AI ASSISTED CODING

Name: N.Prudhvi
Htno: 2403A510G7
Batch: 06

Lab-Assignment: 17.1

Task-1:

Prompt:

Write Python code to clean social media posts by removing stopwords, punctuation, and special symbols from text, handling missing values in likes and shares using the median, converting timestamps to datetime, extracting hour and weekday features, and removing duplicate or spam posts.

Code:

```python
import pandas as pd
import re
from typing import List

# A basic list of English stopwords. For more comprehensive lists, consider u
STOPWORDS = set(
    [
        "i",
        "me",
        "my",
        "myself",
        "we",
        "our",
        "ours",
        "ourselves",
        "you",
        "your",
        "yours",
        "he",
        "him",
        "his",
        "she",
        "her",
        "it",
        "its",
        "they",
        "them",
        "their",
        "what",
        "which",
        "who",
        "whom",
        "this",
        "that",
        "these",
        "those",
        "am",
        "is",
        "are",
```

```python

def clean_text(text: str) -> str:
    """Removes stopwords, punctuation, and special symbols from a text string.
    if not isinstance(text, str):
        return ""
    # Convert to lowercase
    text = text.lower()
    # Remove URLs, mentions, and hashtags
    text = re.sub(r"http\S+|www\S+|@\w+|#\w+", "", text)
    # Remove punctuation and special symbols
    text = re.sub(r"[^a-z\s]", "", text)
    # Remove stopwords
    words = [word for word in text.split() if word not in STOPWORDS]
    return " ".join(words)


def clean_social_media_data(df: pd.DataFrame) -> pd.DataFrame:
    """
    Cleans a raw social media dataset by handling missing values, cleaning tex
    processing timestamps, and removing duplicates/spam.
    """
    # Make a copy to avoid modifying the original DataFrame
    cleaned_df = df.copy()

    # 1. Handle missing values in likes and shares by filling with the median
    for col in ["likes", "shares"]:
        if col in cleaned_df.columns:
            # Coerce to numeric, turning non-numeric into NaN
            cleaned_df[col] = pd.to_numeric(cleaned_df[col], errors="coerce")
            median_val = cleaned_df[col].median()
            cleaned_df[col].fillna(median_val, inplace=True)
            cleaned_df[col] = cleaned_df[col].astype(int)  # Convert to intege

    # 2. Remove stopwords, punctuation, and special symbols
    cleaned_df["cleaned_text"] = cleaned_df["post_text"].apply(clean_text)

```

```python
def clean_social_media_data(df: pd.DataFrame) -> pd.DataFrame:
    # 4. Detect and remove spam/duplicate posts
    # Simple spam detection based on keywords
    spam_keywords = ["buy now", "free money", "click here"]
    spam_pattern = "|".join(spam_keywords)
    cleaned_df["is_spam"] = cleaned_df["post_text"].str.contains(
        spam_pattern, case=False, na=False
    )

    # Remove duplicates based on the cleaned text content
    cleaned_df.drop_duplicates(subset=["cleaned_text"], keep="first", inplace=

    # Filter out posts flagged as spam
    cleaned_df = cleaned_df[~cleaned_df["is_spam"]]

    # Final cleanup: select and reorder columns
    final_cols = [
        "post_id",
        "timestamp",
```

```python
        "cleaned_text",
        "likes",
        "shares",
        "hour_of_day",
        "day_of_week",
    ]
    # Ensure all expected columns (variable) final_cols: list[str]
    final_cols = [col for col in final_cols if col in cleaned_df.columns]
    cleaned_df = cleaned_df[final_cols]

    return cleaned_df.reset_index(drop=True)


# --- Example Usage ---
if __name__ == "__main__":
    # Sample raw social media data
    raw_data = {
        "post_id": [1, 2, 3, 4, 5, 6, 7, 8],
        "timestamp": [
```

