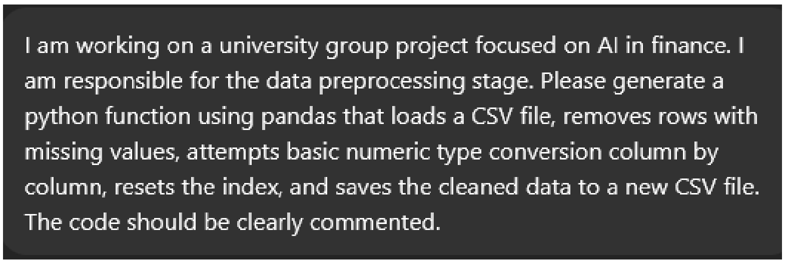
COMP2003 AI Use

Vibe Coding Evidence

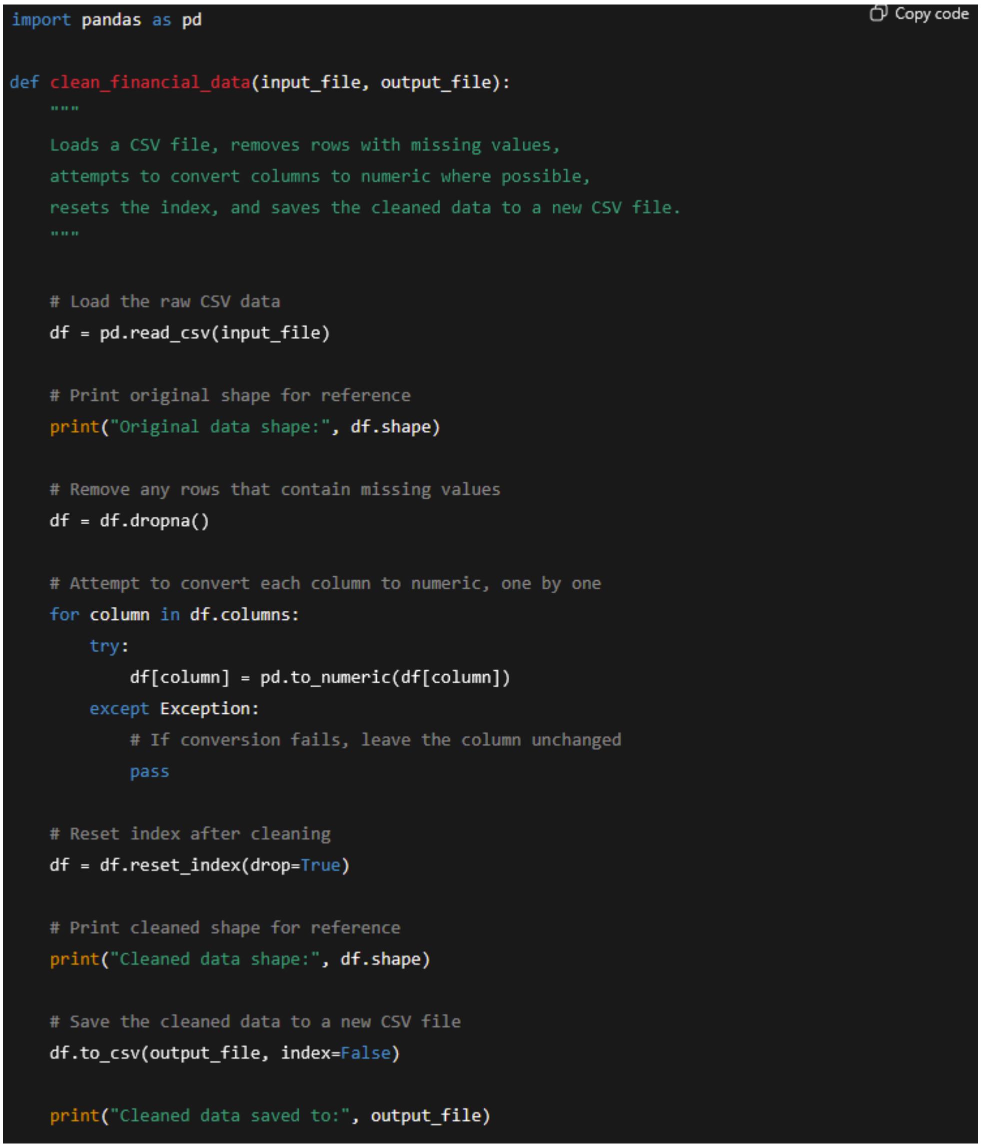
By Daniel Dawdry

Dataset Cleaning

Prompt 1:



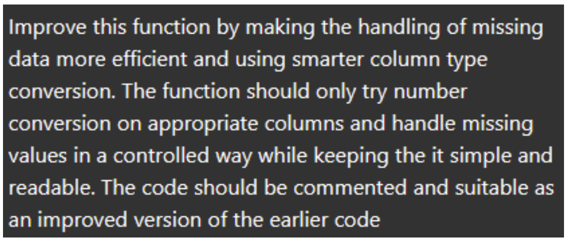
I am working on a university group project focused on AI in finance. I am responsible for the data preprocessing stage. Please generate a python function using pandas that loads a CSV file, removes rows with missing values, attempts basic numeric type conversion column by column, resets the index, and saves the cleaned data to a new CSV file. The code should be clearly commented.



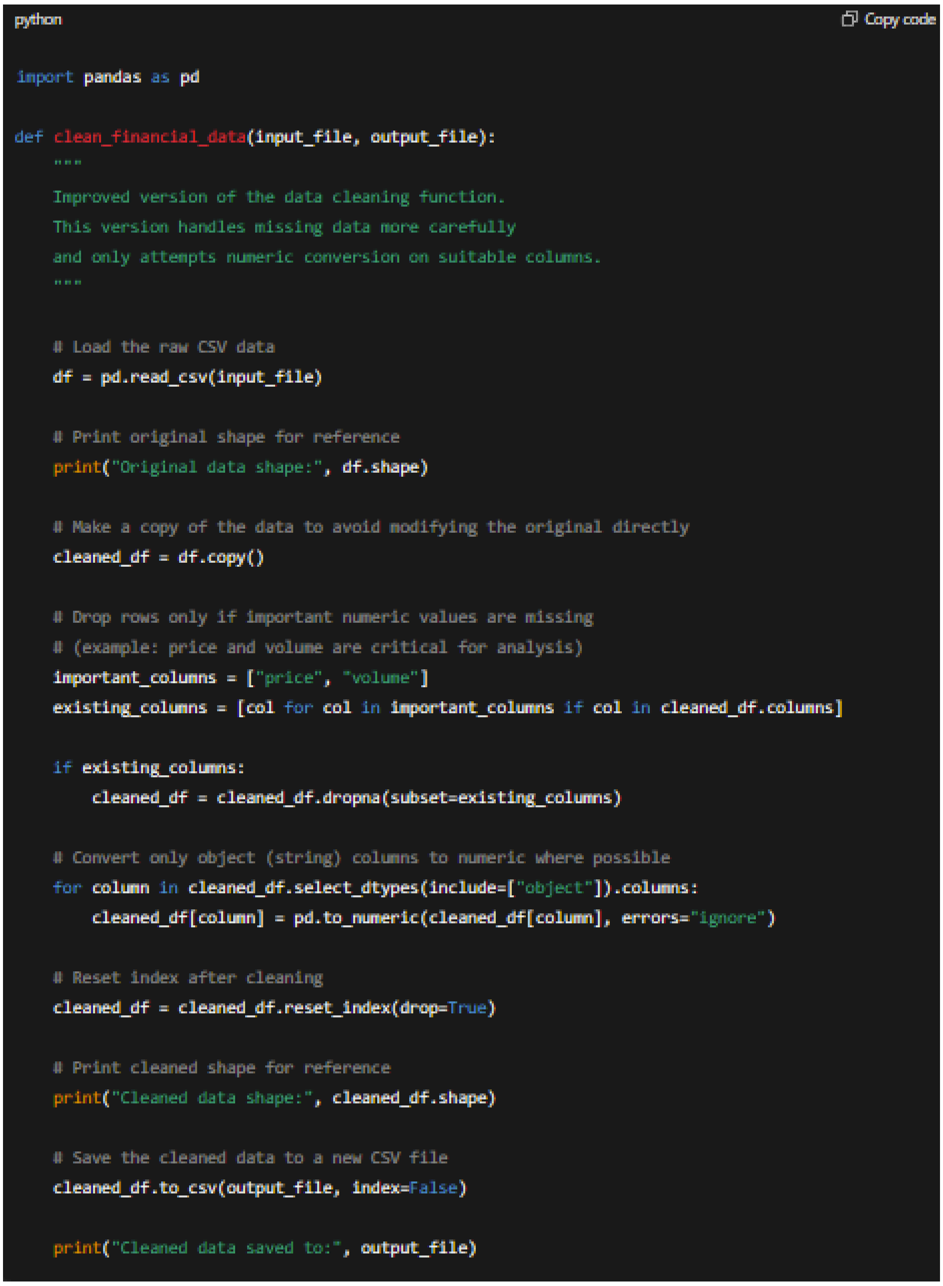
```python
import pandas as pd

def clean_financial_data(input_file, output_file):
    """
    Loads a CSV file, removes rows with missing values,
    attempts to convert columns to numeric where possible,
    resets the index, and saves the cleaned data to a new CSV file.
    """

    # Load the raw CSV data
    df = pd.read_csv(input_file)

    # Print original shape for reference
    print("Original data shape:", df.shape)

    # Remove any rows that contain missing values
    df = df.dropna()

    # Attempt to convert each column to numeric, one by one
    for column in df.columns:
        try:
            df[column] = pd.to_numeric(df[column])
        except Exception:
            # If conversion fails, leave the column unchanged
            pass

    # Reset index after cleaning
    df = df.reset_index(drop=True)

    # Print cleaned shape for reference
    print("Cleaned data shape:", df.shape)

    # Save the cleaned data to a new CSV file
    df.to_csv(output_file, index=False)

    print("Cleaned data saved to:", output_file)
```

What I’ve noticed is that this drops any row with a missing value which can remove lots of data. Another issue was it tries to convert every column which is slow. Other than these it seems to work. So only one more prompt is needed to sort this. Asking it to only drop rows where key columns are missing, and checking the column data type first and only changing it if it needs changing.

Prompt 2:



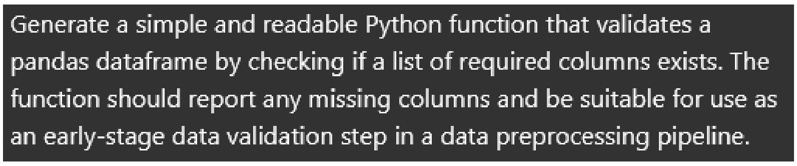
Improve this function by making the handling of missing data more efficient and using smarter column type conversion. The function should only try number conversion on appropriate columns and handle missing values in a controlled way while keeping the it simple and readable. The code should be commented and suitable as an improved version of the earlier code



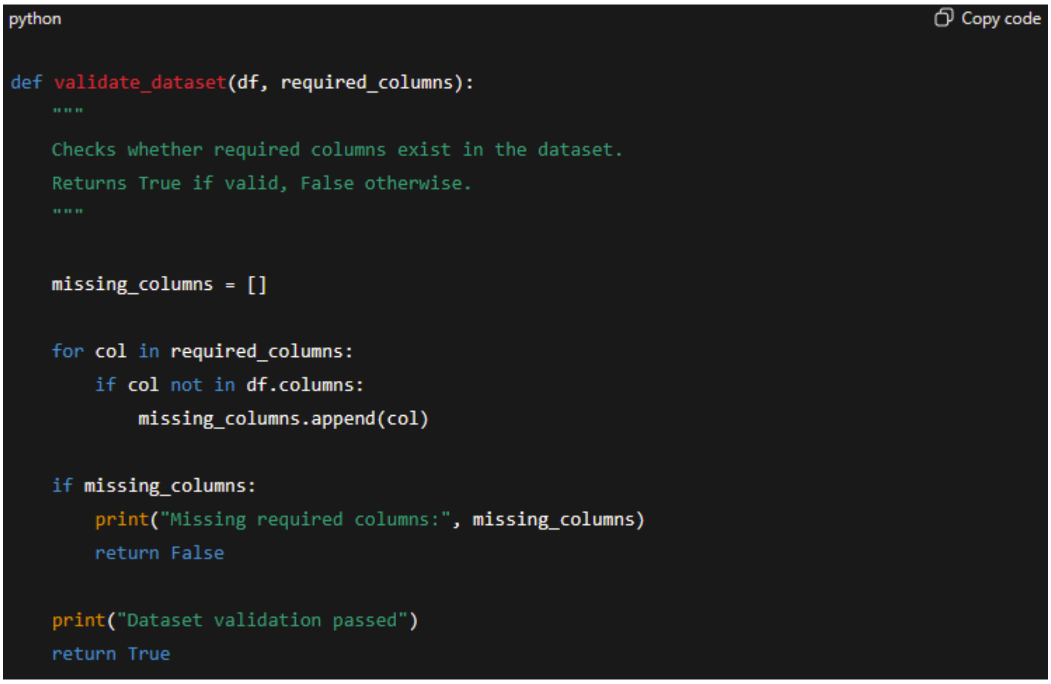
```python
import pandas as pd

def clean_financial_data(input_file, output_file):
    """
    Improved version of the data cleaning function.
    This version handles missing data more carefully
    and only attempts numeric conversion on suitable columns.
    """

    # Load the raw CSV data
    df = pd.read_csv(input_file)

    # Print original shape for reference
    print("Original data shape:", df.shape)

    # Make a copy of the data to avoid modifying the original directly
    cleaned_df = df.copy()

    # Drop rows only if important numeric values are missing
    # (example: price and volume are critical for analysis)
    important_columns = ["price", "volume"]
    existing_columns = [col for col in important_columns if col in cleaned_df.columns]

    if existing_columns:
        cleaned_df = cleaned_df.dropna(subset=existing_columns)

    # Convert only object (string) columns to numeric where possible
    for column in cleaned_df.select_dtypes(include=["object"]).columns:
        cleaned_df[column] = pd.to_numeric(cleaned_df[column], errors="ignore")

    # Reset index after cleaning
    cleaned_df = cleaned_df.reset_index(drop=True)

    # Print cleaned shape for reference
    print("Cleaned data shape:", cleaned_df.shape)

    # Save the cleaned data to a new CSV file
    cleaned_df.to_csv(output_file, index=False)

    print("Cleaned data saved to:", output_file)
```

Data Validation

Prompt 1:



Generate a simple and readable Python function that validates a pandas dataframe by checking if a list of required columns exists. The function should report any missing columns and be suitable for use as an early-stage data validation step in a data preprocessing pipeline.



```python
def validate_dataset(df, required_columns):
    """
    Checks whether required columns exist in the dataset.
    Returns True if valid, False otherwise.
    """

    missing_columns = []

    for col in required_columns:
        if col not in df.columns:
            missing_columns.append(col)

    if missing_columns:
        print("Missing required columns:", missing_columns)
        return False

    print("Dataset validation passed")
    return True
```

Checks if needed columns exist before the pipeline runs. This will help prevent crashes later.