# **Data Visualization, Preprocessing, and Statistical Analysis Lab**

Student: Bhawesh Shrestha

Course: MSCS 634 – Data Analytics and Visualization

Instructor: Satish Penmatsa

Date: 11/02/2025

## **Purpose of the Lab**

The purpose of this lab is to apply data visualization, data preprocessing, and statistical analysis techniques using Jupyter Notebook. The lab demonstrates how to explore, clean, transform, and summarize data to uncover meaningful insights that support data-driven decision-making.

## **Key Steps and Deliverables**

### **1. Data Collection**

- Created a synthetic retail sales dataset of 500 daily records using Python.

# Step 1: Data Collection

import numpy as np

import pandas as pd

# Reproducibility

rng = np.random.default\_rng(42)

# Create a synthetic retail dataset with 500 rows

n = 500

dates = pd.date\_range('2024-01-01', periods=n, freq='D')

stores = rng.choice(['Seattle', 'Dallas', 'Chicago', 'New York', 'San Jose'], size=n)

categories = rng.choice(['Electronics', 'Grocery', 'Clothing', 'Home', 'Sports'], size=n, p=[0.2,0.35,0.2,0.15,0.1])

units\_sold = rng.integers(1, 50, size=n)

unit\_price = np.round(rng.uniform(3, 200, size=n), 2)

discount = np.round(rng.uniform(0, 0.35, size=n), 2)

temp\_f = np.round(rng.normal(65, 15, size=n), 1) # pretend store-day temperature

revenue = np.round(units\_sold \* unit\_price \* (1 - discount), 2)

df = pd.DataFrame({

'date': dates,

'store': stores,

'category': categories,

'units\_sold': units\_sold,

'unit\_price': unit\_price,

'discount': discount,

'temp\_f': temp\_f,

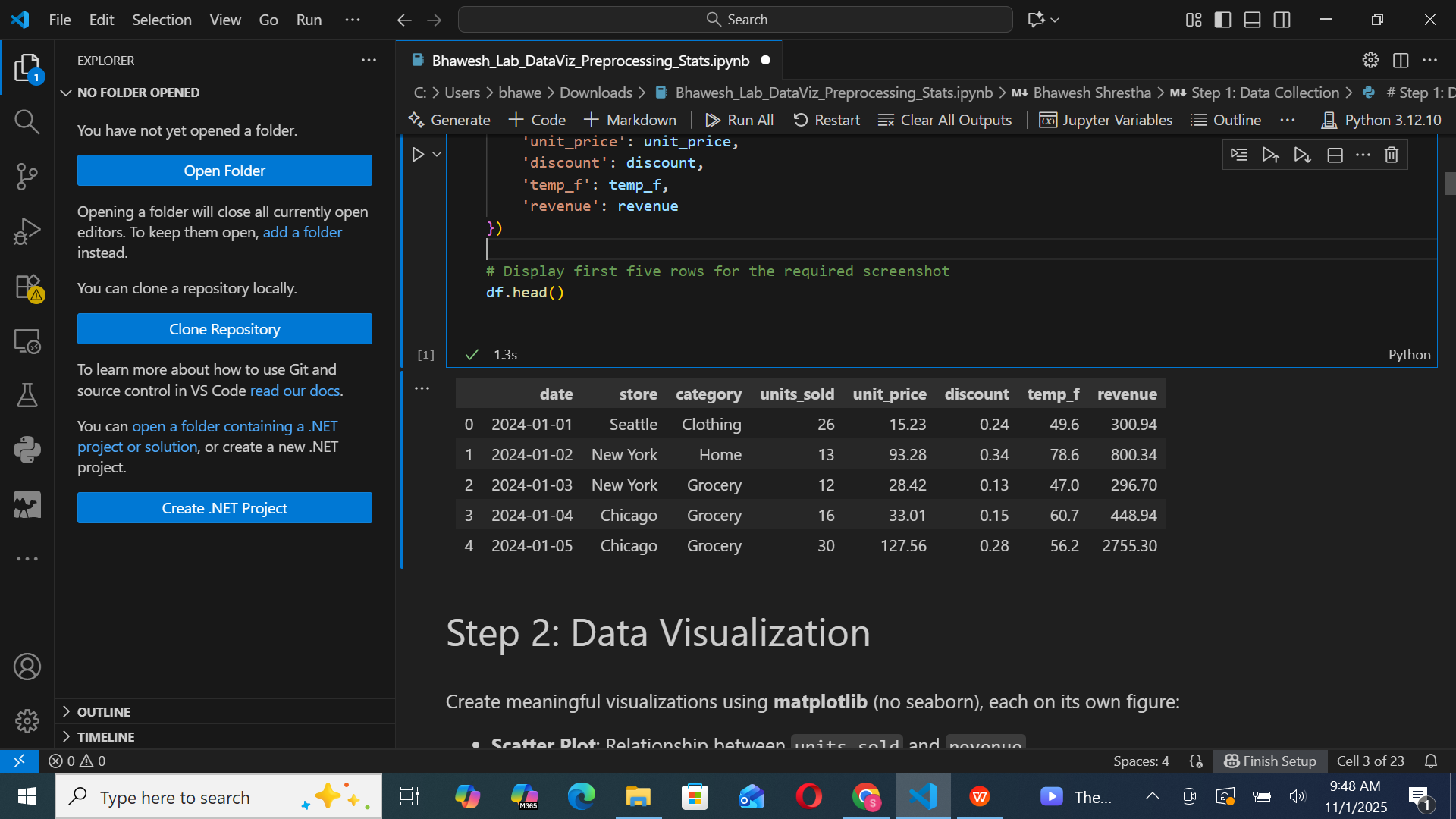
'revenue': revenue

})

# Display first five rows for the required screenshot

df.head()

- Displayed first five rows using df.head().



### **2. Data Visualization**

- Created multiple visualizations with Matplotlib:

# Step 2: Data Visualization

import matplotlib.pyplot as plt

# Ensure date is datetime

df['date'] = pd.to\_datetime(df['date'])

# 1) Scatter: units\_sold vs revenue

plt.figure()

plt.scatter(df['units\_sold'], df['revenue'])

plt.title('Scatter: Units Sold vs Revenue')

plt.xlabel('Units Sold')

plt.ylabel('Revenue')

plt.tight\_layout()

plt.show()

# 2) Line: Daily revenue trend

daily\_rev = df.groupby('date', as\_index=False)['revenue'].sum()

plt.figure()

plt.plot(daily\_rev['date'], daily\_rev['revenue'])

plt.title('Line: Daily Revenue Trend')

plt.xlabel('Date')

plt.ylabel('Revenue')

plt.tight\_layout()

plt.show()

# 3) Bar: Average revenue by category

avg\_rev\_cat = df.groupby('category', as\_index=False)['revenue'].mean().sort\_values('revenue', ascending=False)

plt.figure()

plt.bar(avg\_rev\_cat['category'], avg\_rev\_cat['revenue'])

plt.title('Bar: Average Revenue by Category')

plt.xlabel('Category')

plt.ylabel('Average Revenue')

plt.xticks(rotation=15)

plt.tight\_layout()

plt.show()

# 4) Histogram: Revenue distribution

plt.figure()

plt.hist(df['revenue'], bins=20)

plt.title('Histogram: Revenue Distribution')

plt.xlabel('Revenue')

plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()

# 5) Box Plot: Revenue by category

# Prepare data as a list in category order

cat\_order = sorted(df['category'].unique())

data\_by\_cat = [df[df['category'] == c]['revenue'] for c in cat\_order]

plt.figure()

plt.boxplot(data\_by\_cat, labels=cat\_order, showmeans=True)

plt.title('Box Plot: Revenue by Category')

plt.xlabel('Category')

plt.ylabel('Revenue')

plt.xticks(rotation=15)

plt.tight\_layout()

plt.show()

# 6) Pie Chart: Revenue share by category

share\_rev = df.groupby('category')['revenue'].sum()

plt.figure()

