import pandas as pd

import numpy as np

from scipy import stats

def load\_and\_preprocess\_data(file\_path):

    """

    Load and preprocess the transaction data.

    Parameters:

    file\_path (str): Path to the CSV file containing the transaction data.

    Returns:

    pd.DataFrame: Preprocessed data.

    """

    # Load data

    df = pd.read\_csv(file\_path)

    # Convert date to datetime

    df['date'] = pd.to\_datetime(df['date'])

    # Handle missing values by dropping rows with NaNs

    df = df.dropna()

    return df

def calculate\_statistics(df):

    """

    Calculate basic statistical metrics for transaction amounts by category.

    Parameters:

    df (pd.DataFrame): Preprocessed transaction data.

    Returns:

    pd.DataFrame: DataFrame containing mean, median, and standard deviation for each category.

    """

    # Calculate statistics by category

    stats\_by\_category = df.groupby('category')['amount'].agg(['mean', 'median', 'std'])

    return stats\_by\_category

def detect\_anomalies(df, stats\_by\_category, threshold=1):

    """

    Detect anomalies in transaction data based on Z-score.

    Parameters:

    df (pd.DataFrame): Preprocessed transaction data.

    stats\_by\_category (pd.DataFrame): Statistical metrics for each category.

    threshold (float): The number of standard deviations to use as the cutoff for detecting anomalies.

    Returns:

    list: List of dictionaries containing anomaly details.

    """

    anomalies = []

    for \_, row in df.iterrows():

        category\_stats = stats\_by\_category.loc[row['category']]

        # Z-score method for amount anomalies

        z\_score = (row['amount'] - category\_stats['mean']) / category\_stats['std']

        if abs(z\_score) > threshold:  # Using the specified threshold

            anomalies.append({

                'transaction\_id': row['transaction\_id'],

                'date': row['date'],

                'category': row['category'],

                'amount': row['amount'],

                'reason\_for\_anomaly': 'Unusual amount (Z-score: {:.2f})'.format(z\_score)

            })

    return anomalies

def generate\_report(anomalies):

    """

    Generate a report of detected anomalies and print it.

    Parameters:

    anomalies (list): List of dictionaries containing anomaly details.

    """

    print("Anomaly Report:")

    print("=" \* 80)

    for anomaly in anomalies:

        print(f"Transaction ID: {anomaly['transaction\_id']}")

        print(f"Date: {anomaly['date']}")

        print(f"Category: {anomaly['category']}")

        print(f"Amount: ${anomaly['amount']:.2f}")

        print(f"Reason: {anomaly['reason\_for\_anomaly']}")

        print("-" \* 80)

    print(f"\nTotal anomalies detected: {len(anomalies)}")

def main():

    """

    Main function to execute the anomaly detection process.

    """

    # Assume the data is in a CSV file named 'sam.csv'

    file\_path = '/content/drive/MyDrive/sam.csv'

    # Load and preprocess data

    df = load\_and\_preprocess\_data(file\_path)

    # Calculate statistics

    stats\_by\_category = calculate\_statistics(df)

    # Detect anomalies

    anomalies = detect\_anomalies(df, stats\_by\_category)

    # Generate report

    generate\_report(anomalies)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Methodology**

1. **Data Preprocessing:**
   * **Loading Data:** The dataset is loaded from a CSV file using pandas.read\_csv.
   * **Date Conversion:** The date column is converted to a datetime format to ensure consistency in date handling.
   * **Handling Missing Values:** Any rows with missing values are dropped to ensure clean data for analysis.
2. **Statistical Analysis:**
   * **Grouping by Category:** Transactions are grouped by their category.
   * **Calculating Metrics:** For each category, the mean, median, and standard deviation of the transaction amounts are calculated. These metrics help understand the distribution of transaction amounts within each category.
3. **Anomaly Detection:**
   * **Z-Score Calculation:** The Z-score for each transaction amount is calculated to determine how many standard deviations it is away from the mean of its category.
   * **Threshold for Anomalies:** A threshold of 1 standard deviation is used to identify anomalies. Transactions with a Z-score exceeding this threshold are considered anomalies. This threshold can be adjusted based on the desired sensitivity.
4. **Reporting:**
   * **Generating Report:** An anomaly report is generated, listing each detected anomaly with details such as transaction ID, date, category, amount, and the reason for flagging it as an anomaly.
   * **Printing Report:** The report is printed to the console for review.

**Assumptions**

* The transaction data is assumed to be clean and free from significant errors after handling missing values.
* The distribution of transaction amounts within each category is assumed to be roughly normal, making the Z-score method effective.
* The chosen threshold of 1 standard deviation is assumed to be appropriate for identifying anomalies in this dataset. This can be adjusted based on the specific use case and the desired sensitivity of the anomaly detection.

**Rationale Behind Statistical Methods and Thresholds**

* **Z-Score Method:** The Z-score is a widely used statistical measure that quantifies how far a data point is from the mean, expressed in terms of standard deviations. It is effective for identifying outliers in data that follows a normal distribution.
* **Threshold Selection:** A threshold of 1 standard deviation is used to flag anomalies. This is a relatively sensitive threshold, meaning that even moderate deviations from the mean are considered anomalies. Depending on the application, this threshold can be increased to reduce false positives or decreased to capture more subtle anomalies.

This documentation provides a comprehensive overview of the methodology, assumptions, and rationale for the anomaly detection process implemented in the code.