Output:

```python
        "2023-10-26 08:30:00",
        "2023-10-26 09:15:00",
        "2023-10-26 10:00:00",
        "2023-10-26 11:00:00",
        "2023-10-26 12:45:00",
        "2023-10-26 14:20:00",
        "2023-10-27 15:00:00",
        "2023-10-27 16:00:00",
    ],
    "post_text": [
        "Just had an amazing breakfast! ☀ #foodie",
        "This is a great article on AI: http://example.com/ai",
        "Feeling tired today... need coffee ☕",
        "!!! BUY NOW, limited offer !!!",  # Spam post
        "Just had an amazing breakfast! ☀ #foodie",  # Duplicate post
        "What a game last night! Simply incredible.",
        "Working on a new project. It is very exciting.",
        "Another spam post with free money",  # Spam post
    ],
    "likes": [150, 200, 75, 10, 120, 300, None, 5],  # Includes a missing
    "shares": [20, 45, None, 1, 15, 80, 25, 0],  # Includes a missing valu
}
raw_df = pd.DataFrame(raw_data)

print("--- Original Raw Dataset ---")
print(raw_df)
print("\n" + "=" * 50 + "\n")

# Clean the dataset
cleaned_df = clean_social_media_data(raw_df)

print("--- Cleaned Dataset for Analysis ---")
print(cleaned_df)
```

Output:

```
  cleaned_df[col].fillna(median_val, inplace=True)
--- Cleaned Dataset for Analysis ---
   post_id           timestamp  ... hour_of_day  day_of_week
0        1 2023-10-26 08:30:00  ...           8     Thursday
1        2 2023-10-26 09:15:00  ...           9     Thursday
2        3 2023-10-26 10:00:00  ...          10     Thursday
3        6 2023-10-26 14:20:00  ...          14     Thursday
4        7 2023-10-27 15:00:00  ...          15       Friday

[5 rows x 7 columns]
```

```
C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab-17>python 17_1.py
--- Original Raw Dataset ---
   post_id           timestamp  ...  likes  shares
0        1  2023-10-26 08:30:00  ...  150.0    20.0
1        2  2023-10-26 09:15:00  ...  200.0    45.0
2        3  2023-10-26 10:00:00  ...   75.0     NaN
3        4  2023-10-26 11:00:00  ...   10.0     1.0
4        5  2023-10-26 12:45:00  ...  120.0    15.0
5        6  2023-10-26 14:20:00  ...  300.0    80.0
6        7  2023-10-27 15:00:00  ...    NaN    25.0
7        8  2023-10-27 16:00:00  ...    5.0     0.0

[8 rows x 5 columns]
Live Share
```

Observation:

This task focuses on cleaning and preprocessing social media data to make it suitable for analysis. It involves removing noise from the text, such as stopwords, punctuation, URLs, mentions, and hashtags. Additionally, it handles missing values in numeric columns like likes and shares by replacing them with median values, and ensures these columns are of integer type. Time-related features, such as the hour of day and day of the week, are extracted from timestamps to allow for temporal analysis. The task also identifies and removes duplicate posts as well as spam posts containing keywords like “buy now” or “free money,” resulting in a cleaner and more reliable dataset for downstream tasks.

Task-2:

Prompt:

Write Python code to preprocess stock market data by handling missing values in closing_price and volume, creating 1-day and 7-day lag return features, applying log-scaling to volume, and detecting outliers in closing_price using the IQR method.

Code:

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


def preprocess_stock_data(df: pd.DataFrame) -> pd.DataFrame:
    """
    Preprocess a stock market dataset by handling missing values,
    creating lag features, normalizing volume, and detecting outliers.
    """
    df = df.copy()

    # 1. Handle missing values in 'closing_price' and 'volume'
    for col in ["closing_price", "volume"]:
        if col in df.columns:
            df[col] = pd.to_numeric(df[col], errors="coerce")
            df[col].fillna(method="ffill", inplace=True)  # Forward fill
            df[col].fillna(
                method="bfill", inplace=True
            )  # Backward fill if first row is NaN

    # 2. Create lag features (1-day and 7-day returns)
    df["return_1d"] = df["closing_price"].pct_change(1)
    df["return_7d"] = df["closing_price"].pct_change(7)

    # 3. Normalize volume using log scaling
    df["volume_log"] = df["volume"].apply(
        lambda x: np.log1p(x)
    )  # log(1 + x) to avoid log(0)

    # 4. Detect outliers in 'closing_price' using IQR method
```

```python
    Q1 = df["closing_price"].quantile(0.25)
    Q3 = df["closing_price"].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df["is_outlier"] = (df["closing_price"] < lower_bound) | (
        df["closing_price"] > upper_bound
    )