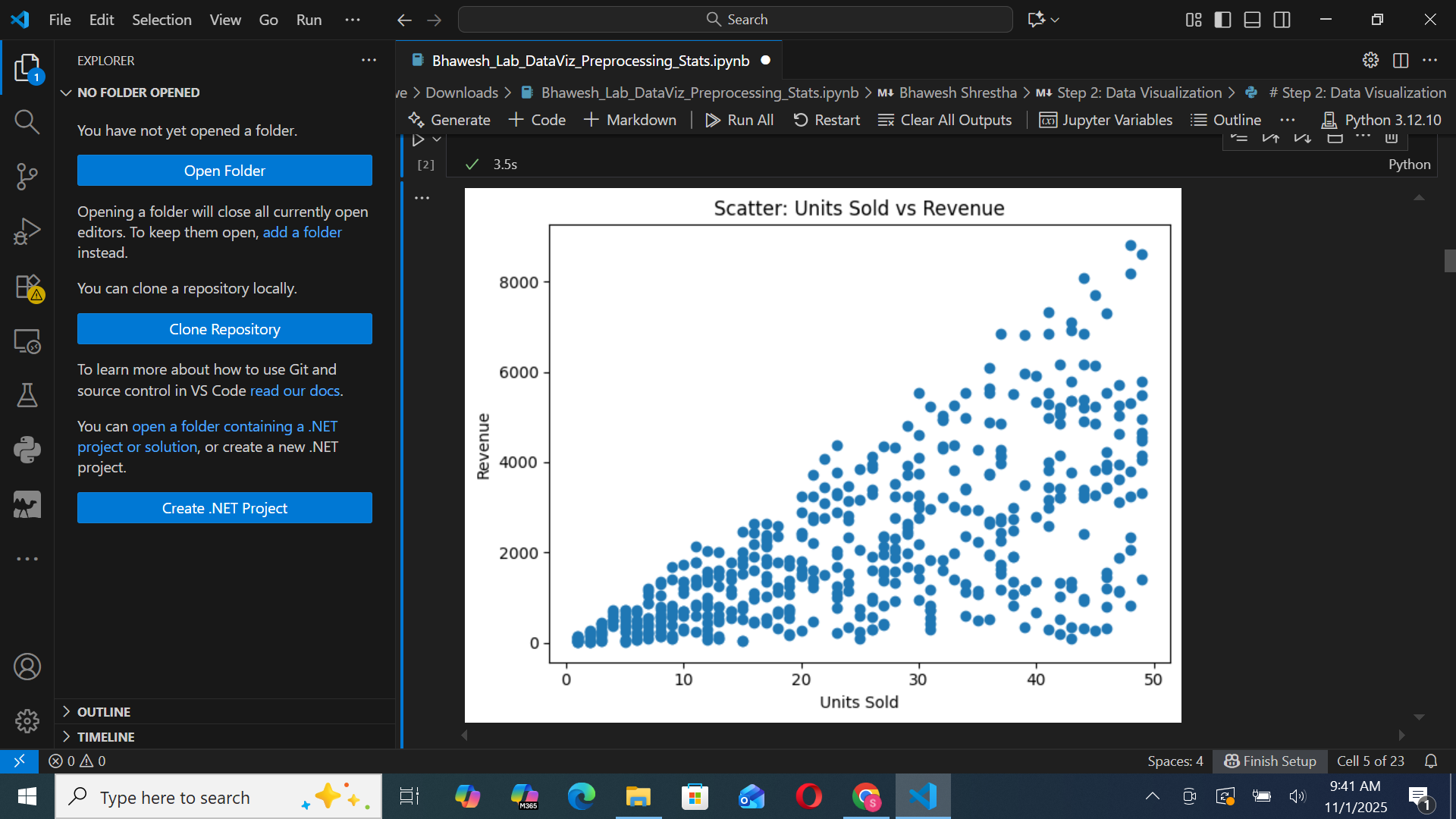
plt.pie(share\_rev.values, labels=share\_rev.index, autopct='%1.1f%%')

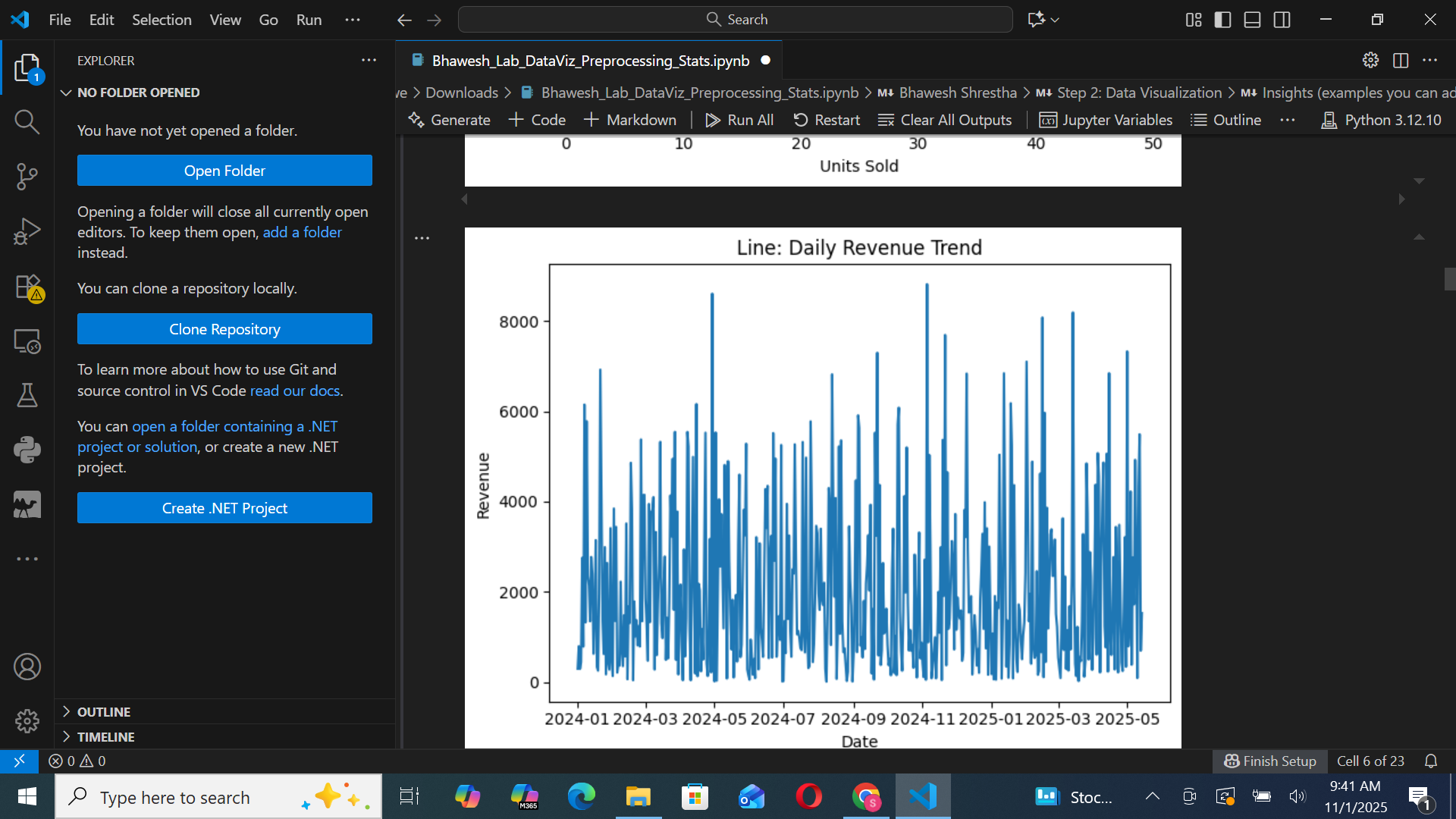
plt.title('Pie: Revenue Share by Category')

