In [54]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import KNNImputer
```

In [59]:
```
# Load dataset
dataset_path = "C:/Users/gundr/Downloads/Day 19_E-Commerce_Data.csv"
df = pd.read_csv(dataset_path)
```

In [61]:
```
# Display basic info and check for missing values
print("Dataset Info:")
df.info()
print("\nMissing Values:")
print(df.isna().sum())
```

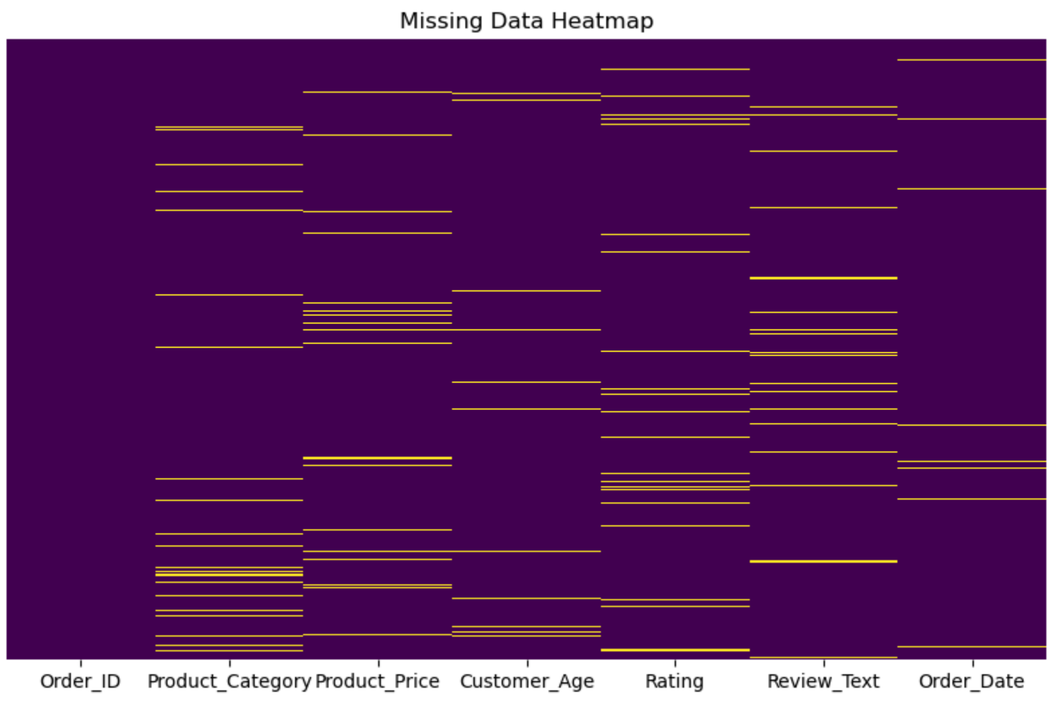
```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 505 entries, 0 to 504
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Order_ID          505 non-null    int64  
 1   Product_Category  481 non-null    object 
 2   Product_Price     485 non-null    float64
 3   Customer_Age      490 non-null    float64
 4   Rating            480 non-null    float64
 5   Review_Text       484 non-null    object 
 6   Order_Date        495 non-null    object 
dtypes: float64(3), int64(1), object(3)
memory usage: 27.7+ KB

Missing Values:
Order_ID            0
Product_Category    24
Product_Price       20
Customer_Age        15
Rating              25
Review_Text         21
Order_Date          10
dtype: int64
```

In [63]:
```
# Compute percentage of missing values
missing_percentage = (df.isna().sum() / len(df)) * 100
print("\nPercentage of Missing Values:")
print(missing_percentage)
```

```
Percentage of Missing Values:
Order_ID            0.000000
Product_Category    4.752475
Product_Price       3.960396
Customer_Age        2.970297
Rating              4.950495
Review_Text         4.158416
Order_Date          1.980198
dtype: float64
```

In [65]:
```
# Visualize missing data pattern
plt.figure(figsize=(10, 6))
sns.heatmap(df.isna(), cmap='viridis', cbar=False, yticklabels=False)
plt.title("Missing Data Heatmap")
plt.show()
```



In [67]:
```
# Handling missing values
# Numerical Columns: Mean/Median Imputation
num_cols = df.select_dtypes(include=['float64', 'int64']).columns
df[num_cols] = df[num_cols].apply(lambda x: x.fillna(x.median()))
```

In [71]:
```
# Date Columns: Forward Fill/Backward Fill (if applicable)
date_cols = [col for col in df.columns if 'date' in col.lower()]
for col in date_cols:
    df[col] = pd.to_datetime(df[col], errors='coerce')  # Convert to datetime
    df[col].fillna(method='ffill', inplace=True)

# KNN Imputation for complex cases
knn_imputer = KNNImputer(n_neighbors=5)
df[num_cols] = knn_imputer.fit_transform(df[num_cols])

# Evaluate impact
print("\nSummary Statistics Before and After Imputation:")
print(df.describe())
```

```
Summary Statistics Before and After Imputation:
         Order_ID  Product_Price  Customer_Age      Rating  \
count  505.000000     505.000000    505.000000  505.000000   
mean   249.899010    2442.748515     42.110891    3.188119   
min      1.000000     108.000000     18.000000    1.000000   
25%    124.000000    1292.000000     30.000000    3.000000   
50%    250.000000    2464.000000     41.000000    3.000000   
75%    375.000000    3588.000000     54.000000    4.000000   
max    500.000000    4993.000000     69.000000    5.000000   
std    144.769438    1387.499472     14.678958    1.117261   

                        Order_Date  
count                          505  
mean   2023-09-02 16:15:12.475247616  
min              2023-01-01 00:00:00  
25%              2023-04-26 00:00:00  
50%              2023-09-01 00:00:00  
75%              2024-01-08 00:00:00  
max              2024-05-14 00:00:00  
std                              NaN  
C:\Users\gundr\AppData\Local\Temp\ipykernel_12256\1820686745.py:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df[col].fillna(method='ffill', inplace=True)
C:\Users\gundr\AppData\Local\Temp\ipykernel_12256\1820686745.py:5: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df[col].fillna(method='ffill', inplace=True)
```

In [73]:
```
# Visualizing imputed data
plt.figure(figsize=(10, 6))
df[num_cols].hist(bins=30, figsize=(12, 8))
plt.suptitle("Histograms of Numerical Features After Imputation")
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```

In [78]:
```
print("\nData cleaning completed. Cleaned dataset saved as 'Cleaned_E_Commerce_Data.csv'")
```

```
Data cleaning completed. Cleaned dataset saved as 'Cleaned_E_Commerce_Data.csv'
```

In [ ]: