**Data Cleansing:**

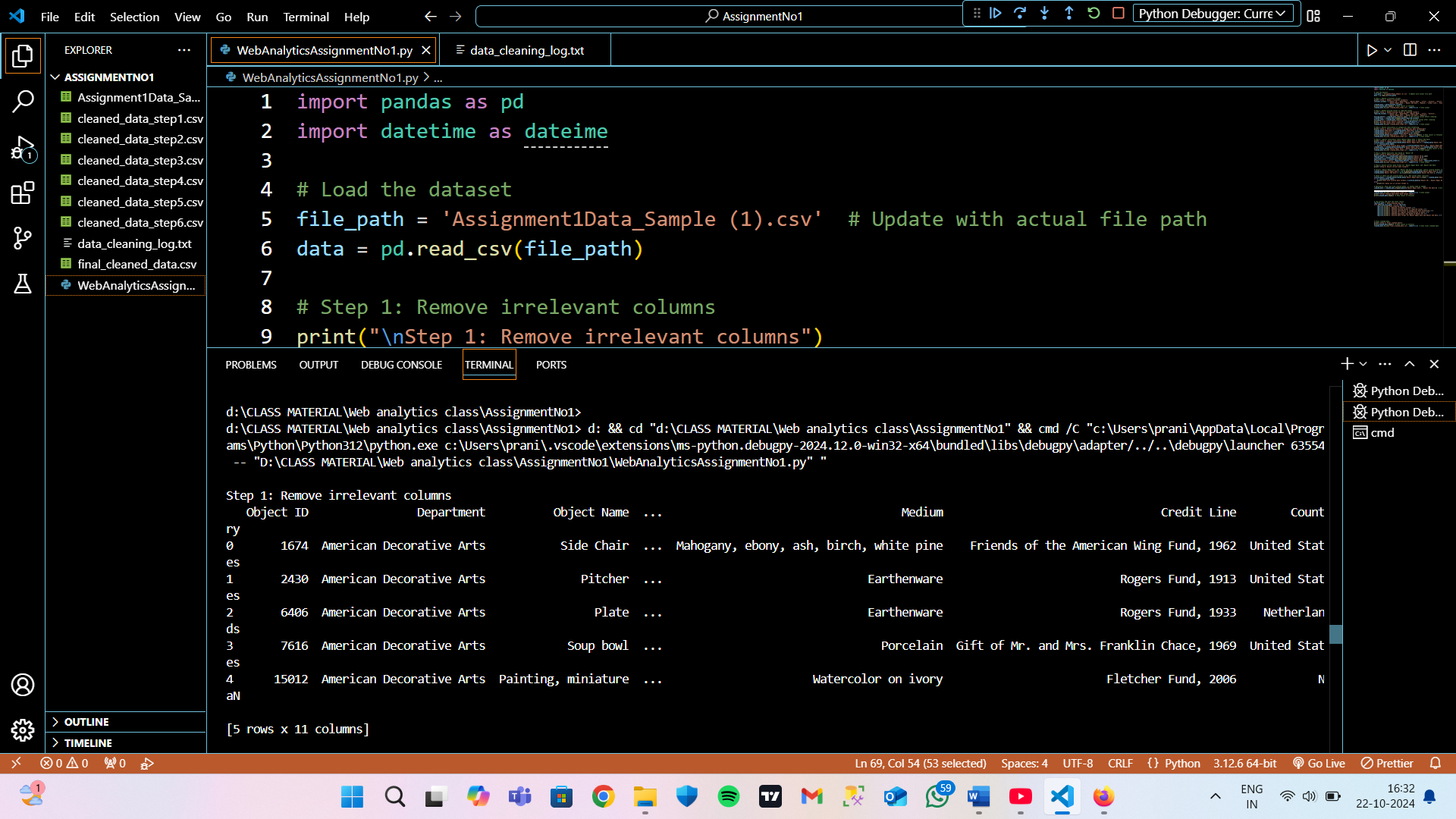
1. Shortlisting only the required columns.

Out of all the columns, we are going to work only on the following columns:

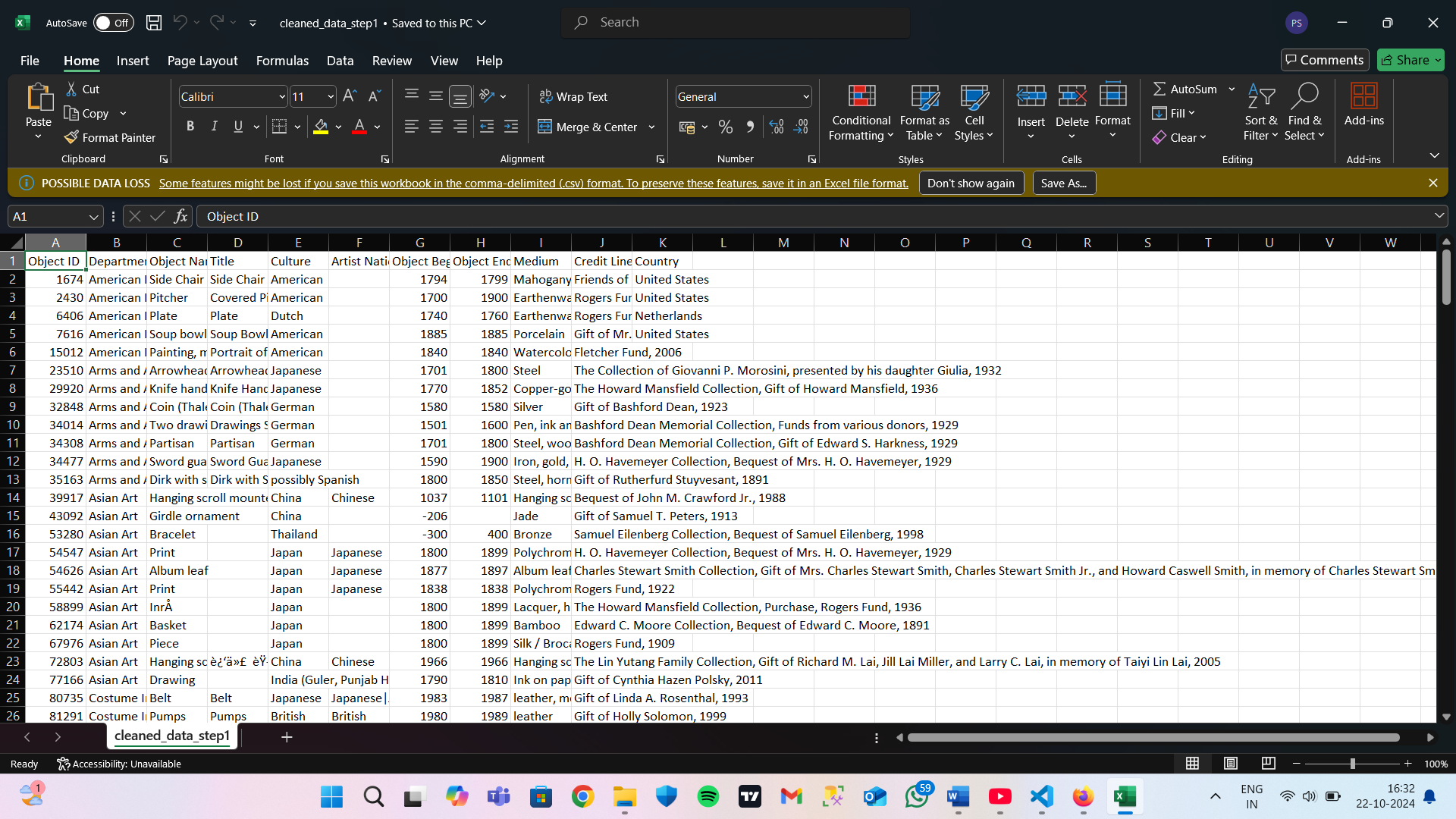
* Object ID
* Department
* Object Name
* Title
* Culture
* Artist Nationality
* Object Begin Date
* Object End Date
* Medium
* Credit Line
* Country