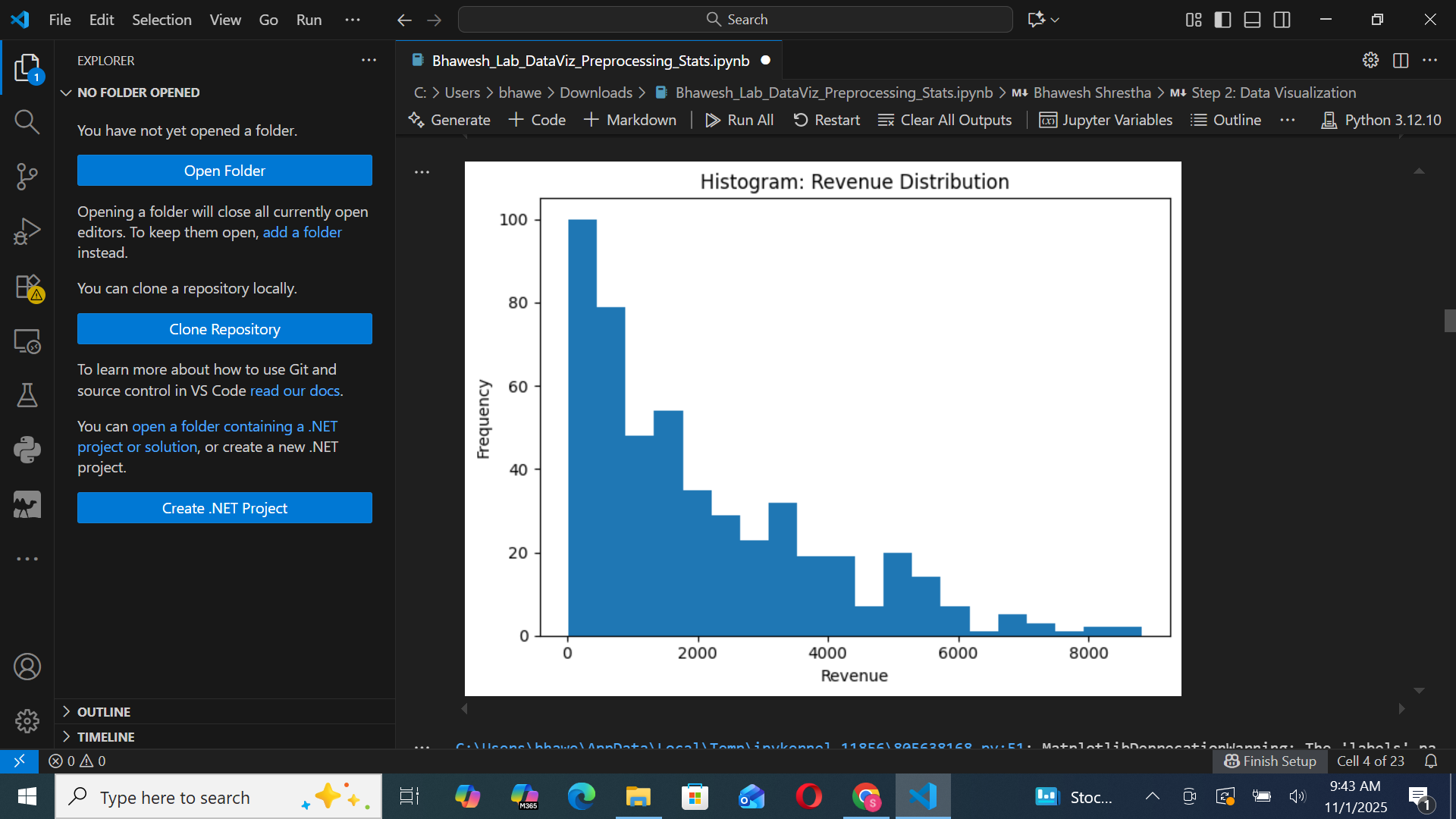
plt.tight\_layout()

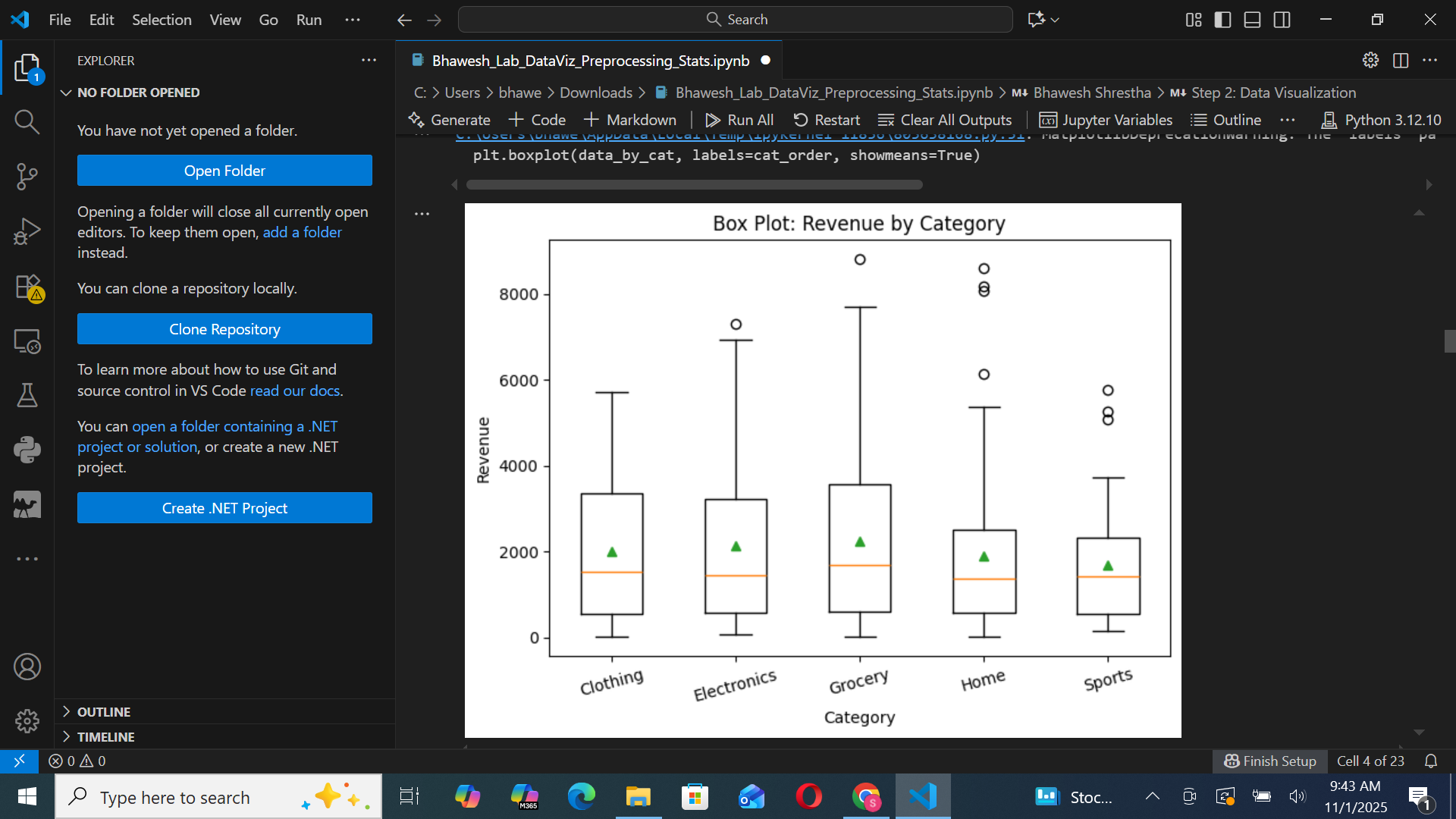
plt.show()

- Scatter Plot: Showed positive correlation between units\_sold and revenue.

