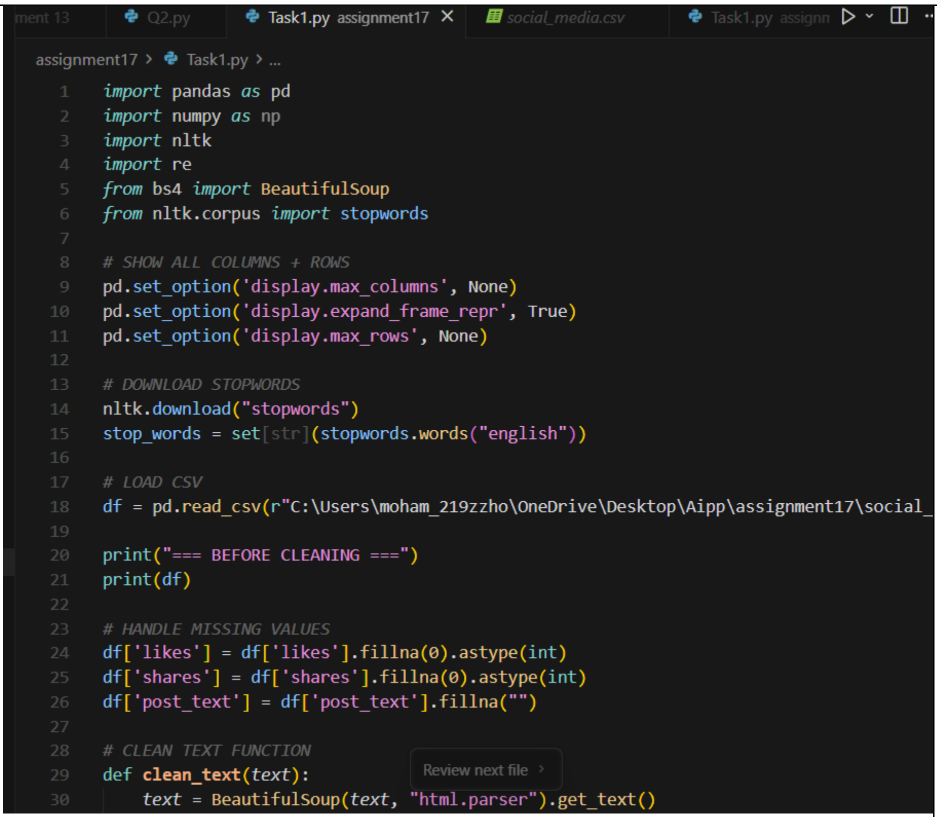
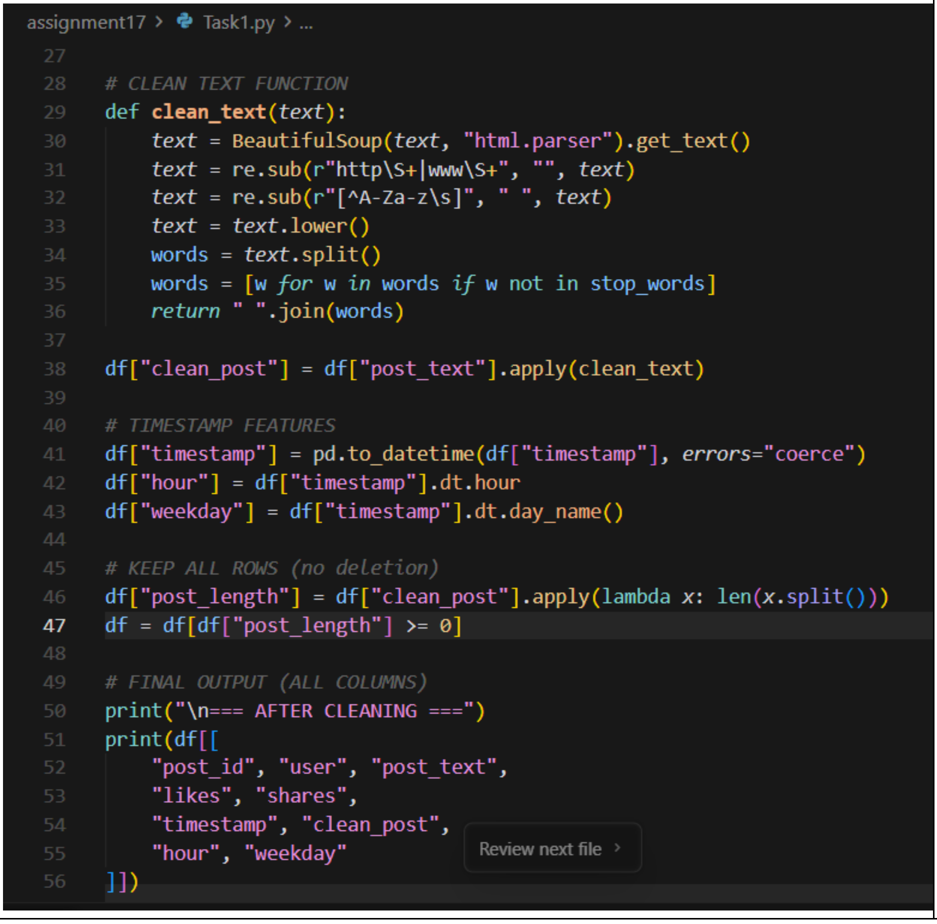
SCHOOLOFCOMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENTOFCOMPUTER SCIENCE ENGINEERING	
ProgramName:M. Tech/MSC/MCA		**AssignmentType: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
CourseCode		**CourseTitle**	AI Assisted Coding
Year/Sem	I/I	**Regulation**	R25
DateandDay of Assignment		**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:17.3(Presentassignmentnumber)/**24**(Totalnumberofassignments)			

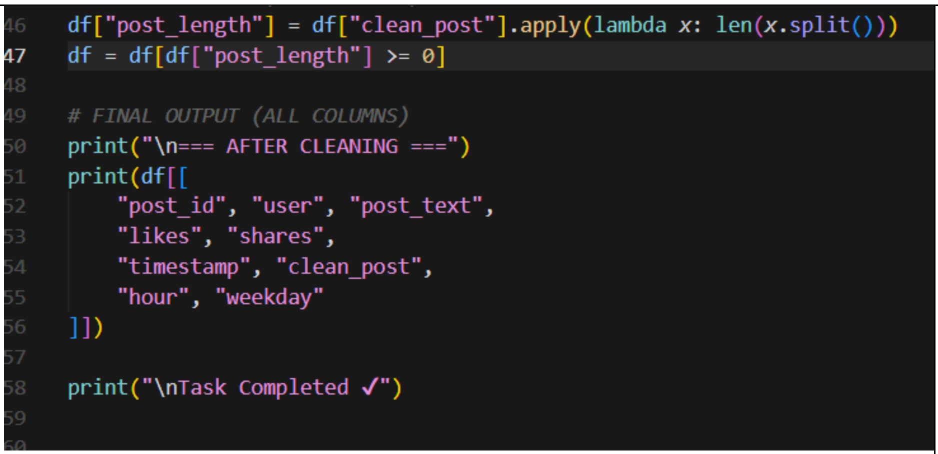
Q.No.	Question	*ExpectedTime to complete*
1	**Lab 17 – AI for Data Processing: Data Cleaning and Preprocessing Scripts** **Lab Objectives:** • Learn how to clean raw datasets using AI-assisted Python scripting. • Apply preprocessing techniques such as handling missing values, encoding categorical data, and normalization. • Automate repetitive data-cleaning tasks with AI-generated code. • Understand how preprocessing impacts model performance. --- **Task 1 – Social Media Data Cleaning** **Task: Clean raw social media posts dataset.** **Instructions:** - Remove stopwords, punctuation, and special symbols from post text. - Handle missing values in likes and shares columns. - Convert timestamp to datetime and extract features (hour, weekday). - Detect and remove spam/duplicate posts. **Expected Output:** A cleaned dataset with structured features for sentiment/engagement analysis. **Code:**	Week9 - Monday



```python
import pandas as pd
import numpy as np
import nltk
import re
from bs4 import BeautifulSoup
from nltk.corpus import stopwords

# SHOW ALL COLUMNS + ROWS
pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', True)
pd.set_option('display.max_rows', None)

# DOWNLOAD STOPWORDS
nltk.download("stopwords")
stop_words = set[str](stopwords.words("english"))

# LOAD CSV
df = pd.read_csv(r"C:\Users\moham_219zzho\OneDrive\Desktop\Aipp\assignment17\social_

print("=== BEFORE CLEANING ===")
print(df)

# HANDLE MISSING VALUES
df['likes'] = df['likes'].fillna(0).astype(int)
df['shares'] = df['shares'].fillna(0).astype(int)
df['post_text'] = df['post_text'].fillna("")

# CLEAN TEXT FUNCTION
def clean_text(text):
    text = BeautifulSoup(text, "html.parser").get_text()
```