**Vs code terminal output:**

Here the print (df\_cleaned.head()) only displays 5 rows by default.



**Excel Sheet Output:**

****

**Python code:**

print("\nStep 1: Remove irrelevant columns")

relevant\_columns = ['Object ID', 'Department', 'Object Name', 'Title', 'Culture', 'Artist Nationality',

                    'Object Begin Date', 'Object End Date', 'Medium', 'Credit Line', 'Country']

cleaned\_data = data[relevant\_columns]

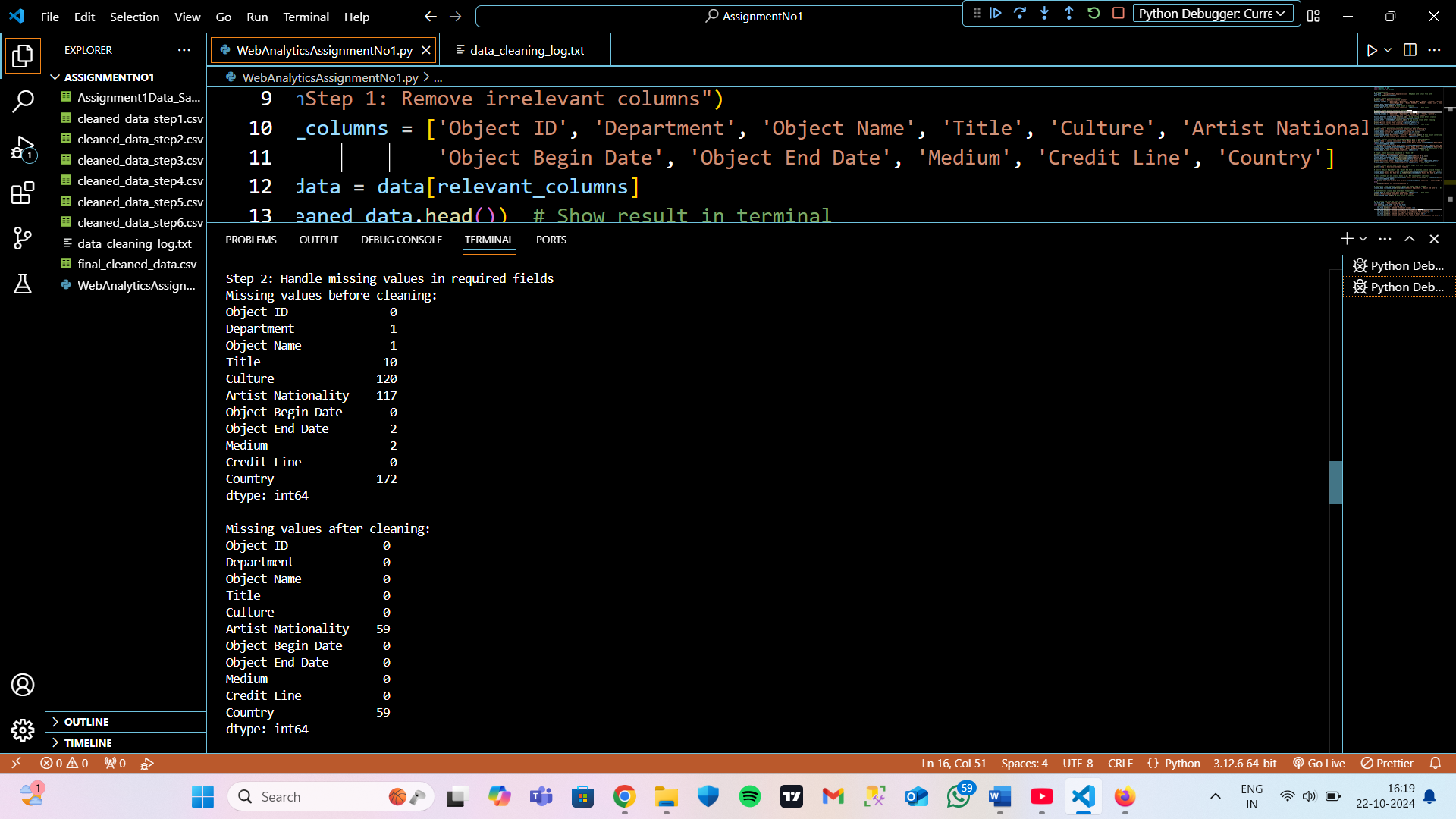
print(cleaned\_data.head())  # Show result in terminal

cleaned\_data.to\_csv('cleaned\_data\_step1.csv', index=False)  # Save output

**2. Removing rows that has missing values from the following columns:**

* Object ID
* Department
* Object Name
* Title
* Culture
* Object Begin Date
* Object End Date
* Medium

**Vs code terminal output:**



**Excel Sheet output:**

****

**Python code:**

print("\nStep 2: Handle missing values in required fields")

required\_columns = ['Object ID', 'Department', 'Object Name', 'Title', 'Culture',

                    'Object Begin Date', 'Object End Date', 'Medium']

missing\_before = cleaned\_data.isnull().sum()  # Log missing values before cleaning

cleaned\_data = cleaned\_data.dropna(subset=required\_columns)

missing\_after = cleaned\_data.isnull().sum()  # Log missing values after cleaning

print(f"Missing values before cleaning:\n{missing\_before}\n")