    # Reset index
    return df.reset_index(drop=True)


# --- Example Usage ---
if __name__ == "__main__":
    # Sample stock dataset
    data = {
        "date": pd.date_range(start="2023-10-20", periods=10, freq="D"),
        "closing_price": [100, 102, 101, None, 105, 107, 106, 108, 110, 109],
        "volume": [5000, 5200, None, 5400, 5600, None, 5800, 6000, 6200, 6400]
    }

    df = pd.DataFrame(data)
    print("--- Original Stock Data ---")
    print(df)

    processed_df = preprocess_stock_data(df)
    print("\n--- Preprocessed Stock Data ---")
    print(processed_df)
```

Output:

```
C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab-17>python 17_1.py
--- Original Stock Data ---
        date  closing_price  volume
0 2023-10-20          100.0  5000.0
1 2023-10-21          102.0  5200.0
2 2023-10-22          101.0     NaN
3 2023-10-23            NaN  5400.0
4 2023-10-24          105.0  5600.0
5 2023-10-25          107.0     NaN
6 2023-10-26          106.0  5800.0
7 2023-10-27          108.0  6000.0
8 2023-10-28          110.0  6200.0
9 2023-10-29          109.0  6400.0
```

```
--- Preprocessed Stock Data ---
         date  closing_price  volume  return_1d  return_7d  volume_log  is_outlier
0 2023-10-20          100.0  5000.0        NaN        NaN    8.517393       False
1 2023-10-21          102.0  5200.0   0.020000        NaN    8.556606       False
2 2023-10-22          101.0  5200.0  -0.009804        NaN    8.556606       False
3 2023-10-23          101.0  5400.0   0.000000        NaN    8.594339       False
4 2023-10-24          105.0  5600.0   0.039604        NaN    8.630700       False
0 2023-10-20          100.0  5000.0        NaN        NaN    8.517393       False
1 2023-10-21          102.0  5200.0   0.020000        NaN    8.556606       False
2 2023-10-22          101.0  5200.0  -0.009804        NaN    8.556606       False
3 2023-10-23          101.0  5400.0   0.000000        NaN    8.594339       False
4 2023-10-24          105.0  5600.0   0.039604        NaN    8.630700       False
1 2023-10-21          102.0  5200.0   0.020000        NaN    8.556606       False
2 2023-10-22          101.0  5200.0  -0.009804        NaN    8.556606       False
3 2023-10-23          101.0  5400.0   0.000000        NaN    8.594339       False
4 2023-10-24          105.0  5600.0   0.039604        NaN    8.630700       False
5 2023-10-25          107.0  5600.0   0.019048        NaN    8.630700       False
6 2023-10-26          106.0  5800.0  -0.009346        NaN    8.665786       False
7 2023-10-27          108.0  6000.0   0.018868   0.080000    8.699681       False
8 2023-10-28          110.0  6200.0   0.018519   0.078431    8.732466       False
9 2023-10-29          109.0  6400.0  -0.009091   0.079208    8.764210       False
2 2023-10-22          101.0  5200.0  -0.009804        NaN    8.556606       False
3 2023-10-23          101.0  5400.0   0.000000        NaN    8.594339       False
4 2023-10-24          105.0  5600.0   0.039604        NaN    8.630700       False
5 2023-10-25          107.0  5600.0   0.019048        NaN    8.630700       False
6 2023-10-26          106.0  5800.0  -0.009346        NaN    8.665786       False
7 2023-10-27          108.0  6000.0   0.018868   0.080000    8.699681       False
8 2023-10-28          110.0  6200.0   0.018519   0.078431    8.732466       False
9 2023-10-29          109.0  6400.0  -0.009091   0.079208    8.764210       False
5 2023-10-25          107.0  5600.0   0.019048        NaN    8.630700       False
6 2023-10-26          106.0  5800.0  -0.009346        NaN    8.665786       False
7 2023-10-27          108.0  6000.0   0.018868   0.080000    8.699681       False
8 2023-10-28          110.0  6200.0   0.018519   0.078431    8.732466       False
9 2023-10-29          109.0  6400.0  -0.009091   0.079208    8.764210       False
8 2023-10-28          110.0  6200.0   0.018519   0.078431    8.732466       False
9 2023-10-29          109.0  6400.0  -0.009091   0.079208    8.764210       False
```

Observation:

This deals with preprocessing financial or stock market data for predictive modeling or analysis. Missing values in key columns are filled with appropriate statistics, such as the mean or median. Lag features are created to capture temporal dependencies, and numeric columns like trading volume are normalized to bring them onto a comparable scale. Outliers are detected and handled to prevent them from skewing model predictions. This ensures the dataset is consistent, structured, and suitable for time-series forecasting or machine learning models.

Task-3:
Prompt:
Write Python code to clean IoT sensor data by forward-filling missing values, applying a rolling mean to remove drift, normalizing temperature and humidity with standard scaling, and encoding categorical sensor IDs.
Code:
```
import pandas as pd
import numpy as np