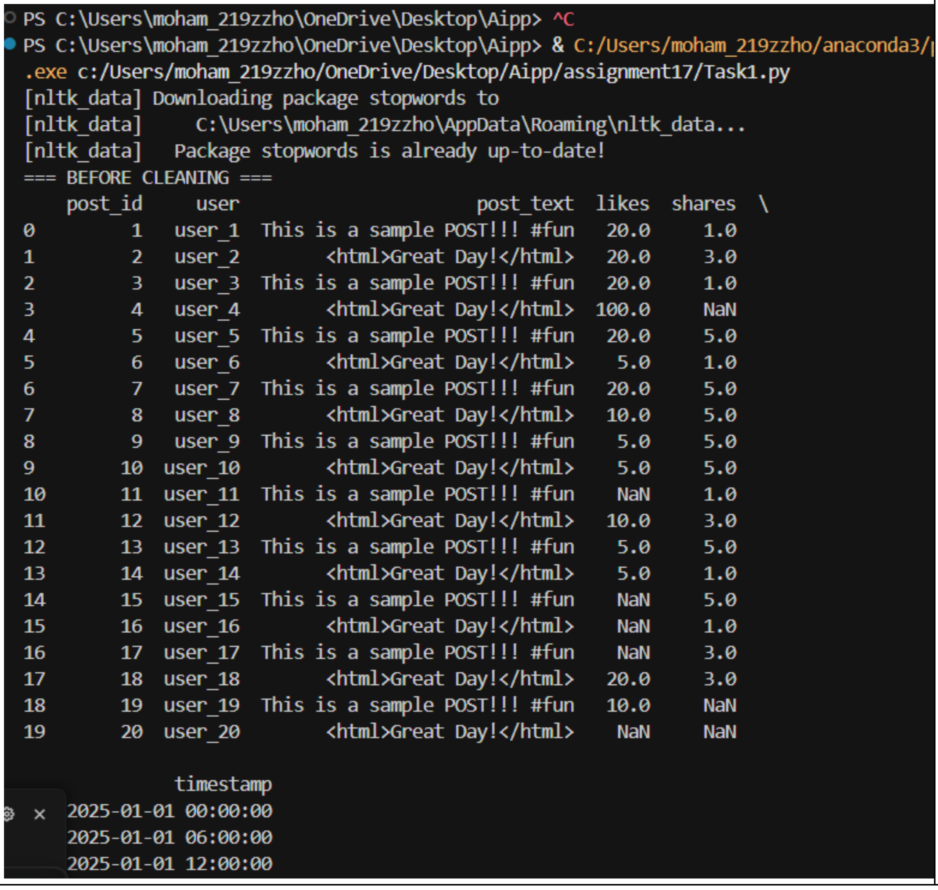


```python

# CLEAN TEXT FUNCTION
def clean_text(text):
    text = BeautifulSoup(text, "html.parser").get_text()
    text = re.sub(r"http\S+|www\S+", "", text)
    text = re.sub(r"[^A-Za-z\s]", " ", text)
    text = text.lower()
    words = text.split()
    words = [w for w in words if w not in stop_words]
    return " ".join(words)

df["clean_post"] = df["post_text"].apply(clean_text)

# TIMESTAMP FEATURES
df["timestamp"] = pd.to_datetime(df["timestamp"], errors="coerce")
df["hour"] = df["timestamp"].dt.hour
df["weekday"] = df["timestamp"].dt.day_name()

# KEEP ALL ROWS (no deletion)
df["post_length"] = df["clean_post"].apply(lambda x: len(x.split()))
df = df[df["post_length"] >= 0]

# FINAL OUTPUT (ALL COLUMNS)
print("\n=== AFTER CLEANING ===")
print(df[[
    "post_id", "user", "post_text",
    "likes", "shares",
    "timestamp", "clean_post",
    "hour", "weekday"
]])
```

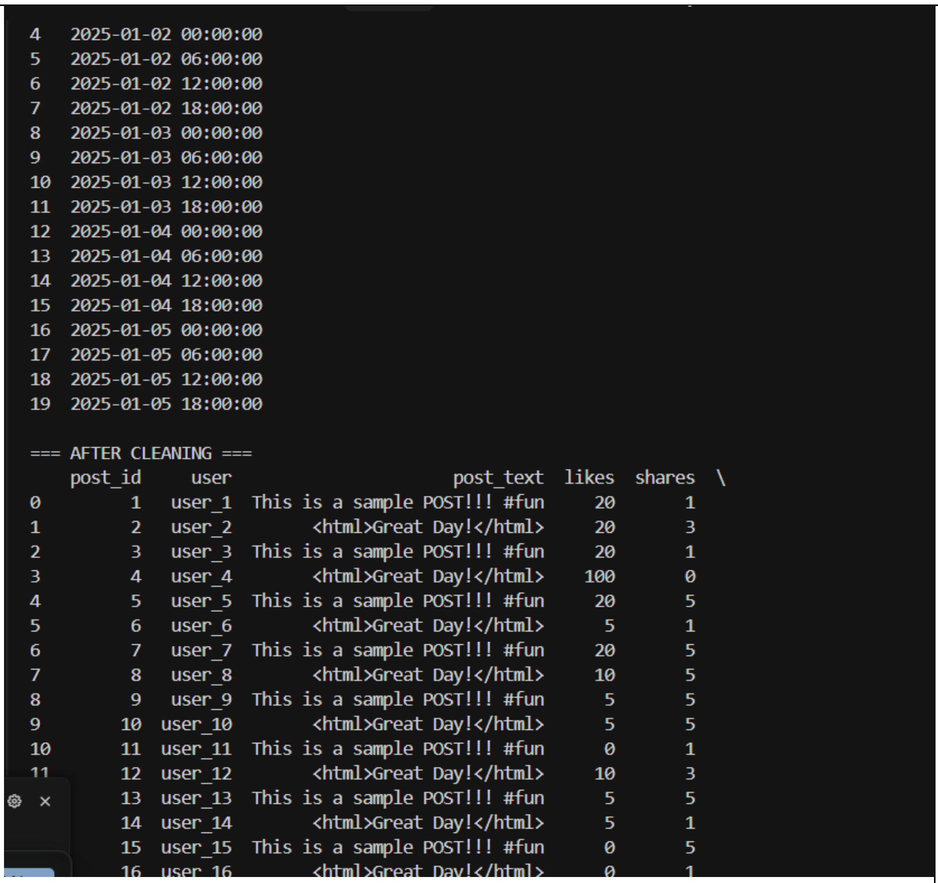


```
df["post_length"] = df["clean_post"].apply(lambda x: len(x.split()))
df = df[df["post_length"] >= 0]

# FINAL OUTPUT (ALL COLUMNS)
print("\n=== AFTER CLEANING ===")
print(df[[
    "post_id", "user", "post_text",
    "likes", "shares",
    "timestamp", "clean_post",
    "hour", "weekday"
]])

print("\nTask Completed ✓")
```