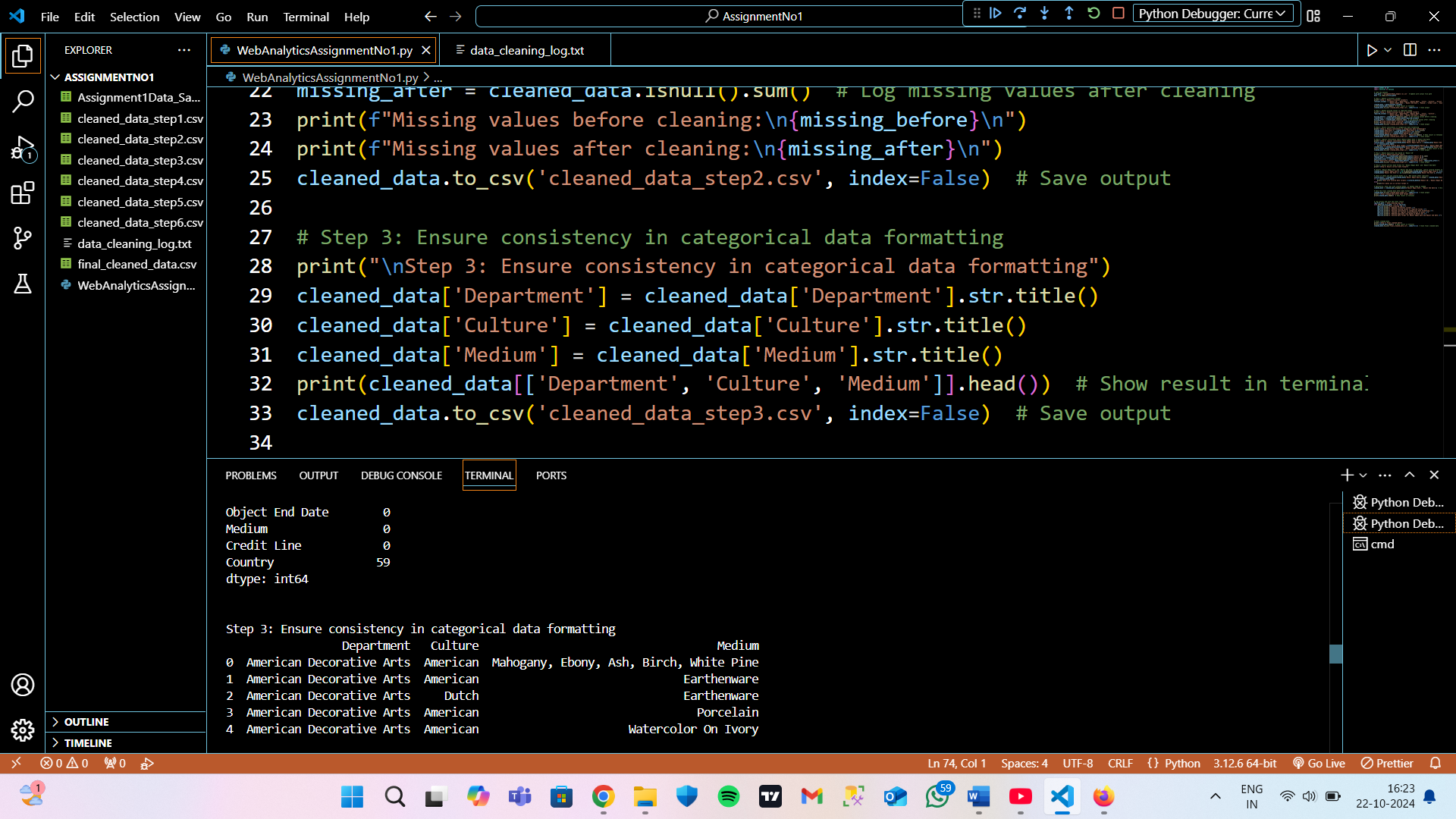
print(f"Missing values after cleaning:\n{missing\_after}\n")

cleaned\_data.to\_csv('cleaned\_data\_step2.csv', index=False)  # Save output

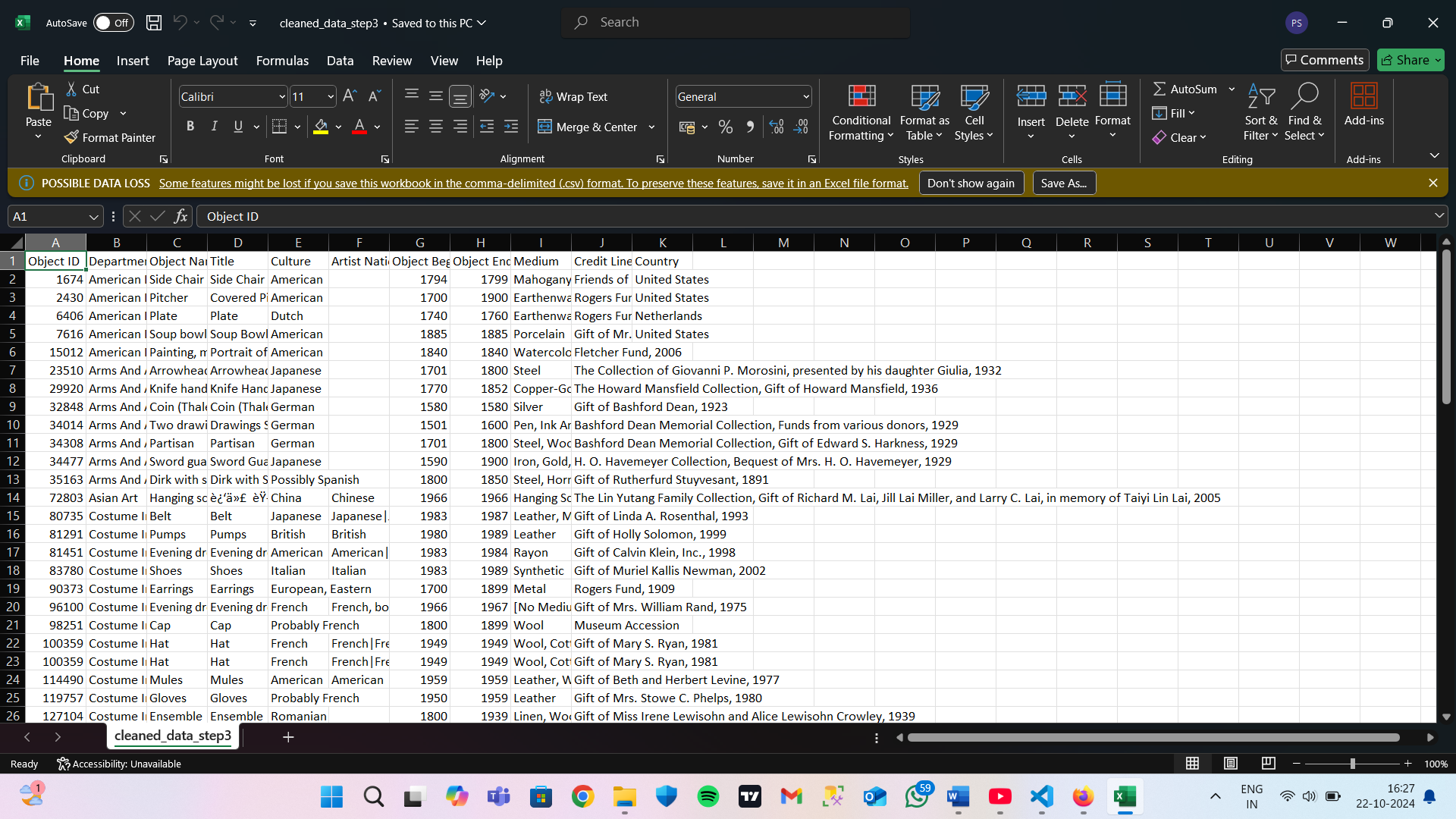
**3. Capitalising the first letter of each data from the following columns to ensure consistency.**