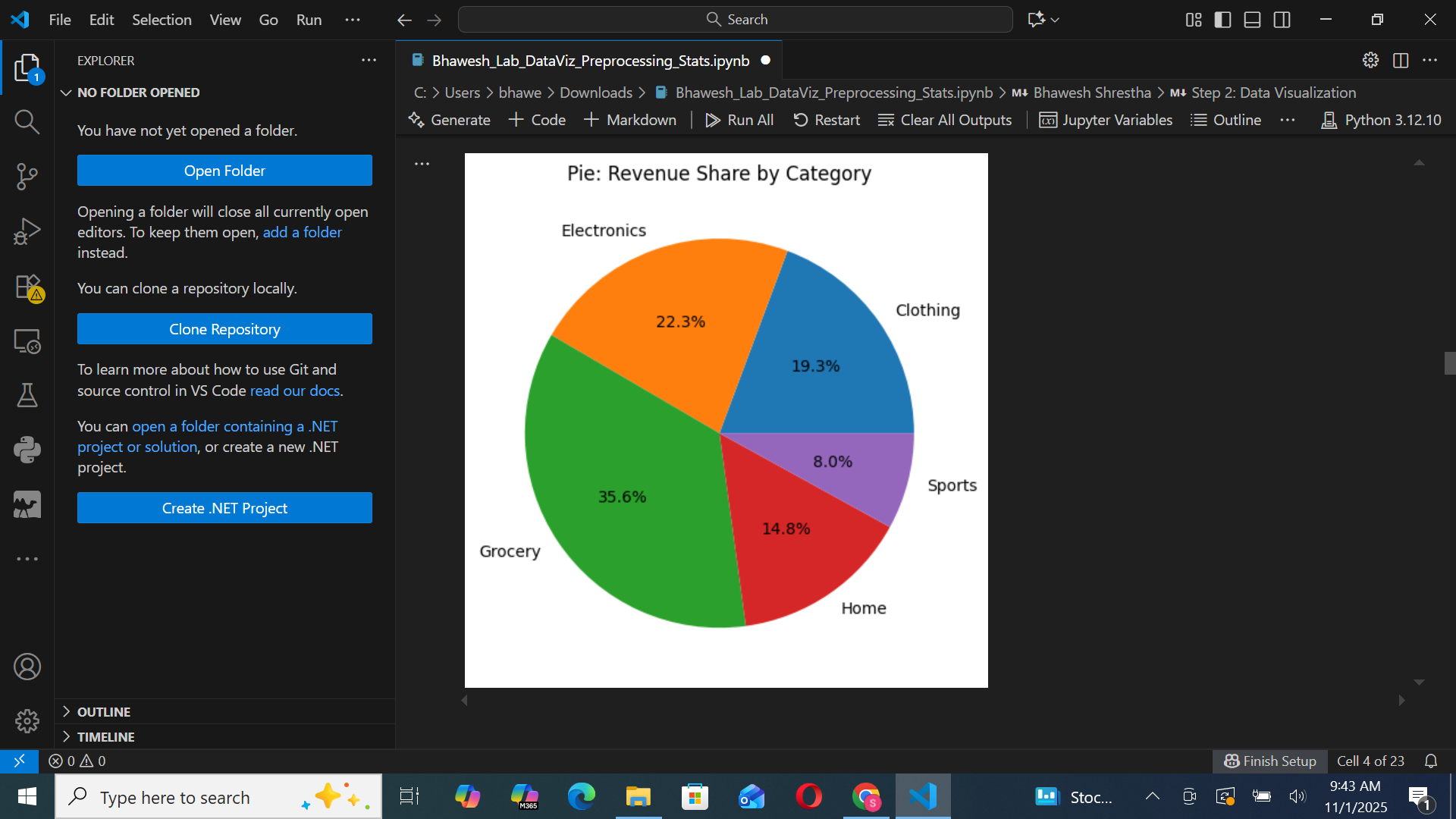
  
 - Line Plot: Revealed stable daily revenue trends with occasional spikes.

  
 - Histogram: Displayed right-skewed revenue distribution.

  
 - Box Plot: Identified revenue spread and potential outliers by category.



- Pie Chart: Illustrated proportional revenue by product category.



### **3. Data Preprocessing**

- Handled missing values: Filled numeric columns with mean values.

# Intentionally introduce a small amount of missingness for demonstration

df\_mv = df.copy()

mask\_rows = df\_mv.sample(frac=0.05, random\_state=7).index

df\_mv.loc[mask\_rows, 'discount'] = np.nan

df\_mv.loc[mask\_rows[:int(len(mask\_rows)/2)], 'unit\_price'] = np.nan

print("Missing values BEFORE:")

print(df\_mv.isna().sum())

# Strategy: fill numeric with mean, categorical with mode, leave date as-is

numeric\_cols = df\_mv.select\_dtypes(include=[np.number]).columns

cat\_cols = df\_mv.select\_dtypes(include=['object']).columns

df\_filled = df\_mv.copy()

for col in numeric\_cols:

df\_filled[col] = df\_filled[col].fillna(df\_filled[col].mean())

for col in cat\_cols:

if df\_filled[col].isna().any():

df\_filled[col] = df\_filled[col].fillna(df\_filled[col].mode()[0])

print("\nMissing values AFTER:")

print(df\_filled.isna().sum())

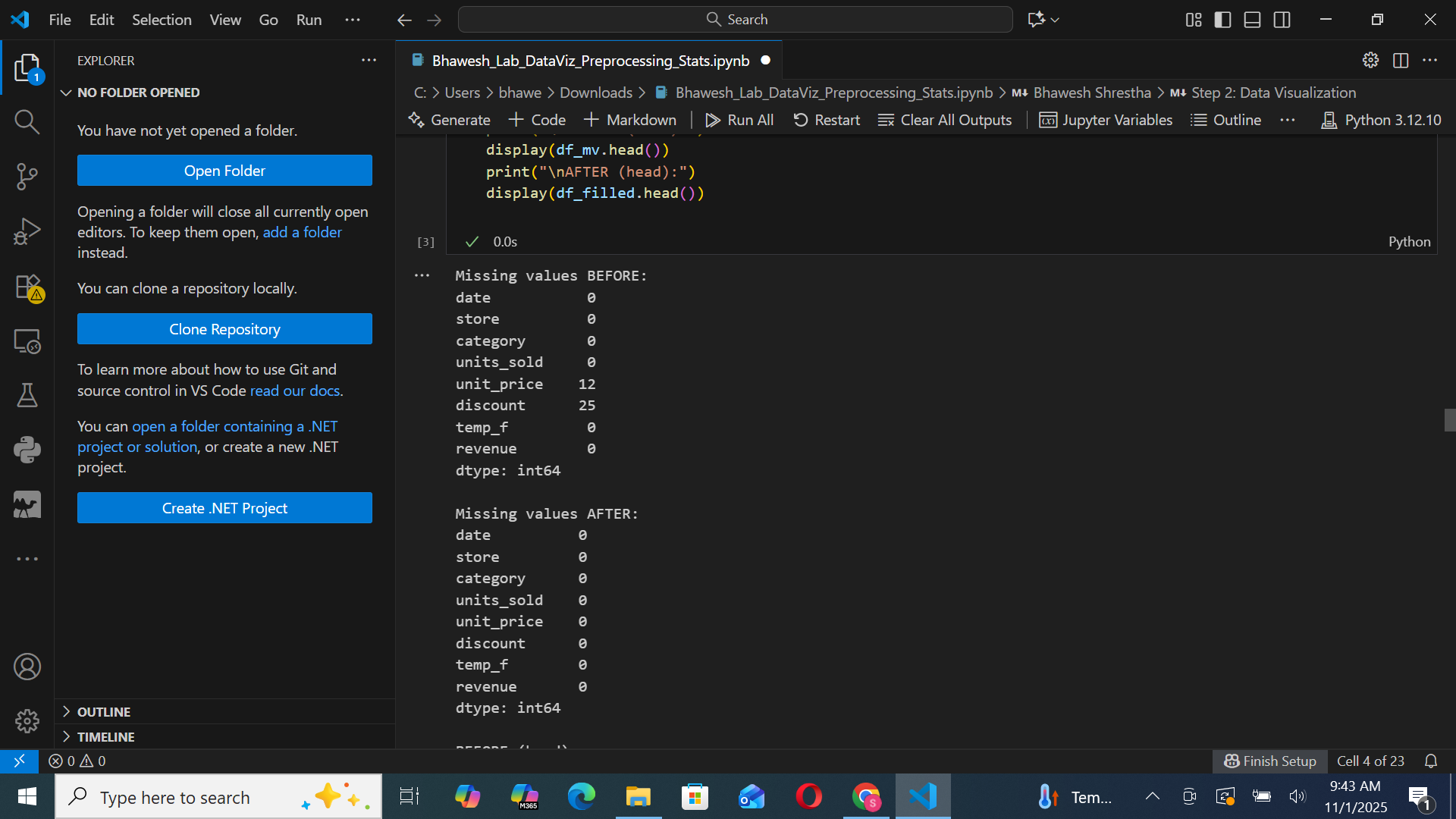
# Show head before/after for screenshot

print("\nBEFORE (head):")

display(df\_mv.head())

print("\nAFTER (head):")

display(df\_filled.head())

  
- Outlier removal: Used Interquartile Range (IQR) method to detect and drop extreme revenue values.