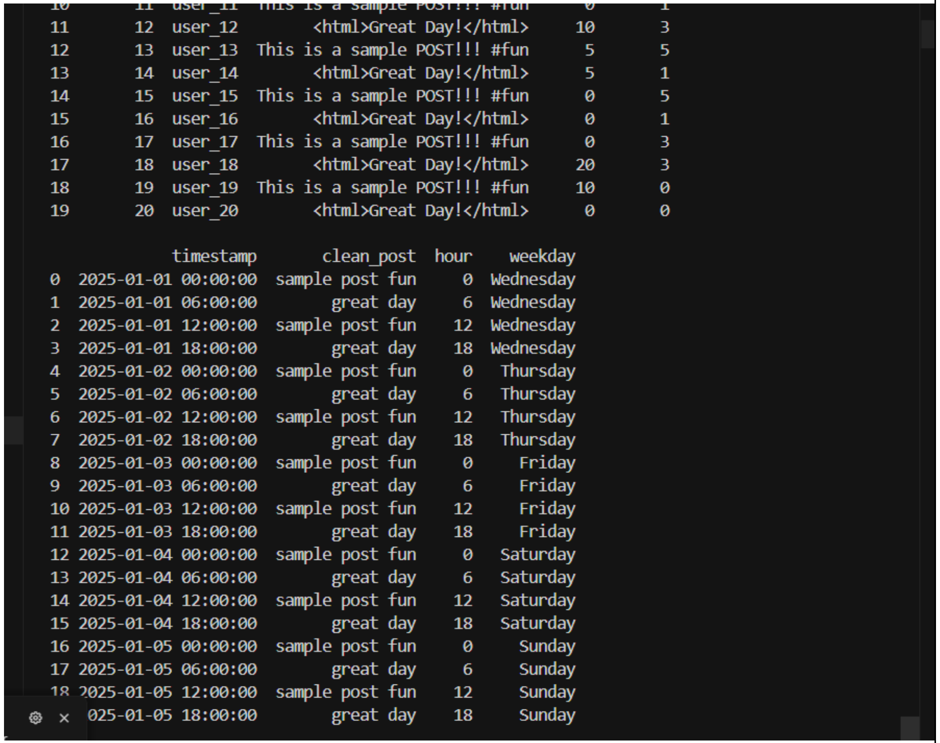
output:



```
PS C:\Users\moham_219zzho\OneDrive\Desktop\Aipp> ^C
PS C:\Users\moham_219zzho\OneDrive\Desktop\Aipp> & C:/Users/moham_219zzho/anaconda3/
.exe c:/Users/moham_219zzho/OneDrive/Desktop/Aipp/assignment17/Task1.py
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\moham_219zzho\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
=== BEFORE CLEANING ===
    post_id     user                      post_text  likes  shares  \
0         1   user_1  This is a sample POST!!! #fun   20.0     1.0
1         2   user_2        <html>Great Day!</html>   20.0     3.0
2         3   user_3  This is a sample POST!!! #fun   20.0     1.0
3         4   user_4        <html>Great Day!</html>  100.0     NaN
4         5   user_5  This is a sample POST!!! #fun   20.0     5.0
5         6   user_6        <html>Great Day!</html>    5.0     1.0
6         7   user_7  This is a sample POST!!! #fun   20.0     5.0
7         8   user_8        <html>Great Day!</html>   10.0     5.0
8         9   user_9  This is a sample POST!!! #fun    5.0     5.0
9        10  user_10        <html>Great Day!</html>    5.0     5.0
10       11  user_11  This is a sample POST!!! #fun    NaN     1.0
11       12  user_12        <html>Great Day!</html>   10.0     3.0
12       13  user_13  This is a sample POST!!! #fun    5.0     5.0
13       14  user_14        <html>Great Day!</html>    5.0     1.0
14       15  user_15  This is a sample POST!!! #fun    NaN     5.0
15       16  user_16        <html>Great Day!</html>    NaN     1.0
16       17  user_17  This is a sample POST!!! #fun    NaN     3.0
17       18  user_18        <html>Great Day!</html>   20.0     3.0
18       19  user_19  This is a sample POST!!! #fun   10.0     NaN
19       20  user_20        <html>Great Day!</html>    NaN     NaN

             timestamp
    2025-01-01 00:00:00
    2025-01-01 06:00:00
    2025-01-01 12:00:00
```



```
4   2025-01-02 00:00:00
5   2025-01-02 06:00:00
6   2025-01-02 12:00:00
7   2025-01-02 18:00:00
8   2025-01-03 00:00:00
9   2025-01-03 06:00:00
10  2025-01-03 12:00:00
11  2025-01-03 18:00:00
12  2025-01-04 00:00:00
13  2025-01-04 06:00:00
14  2025-01-04 12:00:00
15  2025-01-04 18:00:00
16  2025-01-05 00:00:00
17  2025-01-05 06:00:00
18  2025-01-05 12:00:00
19  2025-01-05 18:00:00

=== AFTER CLEANING ===
    post_id     user                   post_text  likes  shares  \
0         1   user_1  This is a sample POST!!! #fun     20       1
1         2   user_2        <html>Great Day!</html>     20       3
2         3   user_3  This is a sample POST!!! #fun     20       1
3         4   user_4        <html>Great Day!</html>    100       0
4         5   user_5  This is a sample POST!!! #fun     20       5
5         6   user_6        <html>Great Day!</html>      5       1
6         7   user_7  This is a sample POST!!! #fun     20       5
7         8   user_8        <html>Great Day!</html>     10       5
8         9   user_9  This is a sample POST!!! #fun      5       5
9        10  user_10        <html>Great Day!</html>      5       5
10       11  user_11  This is a sample POST!!! #fun      0       1
11       12  user_12        <html>Great Day!</html>     10       3
         13  user_13  This is a sample POST!!! #fun      5       5
         14  user_14        <html>Great Day!</html>      5       1
         15  user_15  This is a sample POST!!! #fun      0       5
         16  user_16        <html>Great Day!</html>      0       1
```



```
11       12  user_12        <html>Great Day!</html>     10       3
12       13  user_13  This is a sample POST!!! #fun      5       5
13       14  user_14        <html>Great Day!</html>      5       1
14       15  user_15  This is a sample POST!!! #fun      0       5
15       16  user_16        <html>Great Day!</html>      0       1
16       17  user_17  This is a sample POST!!! #fun      0       3
17       18  user_18        <html>Great Day!</html>     20       3
18       19  user_19  This is a sample POST!!! #fun     10       0
19       20  user_20        <html>Great Day!</html>      0       0

             timestamp       clean_post  hour    weekday
0  2025-01-01 00:00:00  sample post fun     0  Wednesday
1  2025-01-01 06:00:00        great day     6  Wednesday
2  2025-01-01 12:00:00  sample post fun    12  Wednesday
3  2025-01-01 18:00:00        great day    18  Wednesday
4  2025-01-02 00:00:00  sample post fun     0   Thursday
5  2025-01-02 06:00:00        great day     6   Thursday
6  2025-01-02 12:00:00  sample post fun    12   Thursday
7  2025-01-02 18:00:00        great day    18   Thursday
8  2025-01-03 00:00:00  sample post fun     0     Friday
9  2025-01-03 06:00:00        great day     6     Friday
10 2025-01-03 12:00:00  sample post fun    12     Friday
11 2025-01-03 18:00:00        great day    18     Friday
12 2025-01-04 00:00:00  sample post fun     0   Saturday
13 2025-01-04 06:00:00        great day     6   Saturday
14 2025-01-04 12:00:00  sample post fun    12   Saturday
15 2025-01-04 18:00:00        great day    18   Saturday
16 2025-01-05 00:00:00  sample post fun     0     Sunday
17 2025-01-05 06:00:00        great day     6     Sunday
18 2025-01-05 12:00:00  sample post fun    12     Sunday
   025-01-05 18:00:00        great day    18     Sunday
```

Task 2 – Financial Data Preprocessing

Task: Preprocess a stock market dataset.

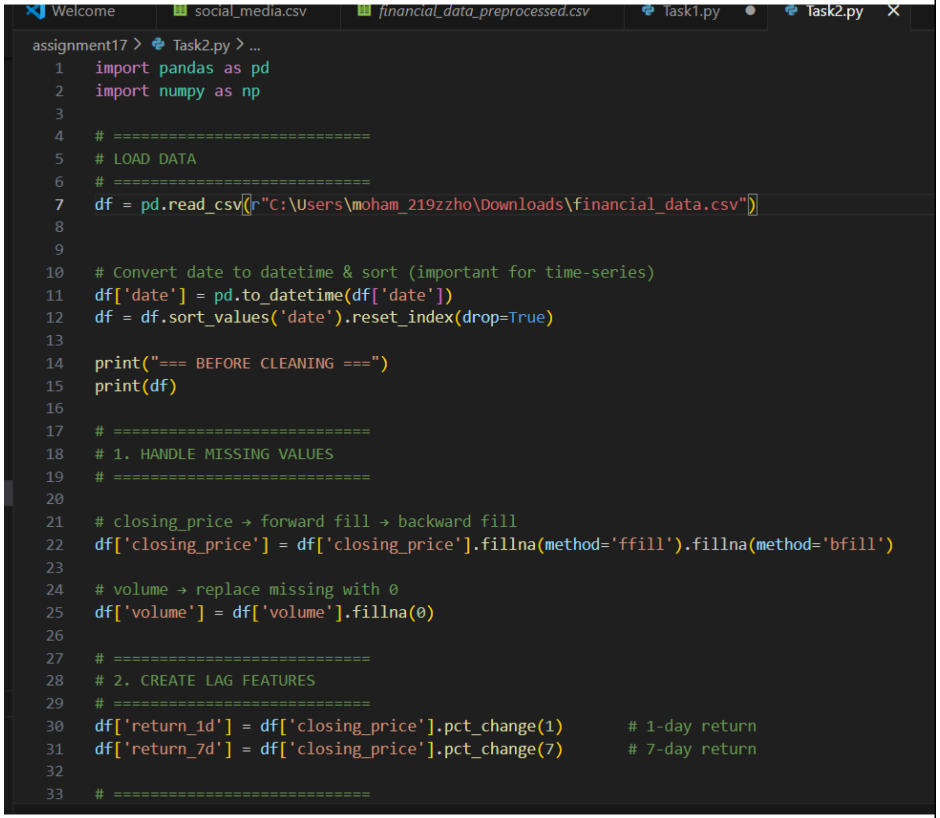
Instructions:

- Handle missing values in closing_price and volume.
- Create lag features (1-day, 7-day returns).
- Normalize volume column using log-scaling.