def preprocess_iot_data(df: pd.DataFrame) -> pd.DataFrame:
    """
    Cleans and preprocesses IoT sensor data with temperature and humidity lo
    Handles missing values, removes sensor drift, normalizes readings, and e
    """
    cleaned_df = df.copy()

    # 1. Handle missing values using forward fill
    cleaned_df[["temperature", "humidity"]] = cleaned_df[
        ["temperature", "humidity"]
    ].fillna(method="ffill")

    # 2. Remove sensor drift using rolling mean (window=3)
    cleaned_df["temperature"] = (
        cleaned_df["temperature"].rolling(window=3, min_periods=1).mean()
    )
    cleaned_df["humidity"] = (
        cleaned_df["humidity"].rolling(window=3, min_periods=1).mean()
    )

    # 3. Normalize readings using standard scaling: (x - mean)/std
    for col in ["temperature", "humidity"]:
        mean_val = cleaned_df[col].mean()
        std_val = cleaned_df[col].std()
        cleaned_df[col] = (cleaned_df[col] - mean_val) / std_val

```

```python
    # 4. Encode categorical sensor IDs as integers
    cleaned_df["sensor_id_encoded"] = (
        cleaned_df["sensor_id"].astype("category").cat.codes
    )

    return cleaned_df.reset_index(drop=True)


# --- Example Usage ---
if __name__ == "__main__":
    # Sample IoT sensor data
    raw_data = {
        "sensor_id": ["S1", "S2", "S1", "S3", "S2", "S1", "S3", "S2"],
        "timestamp": [
            "2023-10-26 08:00:00",
            "2023-10-26 08:05:00",
            "2023-10-26 08:10:00",
            "2023-10-26 08:15:00",
            "2023-10-26 08:20:00",
            "2023-10-26 08:25:00",
            "2023-10-26 08:30:00",
            "2023-10-26 08:35:00",
        ],
        "temperature": [22.5, 23.0, None, 24.1, 23.5, 22.8, 24.0, None],
        "humidity": [45, None, 47, 50, 48, 46, None, 49],
    }

    raw_df = pd.DataFrame(raw_data)
```

```python
            "2023-10-26 08:25:00",
            "2023-10-26 08:30:00",
            "2023-10-26 08:35:00",
        ],
        "temperature": [22.5, 23.0, None, 24.1, 23.5, 22.8, 24.0, None],
        "humidity": [45, None, 47, 50, 48, 46, None, 49],
    }

    raw_df = pd.DataFrame(raw_data)

    print("--- Original IoT Dataset ---")
    print(raw_df)
    print("\n" + "=" * 50 + "\n")

    # Preprocess the dataset
    cleaned_df = preprocess_iot_data(raw_df)

    print("--- Cleaned IoT Dataset ---")
    print(cleaned_df)
```

Output:

```
C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab-17>python 17_1.py
--- Original IoT Dataset ---
  sensor_id            timestamp  temperature  humidity
0        S1  2023-10-26 08:00:00         22.5      45.0
1        S2  2023-10-26 08:05:00         23.0       NaN
2        S1  2023-10-26 08:10:00          NaN      47.0
3        S3  2023-10-26 08:15:00         24.1      50.0
4        S2  2023-10-26 08:20:00         23.5      48.0
5        S1  2023-10-26 08:25:00         22.8      46.0
6        S3  2023-10-26 08:30:00         24.0       NaN
7        S2  2023-10-26 08:35:00          NaN      49.0