# Outlier detection on 'revenue' using IQR

Q1 = df\_filled['revenue'].quantile(0.25)

Q3 = df\_filled['revenue'].quantile(0.75)

IQR = Q3 - Q1

lower = Q1 - 1.5 \* IQR

upper = Q3 + 1.5 \* IQR

print(f"Q1: {Q1:.2f}, Q3: {Q3:.2f}, IQR: {IQR:.2f}")

print(f"Lower bound: {lower:.2f}, Upper bound: {upper:.2f}")

outliers\_mask = (df\_filled['revenue'] < lower) | (df\_filled['revenue'] > upper)

outliers = df\_filled[outliers\_mask]

print(f"Identified outliers: {len(outliers)}")

df\_no\_outliers = df\_filled[~outliers\_mask].copy()

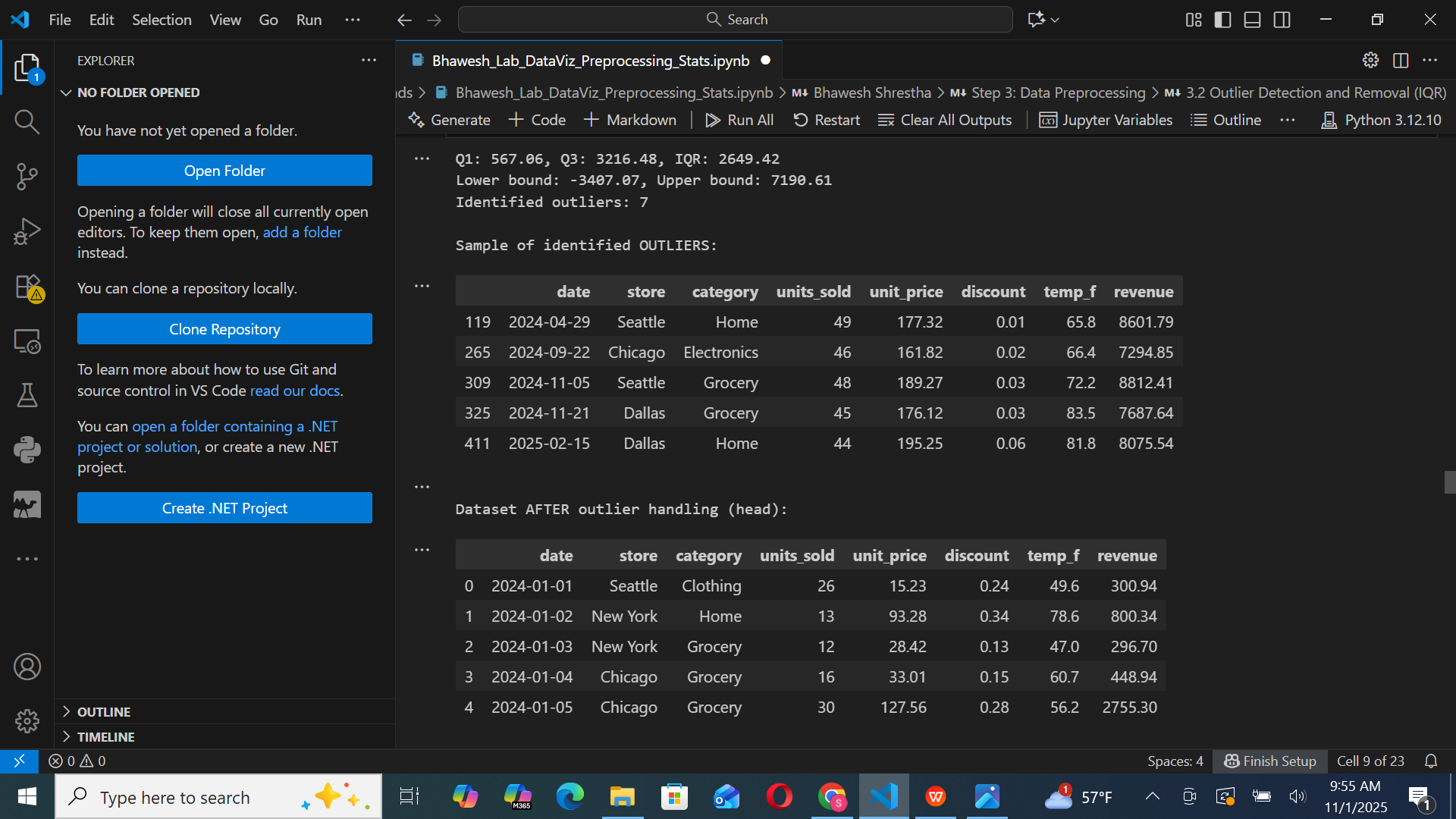
# Show a small sample for screenshots

print("\nSample of identified OUTLIERS:")

display(outliers.head())

print("\nDataset AFTER outlier handling (head):")

display(df\_no\_outliers.head())

  
- Data reduction: Applied sampling and removed less relevant columns.

# Sample 60% of the rows

df\_sampled = df\_no\_outliers.sample(frac=0.6, random\_state=11)

# Drop less relevant columns (example: drop temperature for analysis)

cols\_before = df\_no\_outliers.columns.tolist()

df\_reduced = df\_sampled.drop(columns=['temp\_f'])

cols\_after = df\_reduced.columns.tolist()

print("Columns BEFORE:", cols\_before)

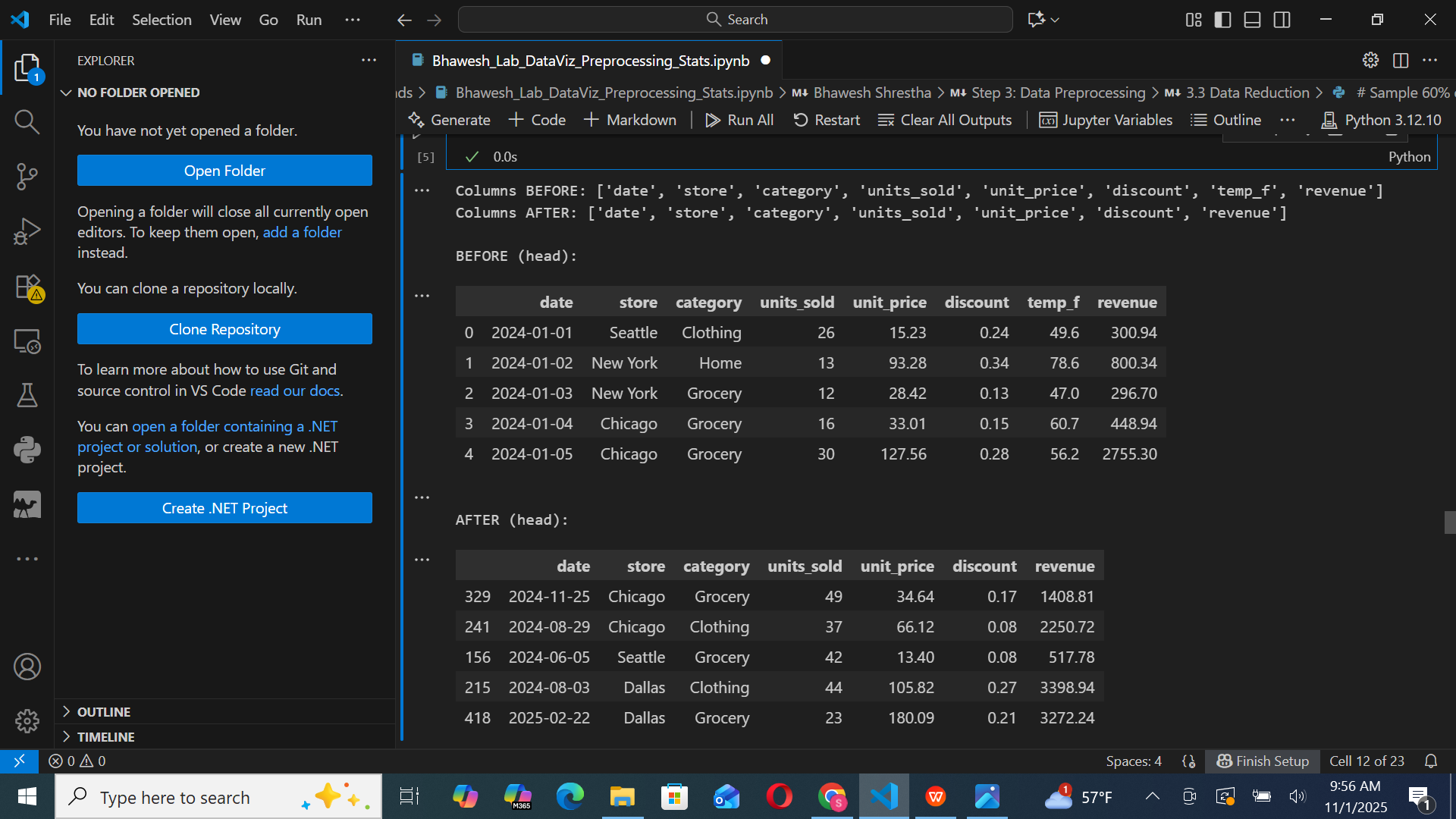
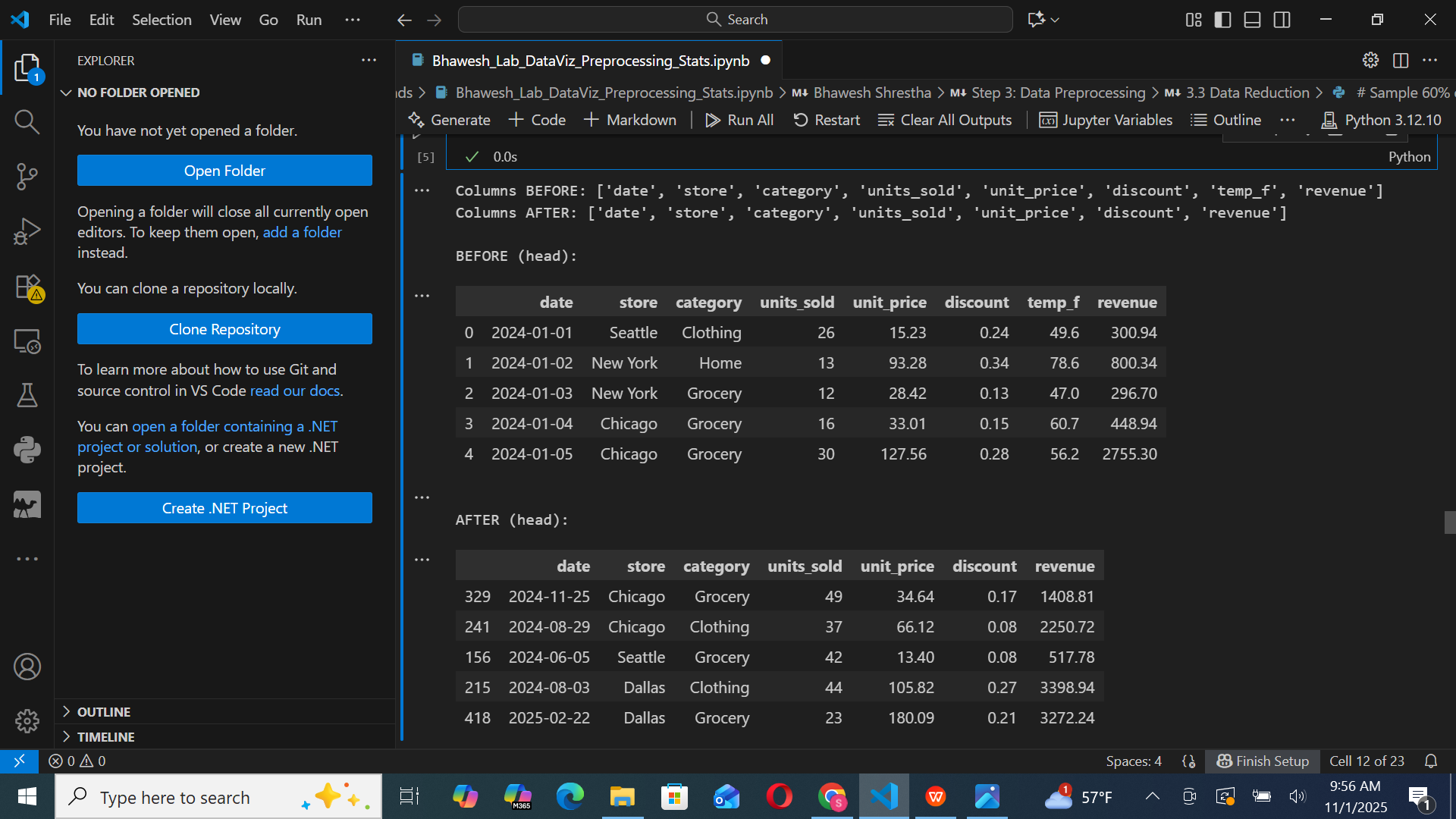
print("Columns AFTER:", cols\_after)

