

AI LAB 17

Name: K. Jashuva

Roll No: 2503A51L16

Batch No: 19

TASK 1 - Employee Data Preprocessing

- Use AI to generate a Python script for cleaning an employee dataset

Instructions:

- Handle missing values in columns (salary, department, joining_date).
- Convert the "joining_date" column into proper datetime format.
- Standardize department names (e.g., "HR", "hr", "Human Resources" → "HR").
- Encode categorical variables (department, job_role)

Prompt:

Clean the employee dataset by filling missing values in salary, department, and joining_date, convert joining_date to datetime, standardize department names (e.g., unify "HR" variants), and encode categorical variables (department, job_role) for modeling.

Code :

```
C:\> Users > HP > Documents > TASK 1.py > ...
1 import pandas as pd
2 import numpy as np
3 from sklearn.preprocessing import LabelEncoder
4
5 # Sample employee data (replace this with your actual dataset)
6 data = [
7     {'employee_id': [1, 2, 3, 4, 5],
8      'salary': [50000, np.nan, 55000, 60000, np.nan],
9      'department': ['HR', 'human resources', 'IT', 'hr', 'Finance'],
10     'joining_date': ['2020-05-01', '2021-06-15', np.nan, '2022-07-26', '2021-08-30'],
11     'job_role': ['Manager', 'Analyst', 'Developer', 'Manager', 'Analyst']}
12 ]
13
14 # Create DataFrame
15 df = pd.DataFrame(data)
16
17 # 1. Handle missing values
18 # Fill missing salary with median salary (without inplace=True)
19 df['salary'] = df['salary'].fillna(df['salary'].median())
20
21 # Fill missing joining_date with a placeholder ('Unknown') or with a default date
22 df['joining_date'] = df['joining_date'].fillna('2020-01-01')
23
24 # 2. Convert 'joining_date' to proper datetime format
25 df['joining_date'] = pd.to_datetime(df['joining_date'], format='%Y-%m-%d')
26
27 # 3. Standardize department names
28 df['department'] = df['department'].replace({
29     'human resources': 'HR',
30     'hr': 'HR',
31     'Human Resources': 'HR'
32 })
33
34 # 4. Encode categorical variables (department, job_role)
35 label_encoder = LabelEncoder()
36
37 # Encode 'department'
38 df['department_encoded'] = label_encoder.fit_transform(df['department'])
39
40 # Encode 'job role'
41 df['job_role_encoded'] = label_encoder.fit_transform(df['job_role'])
42
43 # Final cleaned DataFrame
44 print(df)
45
```

Ln 44, Col 10 Spaces: 4 UFT-8 CRLF () Python 3.13.9 (Microsoft Store) □

Output:

```
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/HP/Documents/TASK 1.py"
employee_id  salary  department  joining_date  job role  department_encoded  job role_encoded
0            1    50000.0        HR  2020-05-01   Manager           1             2
1            2    55000.0        HR  2021-06-15   Analyst           1             0
2            3    55000.0        IT  2020-01-01  Developer           2             1
3            4    60000.0        HR  2022-07-26   Manager           1             2
4            5    55000.0    Finance  2021-08-30   Analyst           0             0
```

Observations:

The dataset contains missing values and inconsistent department names that need cleaning for accuracy. The joining_date column requires conversion to a proper datetime format, and categorical fields like department and job_role should be encoded. After preprocessing, the dataset will be clean, consistent, and ready for analysis or modeling.

Task 2 – Sales Transaction Data Preprocessing

Task: Use AI to generate a script for preprocessing a sales transaction dataset.