* Department
* Culture
* Medium

**Vs code terminal output:**



**Excel sheet output:**



**Python code:**

print("\nStep 3: Ensure consistency in categorical data formatting")

cleaned\_data['Department'] = cleaned\_data['Department'].str.title()

cleaned\_data['Culture'] = cleaned\_data['Culture'].str.title()

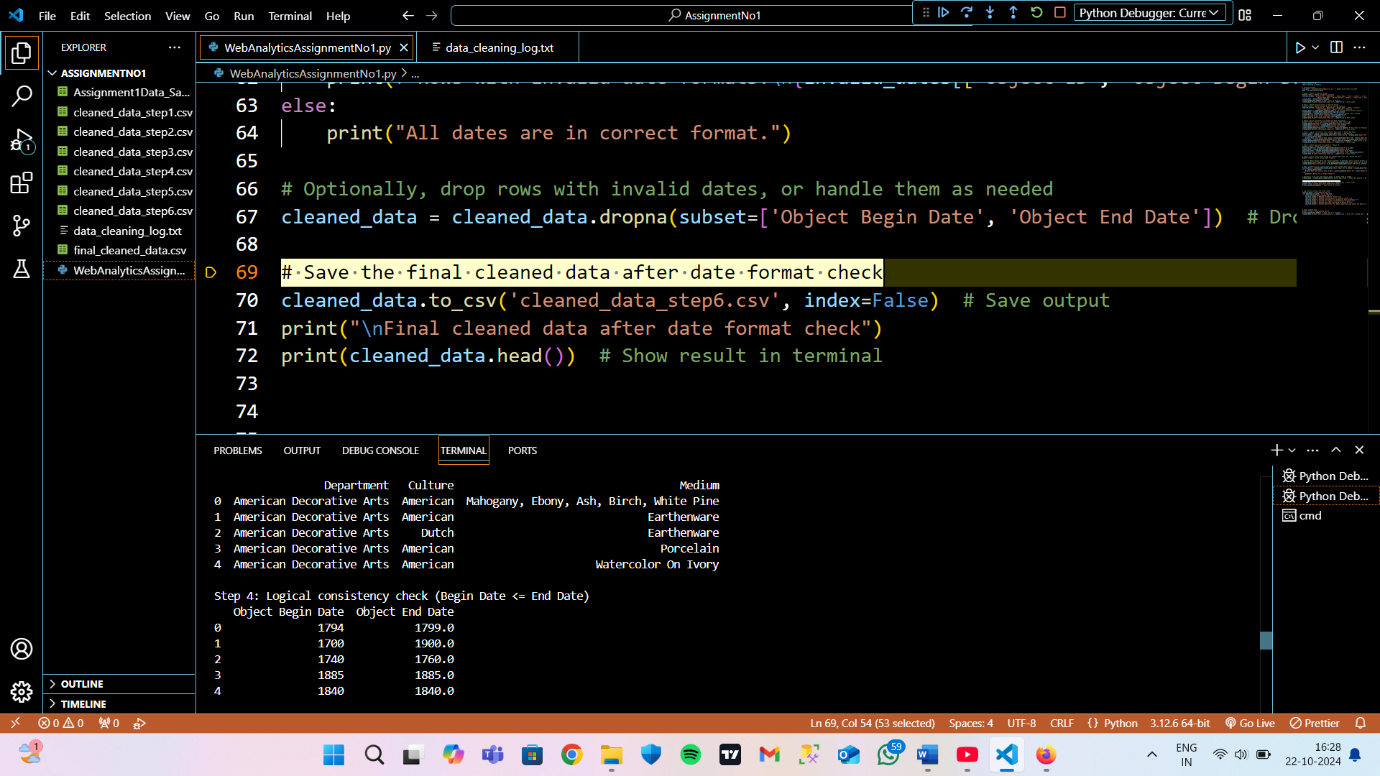
cleaned\_data['Medium'] = cleaned\_data['Medium'].str.title()

print(cleaned\_data[['Department', 'Culture', 'Medium']].head())  # Show result in terminal

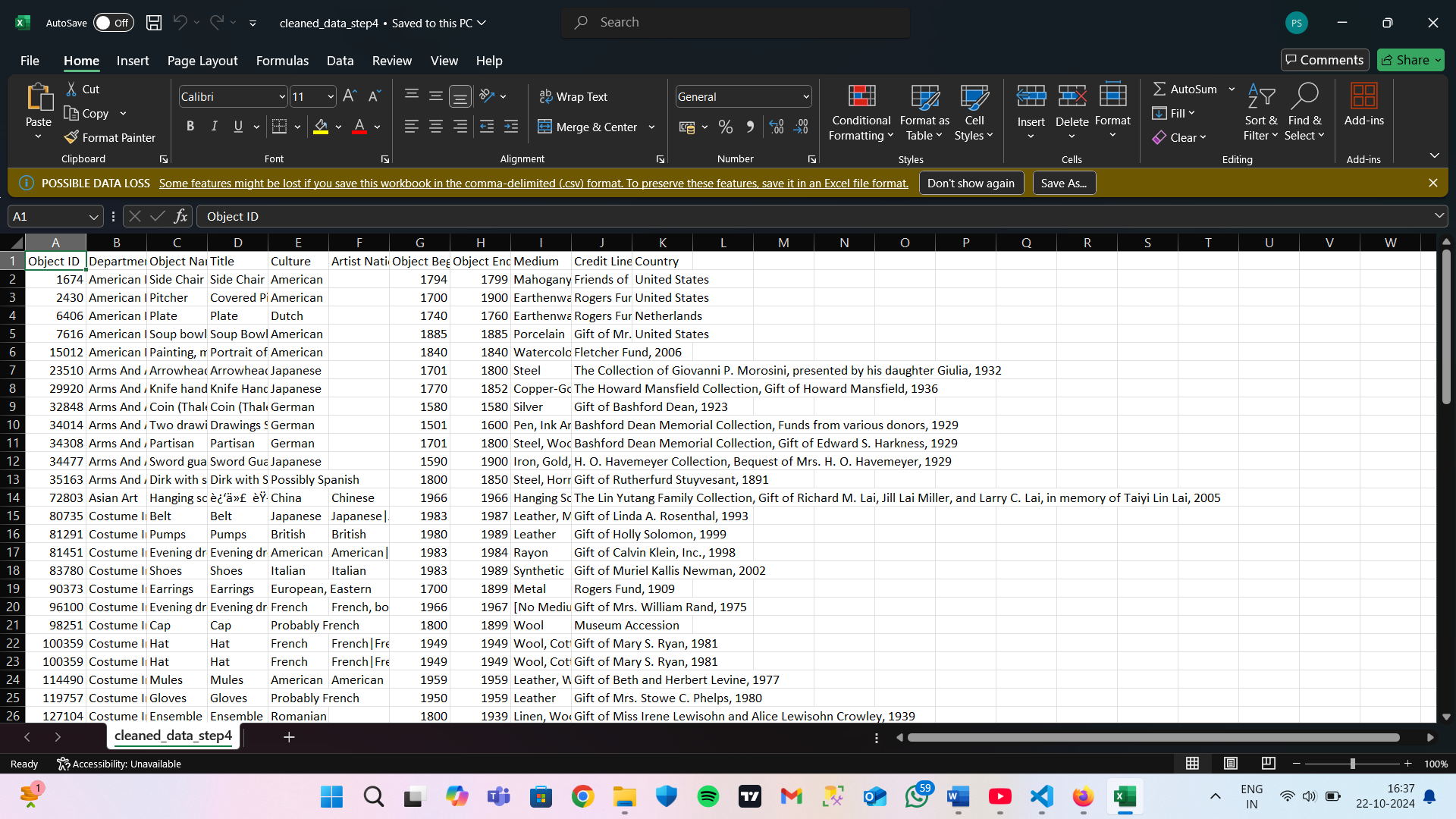
cleaned\_data.to\_csv('cleaned\_data\_step3.csv', index=False)  # Save output