print("\nBEFORE (head):")

display(df\_no\_outliers.head())

print("\nAFTER (head):")

display(df\_reduced.head())

  
- Scaling & discretization: Performed Min–Max normalization and Z-score standardization, and categorized revenue into quartiles.

# Manual Min-Max scaling and Z-score (no sklearn dependency)

scaled = df\_reduced.copy()

for col in ['units\_sold', 'unit\_price', 'discount', 'revenue']:

min\_c, max\_c = scaled[col].min(), scaled[col].max()

scaled[f'{col}\_minmax'] = (scaled[col] - min\_c) / (max\_c - min\_c)

mean\_c, std\_c = scaled[col].mean(), scaled[col].std(ddof=1)

scaled[f'{col}\_zscore'] = (scaled[col] - mean\_c) / std\_c

# Discretize revenue into categories (quartiles)

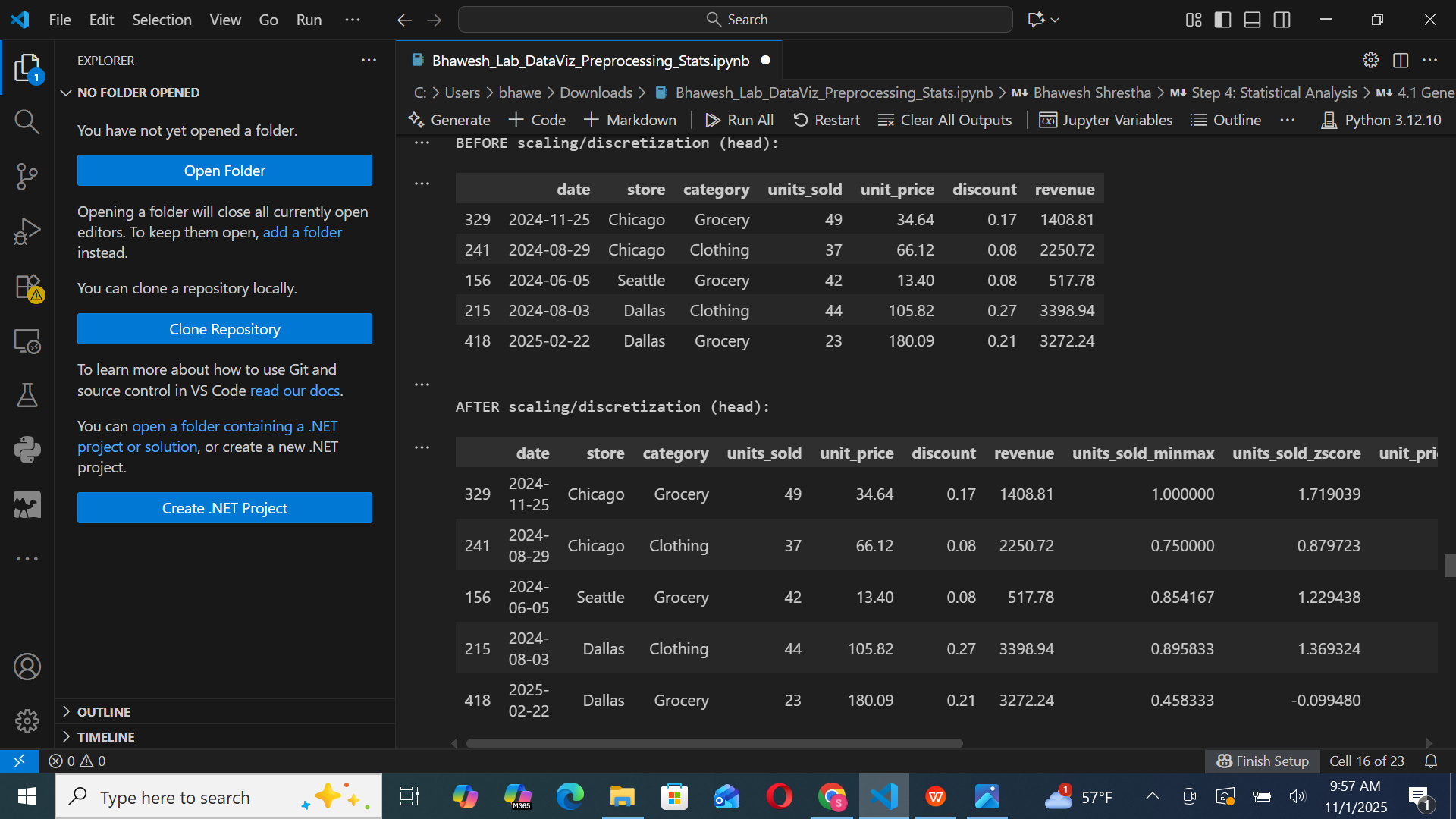
scaled['revenue\_bin'] = pd.qcut(scaled['revenue'], q=4, labels=['Low','Med-Low','Med-High','High'])

print("BEFORE scaling/discretization (head):")

display(df\_reduced.head())