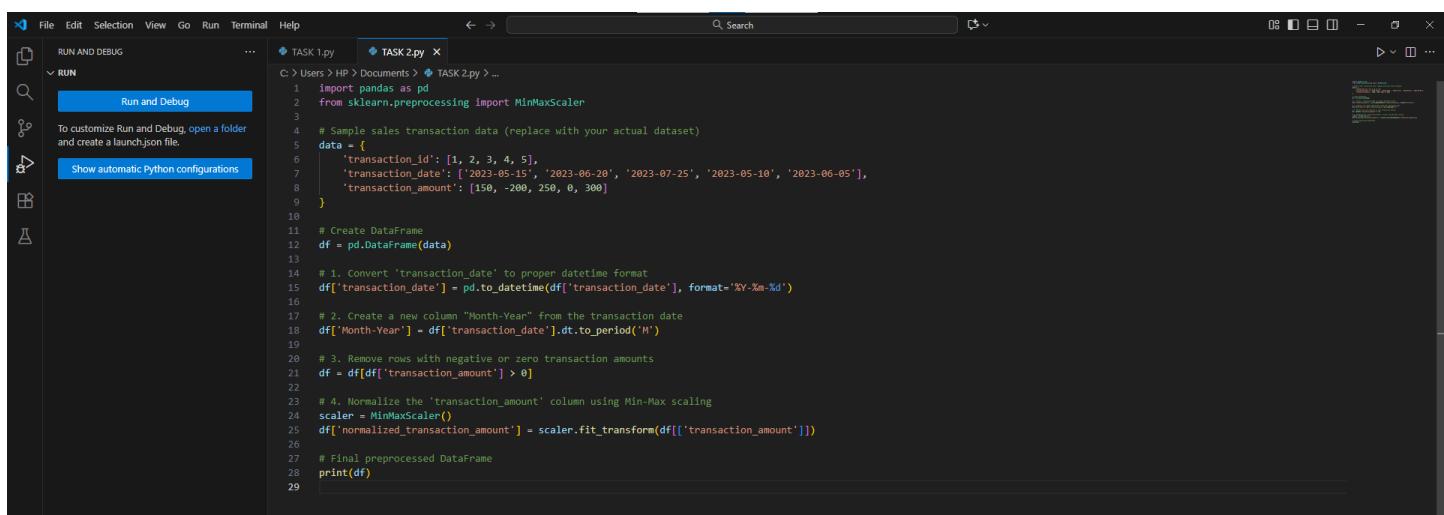
Instructions:

- Convert transaction dates to proper datetime format.
- Create a new column for “Month-Year” from the transaction date.
- Remove rows with negative or zero transaction amounts.
- Normalize the “transaction_amount” column using Min-Max scaling

Prompt :

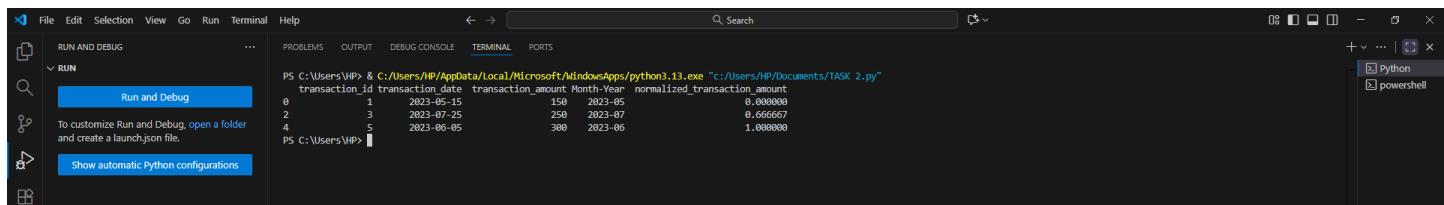
Create a Python script to preprocess a sales transaction dataset by converting transaction dates to datetime format, adding a “Month-Year” column, and removing rows with zero or negative amounts. Normalize the “transaction_amount” column using Min-Max scaling.

Code:



```
C:\> File Edit Selection View Go Run Terminal Help
RUN AND DEBUG ... TASK 1.py TASK 2.py
C:\> Users > HP > Documents > TASK 2.py ...
1 import pandas as pd
2 from sklearn.preprocessing import MinMaxScaler
3
4 # Sample sales transaction data (replace with your actual dataset)
5 data = [
6     {'transaction_id': [1, 2, 3, 4, 5],
7      'transaction_date': ['2023-05-15', '2023-06-20', '2023-07-25', '2023-05-10', '2023-06-05'],
8      'transaction_amount': [150, -200, 250, 0, 300]
9     }
10
11 # Create DataFrame
12 df = pd.DataFrame(data)
13
14 # 1. Convert 'transaction_date' to proper datetime format
15 df['transaction_date'] = pd.to_datetime(df['transaction_date'], format='YY-MM-DD')
16
17 # 2. Create a new column "Month-Year" from the transaction date
18 df['Month-Year'] = df['transaction_date'].dt.to_period('M')
19
20 # 3. Remove rows with negative or zero transaction amounts
21 df = df[df['transaction_amount'] > 0]
22
23 # 4. Normalize the 'transaction_amount' column using Min-Max scaling
24 scaler = MinMaxScaler()
25 df['normalized_transaction_amount'] = scaler.fit_transform(df[['transaction_amount']])
26
27 # Final preprocessed DataFrame
28 print(df)
29
```