**4. Applying logical consistency check by shortlisting only those rows where object begin date is less than or equal to object end date.**

**Vs code terminal output:**



**Excel sheet output:**



**Python code:**

print("\nStep 4: Logical consistency check (Begin Date <= End Date)")

invalid\_dates = cleaned\_data[cleaned\_data['Object Begin Date'] > cleaned\_data['Object End Date']]

if not invalid\_dates.empty:

    print(f"Rows with invalid date ranges:\n{invalid\_dates[['Object ID', 'Object Begin Date', 'Object End Date']]}")

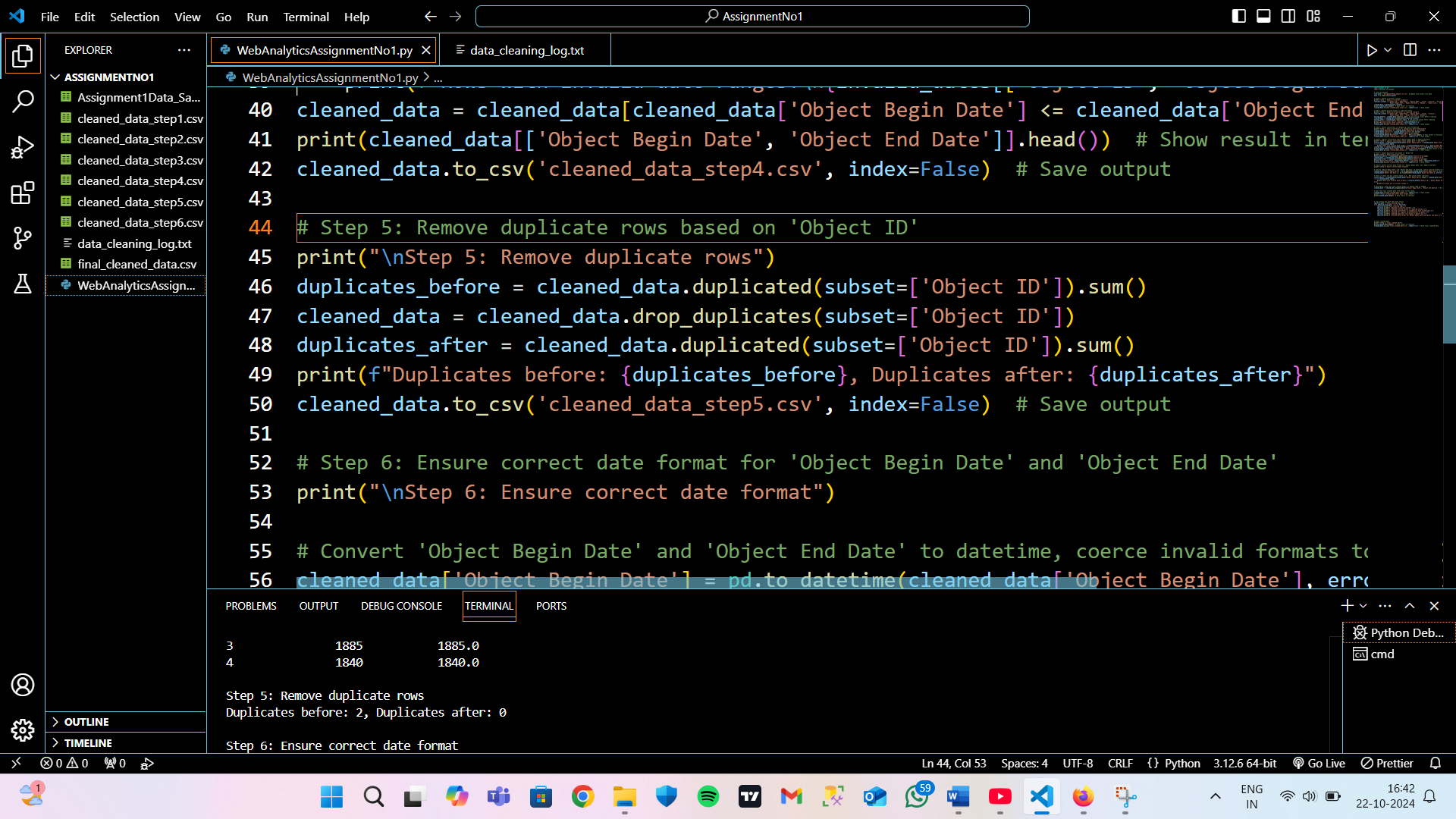
cleaned\_data = cleaned\_data[cleaned\_data['Object Begin Date'] <= cleaned\_data['Object End Date']]

print(cleaned\_data[['Object Begin Date', 'Object End Date']].head())  # Show result in terminal

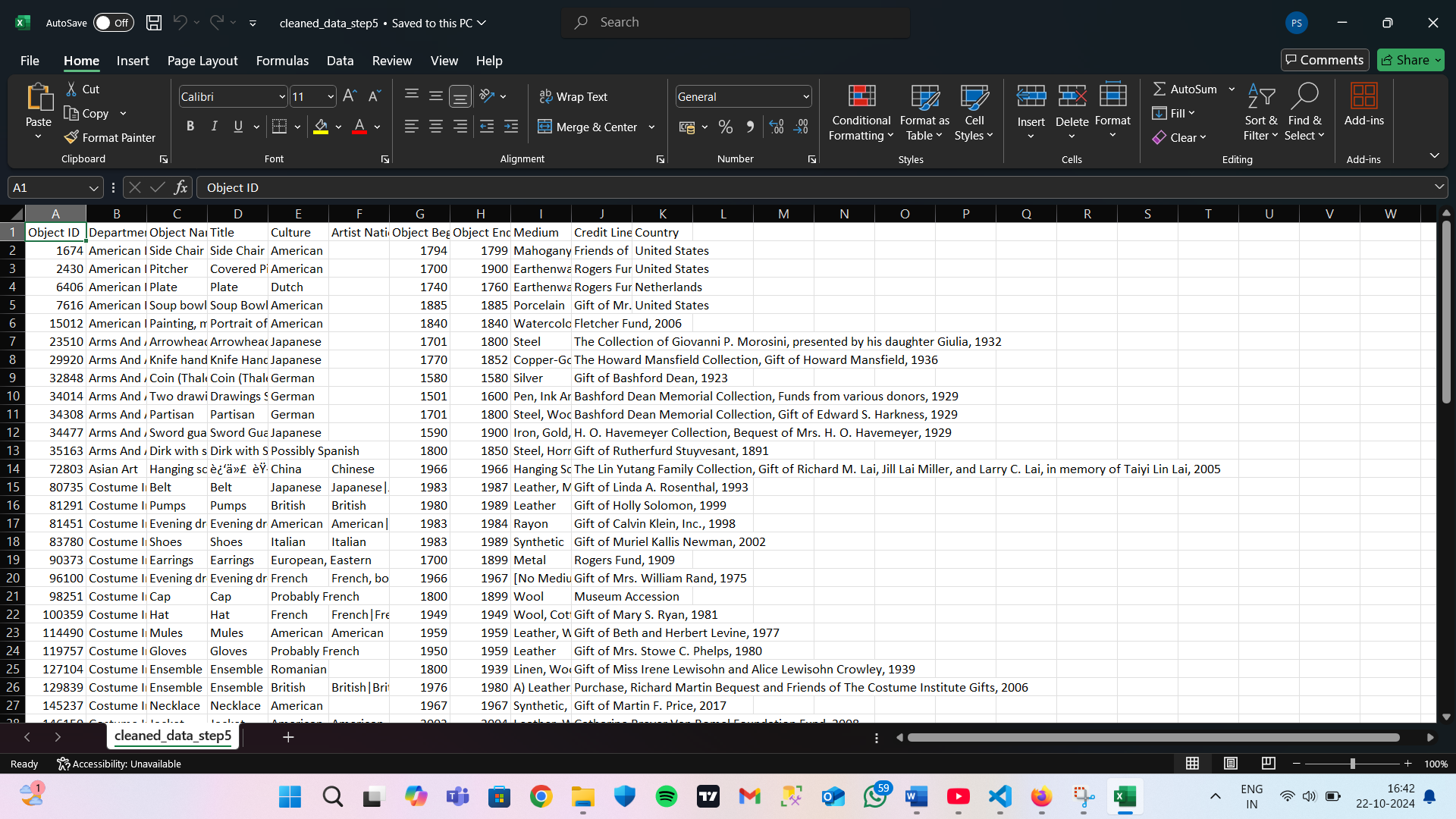
cleaned\_data.to\_csv('cleaned\_data\_step4.csv', index=False)  # Save output

**5. Removing duplicated rows in the ‘Object ID’ column.**

**Vs code terminal output**



**Excel output:**



**Python code:**

print("\nStep 5: Remove duplicate rows")

duplicates\_before = cleaned\_data.duplicated(subset=['Object ID']).sum()

cleaned\_data = cleaned\_data.drop\_duplicates(subset=['Object ID'])

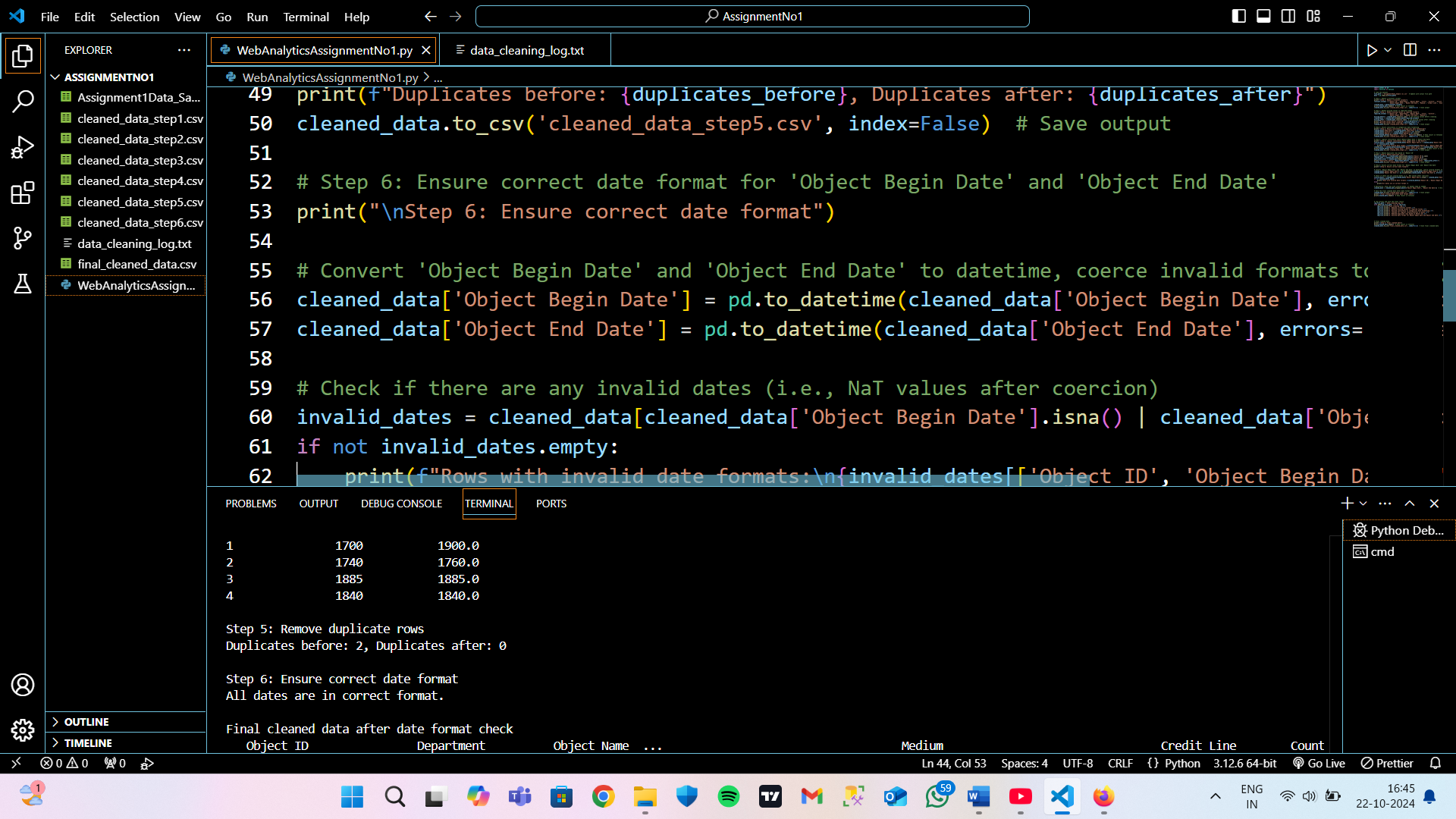
duplicates\_after = cleaned\_data.duplicated(subset=['Object ID']).sum()

print(f"Duplicates before: {duplicates\_before}, Duplicates after: {duplicates\_after}")