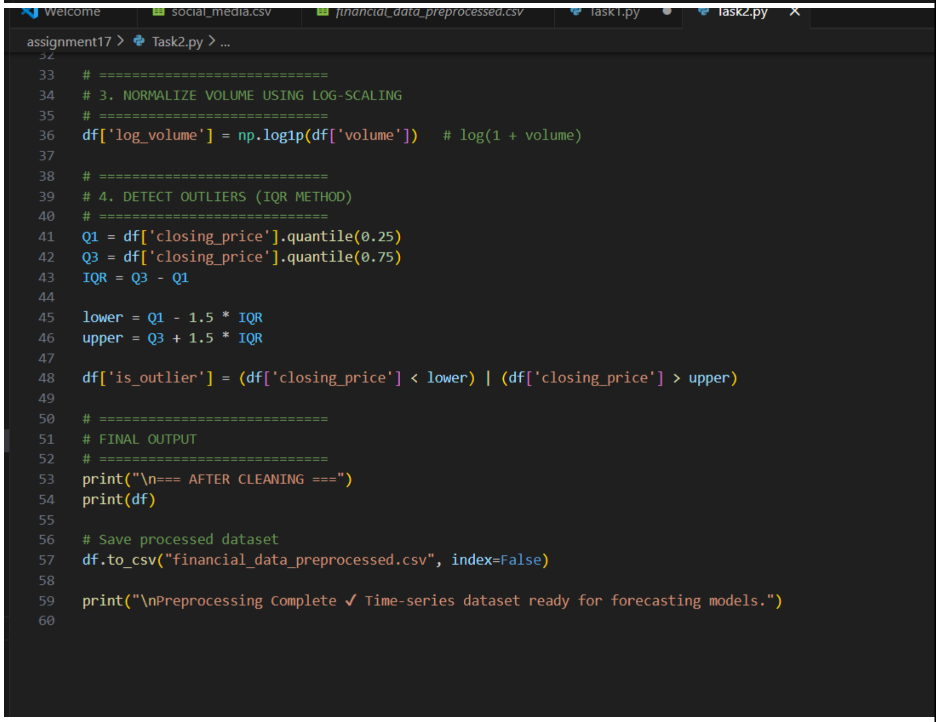
- Detect outliers in closing_price using IQR method.

Expected Output: A time-series dataset ready for forecasting models.

Code:

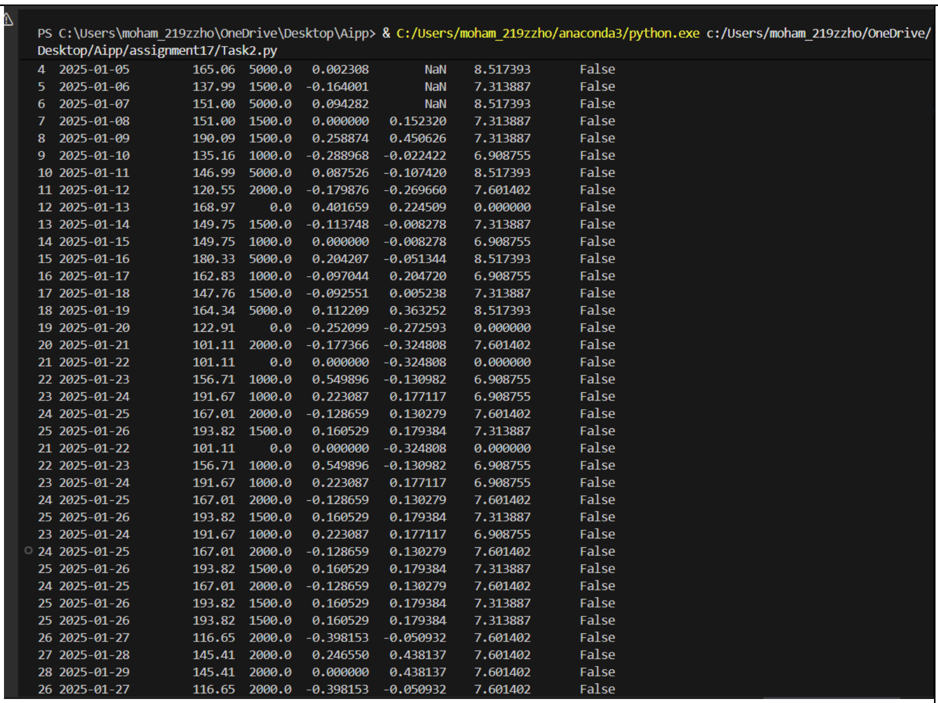

Welcome
social_media.csv
financial_data_preprocessed.csv
Task1.py
Task2.py
assignment17 > Task2.py > ...

```python
import pandas as pd
import numpy as np

# ===========================
# LOAD DATA
# ===========================
df = pd.read_csv(r"C:\Users\moham_219zzho\Downloads\financial_data.csv")


# Convert date to datetime & sort (important for time-series)
df['date'] = pd.to_datetime(df['date'])
df = df.sort_values('date').reset_index(drop=True)

print("=== BEFORE CLEANING ===")
print(df)

# ===========================
# 1. HANDLE MISSING VALUES
# ===========================

# closing_price → forward fill → backward fill
df['closing_price'] = df['closing_price'].fillna(method='ffill').fillna(method='bfill')

# volume → replace missing with 0
df['volume'] = df['volume'].fillna(0)

# ===========================
# 2. CREATE LAG FEATURES
# ===========================
df['return_1d'] = df['closing_price'].pct_change(1)       # 1-day return
df['return_7d'] = df['closing_price'].pct_change(7)       # 7-day return

# ===========================
```

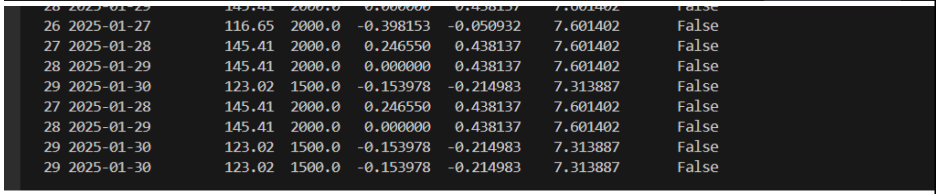

Welcome
social_media.csv
financial_data_preprocessed.csv
Task1.py
Task2.py
assignment17 > Task2.py > ...

```python
# ===========================
# 3. NORMALIZE VOLUME USING LOG-SCALING
# ===========================
df['log_volume'] = np.log1p(df['volume'])   # log(1 + volume)

# ===========================
# 4. DETECT OUTLIERS (IQR METHOD)
# ===========================
Q1 = df['closing_price'].quantile(0.25)
Q3 = df['closing_price'].quantile(0.75)
IQR = Q3 - Q1

lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

df['is_outlier'] = (df['closing_price'] < lower) | (df['closing_price'] > upper)

# ===========================
# FINAL OUTPUT
# ===========================
print("\n=== AFTER CLEANING ===")
print(df)

# Save processed dataset
df.to_csv("financial_data_preprocessed.csv", index=False)

print("\nPreprocessing Complete ✓ Time-series dataset ready for forecasting models.")

```

output:



```
PS C:\Users\moham_219zzho\OneDrive\Desktop\Aipp> & C:/Users/moham_219zzho/anaconda3/python.exe c:/Users/moham_219zzho/OneDrive/
Desktop/Aipp/assignment17/Task2.py
4  2025-01-05        165.06  5000.0  0.002308       NaN   8.517393      False
5  2025-01-06        137.99  1500.0 -0.164001       NaN   7.313887      False
6  2025-01-07        151.00  5000.0  0.094282       NaN   8.517393      False
7  2025-01-08        151.00  1500.0  0.000000  0.152320   7.313887      False
8  2025-01-09        190.09  1500.0  0.258874  0.450626   7.313887      False
9  2025-01-10        135.16  1000.0 -0.288968 -0.022422   6.908755      False
10 2025-01-11        146.99  5000.0  0.087526 -0.107420   8.517393      False
11 2025-01-12        120.55  2000.0 -0.179876 -0.269660   7.601402      False
12 2025-01-13        168.97     0.0  0.401659  0.224509   0.000000      False
13 2025-01-14        149.75  1500.0 -0.113748 -0.008278   7.313887      False
14 2025-01-15        149.75  1000.0  0.000000 -0.008278   6.908755      False
15 2025-01-16        180.33  5000.0  0.204207 -0.051344   8.517393      False
16 2025-01-17        162.83  1000.0 -0.097044  0.204720   6.908755      False
17 2025-01-18        147.76  1500.0 -0.092551  0.005238   7.313887      False
18 2025-01-19        164.34  5000.0  0.112209  0.363252   8.517393      False
19 2025-01-20        122.91     0.0 -0.252099 -0.272593   0.000000      False
20 2025-01-21        101.11  2000.0 -0.177366 -0.324808   7.601402      False
21 2025-01-22        101.11     0.0  0.000000 -0.324808   0.000000      False
22 2025-01-23        156.71  1000.0  0.549896 -0.130982   6.908755      False
23 2025-01-24        191.67  1000.0  0.223087  0.177117   6.908755      False
24 2025-01-25        167.01  2000.0 -0.128659  0.130279   7.601402      False
25 2025-01-26        193.82  1500.0  0.160529  0.179384   7.313887      False
21 2025-01-22        101.11     0.0  0.000000 -0.324808   0.000000      False
22 2025-01-23        156.71  1000.0  0.549896 -0.130982   6.908755      False
23 2025-01-24        191.67  1000.0  0.223087  0.177117   6.908755      False
24 2025-01-25        167.01  2000.0 -0.128659  0.130279   7.601402      False
25 2025-01-26        193.82  1500.0  0.160529  0.179384   7.313887      False
23 2025-01-24        191.67  1000.0  0.223087  0.177117   6.908755      False
24 2025-01-25        167.01  2000.0 -0.128659  0.130279   7.601402      False
25 2025-01-26        193.82  1500.0  0.160529  0.179384   7.313887      False
24 2025-01-25        167.01  2000.0 -0.128659  0.130279   7.601402      False
25 2025-01-26        193.82  1500.0  0.160529  0.179384   7.313887      False
25 2025-01-26        193.82  1500.0  0.160529  0.179384   7.313887      False
26 2025-01-27        116.65  2000.0 -0.398153 -0.050932   7.601402      False
27 2025-01-28        145.41  2000.0  0.246550  0.438137   7.601402      False
28 2025-01-29        145.41  2000.0  0.000000  0.438137   7.601402      False
26 2025-01-27        116.65  2000.0 -0.398153 -0.050932   7.601402      False
```



```
28 2025-01-29        145.41  2000.0  0.000000  0.438137   7.601402      False
26 2025-01-27        116.65  2000.0 -0.398153 -0.050932   7.601402      False
27 2025-01-28        145.41  2000.0  0.246550  0.438137   7.601402      False
28 2025-01-29        145.41  2000.0  0.000000  0.438137   7.601402      False
29 2025-01-30        123.02  1500.0 -0.153978 -0.214983   7.313887      False
27 2025-01-28        145.41  2000.0  0.246550  0.438137   7.601402      False
28 2025-01-29        145.41  2000.0  0.000000  0.438137   7.601402      False
29 2025-01-30        123.02  1500.0 -0.153978 -0.214983   7.313887      False
29 2025-01-30        123.02  1500.0 -0.153978 -0.214983   7.313887      False
```

Task 3 – IoT Sensor Data Preparation

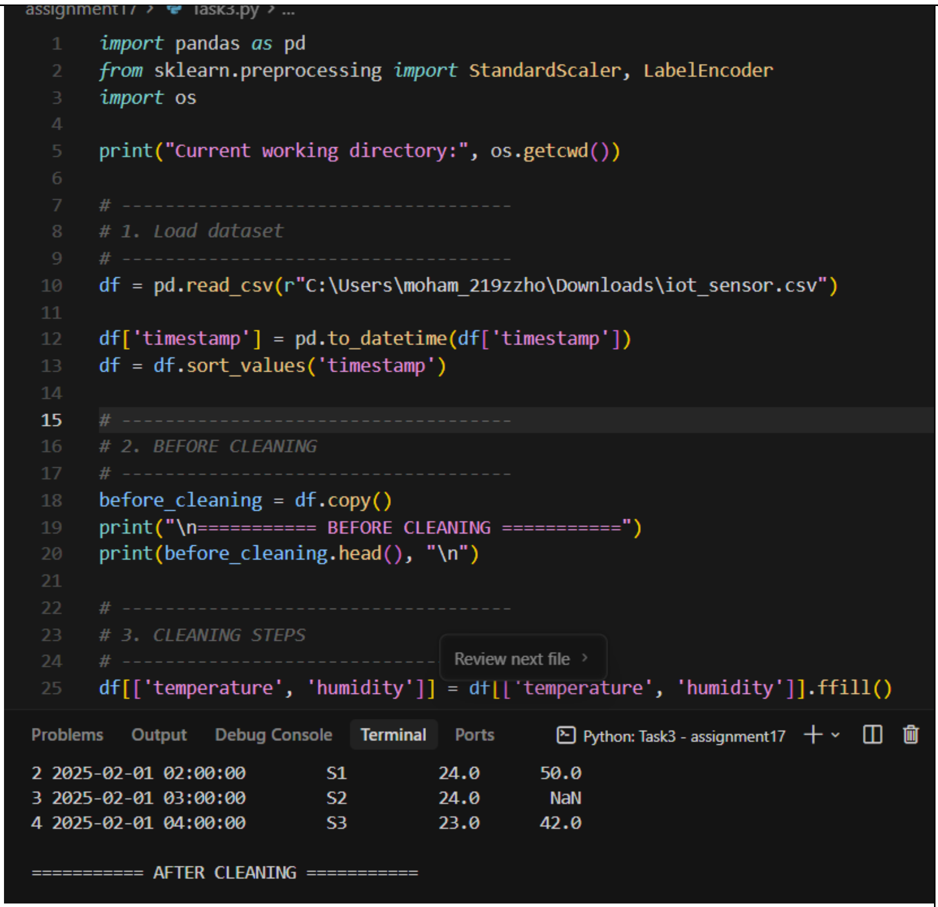
Task: Clean and preprocess IoT temperature and humidity logs.

Instructions:

- Handle missing values using forward fill.
- Remove sensor drift (apply rolling mean).
- Normalize readings using standard scaling.
- Encode categorical sensor IDs.

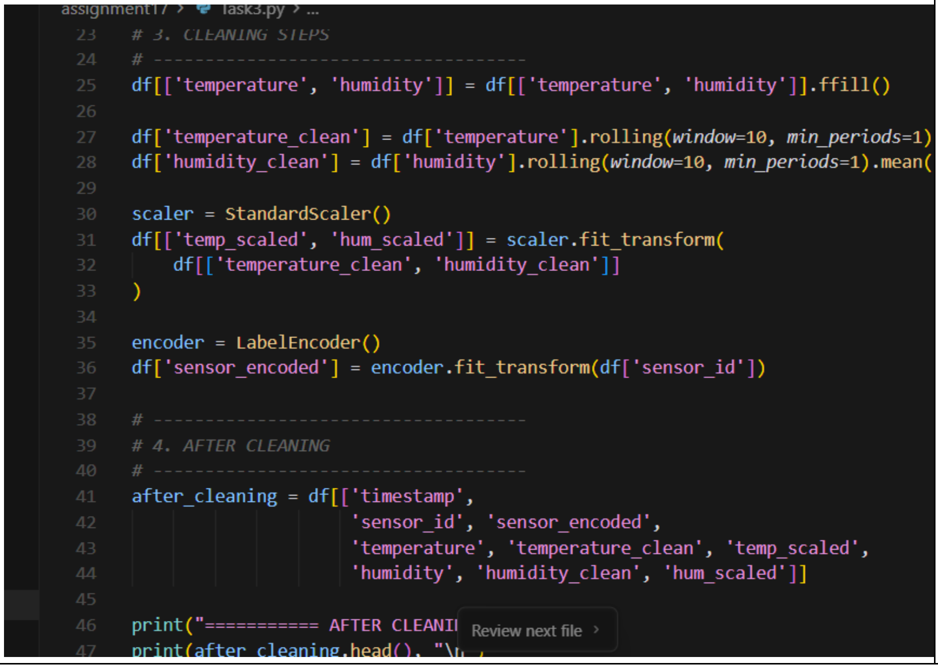
Expected Output: A structured dataset optimized for anomaly detection.

Code:

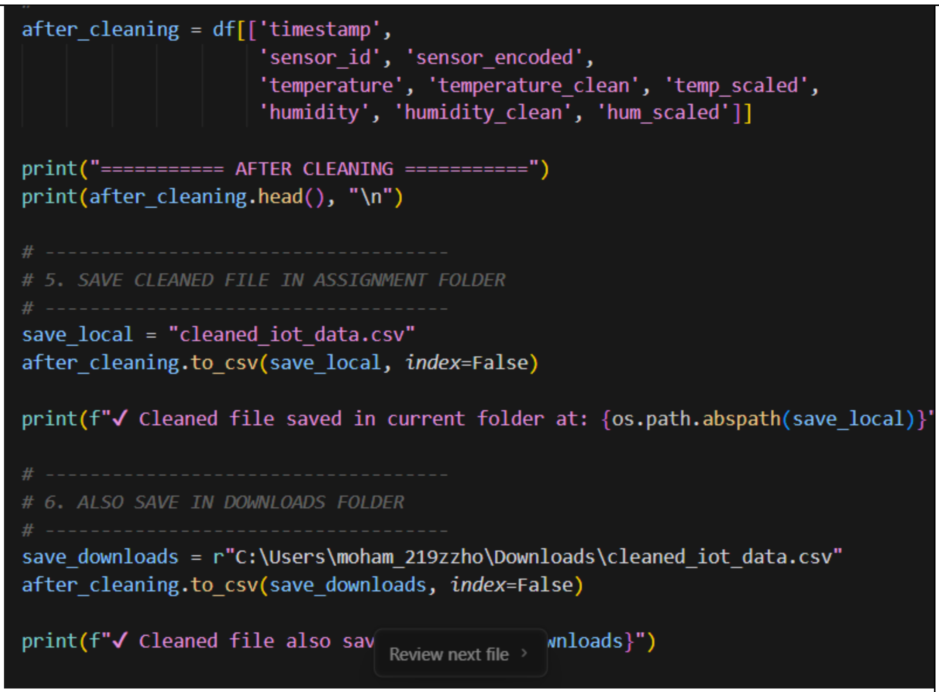


```python
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
import os

print("Current working directory:", os.getcwd())

# ------------------------------------
# 1. Load dataset
# ------------------------------------
df = pd.read_csv(r"C:\Users\moham_219zzho\Downloads\iot_sensor.csv")

df['timestamp'] = pd.to_datetime(df['timestamp'])
df = df.sort_values('timestamp')

# ------------------------------------
# 2. BEFORE CLEANING
# ------------------------------------
before_cleaning = df.copy()
print("\n=========== BEFORE CLEANING ===========")
print(before_cleaning.head(), "\n")

# ------------------------------------
# 3. CLEANING STEPS
# ------------------------------------
df[['temperature', 'humidity']] = df[['temperature', 'humidity']].ffill()
```

```
2 2025-02-01 02:00:00        S1         24.0      50.0
3 2025-02-01 03:00:00        S2         24.0       NaN
4 2025-02-01 04:00:00        S3         23.0      42.0

=========== AFTER CLEANING ===========
```

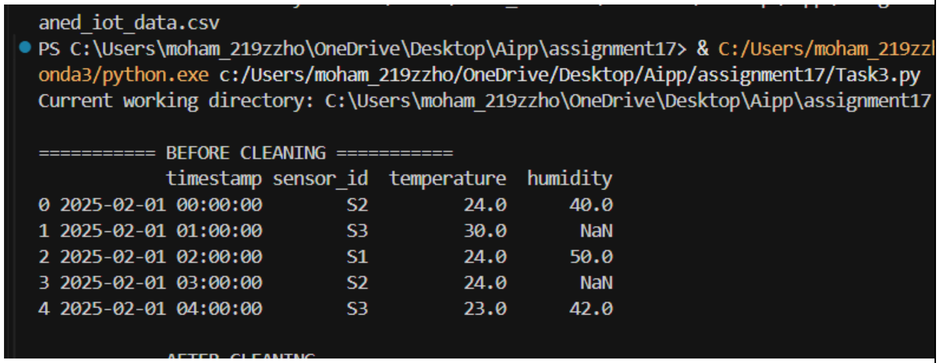


```python
# 3. CLEANING STEPS
# ------------------------------------
df[['temperature', 'humidity']] = df[['temperature', 'humidity']].ffill()

df['temperature_clean'] = df['temperature'].rolling(window=10, min_periods=1)
df['humidity_clean'] = df['humidity'].rolling(window=10, min_periods=1).mean(

scaler = StandardScaler()
df[['temp_scaled', 'hum_scaled']] = scaler.fit_transform(
    df[['temperature_clean', 'humidity_clean']]
)

encoder = LabelEncoder()
df['sensor_encoded'] = encoder.fit_transform(df['sensor_id'])

# ------------------------------------
# 4. AFTER CLEANING
# ------------------------------------
after_cleaning = df[['timestamp',
                     'sensor_id', 'sensor_encoded',
                     'temperature', 'temperature_clean', 'temp_scaled',
                     'humidity', 'humidity_clean', 'hum_scaled']]

print("=========== AFTER CLEANI
print(after_cleaning.head(), "\n
```

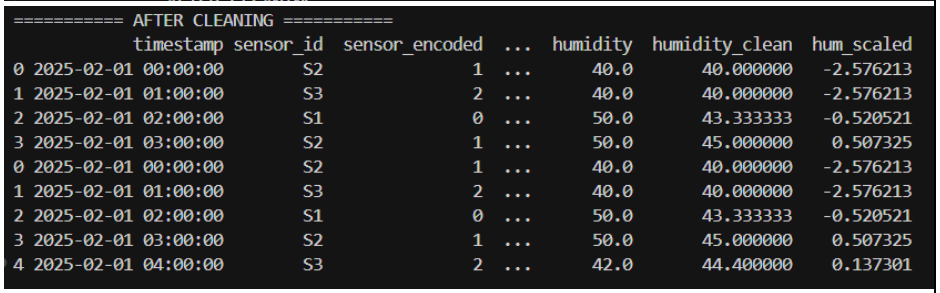


```
after_cleaning = df[['timestamp',
                     'sensor_id', 'sensor_encoded',
                     'temperature', 'temperature_clean', 'temp_scaled',
                     'humidity', 'humidity_clean', 'hum_scaled']]

print("=========== AFTER CLEANING ===========")
print(after_cleaning.head(), "\n")

# ----------------------------------
# 5. SAVE CLEANED FILE IN ASSIGNMENT FOLDER
# ----------------------------------
save_local = "cleaned_iot_data.csv"
after_cleaning.to_csv(save_local, index=False)

print(f"✓ Cleaned file saved in current folder at: {os.path.abspath(save_local)}"

# ----------------------------------
# 6. ALSO SAVE IN DOWNLOADS FOLDER
# ----------------------------------
save_downloads = r"C:\Users\moham_219zzho\Downloads\cleaned_iot_data.csv"
after_cleaning.to_csv(save_downloads, index=False)

print(f"✓ Cleaned file also sav                  wnloads}")
```

output:



```
aned_iot_data.csv
PS C:\Users\moham_219zzho\OneDrive\Desktop\Aipp\assignment17> & C:/Users/moham_219zzl
onda3/python.exe c:/Users/moham_219zzho/OneDrive/Desktop/Aipp/assignment17/Task3.py
Current working directory: C:\Users\moham_219zzho\OneDrive\Desktop\Aipp\assignment17

=========== BEFORE CLEANING ===========
            timestamp sensor_id  temperature  humidity
0 2025-02-01 00:00:00        S2         24.0      40.0
1 2025-02-01 01:00:00        S3         30.0       NaN
2 2025-02-01 02:00:00        S1         24.0      50.0
3 2025-02-01 03:00:00        S2         24.0       NaN
4 2025-02-01 04:00:00        S3         23.0      42.0
```



```
=========== AFTER CLEANING ===========
            timestamp sensor_id  sensor_encoded  ...  humidity  humidity_clean  hum_scaled
0 2025-02-01 00:00:00        S2               1  ...      40.0       40.000000   -2.576213
1 2025-02-01 01:00:00        S3               2  ...      40.0       40.000000   -2.576213
2 2025-02-01 02:00:00        S1               0  ...      50.0       43.333333   -0.520521
3 2025-02-01 03:00:00        S2               1  ...      50.0       45.000000    0.507325
0 2025-02-01 00:00:00        S2               1  ...      40.0       40.000000   -2.576213
1 2025-02-01 01:00:00        S3               2  ...      40.0       40.000000   -2.576213
2 2025-02-01 02:00:00        S1               0  ...      50.0       43.333333   -0.520521
3 2025-02-01 03:00:00        S2               1  ...      50.0       45.000000    0.507325
4 2025-02-01 04:00:00        S3               2  ...      42.0       44.400000    0.137301
```

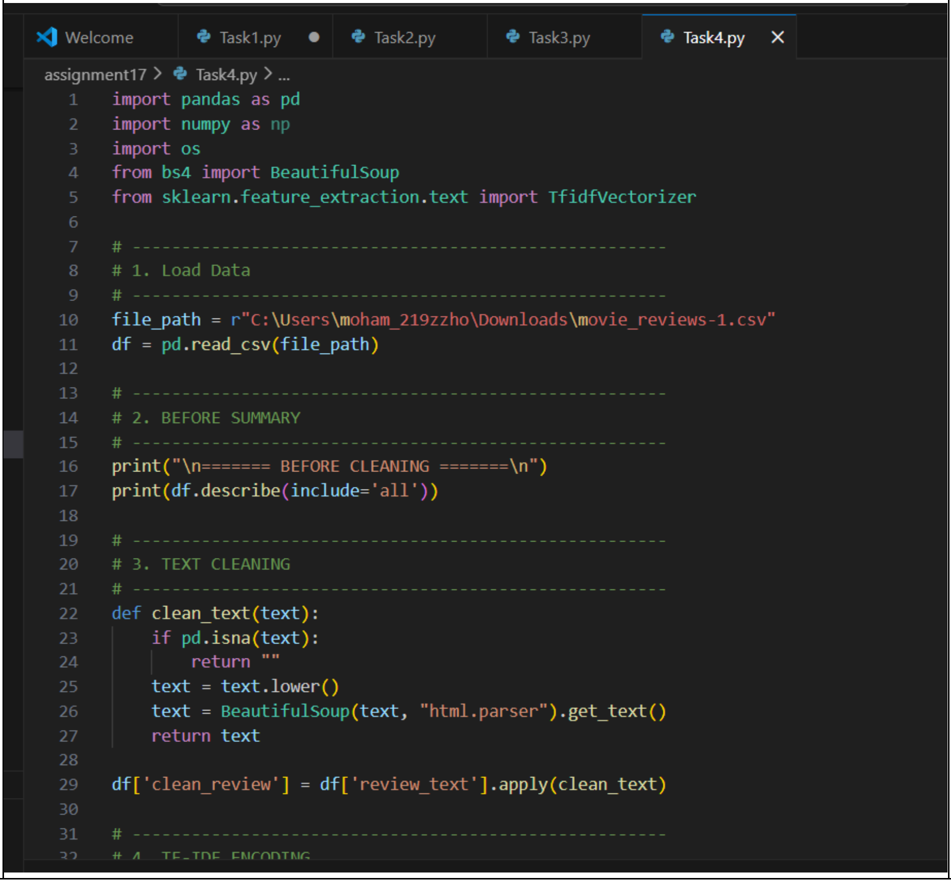
Task 4 – Real-Time Application: Movie Reviews Data Cleaning
Task: A streaming platform wants to analyze customer reviews.
Instructions:
- Standardize text (lowercase, remove HTML tags).
- Tokenize and encode reviews using AI-assisted methods (TF-IDF or embeddings).
- Handle missing ratings (fill with median).
- Normalize ratings (0–10 → 0–1 scale).
- Generate a before vs after summary report.

Expected Output: A cleaned dataset ready for sentiment classification.

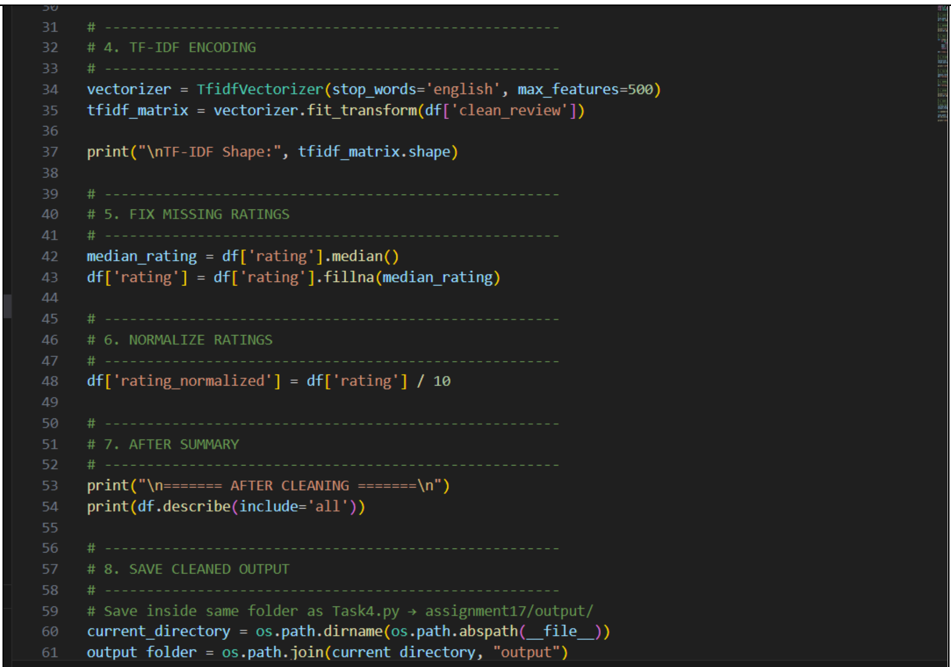
☑ Deliverables (For All Tasks)

1. AI-generated prompts for code and test case generation.
2. At least 3 assert test cases for each task.
3. AI-generated initial code and execution screenshots.
4. Analysis of whether code passes all tests.
5. Improved final version with inline comments and explanations.
6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.

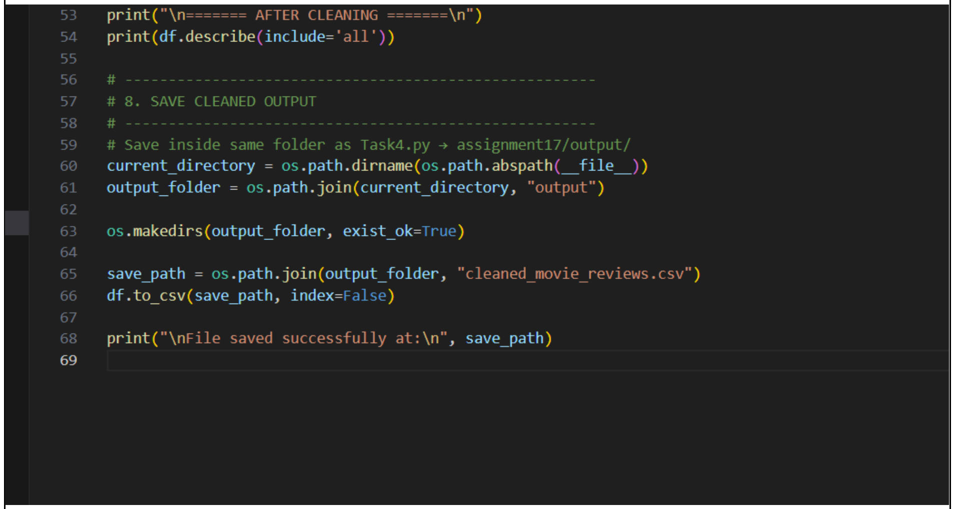
Code:



```python
import pandas as pd
import numpy as np
import os
from bs4 import BeautifulSoup
from sklearn.feature_extraction.text import TfidfVectorizer

# -----------------------------------------------------
# 1. Load Data
# -----------------------------------------------------
file_path = r"C:\Users\moham_219zzho\Downloads\movie_reviews-1.csv"
df = pd.read_csv(file_path)

# -----------------------------------------------------
# 2. BEFORE SUMMARY
# -----------------------------------------------------
print("\n======= BEFORE CLEANING =======\n")
print(df.describe(include='all'))

# -----------------------------------------------------
# 3. TEXT CLEANING
# -----------------------------------------------------
def clean_text(text):
    if pd.isna(text):
        return ""
    text = text.lower()
    text = BeautifulSoup(text, "html.parser").get_text()
    return text

df['clean_review'] = df['review_text'].apply(clean_text)

# -----------------------------------------------------
```



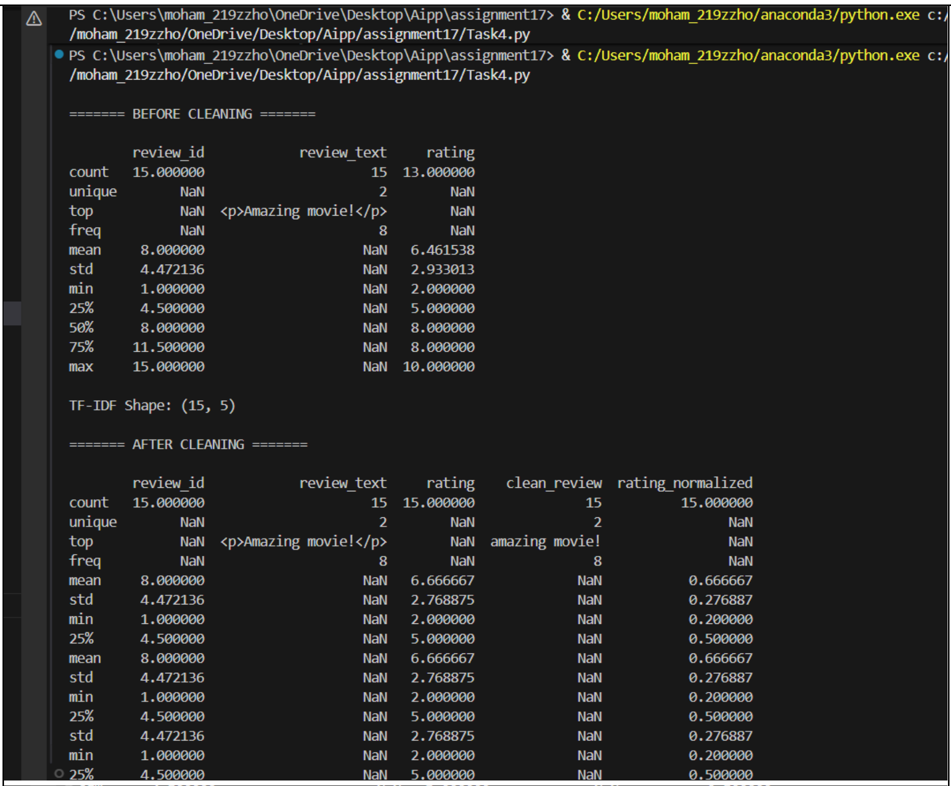
```python
# ----------------------------------------------------
# 4. TF-IDF ENCODING
# ----------------------------------------------------
vectorizer = TfidfVectorizer(stop_words='english', max_features=500)
tfidf_matrix = vectorizer.fit_transform(df['clean_review'])

print("\nTF-IDF Shape:", tfidf_matrix.shape)

# ----------------------------------------------------
# 5. FIX MISSING RATINGS
# ----------------------------------------------------
median_rating = df['rating'].median()
df['rating'] = df['rating'].fillna(median_rating)

# ----------------------------------------------------
# 6. NORMALIZE RATINGS
# ----------------------------------------------------
df['rating_normalized'] = df['rating'] / 10

# ----------------------------------------------------
# 7. AFTER SUMMARY
# ----------------------------------------------------
print("\n======= AFTER CLEANING =======\n")
print(df.describe(include='all'))

# ----------------------------------------------------
# 8. SAVE CLEANED OUTPUT
# ----------------------------------------------------
# Save inside same folder as Task4.py → assignment17/output/
current_directory = os.path.dirname(os.path.abspath(__file__))
output_folder = os.path.join(current_directory, "output")
```



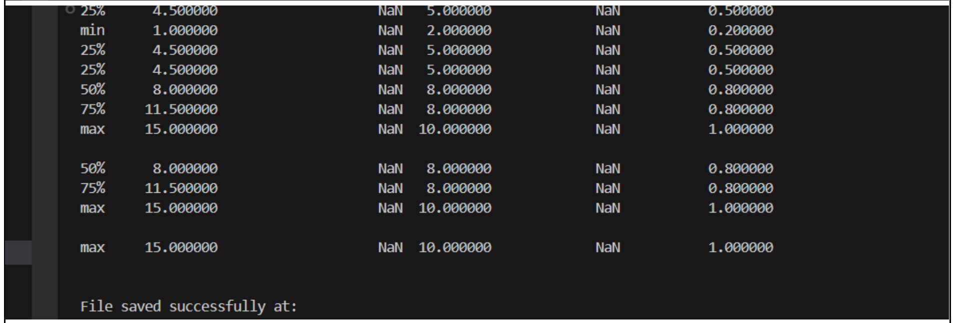
```python
print("\n======= AFTER CLEANING =======\n")
print(df.describe(include='all'))

# ----------------------------------------------------
# 8. SAVE CLEANED OUTPUT
# ----------------------------------------------------
# Save inside same folder as Task4.py → assignment17/output/
current_directory = os.path.dirname(os.path.abspath(__file__))
output_folder = os.path.join(current_directory, "output")
```

```python
os.makedirs(output_folder, exist_ok=True)

save_path = os.path.join(output_folder, "cleaned_movie_reviews.csv")
df.to_csv(save_path, index=False)

print("\nFile saved successfully at:\n", save_path)
```

Output:



```
PS C:\Users\moham_219zzho\OneDrive\Desktop\Aipp\assignment17> & C:/Users/moham_219zzho/anaconda3/python.exe c:/
/moham_219zzho/OneDrive/Desktop/Aipp/assignment17/Task4.py
PS C:\Users\moham_219zzho\OneDrive\Desktop\Aipp\assignment17> & C:/Users/moham_219zzho/anaconda3/python.exe c:/
/moham_219zzho/OneDrive/Desktop/Aipp/assignment17/Task4.py

======= BEFORE CLEANING =======

        review_id          review_text     rating
count   15.000000                   15  13.000000
unique        NaN                    2        NaN
top           NaN  <p>Amazing movie!</p>        NaN
freq          NaN                    8        NaN
mean     8.000000                  NaN   6.461538
std      4.472136                  NaN   2.933013
min      1.000000                  NaN   2.000000
25%      4.500000                  NaN   5.000000
50%      8.000000                  NaN   8.000000
75%     11.500000                  NaN   8.000000
max     15.000000                  NaN  10.000000

TF-IDF Shape: (15, 5)

======= AFTER CLEANING =======

        review_id          review_text     rating   clean_review  rating_normalized
count   15.000000                   15  15.000000             15          15.000000
unique        NaN                    2        NaN              2                NaN
top           NaN  <p>Amazing movie!</p>        NaN  amazing movie!                NaN
freq          NaN                    8        NaN              8                NaN
mean     8.000000                  NaN   6.666667            NaN           0.666667
std      4.472136                  NaN   2.768875            NaN           0.276887
min      1.000000                  NaN   2.000000            NaN           0.200000
25%      4.500000                  NaN   5.000000            NaN           0.500000
mean     8.000000                  NaN   6.666667            NaN           0.666667
std      4.472136                  NaN   2.768875            NaN           0.276887
min      1.000000                  NaN   2.000000            NaN           0.200000
25%      4.500000                  NaN   5.000000            NaN           0.500000
std      4.472136                  NaN   2.768875            NaN           0.276887
min      1.000000                  NaN   2.000000            NaN           0.200000
25%      4.500000                  NaN   5.000000            NaN           0.500000
```



```
25%      4.500000                  NaN   5.000000            NaN           0.500000
min      1.000000                  NaN   2.000000            NaN           0.200000
25%      4.500000                  NaN   5.000000            NaN           0.500000
25%      4.500000                  NaN   5.000000            NaN           0.500000
50%      8.000000                  NaN   8.000000            NaN           0.800000
75%     11.500000                  NaN   8.000000            NaN           0.800000
max     15.000000                  NaN  10.000000            NaN           1.000000

50%      8.000000                  NaN   8.000000            NaN           0.800000
75%     11.500000                  NaN   8.000000            NaN           0.800000
max     15.000000                  NaN  10.000000            NaN           1.000000

max     15.000000                  NaN  10.000000            NaN           1.000000


File saved successfully at:
```