Output:



```
File Edit Selection View Go Run Terminal Help
RUN AND DEBUG ... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
C:\> Users > HP > Documents > TASK 2.py ...
transaction_id transaction_date transaction_amount Month-Year normalized_transaction_amount
1 2023-05-15 150 2023-05 0.000000
2 3 2023-07-25 250 2023-07 0.666667
4 5 2023-06-05 300 2023-06 1.000000
PS C:\Users\HP>
```

Observations:

The sales transaction dataset requires cleaning and transformation to ensure data accuracy. Transaction dates need to be standardized, and a new Month-Year column will help in time-based analysis. Removing invalid transaction amounts and normalizing values will prepare the data for consistent and reliable analysis or modeling.

Task 3 – Healthcare Patient Records Cleaning

Task:

Use AI to generate a script for cleaning healthcare patient records.

Instructions:

- Fill missing values in numeric columns (e.g., blood_pressure, heart_rate) with column mean.
- Standardize units (convert height from cm to meters).
- Correct inconsistent categorical labels (e.g., "M", "Male", "male" → "Male").
- Drop irrelevant columns such as patient_id after cleaning.

Prompt:

Create a Python script to clean healthcare patient records by filling missing numeric values with the column mean, converting height from centimeters to meters, standardizing gender labels (e.g., "M", "Male", "male" → "Male"), and dropping irrelevant columns like patient_id after cleaning.

Code:

```
C:\> Users > HP > Documents > TASK 3.py > ...
1 import pandas as pd
2 import numpy as np
3
4 # Sample healthcare patient records data (replace with your actual dataset)
5 data = [
6     {'patient_id': [101, 102, 103, 104, 105],
7      'age': [45, 60, 38, np.nan, 55],
8      'blood_pressure': [120, np.nan, 130, 140, 115],
9      'heart_rate': [80, 75, np.nan, 72, 85],
10     'height_cm': [170, 180, 160, 175, 165],
11     'gender': ['M', 'Male', 'male', 'M', 'Female']}
12 ]
13
14 # Create DataFrame
15 df = pd.DataFrame(data)
16
17 # 1. Fill missing values in numeric columns with the column mean
18 df['blood_pressure'] = df['blood_pressure'].fillna(df['blood_pressure'].mean())
19 df['heart_rate'] = df['heart_rate'].fillna(df['heart_rate'].mean())
20 df['age'] = df['age'].fillna(df['age'].mean())
21
22 # 2. Standardize units - Convert height from cm to meters
23 df['height_m'] = df['height_cm'] / 100
24
25 # 3. Correct inconsistent categorical labels (e.g., 'M', 'Male', 'male' → 'Male')
26 df['gender'] = df['gender'].replace('M', 'Male', 'male': 'Male')
27
28 # 4. Drop irrelevant columns (e.g., patient_id)
29 df.drop(columns=['patient_id', 'height_cm'], inplace=True)
30
31 # Final cleaned DataFrame
32 print(df)
33
```

Output:

```
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/HP/Documents/TASK 3.py"
age blood_pressure heart_rate gender height_m
0 45.0 120.00 80.0 Male 1.70
1 60.0 126.25 75.0 Male 1.80
2 38.0 130.00 78.0 Male 1.60
3 49.5 140.00 72.0 Male 1.75
4 55.0 115.00 85.0 Female 1.65
PS C:\Users\HP>
```

Observations:

The healthcare dataset contains missing values and inconsistent data formats that need correction for accuracy. Numeric columns require imputation with mean values, and units like height must be standardized for consistency. Cleaning categorical labels and removing irrelevant columns will result in a reliable and well-structured dataset for analysis.

Task 4 – Social Media Sentiment Dataset Preparation

Task:

Use AI to write a script to preprocess a social media text dataset.

Instructions:

- Remove special characters, URLs, and emojis from text.
- Convert all text to lowercase.
- Tokenize and remove stopwords.
- Apply lemmatization for standardizing words

Prompt:

Create a Python script to preprocess a social media sentiment dataset by removing special characters, URLs, and emojis from text. Convert all text to lowercase, tokenize it, remove stopwords, and apply lemmatization to standardize words for accurate sentiment analysis.

