Customer Due Diligence & Suspicious Activity Monitoring System

Project Type: Individual Portfolio Project

Problem

Financial institutions are under constant pressure to detect and prevent fraudulent behavior while maintaining compliance with Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations. However, this can be tough when data is messy, and fraudsters mimic normal behavior. This project aims to simulate a suspicious activity monitoring system in a financial service context.

Objective

- Clean and combine multiple datasets
- Explore trends and detect irregularities through EDA
- Engineer features to highlight risky behavior
- Use unsupervised machine learning to detect behavioral anomalies

Solution & Process

1. Tools Used

- SQL (via DB Browser): data cleaning, standardization, and joins.
- Python (Jupyter Notebooks): EDA, feature engineering, anomaly detection.
  - Libraries: Pandas, Matplotlib, Seaborn, Scikit-learn.

2. Dataset Overview

I worked with three synthetic datasets:

- Customers: Simulated KYC information including customer ID, name, country, occupation, account open date, KYC status, and customer type (e.g., Individual, Business).
- Transactions: Records of financial transactions, including transaction ID, date, amount in USD, country, merchant, and transaction channel (e.g., online, ATM).
- Sanctions: Sanctioned individuals with assigned risk levels (e.g., Low, Medium, High).

These datasets were intentionally “dirtied” to showcase practical data cleaning skills. After cleaning, these datasets were merged on customer_id, providing a single customer-centric view for analysis.

3. Methodology

- Performed comprehensive data cleaning using SQL:

  - Trimmed whitespace from text fields.
  - Standardized country and occupation names.
  - Filled null kyc_status values with 'Pending'.
  - Formatted dates into ISO format (YYYY-MM-DD).
  - Joined cleaned customer and transaction tables with sanctions data.

  Cleaned datasets were exported to Python for further analysis.
- Performed EDA in Python:
  - Average spending by customer type, channel, and country
  - Volume of transactions per channel and country
  - Distribution of risk levels (from sanctions dataset)
  - Outliers in transaction behaviors
  - Time series analysis

  Visualizations include bar plots, box plots, and line graphs
- Created several key features to enhance fraud detection:
  - total_spent: Total amount spent by the customer across all transactions.
  - avg_spent: Average transaction value, capturing typical spending behavior.
  - std_spent: Standard deviation of transaction amounts, reflecting variability or inconsistency in spending.
  - num_txns: Total number of transactions per customer, representing activity level.
  - unique_channels: Number of distinct transaction channels used (e.g., ATM, online, in-store), to gauge behavioral diversity.
  - unique_merchants: Count of unique merchants interacted with, offering insight into purchasing variety.
  - unique_countries: Number of unique countries where transactions were conducted, which can help identify customers with international activity.

  Missing values were replaced with 0 to ensure compatibility with downstream models and analysis. These features allowed for a more robust understanding of customer behavior and served as input to anomaly detection algorithms used later in the project.
- Used Isolation Forest to identify behavioral anomalies:
  - Model trained on engineered features
  - Labeled anomalies with a binary flag

Key Insights

- There were 25 customers flagged as anomalous due to patterns like high spend with low transaction counts or limited variation in merchants and countries, suggesting potential risk behaviors worth further investigation.
- The UK had the highest number of high-risk individuals (11), and high-risk transactions were most conducted via online and wire channels, indicating a potential need for tighter monitoring of digital and cross-border activity.

- The top 10 customers, led by Amber Rios with over $843K in transactions, collectively contribute a significant portion of total volume, highlighting a high-value segment worth closer monitoring.
- Retail customers spend more on average per transaction ($6,108) than corporate customers ($2,570) suggesting that individuals may make fewer but higher-value purchases, whereas businesses transact more frequently with smaller amounts.

Outcome

This project demonstrated how a combination of SQL and Python can be used to clean, analyze, and model financial data for fraud detection purposes. The unsupervised approach offered a starting point for identifying suspicious customer behavior.

Future Improvements

- Incorporate real-world data if available
- Using clustering or supervised learning if fraud labels are introduced
- Developing a dashboard for a risk team

Project Link

GitHub: (SQL/Python Fraud Detection) https://github.com/srdodson22