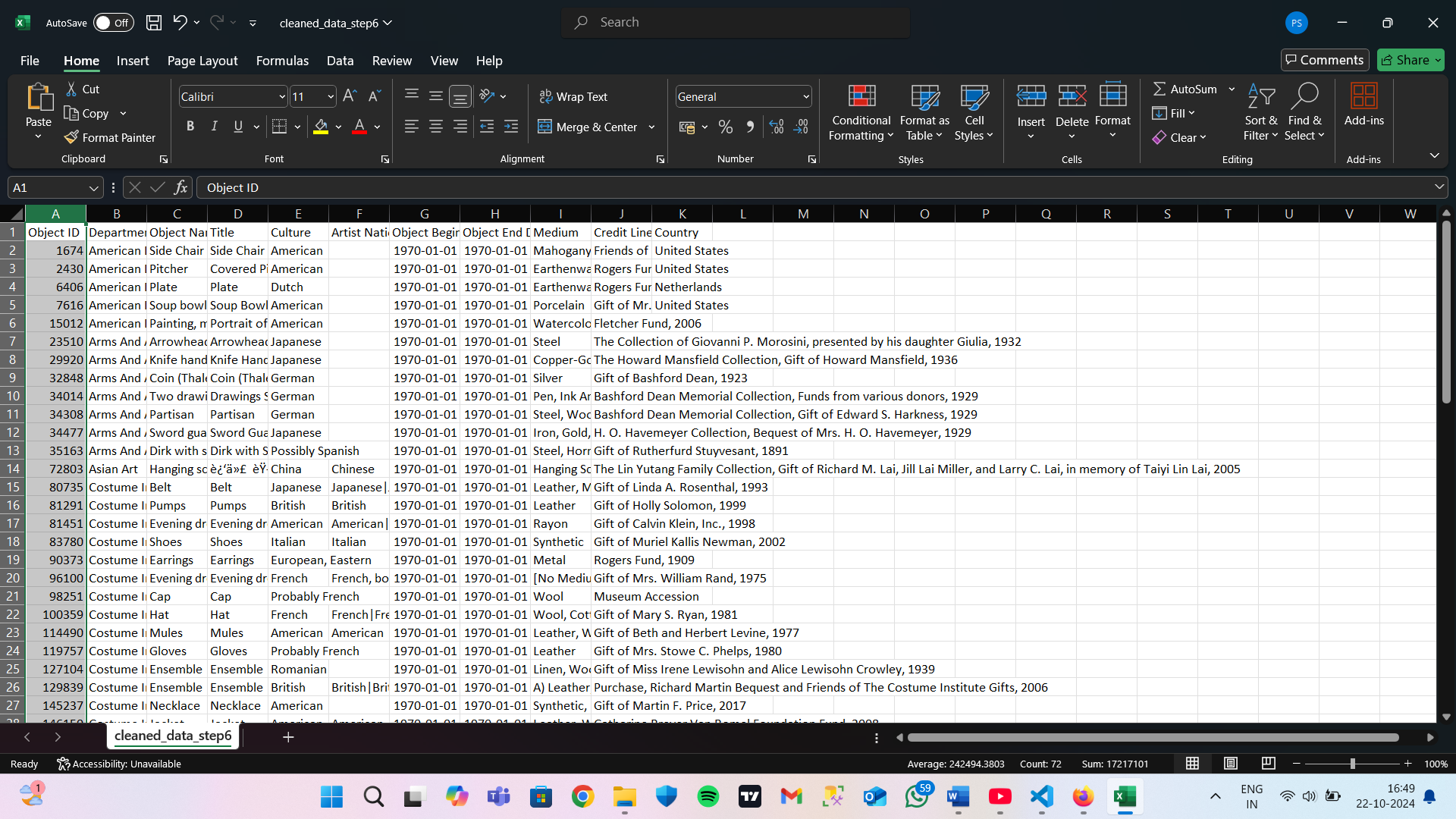
cleaned\_data.to\_csv('cleaned\_data\_step5.csv', index=False)  # Save output

**6. Converting Object begin date and Object end date in correct date format.**

**Vs code terminal output:**



**Excel sheet output:**



**Python code:**

print("\nStep 6: Ensure correct date format")

# Convert 'Object Begin Date' and 'Object End Date' to datetime, coerce invalid formats to NaT (Not a Time)

cleaned\_data['Object Begin Date'] = pd.to\_datetime(cleaned\_data['Object Begin Date'], errors='coerce').dt.date

cleaned\_data['Object End Date'] = pd.to\_datetime(cleaned\_data['Object End Date'], errors='coerce').dt.date

# Check if there are any invalid dates (i.e., NaT values after coercion)

invalid\_dates = cleaned\_data[cleaned\_data['Object Begin Date'].isna() | cleaned\_data['Object End Date'].isna()]

if not invalid\_dates.empty:

    print(f"Rows with invalid date formats:\n{invalid\_dates[['Object ID', 'Object Begin Date', 'Object End Date']]}")

else:

    print("All dates are in correct format.")

# Optionally, drop rows with invalid dates, or handle them as needed

cleaned\_data = cleaned\_data.dropna(subset=['Object Begin Date', 'Object End Date'])  # Drop rows with invalid dates

