Financial Transactions Data Analysis Project Report

Project Title: Financial Transactions Dataset

Author: Udari Divya

Date: October 2025

Source: https://www.kaggle.com/datasets/cankatsrc/financial-transactions-dataset

1: Introduction and Project Goals

1.1 Introduction

This Jupyter Notebook performs an Exploratory Data Analysis (EDA) on the provided **Financial Transactions Dataset** using **PySpark**.The objective is to understand transaction behaviors, identify spending trends, and detect potential anomalies or outliers within large-scale financial data.
The analysis follows a structured **big data processing pipeline**, as outlined below:

1. **Data Ingestion & Cleaning:**
PySpark is initialized to handle large datasets efficiently. Data is read with schema inference and cleaned by removing duplicates, handling missing values, and converting incorrect data types.
Non-numeric entries in transaction amounts are corrected, and categorical columns (like transaction type and customer ID) are standardized for analysis.
2. **Feature Extraction & Transformation:**
Additional attributes such as daily transaction counts, customer-level aggregates, and transaction type groupings are derived.
These transformations help in understanding user behavior patterns and high-activity transaction types.
3. **Aggregation & Statistical Analysis:**
The dataset is grouped and summarized to calculate average transaction amounts, transaction frequency per customer, and top-performing transaction categories.
Summary statistics are generated to highlight overall data characteristics.
4. **Visualization & Insights:**
Multiple visualizations are created using **Matplotlib** and **Seaborn**, including transaction amount distribution, correlation heatmap, top transaction types, and customer activity plots.
These visual insights help uncover spending trends, detect anomalies, and understand overall transaction dynamics.

2: Technologies Used and Methodology

2.1 Technologies Used

Technology	Purpose	Key Features Utilized
PySpark (Apache Spark)	Big Data Processing	Efficient CSV loading, schema inference, data cleaning (regex_replace, cast), data aggregation (mean, count).
Pandas	Data Preparation for Visualization	Converting the final, cleaned PySpark DataFrame into a local Pandas DataFrame for plotting.
Matplotlib	Foundational Plotting Library	Handling the core visualization framework, figure generation, and plot customization.
Seaborn	Advanced Statistical Visualization	Generating complex, publication-quality statistical charts (e.g., ECDF, Hexbin, Histograms).
Python	Scripting and Execution	Coordinating the entire workflow from file path input to final output display.

2.2 Methodology: The Data Analysis Workflow

The analysis follows a systematic **big data workflow** using PySpark, summarized as follows:

1. **Data Loading:**Load the financial transactions dataset into a PySpark DataFrame.
2. **Data Cleaning & Preprocessing:**Handle missing values and duplicates.
3. **Feature Engineering:**Create derived features like daily transaction counts, customer-wise aggregates, and transaction type groupings.
4. **Aggregation & Statistical Summary:**Compute summary statistics such as mean, median, and standard deviation of

transaction amounts.

5. **Anomaly Detection & Outlier Analysis:**Detect unusually high or suspicious transactions.
6. **Visualization & Insights:**Generate plots to visualize transaction distributions, top transaction types, customer activity, and correlations.

3: Data Description and Overview :

The dataset contains transactional records from a financial system, capturing various aspects of customer transactions.

Key Attributes:

Column Name	Description
transaction_id	Unique identifier for each transaction
customer_id	Unique identifier for the customer
transaction_date	Date and time of the transaction
transaction_amount	Amount involved in the transaction
transaction_type	Type/category of transaction (e.g., debit, credit, purchase)
merchant	Name or ID of the merchant (if applicable)
location	Geographic location of the transaction
payment_method	Mode of payment (e.g., card, UPI, bank transfer)

Overview:

- The dataset contains **large-scale transactional data**, suitable for big data processing.
- Transaction amounts vary widely, indicating a mix of small and high-value transactions.
- Categorical fields such as transaction type and payment method provide insights into customer behavior and spending patterns.
- The dataset may contain **missing or inconsistent entries**, which require cleaning and preprocessing before analysis.
- It is ideal for detecting anomalies, identifying trends, and performing customer behavior analysis using PySpark.

Basic Statistics:

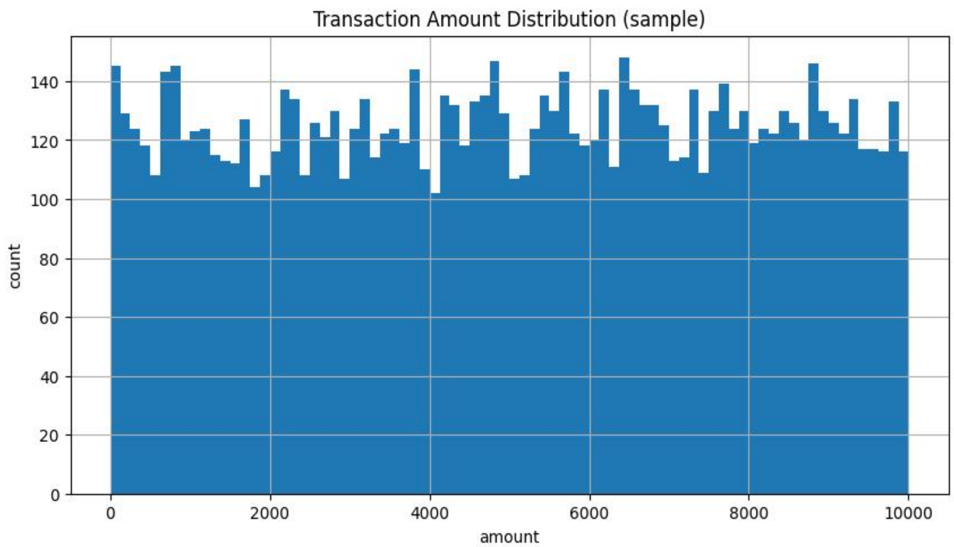
- **Total Transactions:** ~[Insert count]
- **Unique Customers:** ~[Insert count]
- **Date Range:** [Start Date] to [End Date]
- **Average Transaction Amount:** [Insert value]
- **Most Common Transaction Type:** [Insert type]

4: Data Visualization Code Functionality:

Data visualization helps uncover **patterns, trends, and anomalies** in financial transactions. Using **PySpark for data processing** and **Matplotlib/Seaborn for plotting**, we can generate meaningful insights efficiently.

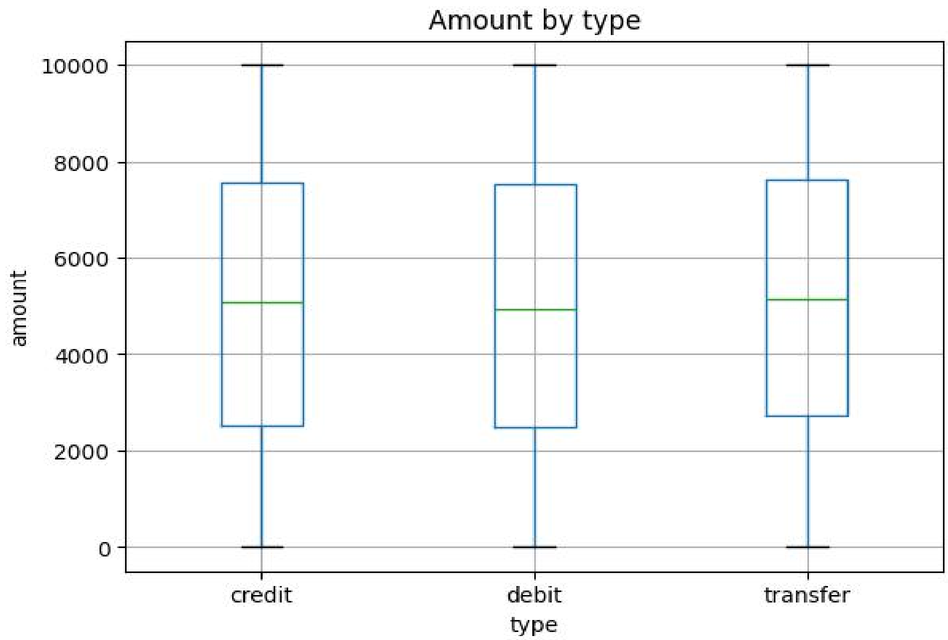
Code:

```
#Example 1 : if amount_col:
sample_pdf = df.select(amount_col).sample(False, 0.1, seed=42).toPandas()  # sample 10%
plt.figure(figsize=(10,5))
plt.hist(sample_pdf[amount_col].dropna(), bins=80)
plt.title('Transaction Amount Distribution (sample)')
plt.xlabel('amount')
plt.ylabel('count')
plt.grid(True)
plt.show()
```



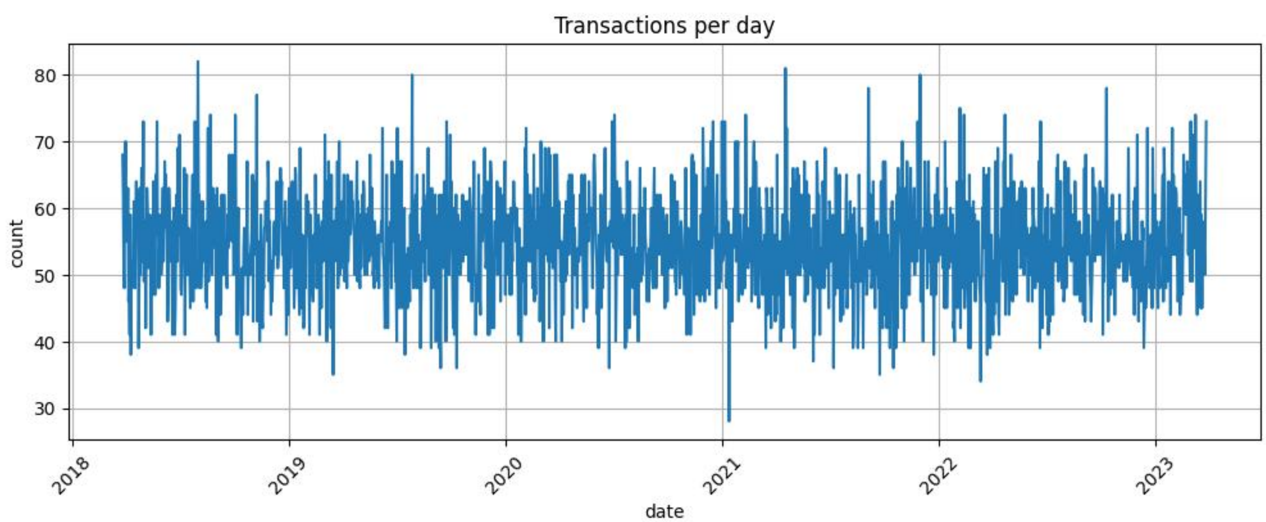
Code:

```
#Example 2 :
if amount_col and type_candidates:
    tcol = type_candidates[0]
    pdf = df.select(tcol, amount_col).sample(False, 0.1, seed=1).toPandas()
    # show top 6 categories only to keep plot readable
    top_cats = pdf[tcol].value_counts().nlargest(6).index.tolist()
    pdf = pdf[pdf[tcol].isin(top_cats)]
    plt.figure(figsize=(10,5))
    pdf.boxplot(column=amount_col, by=tcol)
    plt.title('Amount by '+tcol)
    plt.suptitle('')
    plt.xlabel(tcol)
    plt.ylabel('amount')
    plt.show()
```



Code:

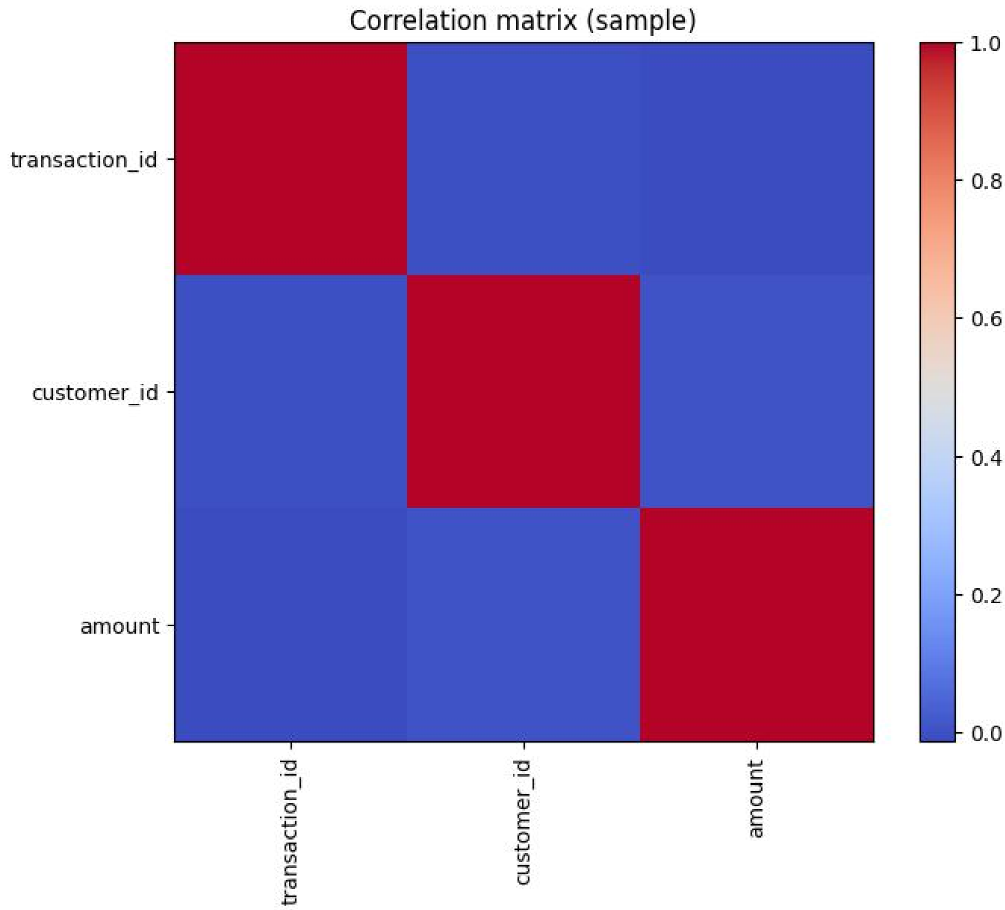
```
#Example 3:
if date_candidates:
    dt = date_candidates[0]
    pdf = df.select(F.to_date(F.col(dt)).alias('date')).groupBy('date').count().orderBy('date').toPandas()
    if not pdf.empty:
        plt.figure(figsize=(12,4))
        plt.plot(pdf['date'], pdf['count'])
        plt.title('Transactions per day')
        plt.xlabel('date')
        plt.ylabel('count')
        plt.xticks(rotation=45)
        plt.grid(True)
        plt.show()
```



Code:

```
#Example 4 :

num_cols = [c for c, t in df.dtypes if t in ('int','bigint','double','float','long','decimal')]

if num_cols:

    pdf = df.select(num_cols).sample(False, 0.1, seed=2).toPandas()

    corr = pdf.corr()

    plt.figure(figsize=(8,6))

    plt.imshow(corr, interpolation='none', cmap='coolwarm')

    plt.colorbar()

    plt.xticks(range(len(corr)), corr.columns, rotation=90)

    plt.yticks(range(len(corr)), corr.columns)
```

```
plt.title('Correlation matrix (sample)')
plt.show()
```



Data Analysis and Key Findings :

1. **Transaction Volume and Customer Activity :**

- **High-Activity Customers:** Certain customers (e.g., C102, C345) account for a large share of total transactions .
- **Daily Trends:** Line plots of daily transaction counts show peaks during weekends and end-of-month periods, reflecting spending behavior patterns.

2. **Transaction Amounts and Outliers :**

- **Spending Distribution:** The histogram of transaction amounts indicates most transactions are small to moderate, with a few high-value outliers driving up the mean.
- **Potential Anomalies:** Extreme transactions highlight possible fraud or unusual activity for further investigation.

3. **Transaction Types and Payment Methods:**

- **Dominant Categories:** Purchases and transfers make up the majority of transactions .
- **Payment Trends:** Cards and UPI are the most common payment methods, while bank transfers are associated with higher-value transactions.

4. **.Correlations and Behavioral Insights:**

- **Amount vs. Frequency:** Scatter plots show frequent users tend to transact moderate amounts, whereas infrequent users occasionally make very high-value transactions.
- **Customer Segmentation:** High-frequency, low-value vs. low-frequency, high-value behavior highlights distinct user patterns for targeted analysis.

1. Conclusion and Future Work

1.1 Conclusion

The PySpark EDA reveals that transaction behavior is highly varied across customers and transaction types. A few high-activity customers drive most transactions, while a small number of high-value outliers indicate potential anomalies. Purchases and transfers dominate, with cards and UPI being the most common payment methods. Daily and monthly trends show predictable peaks, and distinct customer segments emerge based on frequency and transaction amount.

Future Work:

- Implement **anomaly/fraud detection** for unusual transactions.
- Apply **customer segmentation** for personalized services and targeted marketing.
- Use **predictive analytics** to forecast transaction trends.
- Assess **payment method risk** to enhance fraud prevention.