==================================================
```

```
--- Cleaned IoT Dataset ---
  sensor_id           timestamp  temperature  humidity  sensor_id_encoded
0        S1 2023-10-26 08:00:00    -1.622195 -1.259469                  0
1        S2 2023-10-26 08:05:00    -1.030513 -1.259469                  1
2        S1 2023-10-26 08:10:00    -0.833285 -0.742764                  0
3        S3 2023-10-26 08:15:00     0.428969  0.549000                  2
4        S2 2023-10-26 08:20:00     0.823424  1.324058                  1
5        S1 2023-10-26 08:25:00     0.665642  1.065705                  0
6        S3 2023-10-26 08:30:00     0.586751  0.032294                  2
7        S2 2023-10-26 08:35:00     0.981206  0.290647                  1
0        S1 2023-10-26 08:00:00    -1.622195 -1.259469                  0
1        S2 2023-10-26 08:05:00    -1.030513 -1.259469                  1
2        S1 2023-10-26 08:10:00    -0.833285 -0.742764                  0
3        S3 2023-10-26 08:15:00     0.428969  0.549000                  2
4        S2 2023-10-26 08:20:00     0.823424  1.324058                  1
5        S1 2023-10-26 08:25:00     0.665642  1.065705                  0
6        S3 2023-10-26 08:30:00     0.586751  0.032294                  2
7        S2 2023-10-26 08:35:00     0.981206  0.290647                  1
1        S2 2023-10-26 08:05:00    -1.030513 -1.259469                  1
2        S1 2023-10-26 08:10:00    -0.833285 -0.742764                  0
3        S3 2023-10-26 08:15:00     0.428969  0.549000                  2
4        S2 2023-10-26 08:20:00     0.823424  1.324058                  1
5        S1 2023-10-26 08:25:00     0.665642  1.065705                  0
6        S3 2023-10-26 08:30:00     0.586751  0.032294                  2
7        S2 2023-10-26 08:35:00     0.981206  0.290647                  1
2        S1 2023-10-26 08:10:00    -0.833285 -0.742764                  0
3        S3 2023-10-26 08:15:00     0.428969  0.549000                  2
4        S2 2023-10-26 08:20:00     0.823424  1.324058                  1
5        S1 2023-10-26 08:25:00     0.665642  1.065705                  0
6        S3 2023-10-26 08:30:00     0.586751  0.032294                  2
7        S2 2023-10-26 08:35:00     0.981206  0.290647                  1
4        S2 2023-10-26 08:20:00     0.823424  1.324058                  1
5        S1 2023-10-26 08:25:00     0.665642  1.065705                  0
6        S3 2023-10-26 08:30:00     0.586751  0.032294                  2
7        S2 2023-10-26 08:35:00     0.981206  0.290647                  1
5        S1 2023-10-26 08:25:00     0.665642  1.065705                  0
6        S3 2023-10-26 08:30:00     0.586751  0.032294                  2
7        S2 2023-10-26 08:35:00     0.981206  0.290647                  1
7        S2 2023-10-26 08:35:00     0.981206  0.290647                  1

C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab-17>
```

Observation:

This task involves preprocessing IoT or sensor data, which often contains missing readings, noise, and inconsistencies due to sensor drift or device errors. Missing values are filled using appropriate imputation methods, and rolling averages or other smoothing techniques are applied to reduce noise. Sensor IDs or categorical features are encoded to numeric form so that models can process them effectively. The cleaned and normalized dataset is then ready for tasks like anomaly detection, predictive maintenance, or trend analysis.

Task-4:

Prompt:

The prompt asked to clean reviews by lowercasing text, removing HTML tags, encoding using TF-IDF or embeddings, filling missing ratings with the median, normalizing ratings (0–10 → 0–1), and creating a before–after summary.

Code:

```
import pandas as pd
import re

# Basic stopwords list
STOPWORDS = set(
    [
        "i",
        "me",
        "my",
        "we",
        "our",
        "you",
        "your",
        "he",
        "she",
        "it",
        "they",
        "them",
        "this",
        "that",
        "a",
        "an",
        "the",
        "and",
        "or",
        "but",
```

