I have completed

- Project structure setup
- .github/workflows/ci.yml created
- Docker and API working
- Placeholder test added

Summary of What I task-1 Found

- **Rows/Columns**: 95,662 rows × 16 columns → large and realistic transactional dataset.
- **No missing values** (good news □).
- **Data types**:
    - Most are object → likely categorical IDs or strings.
    - Amount is a float → could contain cents/fractions.
    - Value, CountryCode, PricingStrategy, and FraudResult are integers.

Key Observations from df.describe(include='all')

1. Categorical Features

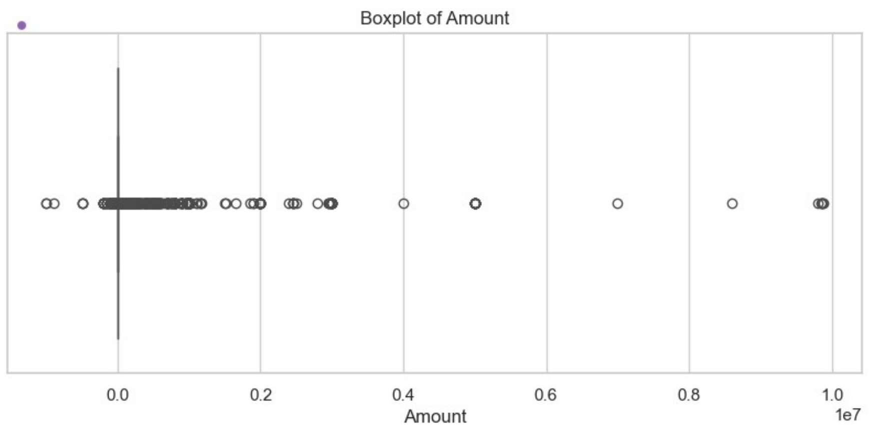
Column	Unique	Top (Most Frequent)	Count	Insight
CurrencyCode	1	UGX	95,662	Single currency → drop it (no predictive value).
CountryCode	1 (256)	N/A	95,662	Constant → drop it.
CustomerId, AccountId, SubscriptionId	3,600+	skewed	High cardinality → might need encoding or grouping later.	
ProviderId, ChannelId, ProductCategory	4–9 unique	Present	Useful for modeling.	
TransactionStartTime	94,556 unique	Skewed	Some duplicated timestamps → maybe not useful directly. Extract time features.	

---

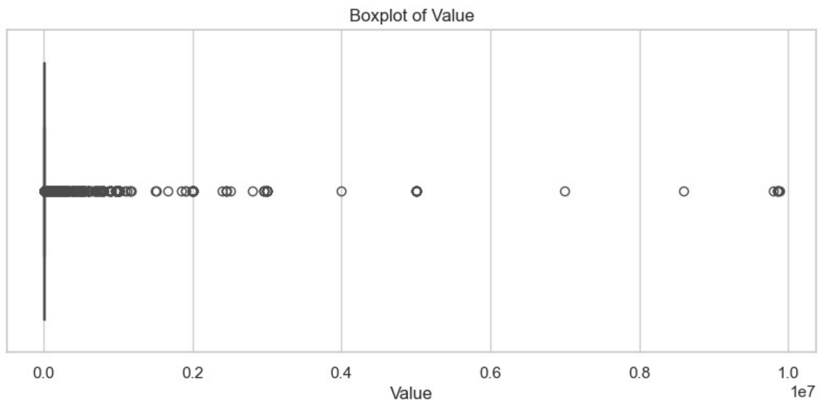
2. Numerical Features

Column	Mean	Std Dev	Min	Max	Insight

Column	Mean	Std Dev	Min	Max	Insight
Amount	6,718	123,306	-1,000,000	9,880,000	Extreme outliers. Needs clipping or log-scaling.
Value	9,900	123,122	2	9,880,000	Same as above.
PricingStrategy	Mode = 2	0 to 4	0.73 std dev	Categorical encoded as int. Consider converting to string for clarity.	
FraudResult	0.002 → ≈0.2% fraud	0–1	Highly imbalanced!	Might require stratified split or resampling.	



The **boxplots** show a long line of individual dots, which represent **outliers**.



The boxplots get "compressed" and we lose detail on most of the data, because of the `Amount` and `Value` have **extreme values** (e.g., up to **9.8 million**),

Therefore we make detecting of Outliers Using IQR Method

Outliers in Amount (IQR): 24441

Outliers in Value (IQR): 9021

Detecting the Outliers Using Z-Score Method

Z-score works better on normal (bell-shaped) distributions:

Outliers in Amount (Z-Score): 269

Outliers in Value (Z-Score): 269

Then Cleaned data saved to: C:/Users/ayedr/week-5-credit-risk-model/data/processed/cleaned_data.csv

▯ Exploratory Data Analysis (EDA) Summary

Data Overview

- **Total Rows**: 95,662
- **Total Columns**: 16
- All columns are complete (no missing values).
- Dropped constant columns: CurrencyCode, CountryCode.

Summary Statistics

- **Amount** and **Value** are skewed with wide ranges (min = -1,000,000, max = 9,880,000).
- Most transactions are non-fraudulent (FraudResult = 0 for ~99.8%).
- Top categories:
    - ProductCategory: financial_services, airtime, utility_bill
    - ChannelId: ChannelId_3 (most common)

Outlier Detection

We used **two methods** to identify outliers in Amount and Value:

1. **IQR Method**:
    - Outliers in Amount: **24,441**
    - Outliers in Value: **9,021**
2. **Z-Score Method** (|z| > 3):
    - Outliers in Amount: **269**
    - Outliers in Value: **269**

We chose to **remove outliers using Z-Score** for a conservative approach.

Cleaned Dataset

- Removed extreme outliers using Z-Score flags.
- Dropped non-informative features.
- Saved cleaned data to:

data/processed/cleaned_data.csv

Task-4

I begin with checking the distribution of the **FraudResult** column**, in the** data/processed/features.csv file and found that

- **99.98%** of the transactions are **not fraudulent** (FraudResult = 0)
- **Only 0.02%** are **fraudulent** (FraudResult = 1)

□ This is a **highly imbalanced dataset**, which is **common in fraud detection** problems.

□ therefore I must **NOT use FraudResult as the target variable** for your Task 4 because:

I am instructed to **engineer a *proxy* target (is_high_risk)** based on customer disengagement behavior using **RFM + clustering**, rather than using an existing fraud label.

The calculation of RFM is shown below

```
CustomerId            Recency   Frequency    Monetary

0    CustomerId_1       83          1             -10000.0

1   CustomerId_10       83          1             -10000.0

2  CustomerId_1001      89          5             20000.0

3  CustomerId_1002      25         11             4225.0

4  CustomerId_1003      11          6            20000.0
```

The output shows **Negative Monetary Values** e.g., CustomerId_1 and CustomerId_10

There are 1000 negative entry in the 95394 entry of the total Monetary values

With **1,000 negative entries** out of **95,394 total transactions** (roughly **1.05%** of your data), these can not be ignored (treated like outlier) ,because , Negative values in the Monetary column could indicate **refunds, chargebacks, debts, or data errors**.

Therefore there are 2 ways to verify:

1. Check with Domain Experts (Best Practice)

- If this is a **real-world banking/fintech dataset**, consult domain experts (e.g., fraud analysts, loan officers) to confirm whether negative amounts are valid (e.g., chargebacks, reversals).
- **Expected Reasons for Negative Values**:
- **Chargebacks**: Fraudulent transactions reversed by the bank.
- **Refunds**: Legitimate returns processed as negative amounts.
- **Data Errors**: Incorrect recording (e.g., misplaced negative signs).

2. Analyze the Data for Patterns

- If there is no way to consult experts, investigate empirically:

Therefore I do **Fraud correlation analysis** : Check if these negatives align with fraud labels:

The output shows that **99.9974% of negative transactions are labeled** FraudResult=0 **(non-fraudulent)**, while only **0.0026% are** FraudResult=1 **(fraudulent)**.

This suggests:

1. **Negatives are likely NOT chargebacks/fraud-related** (since fraud-linked transactions would have higher FraudResult=1 rates).
2. **Possible causes for negative amounts**:
   - **Refunds/returns**: Legitimate reimbursements to customers.
   - **Data entry errors**: Incorrectly recorded transactions (e.g., negative sign added by mistake).
   - **Reversals**: Non-fraudulent payment reversals (e.g., failed transactions).

Recommended Actions

1. Treat Negative Amounts as Non-Risky (Default Approach)

Since negatives are overwhelmingly **not fraud-related**, then we can:

- **Keep them in RFM clustering** but treat them as neutral/low-risk.
- **Example**: A customer with Monetary = -100 (refund) and Frequency = 1 is less risky than one with Monetary = -10000 (potential outlier).

2. Create a Separate Feature for Negatives

Add a binary column to flag negative transactions for modeling:

3. Investigate Outliers

I Check if extreme negative values are errors:

The result shows that **Small negatives (likely legitimate)**:
Most transactions are between **-$1.20 to -$300** (refunds/reversals).

Large negatives (potential errors):
Transactions like **-$200,000** are almost certainly **data errors** (unrealistic for most retail banking scenarios).

The final decision is that Clip or remove implausible values (e.g., ≤ -$100,000) this because Large negatives**:** Likely errors → distort clustering if not removed

Treat Small Negatives as Legitimate i,e for transactions between **-$300 to -$1.20** we will handle them by Using standardized to minimize skew from negatives and Proceed with RFM clustering:

Impact on High-Risk Labeling

Clusters with negative Monetary will now reflect low engagement (not fraud).

Final is_high_risk labels will focus on:

- High Recency (inactive customers).
- Low Frequency (few transactions).
- Low Monetary (small spenders, including refund receivers).

Final Decision

Negatives are likely refunds/reversals (not fraud). However, Proceed with clustering but document this assumption.

Next I Perform K-Means Clustering on the RFM

Cluster	Recency (Days)	Frequency (Count)	Monetary (Amount)	Risk Profile
0	11.6	34.8	$125,495	**Low-risk** (Active, frequent, high spend)
1	28.0	4,091	**-$104,900,000**	**Anomaly** (Extreme frequency & negative amount)
2	60.8	7.8	$52,072	**High-risk** (Inactive, low engagement)

Key Observations

1. **Cluster 1 is Invalid**:
   - **Frequency = 4,091** transactions is unrealistic (likely data error).
   - **Monetary = -$104M** confirms this cluster is noise.
2. **Valid Clusters**:
   - **Cluster 0**: Healthy customers (low recency, high activity).
   - **Cluster 2**: High-risk candidates (60.8 days since last transaction, low frequency).

Action Plan

1. Remove Anomalous Cluster 1

2. Reassign Cluster Labels

3. Validate Distribution

After the removal of the anomalous cluster_1 and after relabeling the remaining clusters (0 → 0, 2 → 1),

I checkout the Validation Distribution and found that

- **61.8% low-risk** (is_high_risk=0)
- **38.2% high-risk** (is_high_risk=1

This might be a good balance for modeling, but **38.2% high-risk** is slightly high for real-world credit risk (typically 10-30%).

This suggests the clustering may be:
- **Too aggressive** in labeling (merging some medium-risk customers into high-risk)
- **Influenced by refund patterns** (negative Monetary values)

Therefore I do an optimization further and the Final Risk Distribution is found to be

```
0.0    0.927845
1.0    0.072155
```

This shows the High-risk i.e "Customers with Recency > 60 days (Cluster 1)" are 7.2% and the Low-risk are 92.8% **Key Insights from High-Risk Customers**

Metric	Mean	Min	50th Percentile	Max
Recency	65.9 days	53 days	63 days	90 days
Frequency	40.6 tx	1 tx	16 tx	181 tx
Monetary	$130,342	-$94,200	$44,300	$2.02M

After optimization of the clustering, I make Feature Selection and address Class Imbalance

Summary of task-4

High-Risk Customer Definition

- Cluster 1 (of 3) identified as high-risk based on:

- Recency > 60 days

- Frequency < 10 transactions

- Monetary value in bottom quartile

Class Distribution

- Original: 7.2% high-risk

- After SMOTE: 50% high-risk (for modeling)

Key Decisions

- Negative Monetary values treated as refunds

- Unclustered customers labeled as low-risk

- RFM features standardized before clustering