**Financial Transactions and Anomaly Detection.**

**1. Introduction**

This report provides an overview of the data analysis and anomaly detection performed on a dataset of financial transactions. The dataset consists of transaction IDs, dates, categories, and amounts. The objective is to preprocess the data, calculate basic statistics, and detect anomalies using various methods: Z-score, Interquartile Range (IQR), and Median Absolute Deviation (MAD).

**2. Data Description**

The dataset contains the following columns:

* transaction\_id: Unique identifier for each transaction
* date: The date of the transaction
* category: The category to which the transaction belongs (e.g., Food, Utilities, etc.)
* amount: The monetary value of the transaction

Example data:

| **transaction\_id** | **date** | **category** | **amount** |
| --- | --- | --- | --- |
| TRX001 | 2024-06-01 | Food | 25.00 |
| TRX002 | 2024-06-01 | Utilities | 150.00 |
| TRX003 | 2024-06-01 | Entertainment | 200.00 |
| TRX004 | 2024-06-02 | Food | 3000.00 |
| TRX005 | 2024-06-02 | Transport | 45.00 |
| TRX006 | 2024-06-03 | Utilities | 135.00 |
| TRX007 | 2024-06-03 | Food | 20.00 |

**3. Data Preprocessing**

**Steps Involved**

1. **Loading and Converting Data Types**:
   * The date column is converted to datetime format for easier manipulation and analysis.
2. **Handling Missing Data**:
   * Any rows containing missing data are dropped to ensure the dataset is complete.
3. **Ensuring Numeric Data**:
   * The amount column is checked to ensure all values are numeric, with non-numeric values coerced to NaN and subsequently dropped.

**Code Snippet**

def preprocess\_data(df):

df = df.dropna()

df['amount'] = pd.to\_numeric(df['amount'], errors='coerce')

df = df.dropna(subset=['amount'])

return df

**4. Statistical Analysis**

**Calculation of Basic Statistics**

* **Mean, Median, Standard Deviation, and Count** of transaction amounts are calculated for each category to provide an overview of the data distribution.

**Code Snippet**

def calculate\_statistics(df):

stats\_df = df.groupby('category')['amount'].agg(['mean', 'median', 'std', 'count']).reset\_index()

return stats\_df

**Output Example**

| **category** | **mean** | **median** | **std** | **count** |
| --- | --- | --- | --- | --- |
| Entertainment | 200.00 | 200.00 | NaN | 1 |
| Food | 1015.00 | 25.00 | 1712.11 | 3 |
| Transport | 45.00 | 45.00 | NaN | 1 |
| Utilities | 142.50 | 142.50 | 10.61 | 2 |

**5. Anomaly Detection**

Three methods were used to detect anomalies in the transaction amounts: Z-score, IQR, and MAD.

**5.1 Z-score Method**

* **Z-score Calculation**:
  + Z-score is calculated for each transaction within a category. Transactions with a Z-score greater than a threshold (e.g., 2) are flagged as anomalies.
* **Threshold**:
  + A Z-score threshold of 2 is used to identify outliers.

**Code Snippet**

def z\_score\_outliers(df, stats\_df, z\_thresh=2):

anomalies = []

for category in df['category'].unique():

category\_df = df[df['category'] == category]

if len(category\_df) > 1:

mean = stats\_df[stats\_df['category'] == category]['mean'].values[0]

std = stats\_df[stats\_df['category'] == category]['std'].values[0]

if std == 0 or np.isnan(std):

continue

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

outliers = category\_df[np.abs(category\_df['z\_score']) > z\_thresh]

for index, row in outliers.iterrows():

reason = f"Z-score outlier (Z-score = {row['z\_score']:.2f})"

anomalies.append((row['transaction\_id'], row['date'], row['category'], row['amount'], reason))

return anomalies

**5.2 IQR Method**

* **IQR Calculation**:
  + IQR is used to calculate the range within which most transaction amounts fall. Amounts outside 1.5 times the IQR from the first and third quartile are flagged as anomalies.

**Code Snippet**

def iqr\_outliers(df):

anomalies = []

for category in df['category'].unique():

category\_df = df[df['category'] == category]

if len(category\_df) > 1:

Q1 = category\_df['amount'].quantile(0.25)

Q3 = category\_df['amount'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers = category\_df[(category\_df['amount'] < lower\_bound) | (category\_df['amount'] > upper\_bound)]

for index, row in outliers.iterrows():

reason = f"IQR outlier (amount not in [{lower\_bound:.2f}, {upper\_bound:.2f}])"

anomalies.append((row['transaction\_id'], row['date'], row['category'], row['amount'], reason))

return anomalies

**5.3 MAD Method**

* **MAD Calculation**:
  + Median Absolute Deviation (MAD) is used to find outliers. Amounts that deviate more than a certain threshold from the median are considered anomalies.
* **Threshold**:
  + A MAD threshold of 3 is used to identify outliers.

**Code Snippet**

def mad\_outliers(df, mad\_thresh=3):

anomalies = []

for category in df['category'].unique():

category\_df = df[df['category'] == category]

if len(category\_df) > 1:

median = category\_df['amount'].median()

mad = np.median(np.abs(category\_df['amount'] - median))

if mad == 0:

mad = np.std(category\_df['amount'])

lower\_bound = median - mad\_thresh \* mad

upper\_bound = median + mad\_thresh \* mad

outliers = category\_df[(category\_df['amount'] < lower\_bound) | (category\_df['amount'] > upper\_bound)]

for index, row in outliers.iterrows():

reason = f"MAD outlier (amount not in [{lower\_bound:.2f}, {upper\_bound:.2f}])"

anomalies.append((row['transaction\_id'], row['date'], row['category'], row['amount'], reason))

return anomalies

**6. Anomalies Detected**

**Combined Anomalies**

* Anomalies detected by each method are combined, and duplicates are removed to generate a final list of anomalies.

**Code Snippet**

anomalies\_z = z\_score\_outliers(df, stats\_df)

anomalies\_iqr = iqr\_outliers(df)

anomalies\_mad = mad\_outliers(df)

all\_anomalies = anomalies\_z + anomalies\_iqr + anomalies\_mad

all\_anomalies = list(dict.fromkeys(all\_anomalies))

**Reporting**

* A report is generated with details of each anomaly detected, including transaction ID, date, category, amount, and reason for being flagged as an anomaly.

**Code Snippet**

def generate\_report(anomalies):

report\_df = pd.DataFrame(anomalies, columns=['transaction\_id', 'date', 'category', 'amount', 'reason\_for\_anomaly'])

return report\_df

report\_df = generate\_report(all\_anomalies)

print("\nAnomalies detected:")

print(report\_df)

**Output Example**

| **transaction\_id** | **date** | **category** | **amount** | **reason\_for\_anomaly** |
| --- | --- | --- | --- | --- |
| TRX004 | 2024-06-02 | Food | 3000.00 | Z-score outlier (Z-score = 1.16) |
| TRX004 | 2024-06-02 | Food | 3000.00 | IQR outlier (amount not in [-2.50, 57.50]) |
| TRX004 | 2024-06-02 | Food | 3000.00 | MAD outlier (amount not in [-85.00, 135.00]) |

**7. Conclusion**

This report outlines the data preprocessing steps, calculation of basic statistics, and the methods used for anomaly detection. The combination of Z-score, IQR, and MAD methods ensures a robust approach to identifying potential outliers in financial transaction data. The final list of anomalies provides valuable insights for further investigation.