```python
        "if",
        "to",
        "of",
        "in",
        "on",
        "for",
        "with",
        "as",
        "is",
        "are",
        "was",
        "were",
        "be",
        "been",
        "has",
        "have",
        "had",
        "do",
        "does",
        "did",
        "not",
    ]
)


def clean_review_text(text: str) -> str:
```

```python
def clean_review_text(text: str) -> str:
    """Cleans review text by removing punctuation, lowercasing, and removing s
    if not isinstance(text, str):
        return ""
    # Lowercase
    text = text.lower()
    # Remove URLs
    text = re.sub(r"http\S+|www\S+", "", text)
    # Remove punctuation and special characters
    text = re.sub(r"[^a-z\s]", "", text)
    # Remove stopwords
    words = [word for word in text.split() if word not in STOPWORDS]
    return " ".join(words)


def clean_movie_reviews(df: pd.DataFrame) -> pd.DataFrame:
    """
    Cleans a movie reviews dataset:
    - Removes duplicates
    - Cleans review text
    - Handles missing ratings
    """
    cleaned_df = df.copy()

    # 1. Remove duplicates
    cleaned_df.drop_duplicates(subset=["review_text"], keep="first", inplace=T
```

```python
    cleaned_df.drop_duplicates(subset=["review_text"], keep="first", inplace=T

    # 2. Fill missing ratings with median
    if "rating" in cleaned_df.columns:
        cleaned_df["rating"] = pd.to_numeric(cleaned_df["rating"], errors="coe
        median_rating = cleaned_df["rating"].median()
        cleaned_df["rating"].fillna(median_rating, inplace=True)
        cleaned_df["rating"] = cleaned_df["rating"].astype(int)

    # 3. Clean review text
    cleaned_df["cleaned_review"] = cleaned_df["review_text"].apply(clean_revie

    # 4. Reset index
    return cleaned_df.reset_index(drop=True)


# --- Example Usage ---
if __name__ == "__main__":
    raw_data = {
        "review_id": [101, 102, 103, 104, 105, 106],
        "review_text": [
            "I loved this movie! It was fantastic. http://example.com",
            "Terrible movie... would not recommend! #fail",
            "I loved this movie! It was fantastic.",  # Duplicate
            "Average movie, some good parts, some bad.",
```

ROBLEMS OUTPUT DEBUG CONSOLE PORTS AZURE QUERY RESULTS POSTGRESQL QUERY RESULTS TERMINAL

```python
if __name__ == "__main__":
    raw_data = {
        "review_id": [101, 102, 103, 104, 105, 106],
        "review_text": [
            "I loved this movie! It was fantastic. http://example.com",
            "Terrible movie... would not recommend! #fail",
            "I loved this movie! It was fantastic.",  # Duplicate
            "Average movie, some good parts, some bad.",
            None,
            "Best movie ever! A must-watch!",
        ],
        "rating": [5, 1, 5, None, 3, 5],
    }

    raw_df = pd.DataFrame(raw_data)

    print("--- Original Movie Reviews Dataset ---")
    print(raw_df)
    print("\n" + "=" * 50 + "\n")

    # Clean dataset
    cleaned_df = clean_movie_reviews(raw_df)

    print("--- Cleaned Movie Reviews Dataset ---")
    print(cleaned_df)

```

Output:

```
C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab-17>python 17_1.py
--- Original Movie Reviews Dataset ---
   review_id                                        review_text  rating
0        101  I loved this movie! It was fantastic. http://e...     5.0
1        102       Terrible movie... would not recommend! #fail     1.0
2        103              I loved this movie! It was fantastic.     5.0
3        104          Average movie, some good parts, some bad.     NaN
4        105                                               None     3.0
5        106                     Best movie ever! A must-watch!     5.0


==================================================
```

```
--- Cleaned Movie Reviews Dataset ---
   review_id  ...                       cleaned_review
0        101  ...               loved movie fantastic
1        102  ...     terrible movie would recommend fail
2        103  ...               loved movie fantastic
3        104  ...  average movie some good parts some bad
4        105  ...
5        106  ...            best movie ever mustwatch

[6 rows x 4 columns]

C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab-17>
```

Observation:

This focuses on cleaning and preparing textual review data, such as movie or product reviews, for sentiment analysis or recommendation systems. Text preprocessing includes lowercasing, removing HTML tags, and optionally removing stopwords or special characters. The text is then encoded using methods like TF-IDF or embeddings to transform it into numeric features usable by machine learning models. Ratings are preprocessed by filling missing values with the median and normalizing them to a 0–1 scale. The task may also include generating before-and-after summaries to compare the raw and cleaned datasets, ensuring the text is ready for further analysis or modeling.