AI ASSISTED CODING

NAME : Likhitha Pothunuri

ROLLNO:2403A510D1

BATCH : 05

ASSIGNMENT : 17.1

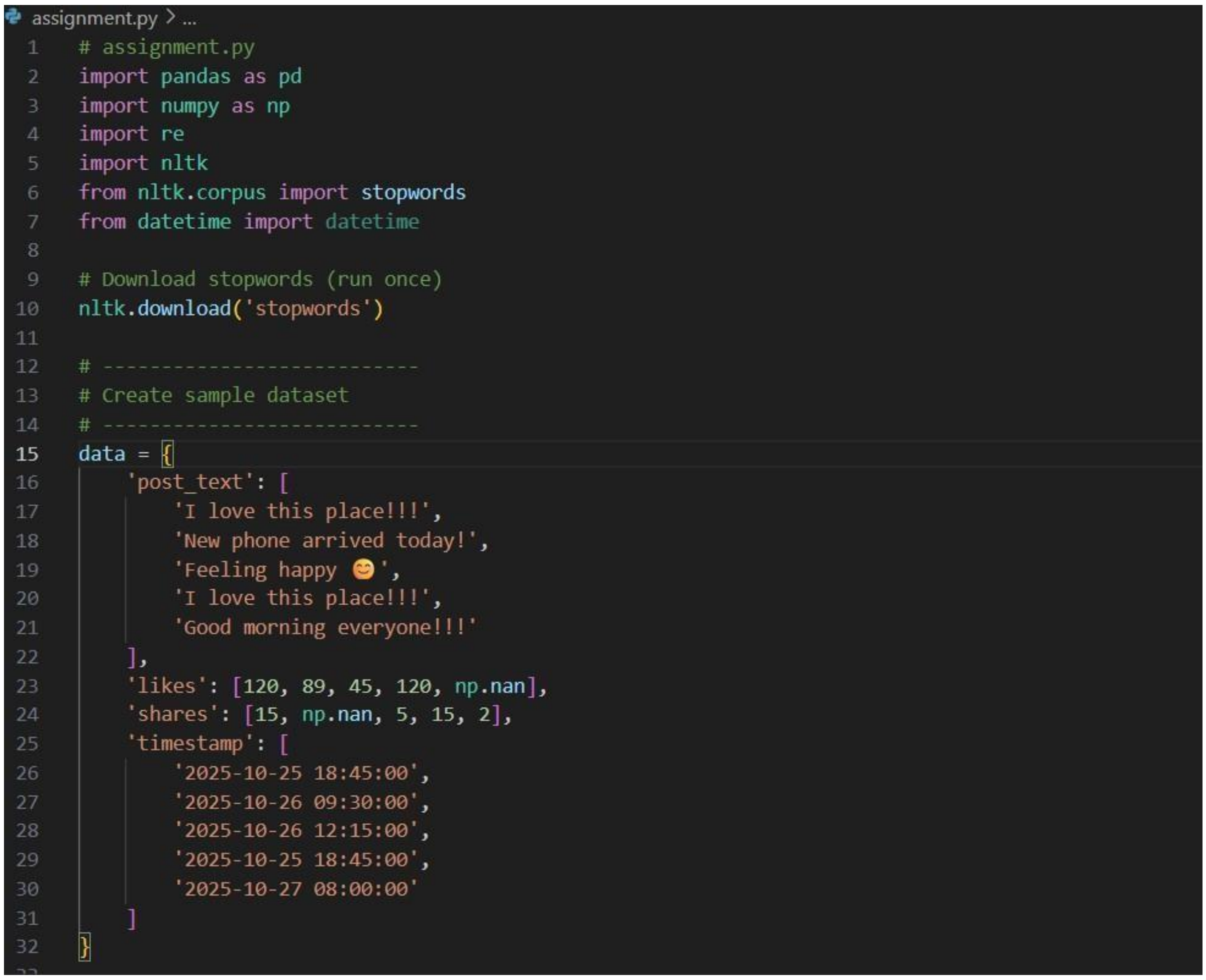
#TASK-1

PROMPT :

Write a python program to Clean raw social media posts dataset.
Instructions:

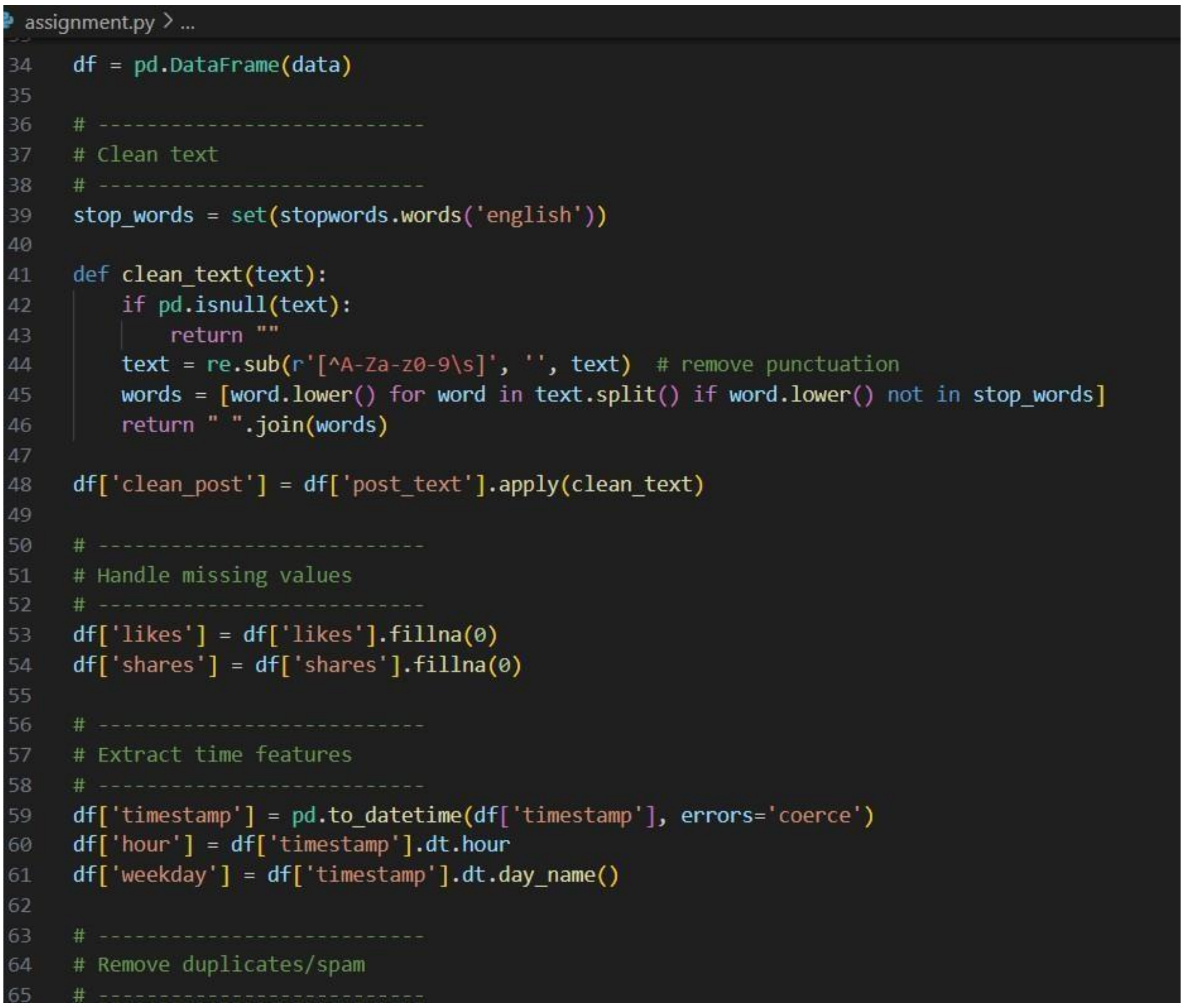
- Remove stopwords, punctuation, and special symbols from post text.
- Handle missing values in likes and shares columns.
- Convert timestamp to datetime and extract features (hour, weekday).
- Detect and remove spam/duplicate posts.

CODE :

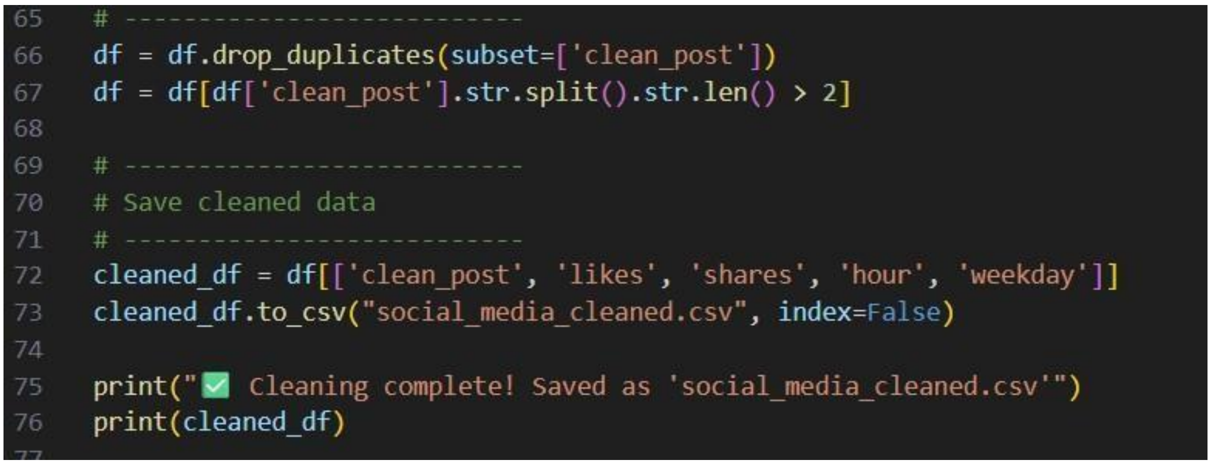


assignment.py > ...

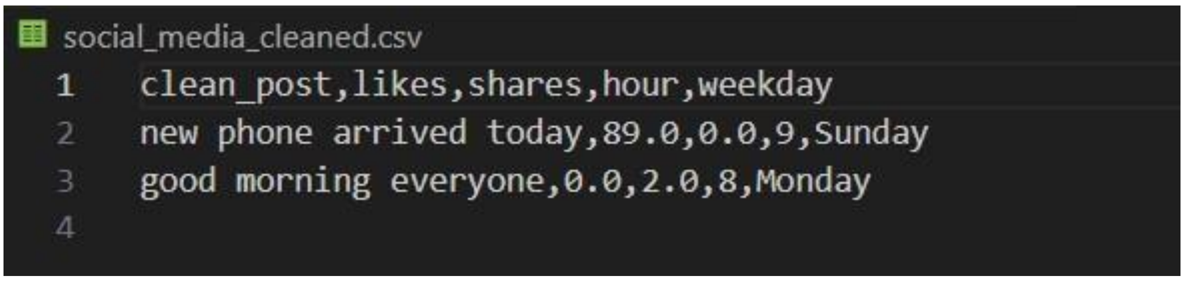
```python
# assignment.py
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from datetime import datetime

# Download stopwords (run once)
nltk.download('stopwords')

# ---------------------------
# Create sample dataset
# ---------------------------
data = {
    'post_text': [
        'I love this place!!!',
        'New phone arrived today!',
        'Feeling happy 😊',
        'I love this place!!!',
        'Good morning everyone!!!'
    ],
    'likes': [120, 89, 45, 120, np.nan],
    'shares': [15, np.nan, 5, 15, 2],
    'timestamp': [
        '2025-10-25 18:45:00',
        '2025-10-26 09:30:00',
        '2025-10-26 12:15:00',
        '2025-10-25 18:45:00',
        '2025-10-27 08:00:00'
    ]
}
```



```python
df = pd.DataFrame(data)

# --------------------------
# Clean text
# --------------------------
stop_words = set(stopwords.words('english'))

def clean_text(text):
    if pd.isnull(text):
        return ""
    text = re.sub(r'[^A-Za-z0-9\s]', '', text)  # remove punctuation
    words = [word.lower() for word in text.split() if word.lower() not in stop_words]
    return " ".join(words)

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

# --------------------------
# Handle missing values
# --------------------------
df['likes'] = df['likes'].fillna(0)
df['shares'] = df['shares'].fillna(0)

# --------------------------
# Extract time features
# --------------------------
df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')
df['hour'] = df['timestamp'].dt.hour
df['weekday'] = df['timestamp'].dt.day_name()

# --------------------------
# Remove duplicates/spam
# --------------------------
```

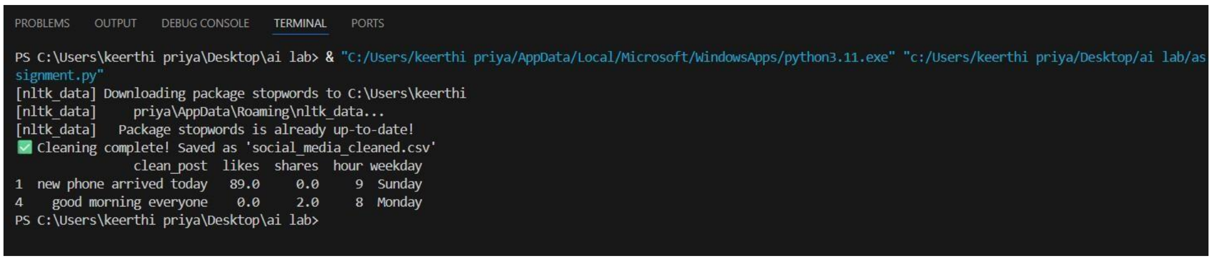


```python
# --------------------------
df = df.drop_duplicates(subset=['clean_post'])
df = df[df['clean_post'].str.split().str.len() > 2]

# --------------------------
# Save cleaned data
# --------------------------
cleaned_df = df[['clean_post', 'likes', 'shares', 'hour', 'weekday']]
cleaned_df.to_csv("social_media_cleaned.csv", index=False)

print("✅ Cleaning complete! Saved as 'social_media_cleaned.csv'")
print(cleaned_df)
```



```
clean_post,likes,shares,hour,weekday
new phone arrived today,89.0,0.0,9,Sunday
good morning everyone,0.0,2.0,8,Monday
```

OUTPUT :



```
PS C:\Users\keerthi priya\Desktop\ai lab> & "C:/Users/keerthi priya/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/keerthi priya/Desktop/ai lab/as
signment.py"
[nltk_data] Downloading package stopwords to C:\Users\keerthi
[nltk_data]     priya\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
✅ Cleaning complete! Saved as 'social_media_cleaned.csv'
                clean_post  likes  shares  hour weekday
1  new phone arrived today   89.0     0.0     9  Sunday
4    good morning everyone    0.0     2.0     8  Monday
PS C:\Users\keerthi priya\Desktop\ai lab>
```

OBSERVATION :

1.Modules used are pandas , re , numpy , nltk and datetime for data clean task

2.nltk stopwords downloaded

3. Cleaned the text by removing punctuation, symbols, and stopwords.

4. Handled missing values by replacing empty likes/shares with 0.

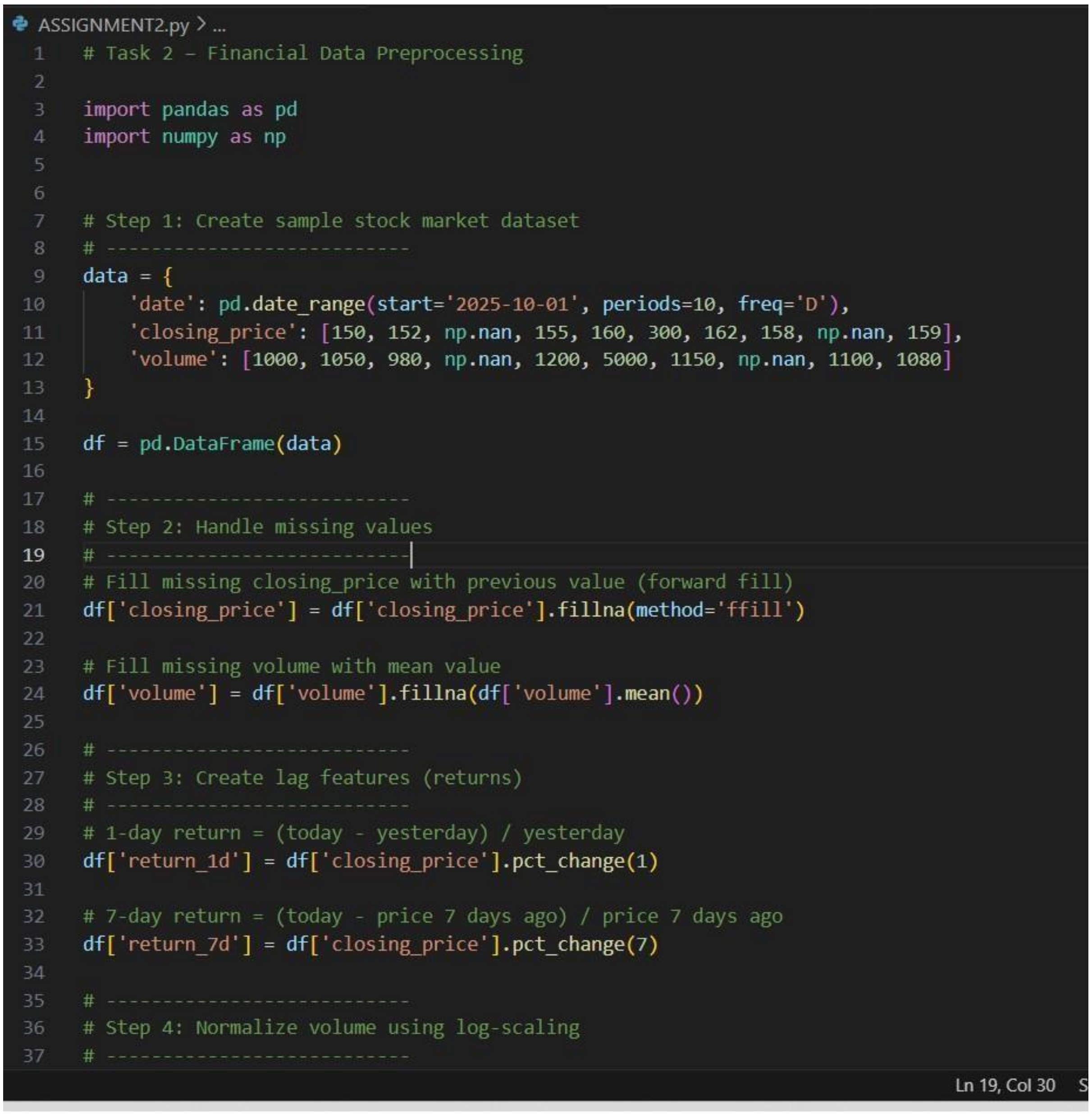
5. Saved the cleaned data into a new file named social_media_cleaned.csv.

#TASK-2

PROMPT :

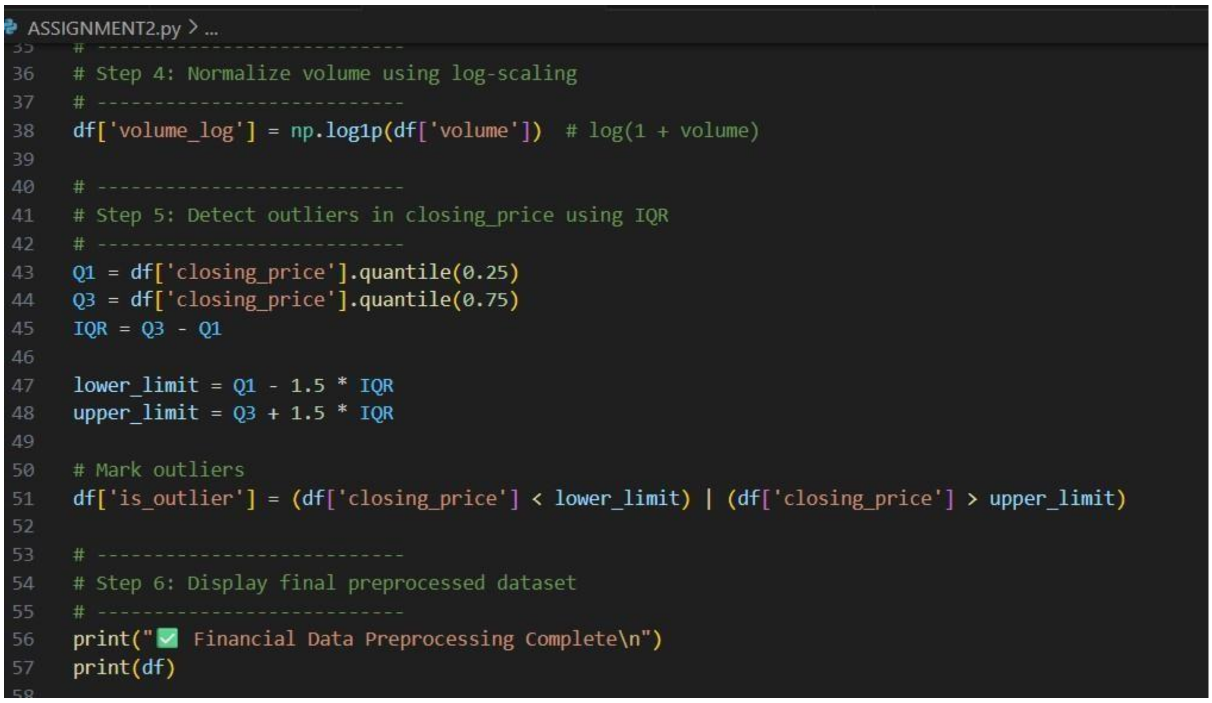
Write a python program to Preprocess a stock market dataset.
Instructions:

- 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 CODE :


ASSIGNMENT2.py > ...

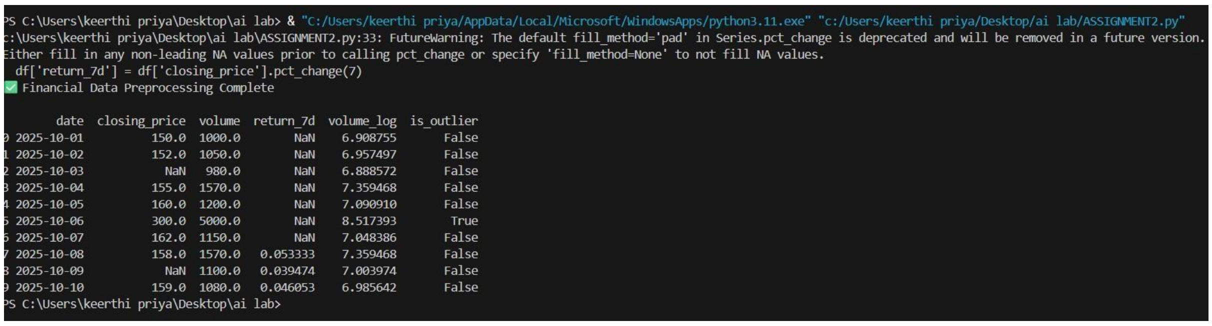
```python
# Task 2 – Financial Data Preprocessing

import pandas as pd
import numpy as np


# Step 1: Create sample stock market dataset
# ---------------------------
data = {
    'date': pd.date_range(start='2025-10-01', periods=10, freq='D'),
    'closing_price': [150, 152, np.nan, 155, 160, 300, 162, 158, np.nan, 159],
    'volume': [1000, 1050, 980, np.nan, 1200, 5000, 1150, np.nan, 1100, 1080]
}

df = pd.DataFrame(data)

# ---------------------------
# Step 2: Handle missing values
# ---------------------------
# Fill missing closing_price with previous value (forward fill)
df['closing_price'] = df['closing_price'].fillna(method='ffill')

# Fill missing volume with mean value
df['volume'] = df['volume'].fillna(df['volume'].mean())

# ---------------------------
# Step 3: Create lag features (returns)
# ---------------------------
# 1-day return = (today - yesterday) / yesterday
df['return_1d'] = df['closing_price'].pct_change(1)

# 7-day return = (today - price 7 days ago) / price 7 days ago
df['return_7d'] = df['closing_price'].pct_change(7)

# ---------------------------
# Step 4: Normalize volume using log-scaling
# ---------------------------
```

Ln 19, Col 30



```
ASSIGNMENT2.py > ...
    # ---------------------------
    # Step 4: Normalize volume using log-scaling
    # ---------------------------
    df['volume_log'] = np.log1p(df['volume'])  # log(1 + volume)

    # ---------------------------
    # Step 5: Detect outliers in closing_price using IQR
    # ---------------------------
    Q1 = df['closing_price'].quantile(0.25)
    Q3 = df['closing_price'].quantile(0.75)
    IQR = Q3 - Q1

    lower_limit = Q1 - 1.5 * IQR
    upper_limit = Q3 + 1.5 * IQR

    # Mark outliers
    df['is_outlier'] = (df['closing_price'] < lower_limit) | (df['closing_price'] > upper_limit)

    # ---------------------------
    # Step 6: Display final preprocessed dataset
    # ---------------------------
    print("✅ Financial Data Preprocessing Complete\n")
    print(df)
```

OUTPUT :



```
PS C:\Users\keerthi priya\Desktop\ai lab> & "C:/Users/keerthi priya/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/keerthi priya/Desktop/ai lab/ASSIGNMENT2.py"
c:\Users\keerthi priya\Desktop\ai lab\ASSIGNMENT2.py:33: FutureWarning: The default fill_method='pad' in Series.pct_change is deprecated and will be removed in a future version.
Either fill in any non-leading NA values prior to calling pct_change or specify 'fill_method=None' to not fill NA values.
  df['return_7d'] = df['closing_price'].pct_change(7)
✅ Financial Data Preprocessing Complete

        date  closing_price  volume  return_7d  volume_log  is_outlier
0 2025-10-01          150.0  1000.0        NaN    6.908755       False
1 2025-10-02          152.0  1050.0        NaN    6.957497       False
2 2025-10-03            NaN   980.0        NaN    6.888572       False
3 2025-10-04          155.0  1570.0        NaN    7.359468       False
4 2025-10-05          160.0  1200.0        NaN    7.090910       False
5 2025-10-06          300.0  5000.0        NaN    8.517393        True
6 2025-10-07          162.0  1150.0        NaN    7.048386       False
7 2025-10-08          158.0  1570.0   0.053333    7.359468       False
8 2025-10-09            NaN  1100.0   0.039474    7.003974       False
9 2025-10-10          159.0  1080.0   0.046053    6.985642       False
PS C:\Users\keerthi priya\Desktop\ai lab>
```

OBSERVATION :

1. Imported pandas and numpy for data handling and calculations.

2.Created a sample stock dataset with date, closing price, and volume.

3. Filled missing closing_price using fillna and volume with mean value.

4. Applied log normalization to the volume column to reduce skewness.

5. Added a new column is_outlier showing whether a price value is unusually high/low.

6. Final dataset is clean, normalized, and feature-rich for stock analysis or modelling

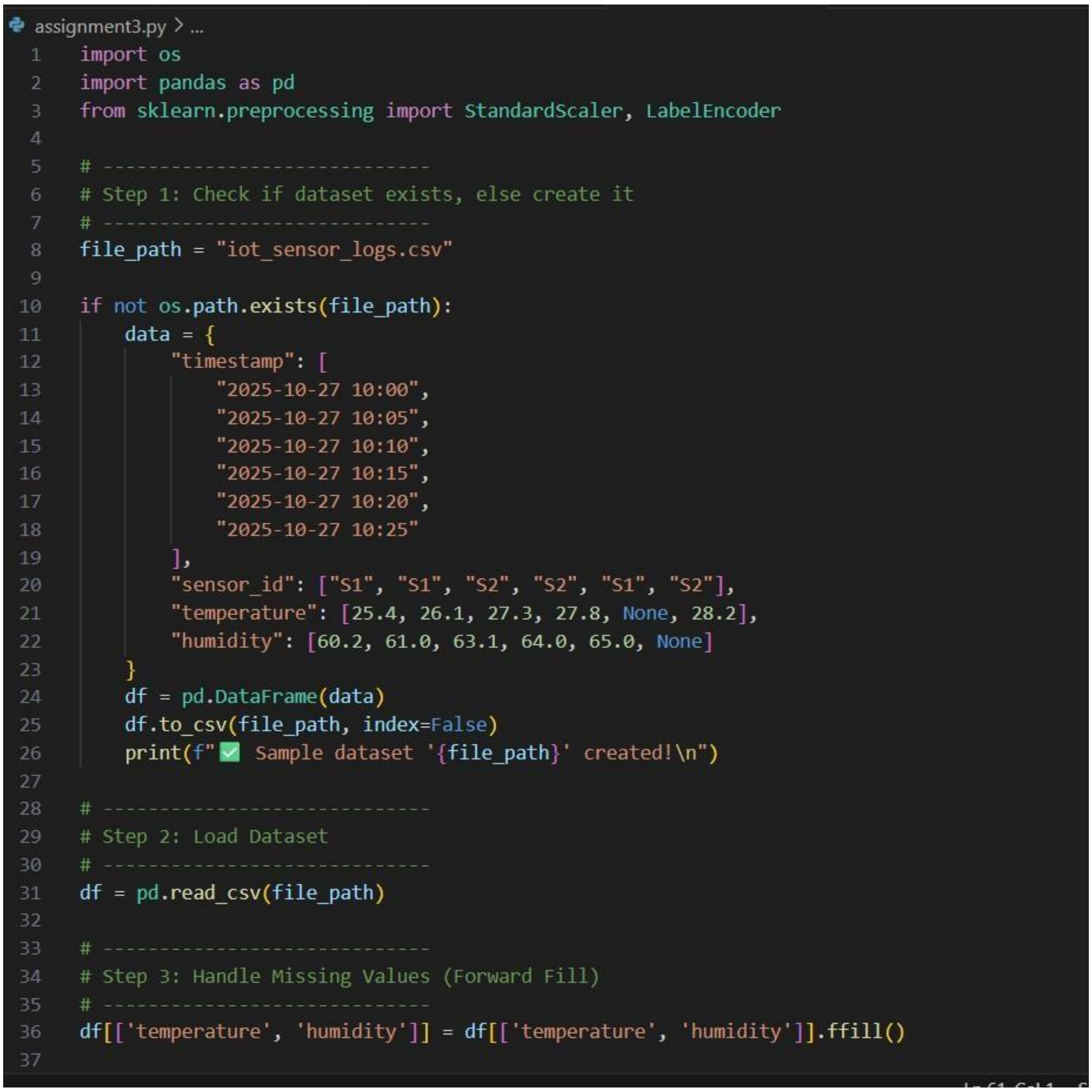
#TASK-3

PROMPT :

Write a python program to Clean and preprocess IoT temperature and humidity logs.
Instructions:

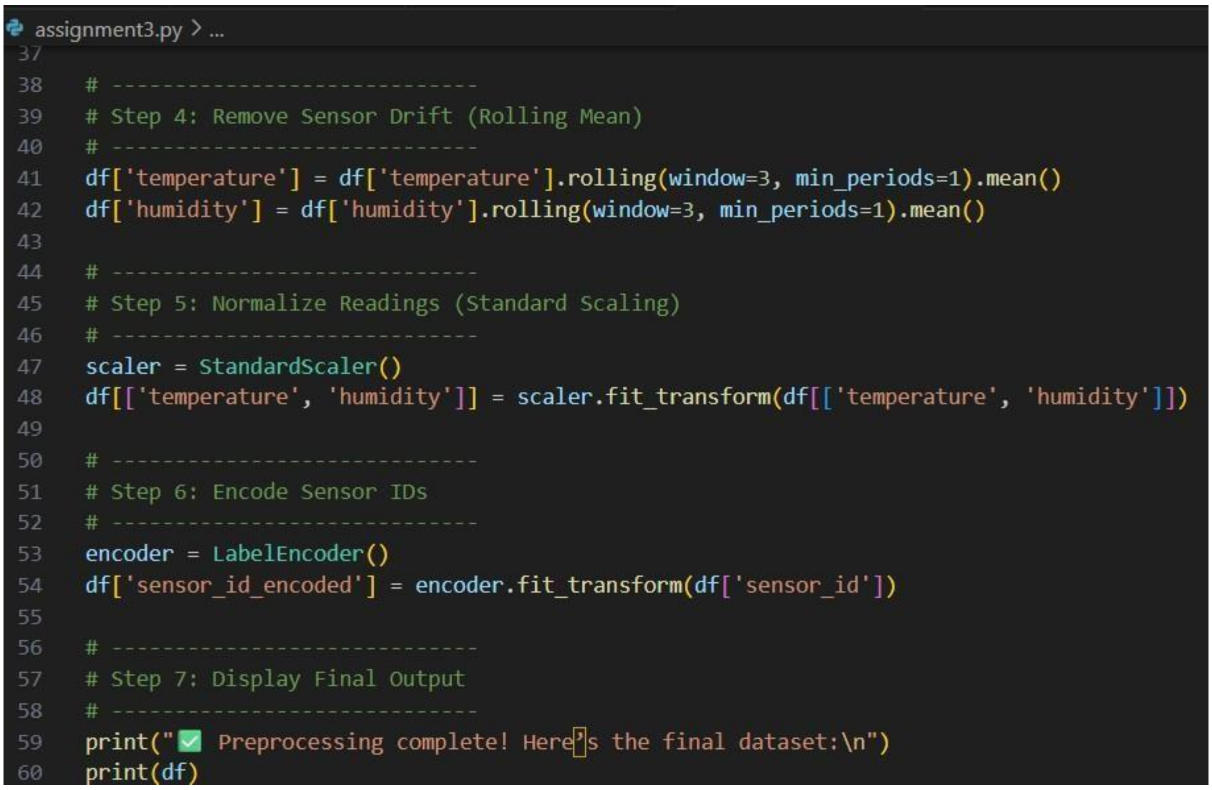
- Handle missing values using forward fill.
- Remove sensor drift (apply rolling mean).
- Normalize readings using standard scaling. - Encode categorical sensor IDs.

CODE :



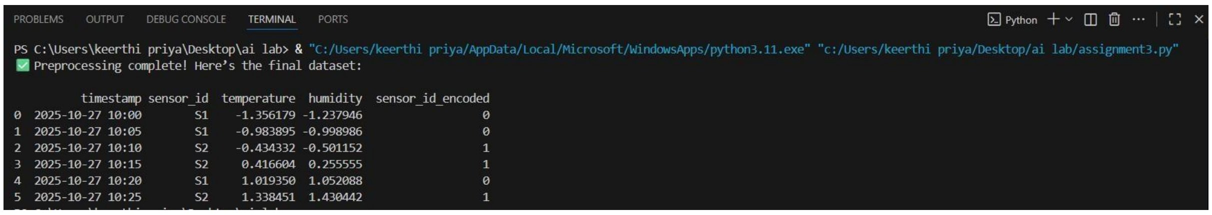
assignment3.py > ...

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

# ----------------------------
# Step 1: Check if dataset exists, else create it
# ----------------------------
file_path = "iot_sensor_logs.csv"

if not os.path.exists(file_path):
    data = {
        "timestamp": [
            "2025-10-27 10:00",
            "2025-10-27 10:05",
            "2025-10-27 10:10",
            "2025-10-27 10:15",
            "2025-10-27 10:20",
            "2025-10-27 10:25"
        ],
        "sensor_id": ["S1", "S1", "S2", "S2", "S1", "S2"],
        "temperature": [25.4, 26.1, 27.3, 27.8, None, 28.2],
        "humidity": [60.2, 61.0, 63.1, 64.0, 65.0, None]
    }
    df = pd.DataFrame(data)
    df.to_csv(file_path, index=False)
    print(f"✅ Sample dataset '{file_path}' created!\n")

# ----------------------------
# Step 2: Load Dataset
# ----------------------------
df = pd.read_csv(file_path)

# ----------------------------
# Step 3: Handle Missing Values (Forward Fill)
# ----------------------------
df[['temperature', 'humidity']] = df[['temperature', 'humidity']].ffill()

```



```python
assignment3.py > ...

# ---------------------------
# Step 4: Remove Sensor Drift (Rolling Mean)
# ---------------------------
df['temperature'] = df['temperature'].rolling(window=3, min_periods=1).mean()
df['humidity'] = df['humidity'].rolling(window=3, min_periods=1).mean()

# ---------------------------
# Step 5: Normalize Readings (Standard Scaling)
# ---------------------------
scaler = StandardScaler()
df[['temperature', 'humidity']] = scaler.fit_transform(df[['temperature', 'humidity']])

# ---------------------------
# Step 6: Encode Sensor IDs
# ---------------------------
encoder = LabelEncoder()
df['sensor_id_encoded'] = encoder.fit_transform(df['sensor_id'])

# ---------------------------
# Step 7: Display Final Output
# ---------------------------
print("✅ Preprocessing complete! Here's the final dataset:\n")
print(df)
```

OUTPUT :



```
PS C:\Users\keerthi priya\Desktop\ai lab> & "C:/Users/keerthi priya/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/keerthi priya/Desktop/ai lab/assignment3.py"
✅ Preprocessing complete! Here's the final dataset:

          timestamp sensor_id  temperature  humidity  sensor_id_encoded
0  2025-10-27 10:00        S1    -1.356179 -1.237946                  0
1  2025-10-27 10:05        S1    -0.983895 -0.998986                  0
2  2025-10-27 10:10        S2    -0.434332 -0.501152                  1
3  2025-10-27 10:15        S2     0.416604  0.255555                  1
4  2025-10-27 10:20        S1     1.019350  1.052088                  0
5  2025-10-27 10:25        S2     1.338451  1.430442                  1
```

OBSERVATION :

1. Missing values were filled using forward fill to preserve temporal continuity.

2. Rolling mean removes short-term noise and drift in sensor readings.

3.Standard scaling ensures both temperature and humidity have mean = 0 and standard deviation = 1 (good for anomaly models).

4. Sensor ID encoding converts categorical identifiers into numeric format required for ML algorithms.

#TASK- 4

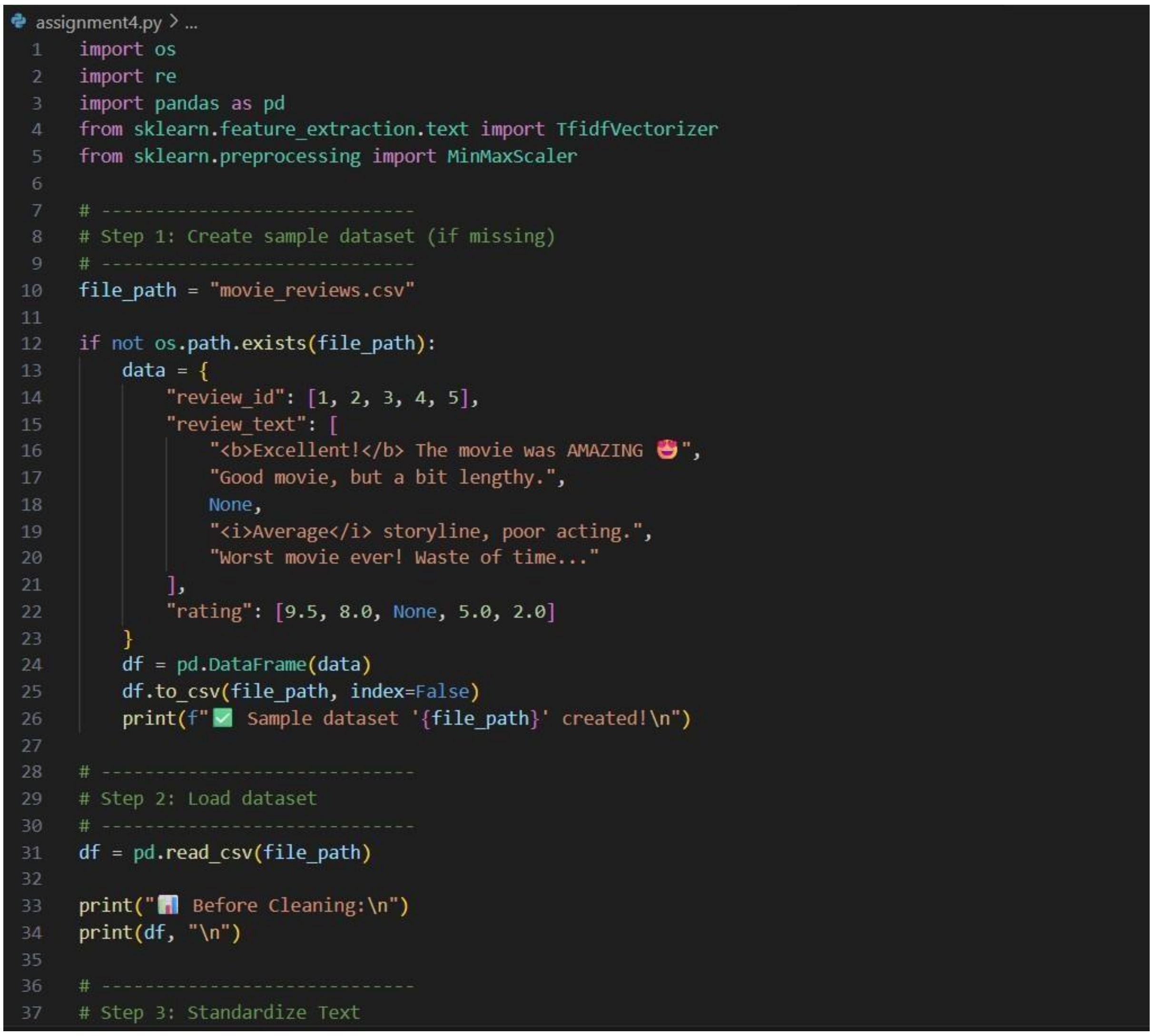
PROMPT :

A streaming platform wants to analyze customer reviews.

Instructions:

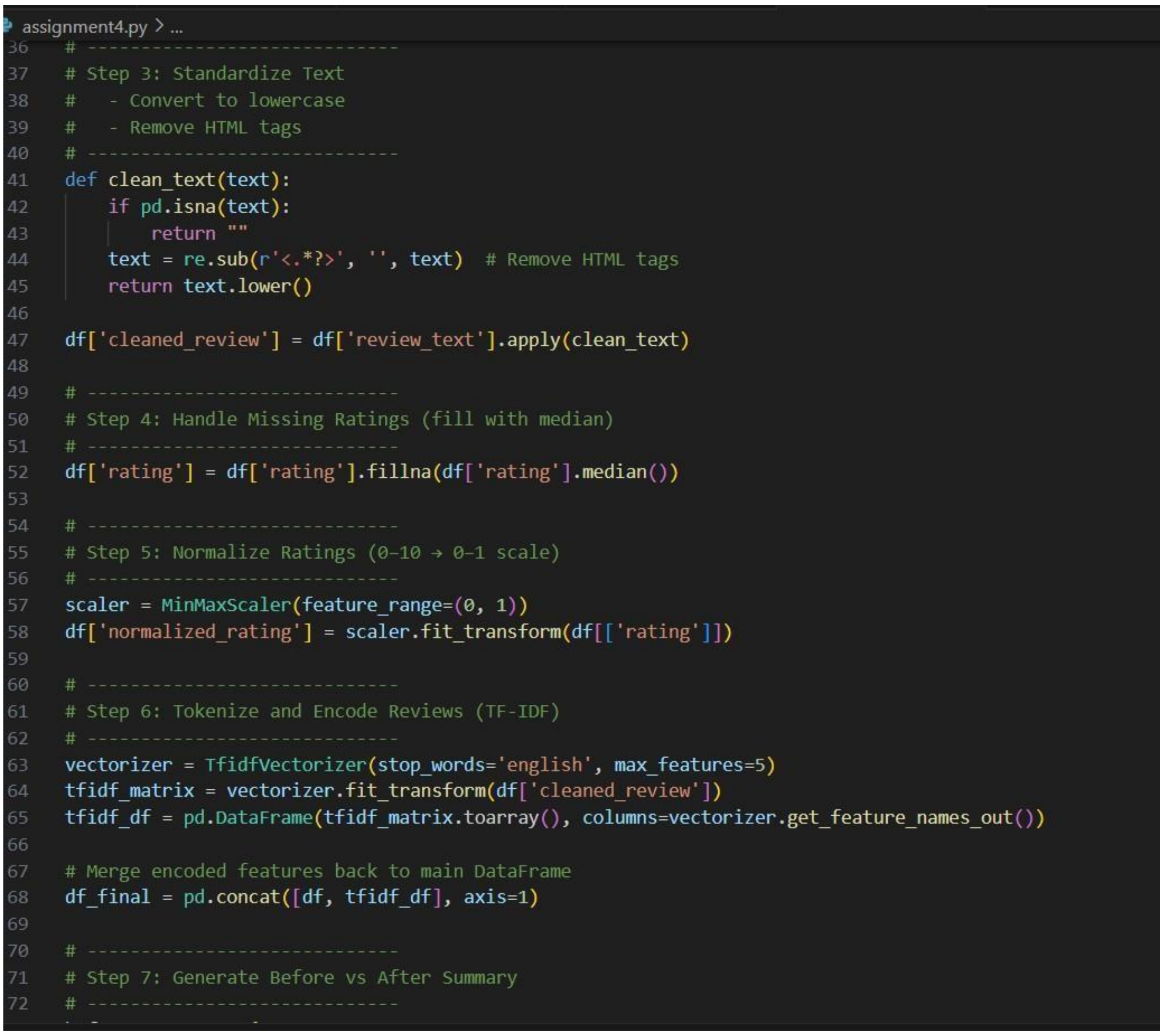
- Standardize text (lowercase, remove HTML tags).
- Tokenize and encode reviews using AI-assisted methods (TF-IDF or embeddings).
- Handle missing ratings (fill with median).
- Normalize ratings (0–10 → 0–1 scale).
- Generate a before vs after summary report.

CODE :

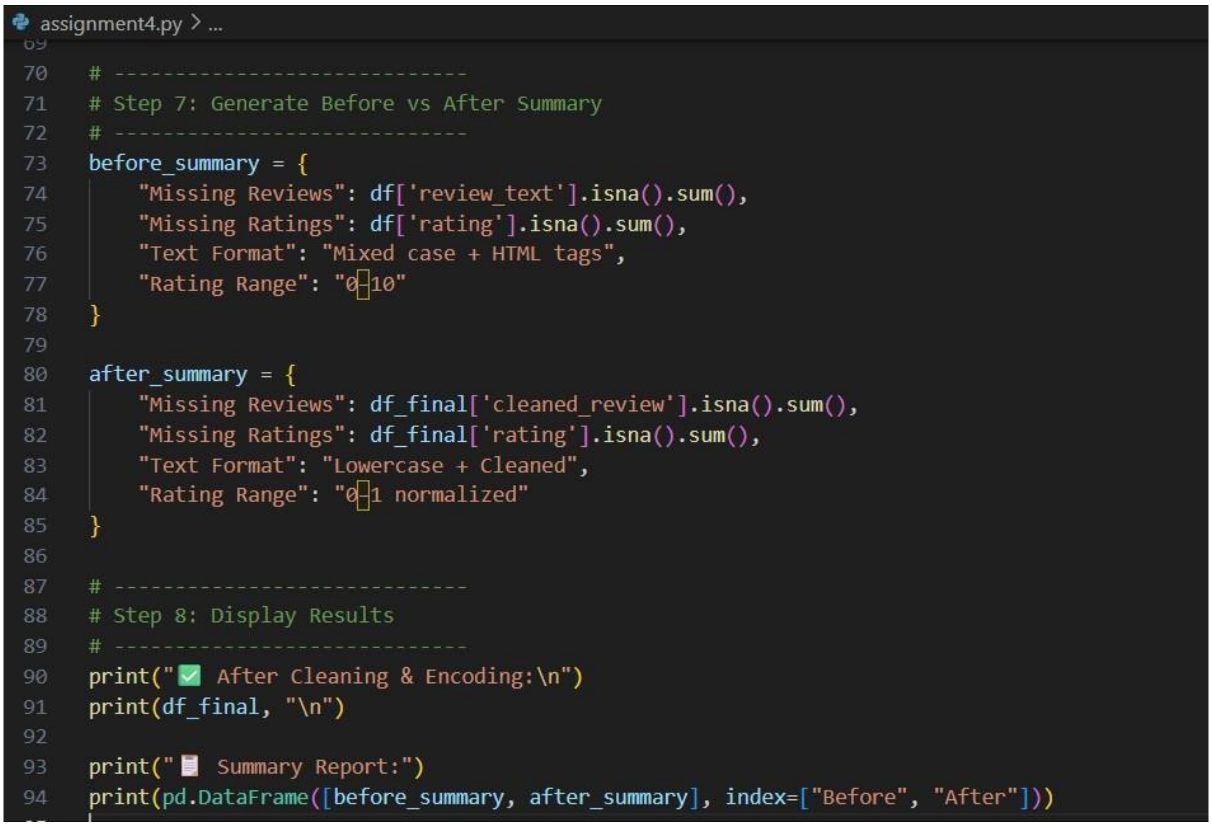


assignment4.py > ...