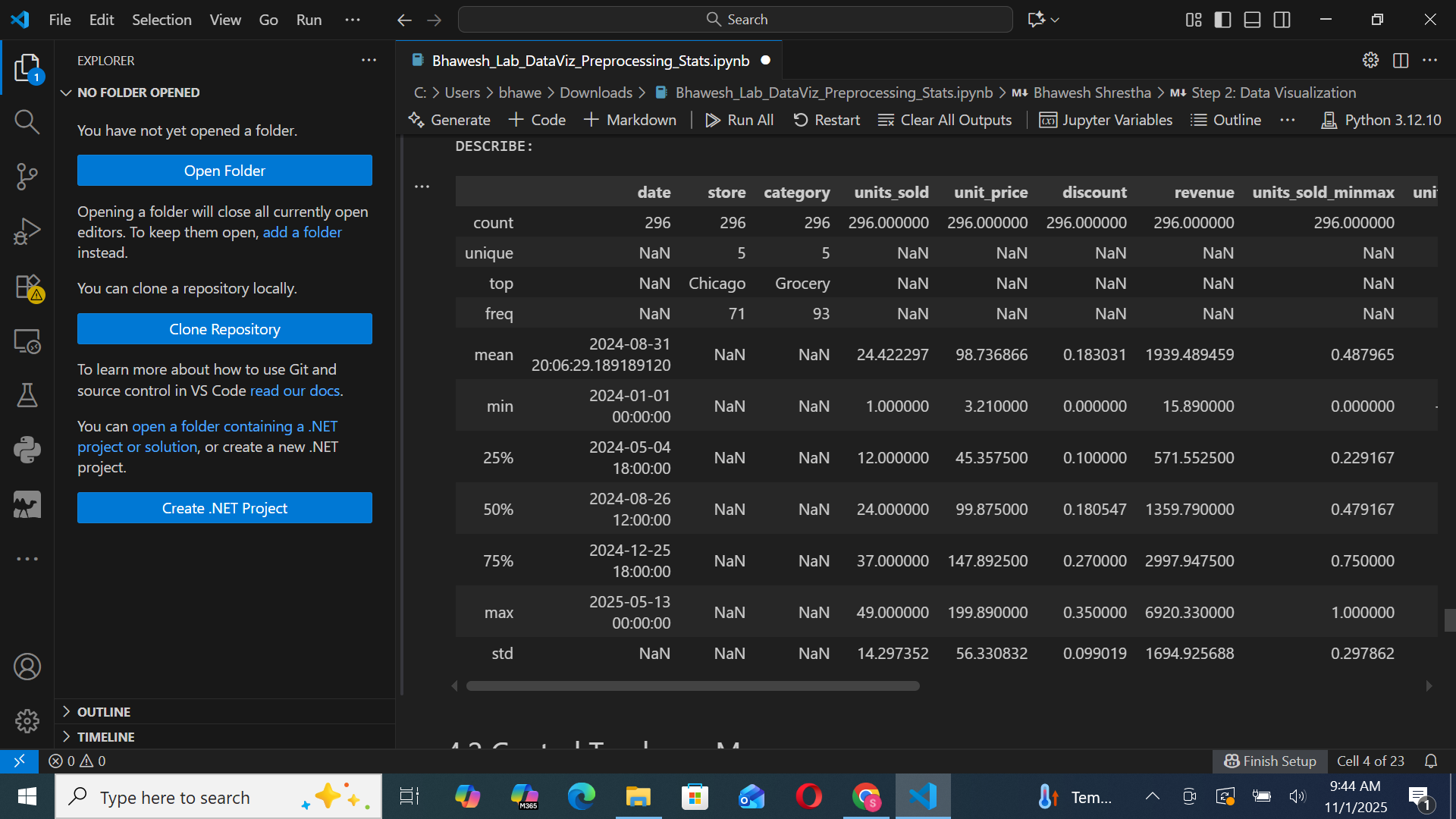
print("\nAFTER scaling/discretization (head):")

display(scaled.head())