Code:

The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The left sidebar has sections for EXPLORER, OUTLINE, and TIMELINE, with an Open Folder button. The main area displays a Python script named TASK 4.py. The script imports pandas, re, and spacy, loads the English model, and defines a sample dataset of 4 tweets. It then defines functions to clean text (removing URLs and emojis, keeping letters and spaces), preprocess text (using spaCy to process, doc, tokens, and token.lemma_), and finally tokenizes, removes stopwords, and lemmatizes the cleaned text. The script prints the resulting DataFrame with columns 'id' and 'processed_text'. The status bar at the bottom shows Line 44, Column 1, Spaces: 4, UTF-8, CR/LF, Python, and 3.13.9 (Microsoft Store).

```
File Edit Selection View Go Run Terminal Help Search
EXPLORER NO FOLDER OPENED
You have not yet opened a folder. Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
TASK 1.py TASK 2.py TASK 3.py TASK 4.py
C:\Users\HP\Documents> python TASK 4.py ...
1 import pandas as pd
2 import re
3 import spacy
4
5 # Load Spacy English model (run once in terminal: python -m spacy download en_core_web_sm)
6 nlp = spacy.load("en_core_web_sm")
7
8 # Sample social media text dataset
9 data = [
10     {"id": [1, 2, 3, 4],
11      "text": [
12          "I love this product! 😊 Check it out at http://example.com",
13          "Worst purchase ever 😡!!! #angry #frustrated http://mywebsite.com",
14          "Highly Recommend! Great quality and fast shipping. ▲😊",
15          "Not worth it! Wouldn't buy again! 😢 http://website.com",
16      ],
17  }
18
19 df = pd.DataFrame(data)
20
21 def clean_text(text):
22     # Remove URLs
23     text = re.sub(r"http\S+|www\S+", "", text)
24     # Remove emojis and special characters (keep letters and spaces)
25     text = re.sub(r"[^\u2000-\u20ff]", "", text)
26     # Convert to lowercase and strip whitespace
27     text = text.lower().strip()
28     return text
29
30 def preprocess_text(text):
31     # Process text with spaCy
32     doc = nlp(text)
33     # Lemmatize, remove stopwords and non-alphabetic tokens
34     tokens = [token.lemma_ for token in doc if not token.is_stop and token.is_alpha]
35     return tokens
36
37 # Clean the text column
38 df["cleaned_text"] = df["text"].apply(clean_text)
39
40 # Tokenize, remove stopwords, and lemmatize
41 df["processed_text"] = df["cleaned_text"].apply(preprocess_text)
42
43 print(df[["id", "processed_text"]])
44
```

Output:

The screenshot shows a terminal window with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The left sidebar has sections for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS, with an Open Folder button. The main area shows the command PS C:\Users\HP> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/HP/Documents/TASK 4.py" followed by the output of the script. The output shows the processed text for each tweet, where emojis and URLs are removed, and the text is converted to lowercase and tokenized. The status bar at the bottom shows Line 44, Column 1, Spaces: 4, UTF-8, CR/LF, Python, and 3.13.9 (Microsoft Store).

```
File Edit Selection View Go Run Terminal Help Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
EXPLORER NO FOLDER OPENED
You have not yet opened a folder. Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/HP/Documents/TASK 4.py"
0 1 [love, product, check]
1 2 [bad, purchase, angry, frustrate]
2 3 [highly, recommend, great, quality, fast, ship...]
3 4 [worth, not, buy]
PS C:\Users\HP>
```

Observations:

The social media text data contains noise such as emojis, URLs, and special characters that can affect sentiment accuracy. Standardizing text through lowercasing, tokenization, and lemmatization ensures consistency. After preprocessing, the dataset will be cleaner and more suitable for effective sentiment analysis and model training.

Task 5 – Financial Dataset Feature Engineering

Task:

Use AI to create a preprocessing script for a financial dataset.