```python
import os
import re
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import MinMaxScaler

# ----------------------------
# Step 1: Create sample dataset (if missing)
# ----------------------------
file_path = "movie_reviews.csv"

if not os.path.exists(file_path):
    data = {
        "review_id": [1, 2, 3, 4, 5],
        "review_text": [
            "<b>Excellent!</b> The movie was AMAZING 😍",
            "Good movie, but a bit lengthy.",
            None,
            "<i>Average</i> storyline, poor acting.",
            "Worst movie ever! Waste of time..."
        ],
        "rating": [9.5, 8.0, None, 5.0, 2.0]
    }
    df = pd.DataFrame(data)
    df.to_csv(file_path, index=False)
    print(f"✅ Sample dataset '{file_path}' created!\n")

# ----------------------------
# Step 2: Load dataset
# ----------------------------
df = pd.read_csv(file_path)

print("📊 Before Cleaning:\n")
print(df, "\n")

# ----------------------------
# Step 3: Standardize Text
```


assignment4.py > ...

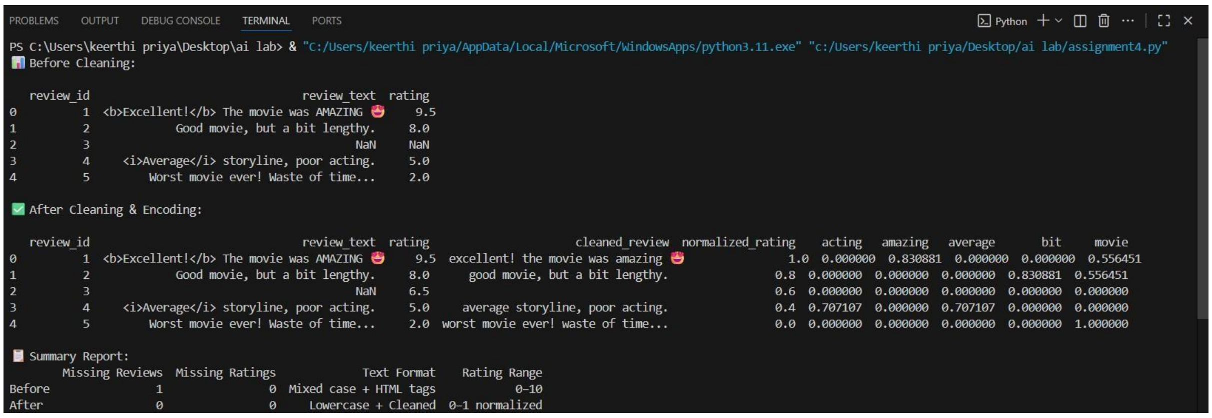
```python
# ----------------------------
# Step 3: Standardize Text
#   - Convert to lowercase
#   - Remove HTML tags
# ----------------------------
def clean_text(text):
    if pd.isna(text):
        return ""
    text = re.sub(r'<.*?>', '', text)  # Remove HTML tags
    return text.lower()

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

# ----------------------------
# Step 4: Handle Missing Ratings (fill with median)
# ----------------------------
df['rating'] = df['rating'].fillna(df['rating'].median())

# ----------------------------
# Step 5: Normalize Ratings (0–10 → 0–1 scale)
# ----------------------------
scaler = MinMaxScaler(feature_range=(0, 1))
df['normalized_rating'] = scaler.fit_transform(df[['rating']])

# ----------------------------
# Step 6: Tokenize and Encode Reviews (TF-IDF)
# ----------------------------
vectorizer = TfidfVectorizer(stop_words='english', max_features=5)
tfidf_matrix = vectorizer.fit_transform(df['cleaned_review'])
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())

# Merge encoded features back to main DataFrame
df_final = pd.concat([df, tfidf_df], axis=1)

# ----------------------------
# Step 7: Generate Before vs After Summary
# ----------------------------
```



assignment4.py > ...

```python
# ----------------------------
# Step 7: Generate Before vs After Summary
# ----------------------------
before_summary = {
    "Missing Reviews": df['review_text'].isna().sum(),
    "Missing Ratings": df['rating'].isna().sum(),
    "Text Format": "Mixed case + HTML tags",
    "Rating Range": "0–10"
}

after_summary = {
    "Missing Reviews": df_final['cleaned_review'].isna().sum(),
    "Missing Ratings": df_final['rating'].isna().sum(),
    "Text Format": "Lowercase + Cleaned",
    "Rating Range": "0–1 normalized"
}

# ----------------------------
# Step 8: Display Results
# ----------------------------
print("✅ After Cleaning & Encoding:\n")
print(df_final, "\n")

print("📋 Summary Report:")
print(pd.DataFrame([before_summary, after_summary], index=["Before", "After"]))
```

OUTPUT :



```
PS C:\Users\keerthi priya\Desktop\ai lab> & "C:/Users/keerthi priya/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/keerthi priya/Desktop/ai lab/assignment4.py"
📊 Before Cleaning:

   review_id                               review_text  rating
0          1  <b>Excellent!</b> The movie was AMAZING 😍     9.5
1          2              Good movie, but a bit lengthy.     8.0
2          3                                         NaN     NaN
3          4     <i>Average</i> storyline, poor acting.     5.0
4          5          Worst movie ever! Waste of time...     2.0

✅ After Cleaning & Encoding:

   review_id                               review_text  rating                         cleaned_review  normalized_rating    acting   amazing   average       bit     movie
0          1  <b>Excellent!</b> The movie was AMAZING 😍     9.5  excellent! the movie was amazing 😍                1.0  0.000000  0.830881  0.000000  0.000000  0.556451
1          2              Good movie, but a bit lengthy.     8.0     good movie, but a bit lengthy.                0.8  0.000000  0.000000  0.000000  0.830881  0.556451
2          3                                         NaN     6.5                                               0.6  0.000000  0.000000  0.000000  0.000000  0.000000
3          4     <i>Average</i> storyline, poor acting.     5.0    average storyline, poor acting.                0.4  0.707107  0.000000  0.707107  0.000000  0.000000
4          5          Worst movie ever! Waste of time...     2.0  worst movie ever! waste of time...                0.0  0.000000  0.000000  0.000000  0.000000  1.000000

📋 Summary Report:
        Missing Reviews  Missing Ratings             Text Format     Rating Range
Before                1                0  Mixed case + HTML tags             0-10
After                 0                0     Lowercase + Cleaned  0-1 normalized
```

OBSERVATION :

1. All reviews were converted to lowercase for consistency.

2. HTML tags such as <b>, <i> were successfully removed.

3.Missing review texts were replaced with empty strings ("").

4.Removed inconsistencies in text format and missing values.

5.The final dataset is clean, consistent, and AI-ready for sentiment or anomaly detection tasks.