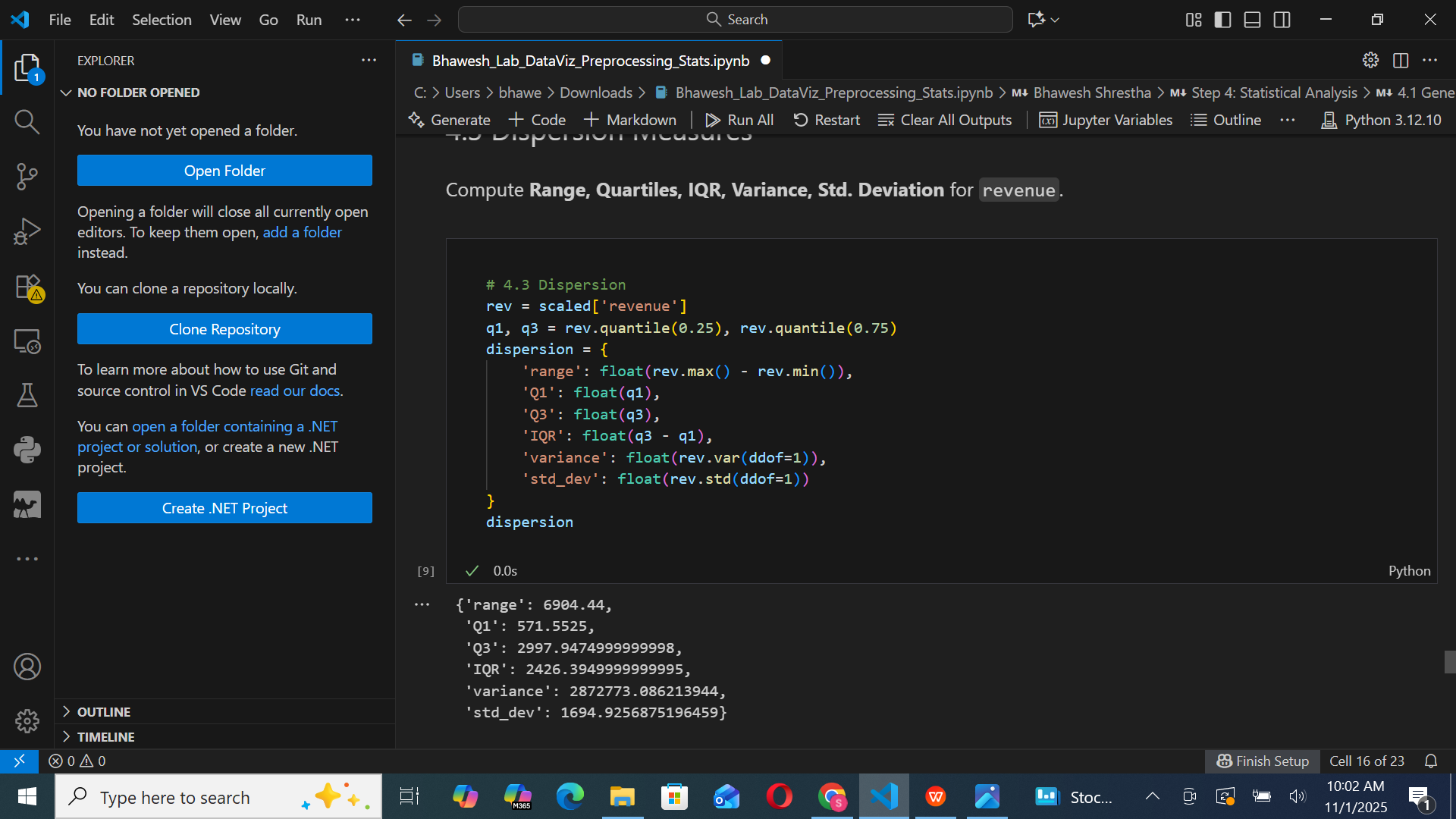
### **4. Statistical Analysis**

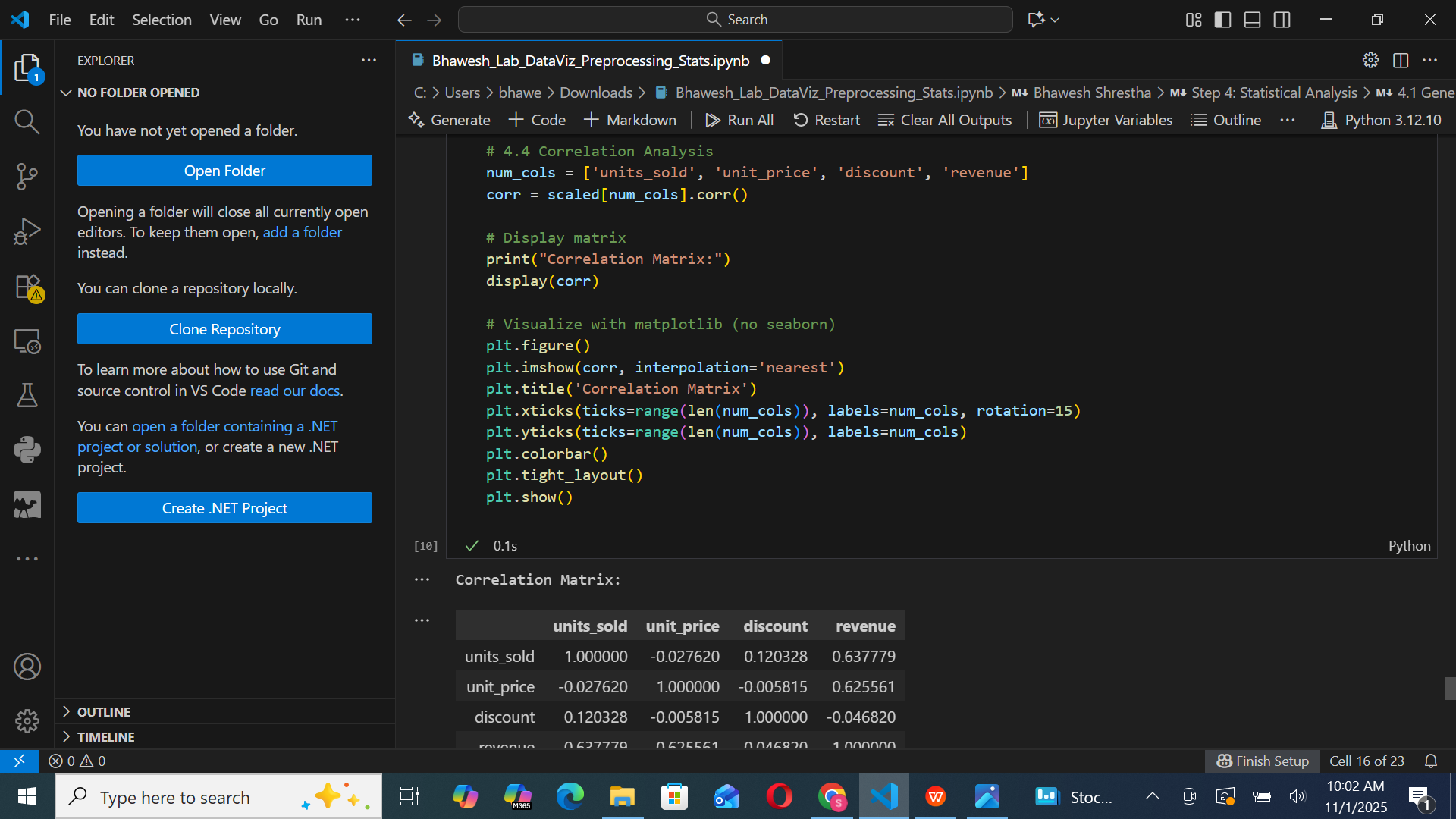
- General Overview of Data: Use .info() and .describe() to explore dataset characteristics.

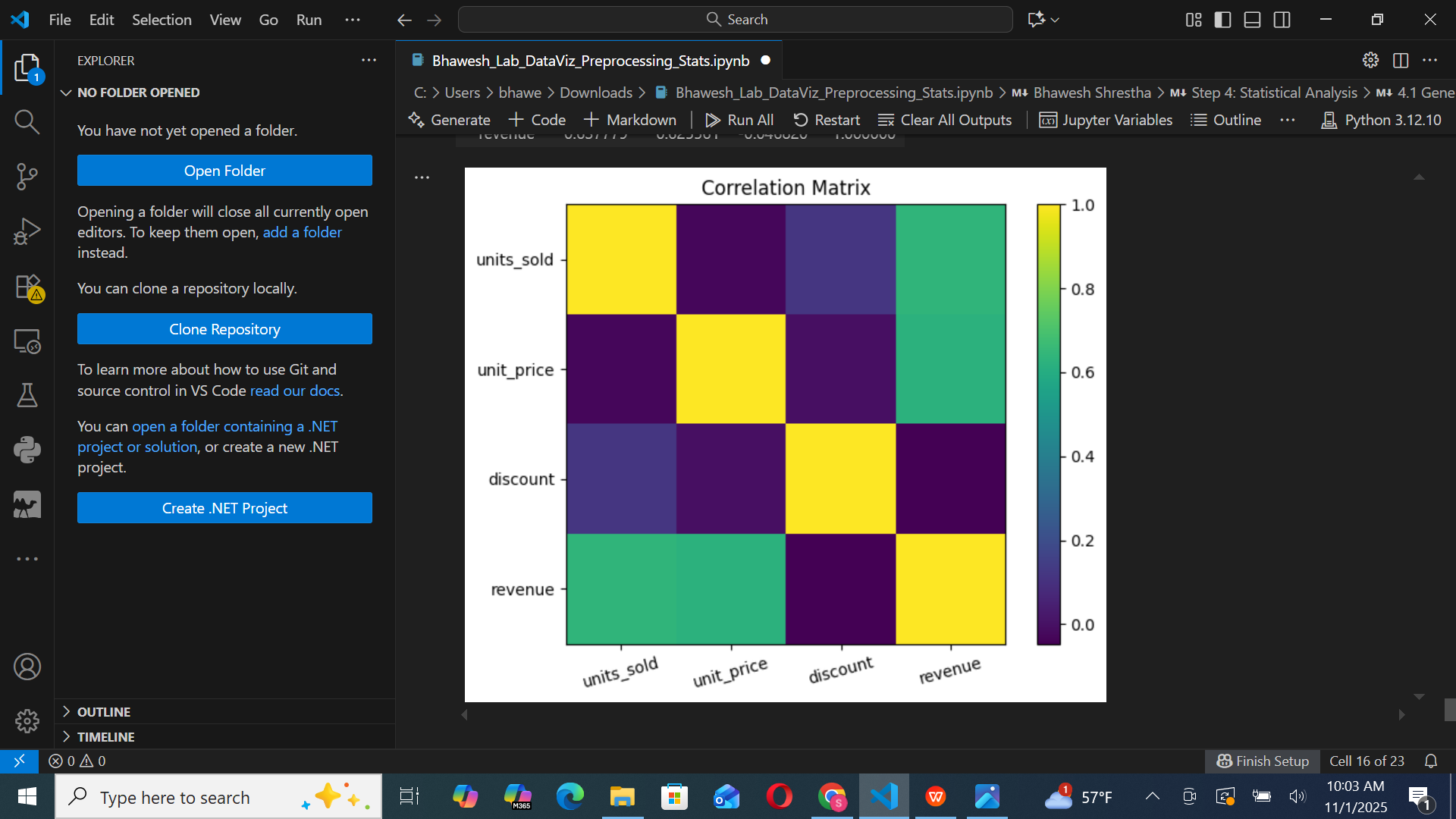


- Central Tendency: Calculated Min, Max, Mean, Median, and Mode.

  
- Dispersion: Found Range, IQR, Variance, and Standard Deviation.

  
- Correlation Analysis: Revealed strong relationships between units\_sold and revenue.





## **Key Insights**

- Sales–Revenue Relationship: Clear positive correlation between units sold and total revenue.  
- High Variability: Electronics category showed greater revenue variance.  
- Data Quality Impact: Outlier removal and scaling improved statistical clarity.  
- Normalization Benefit: Min–Max scaling ensured balanced comparison across variables.

## **Challenges and Decisions**

- Managed missing data using mean and mode to retain dataset integrity.  
- Balanced dataset reduction to simplify analysis without losing insights.  
- Ensured all plots were generated with Matplotlib only, avoiding external dependencies.

## **How to Run**

1. Clone the repository:  
 git clone <https://github.com/<your-username>/MSCS_634_Lab_1.git>2. Install dependencies:  
 pip install notebook pandas matplotlib numpy  
3. Open and run the notebook:  
 jupyter notebook Bhawesh\_Lab\_DataViz\_Preprocessing\_Stats.ipynb

## **Submission**

GitHub Repository Link: