, customer_name, last_date, last_sales, next_month, next_year, predicted_sales[0]))

predictions_df = pd.DataFrame(predictions, columns=['Email', 'Customer Name', 'Last_Billing_Date', 'Last_Sales', 'Next_Month', 'Next_Year', 'Predicted_Sal

# Save the predictions to a CSV file
predictions_df.to_csv('customer_next_month_predictions.csv', index=False)

print(f"Best model: {best_model_name}")
print(predictions_df.head())

```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegress
  warnings.warn(
Best model: Linear Regression
                 Email    Customer Name   Last_Billing_Date  Last_Sales  \
0  aabbott@example.net      Dawn Arroyo 2024-01-12 14:41:35         310   
1   aadams@example.net  Brandon Schmidt 2024-06-10 11:37:44         618   
2   aadams@example.org       Anita Soto 2024-06-24 14:55:02         608   
3   aadams@example.org    Cindy Michael 2024-03-01 22:22:37         839   
4  aadkins@example.com      Mark Barnes 2024-04-01 17:59:30           0   

   Next_Month  Next_Year  Predicted_Sales  
0           2       2024       276.441892  
1           7       2024       277.664137  
2           7       2024       277.664137  
3           4       2024       276.930790  
4           5       2024       277.175239  
```

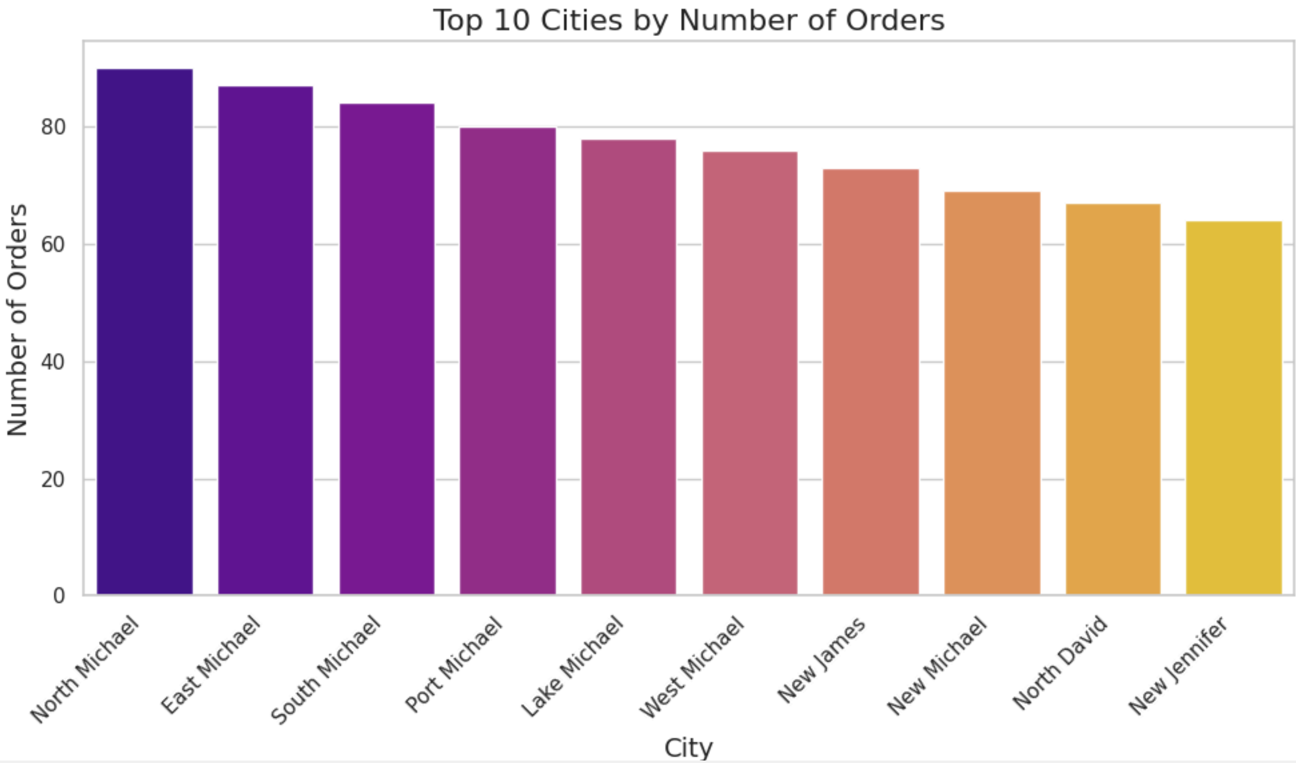
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Set the style for seaborn plots
sns.set(style='whitegrid')

# 1. Product Popularity Analysis
plt.figure(figsize=(10, 10))
top_products = df['Product name'].value_counts().head(10)
sns.barplot(x=top_products.index, y=top_products.values, palette='viridis')
plt.title('Top 10 Most Popular Products', fontsize=16)
plt.xlabel('Product Name', fontsize=14)
plt.ylabel('Number of Orders', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
<ipython-input-22-14fc856ef185>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  sns.barplot(x=top_products.index, y=top_products.values, palette='viridis')
```



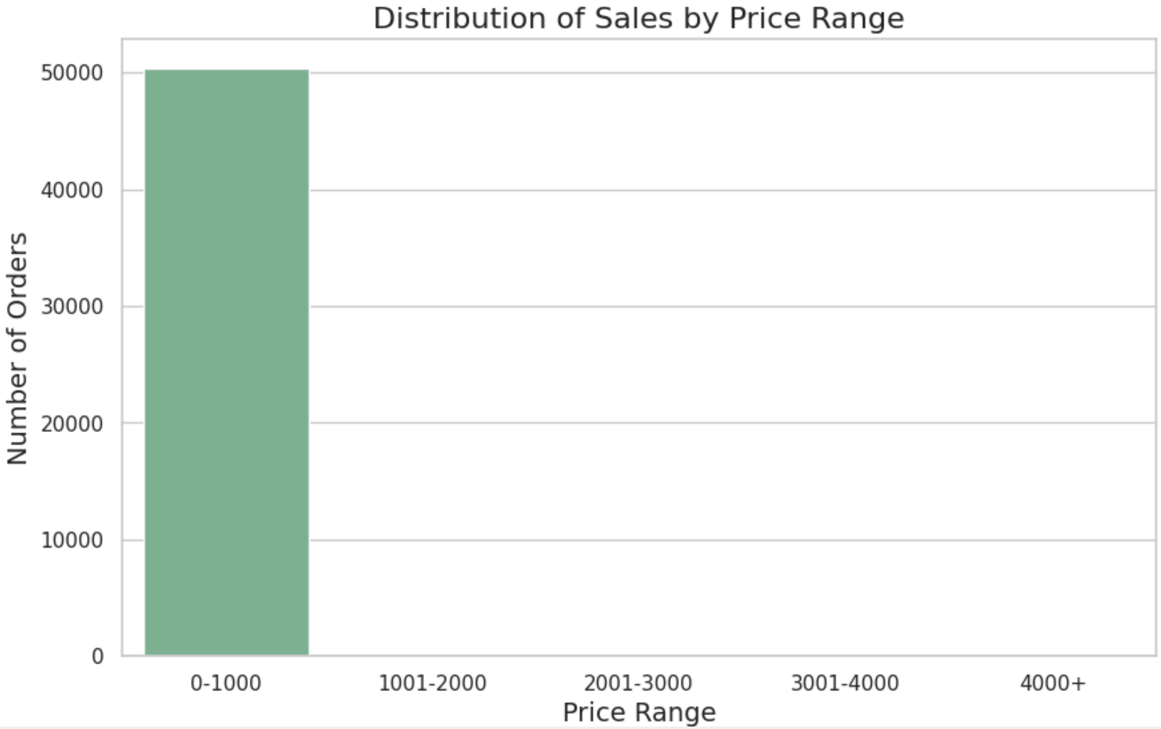
```
# 2. Customer Segmentation by Location
plt.figure(figsize=(10, 6))
top_cities = df['Customer City'].value_counts().head(10)
sns.barplot(x=top_cities.index, y=top_cities.values, palette='plasma')
plt.title('Top 10 Cities by Number of Orders', fontsize=16)
plt.xlabel('City', fontsize=14)
plt.ylabel('Number of Orders', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()




```