# Save the final cleaned data after date format check

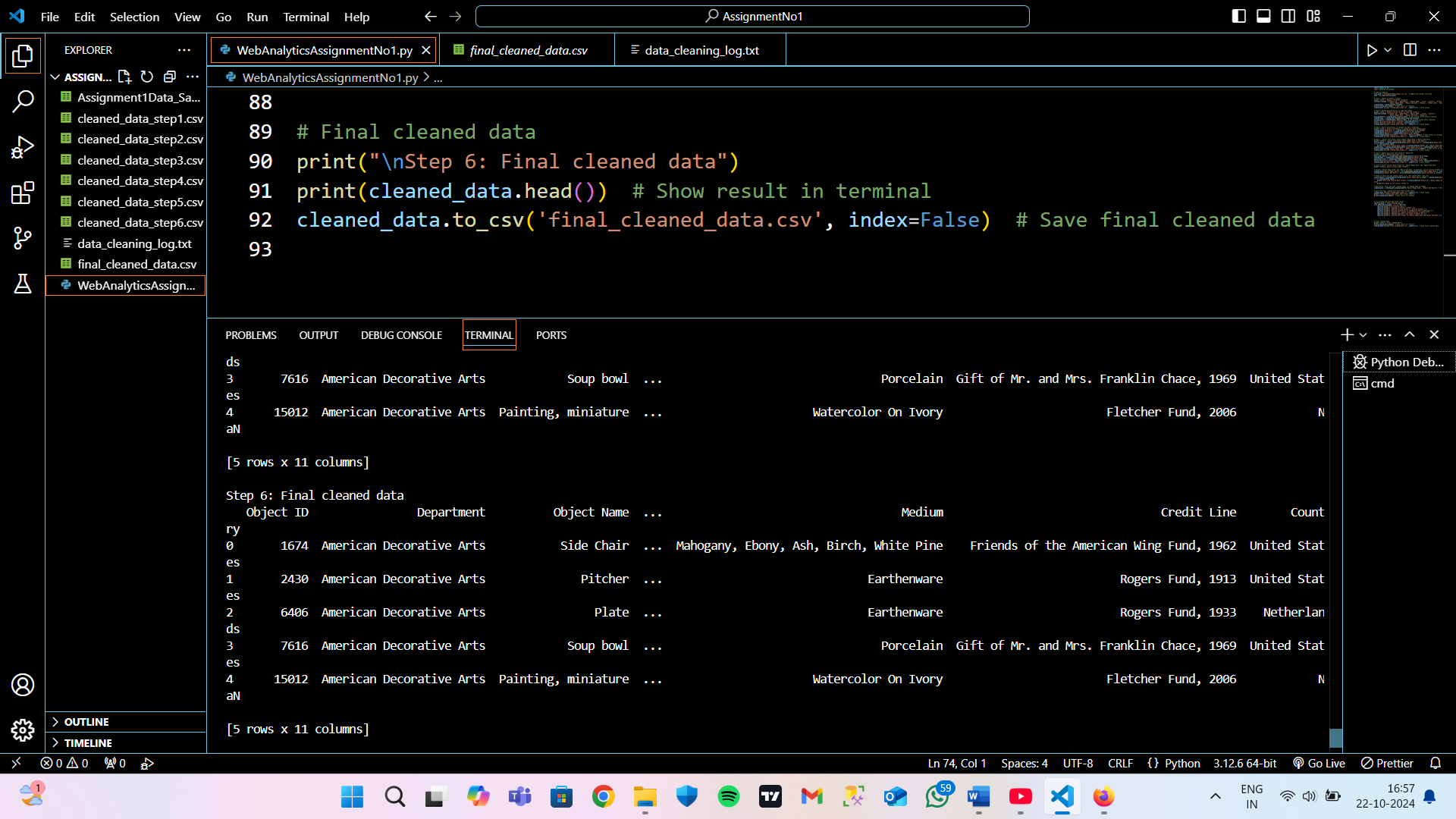
cleaned\_data.to\_csv('cleaned\_data\_step6.csv', index=False)  # Save output

print("\nFinal cleaned data after date format check")

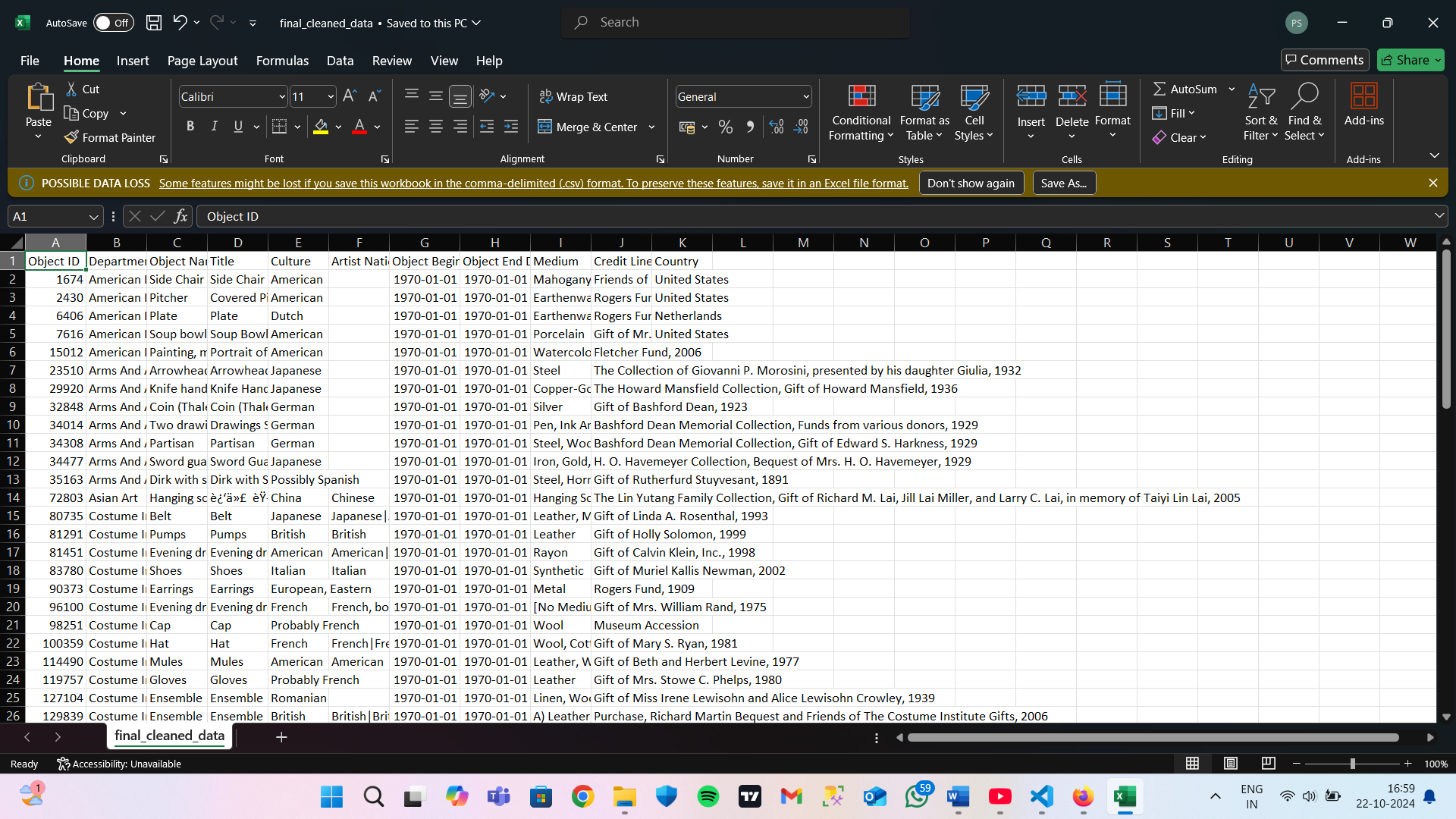
print(cleaned\_data.head())  # Show result in terminal

**FINAL CLEANED DATA:**

**Vs code output:**



**Excel sheet output:**



**Python code:**

print("\nStep 6: Final cleaned data")

print(cleaned\_data.head())  # Show result in terminal

cleaned\_data.to\_csv('final\_cleaned\_data.csv', index=False)  # Save final cleaned data