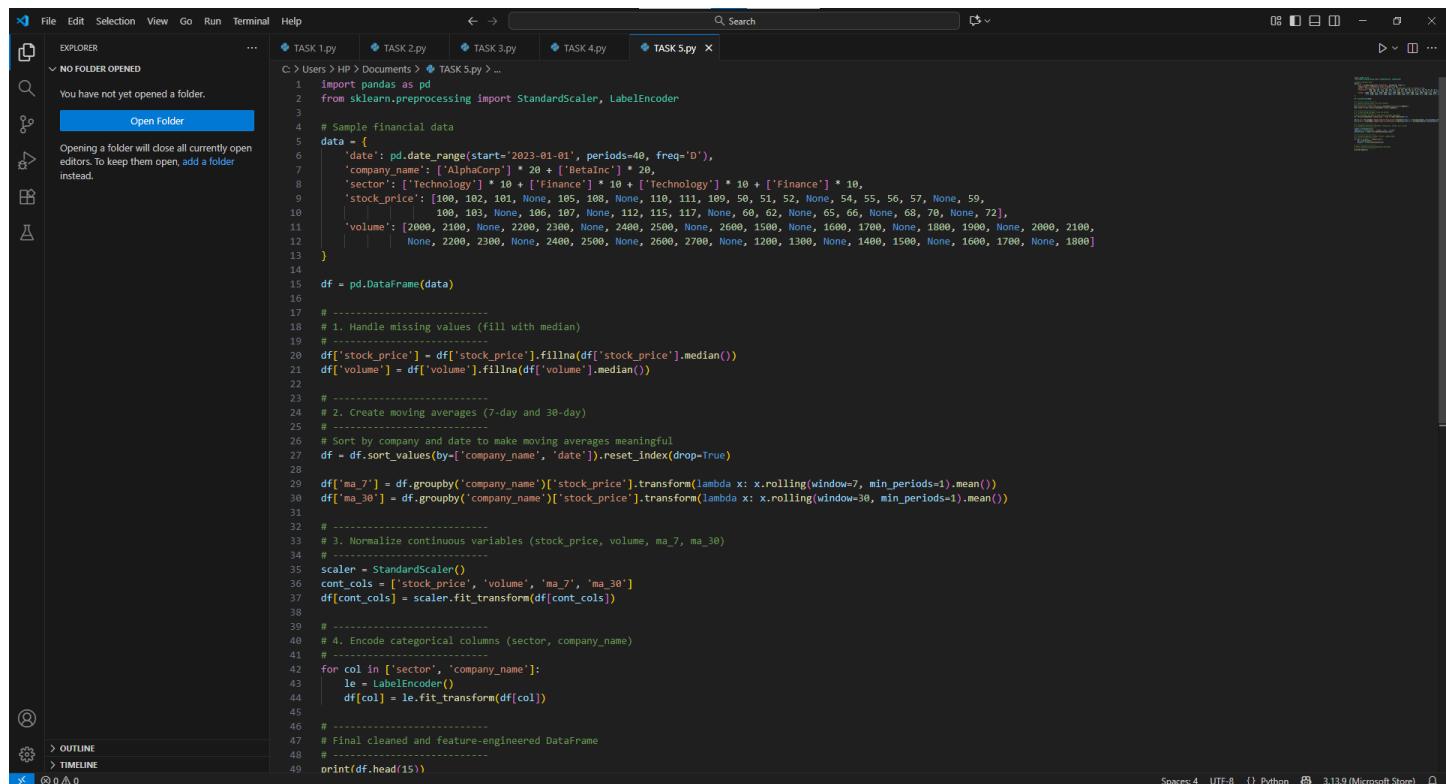
Instructions:

- Handle missing values in stock price and volume.
- Create new features such as moving average (7-day, 30-day).
- Normalize continuous variables using StandardScaler.
- Encode categorical columns (sector, company_name).

Prompt:

Create a Python script to preprocess a financial dataset by handling missing values in stock price and volume, generating new features like 7-day and 30-day moving averages, normalizing continuous variables using StandardScaler, and encoding categorical columns such as sector and company_name.

Code:



The screenshot shows a Microsoft Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows a folder structure with 'NO FOLDER OPENED' and an 'Open Folder' button.
- Code Editor:** Displays a Python script named 'TASK 5.py'. The code is as follows:

```
File Edit Selection View Go Run Terminal Help < - > Search C > Users > HP > Documents > TASK 5.py ... EXPLORER NO FOLDER OPENED Open Folder You have not yet opened a folder. Opening a folder will close all currently open editors. To keep them open, add a folder instead. TASK 1.py TASK 2.py TASK 3.py TASK 4.py TASK 5.py Search ... C > Users > HP > Documents > TASK 5.py ... 1 import pandas as pd 2 from sklearn.preprocessing import StandardScaler, LabelEncoder 3 4 # Sample financial data 5 data = { 6     'date': pd.date_range(start='2023-01-01', periods=40, freq='D'), 7     'company_name': ['AlphaCorp'] * 20 + ['BetaInc'] * 20, 8     'sector': ['Technology'] * 10 + ['Finance'] * 10 + ['Technology'] * 10 + ['Finance'] * 10, 9     'stock_price': [100, 102, 101, None, 105, 108, None, 110, 111, 109, 50, 51, 52, None, 54, 55, 56, 57, None, 59, 100, 103, None, 106, 107, None, 112, 115, 117, None, 60, 62, None, 65, 66, None, 68, 70, None, 72], 10    'volume': [2000, 2100, None, 2200, 2300, None, 2400, 2500, None, 2600, 1500, None, 1600, 1700, None, 1800, 1900, None, 2000, 2100, None, 2200, 2300, None, 2400, 2500, None, 2600, 2700, None, 1200, 1300, None, 1400, 1500, None, 1600, 1700, None, 1800] 11 } 12 13 df = pd.DataFrame(data) 14 15 # ----- 16 # 1. Handle missing values (fill with median) 17 # ----- 18 df['stock_price'] = df['stock_price'].fillna(df['stock_price'].median()) 19 df['volume'] = df['volume'].fillna(df['volume'].median()) 20 21 # ----- 22 # 2. Create moving averages (7-day and 30-day) 23 # ----- 24 df = df.sort_values(by=['company_name', 'date']).reset_index(drop=True) 25 26 df['ma_7'] = df.groupby('company_name')['stock_price'].transform(lambda x: x.rolling(window=7, min_periods=1).mean()) 27 df['ma_30'] = df.groupby('company_name')['stock_price'].transform(lambda x: x.rolling(window=30, min_periods=1).mean()) 28 29 # ----- 30 # 3. Normalize continuous variables (stock_price, volume, ma_7, ma_30) 31 # ----- 32 scaler = StandardScaler() 33 cont_cols = ['stock_price', 'volume', 'ma_7', 'ma_30'] 34 df[cont_cols] = scaler.fit_transform(df[cont_cols]) 35 36 # ----- 37 # 4. Encode categorical columns (sector, company_name) 38 # ----- 39 for col in ['sector', 'company_name']: 40     le = LabelEncoder() 41     df[col] = le.fit_transform(df[col]) 42 43 # ----- 44 # Final cleaned and feature-engineered DataFrame 45 # ----- 46 df.head(15)
```

Bottom status bar: Spaces:4 UTF-8 Python 3.13.9 (Microsoft Store)

Output:

```
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/HP/Documents/TASK_5.py"
   date      stock_id company_name    sector  stock_price  volume          timestamp
0 2023-01-01        0       ABBV       Tech     0.752621  0.007158  0.791154  0.825911
1 2023-01-02        0       ABBV       Tech     0.847142  0.293164  0.863633  0.979835
2 2023-01-03        0       ABBV       Tech     0.799882  0.007158  0.863633  0.979835
3 2023-01-04        0       ABBV       Tech     0.098976  0.579178  0.693887  0.402619
4 2023-01-05        0       ABBV       Tech     0.988923  0.865191  0.718779  0.641281
5 2023-01-06        0       ABBV       Tech     1.138704  0.007158  0.817314  0.877219
6 2023-01-07        0       ABBV       Tech     0.098976  1.151285  0.675847  0.562648
7 2023-01-08        0       ABBV       Tech     1.225225  1.437211  0.774303  0.787429
8 2023-01-09        0       ABBV       Tech     1.272426  0.007158  0.863633  0.979835
9 2023-01-10        0       ABBV       Tech     1.177965  0.723232  0.943838  1.102774
10 2023-01-11        0       ABBV       Tech     1.042082  1.151285  0.675847  0.562648
11 2023-01-12        0       ABBV       Tech     -1.563137  0.007158  0.449735  -0.213079
12 2023-01-13        0       ABBV       Tech     -1.515876  -1.126904  0.506997  -0.701492
13 2023-01-14        0       ABBV       Tech     0.098976  -0.858898  0.506997  -0.746316
14 2023-01-15        0       ABBV       Tech     -1.421355  0.007158  -1.061938  -1.113536
```

PS C:\Users\HP>

Observations:

The financial dataset requires handling missing values to maintain data reliability. Creating moving average features will help capture market trends, while normalization ensures balanced scaling for modeling. Encoding categorical variables makes the dataset suitable for machine learning and predictive analytics.