```
<ipython-input-23-2006098d045a>:4: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(x=top_cities.index, y=top_cities.values, palette='plasma')
```



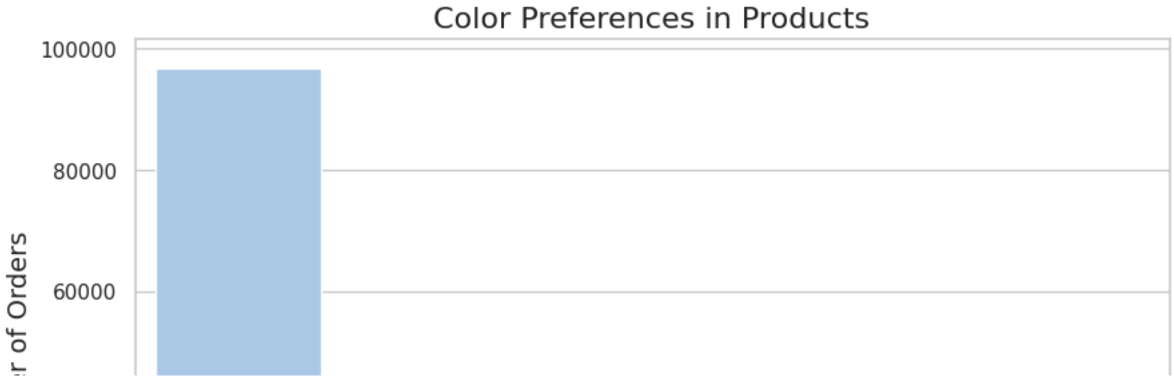
```
# 4. Price Point Analysis
df['Sales'] = pd.to_numeric(df['Sales'], errors='coerce')
price_ranges = pd.cut(df['Sales'], bins=[0, 1000, 2000, 3000, 4000, float('inf')],
                      labels=['0-1000', '1001-2000', '2001-3000', '3001-4000', '4000+'])
price_range_counts = price_ranges.value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(x=price_range_counts.index, y=price_range_counts.values, palette='crest')
plt.title('Distribution of Sales by Price Range', fontsize=16)
plt.xlabel('Price Range', fontsize=14)
plt.ylabel('Number of Orders', fontsize=14)
plt.show()

```

```
<ipython-input-25-f7feba57bb0b>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(x=price_range_counts.index, y=price_range_counts.values, palette='crest')
```

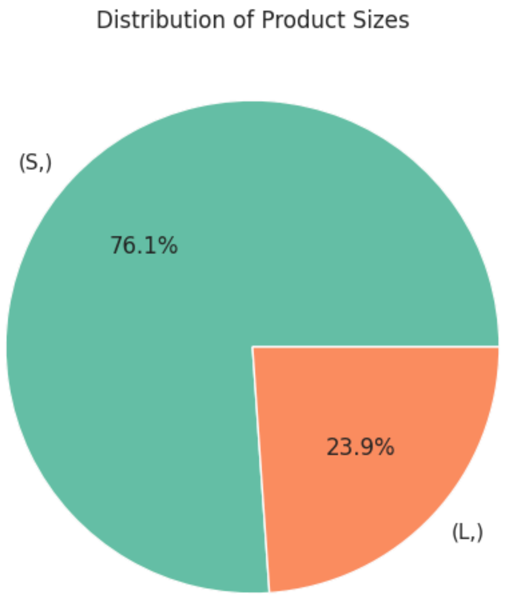


```python

# 5. Time-based Analysis
# df['Date'] = pd.to_datetime(df['Date'])
# orders_by_hour = df['Date'].dt.hour.value_counts().sort_index()
# plt.figure(figsize=(14, 7))
# sns.lineplot(x=orders_by_hour.index, y=orders_by_hour.values, marker='o', color='royalblue')
# plt.title('Number of Orders by Hour of Day', fontsize=16)
# plt.xlabel('Hour of Day', fontsize=14)
# plt.ylabel('Number of Orders', fontsize=14)
# plt.grid(True)
# plt.show()

# 6. Color Preference Analysis
def extract_color(product_name):
    colors = ['Blue', 'Red', 'Pink', 'Green', 'Grey', 'White', 'Ivory', 'Navy', 'Rose', 'Lavender', 'Mustard']
    for color in colors:
        if color.lower() in product_name.lower():
            return color
    return 'Other'

df['Color'] = df['Product name'].apply(extract_color)
color_counts = df['Color'].value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(x=color_counts.index, y=color_counts.values, palette='pastel')
plt.title('Color Preferences in Products', fontsize=16)
plt.xlabel('Color', fontsize=14)
plt.ylabel('Number of Orders', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.show()



```

```
<ipython-input-26-eb013bfabf40>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(x=color_counts.index, y=color_counts.values, palette='pastel')
```



```
# 3. Size Analysis
plt.figure(figsize=(8, 6))
size_counts = df['Product name'].str.extract(r'(\d+X|[XLS])').value_counts()
size_counts.plot(kind='pie', autopct='%1.1f%%', colors=sns.color_palette('Set2'))
plt.title('Distribution of Product Sizes', fontsize=12)
plt.ylabel('')
plt.show()
```


Distribution of Product Sizes
(S,)
(L,)