IRMAI CODING CHALLENGE DOCUMENTATION

<u>1.Stock Trade Outlier Analysis using Graph Database</u>

Introduction

This project analyzes foreign exchange (FX) trades to detect outliers using statistical methods and visualizes the trade relationships using a graph database approach. The system identifies unusual trades based on volume and price deviations and presents them in an interactive graph and table format.

Objectives

- **Analyze FX Trades** – Detect anomalies in trading data based on statistical measures.
- **Graph-Based Trade Model** – Represent trades as nodes and relationships in a directed graph.
- **Outlier Detection** – Identify unusual trades using Z-score analysis.
- **Comparison with Expected Patterns** – Detects deviations from normal trading behavior.
- **Data Visualization** – Display the trade network and outliers interactively.

Tools & Technologies

- **Python** – Primary programming language
- **Gradio** – For interactive UI
- **NetworkX** – Graph visualization
- **Matplotlib** – Plotting graphs
- **Pandas & NumPy** – Data processing

Implementation Details

1. Data Collection

- A synthetic dataset is created with 15 FX trade records.
- Trades involve currency pairs (EUR/USD, GBP/USD) with varying volumes and prices.
- Some extreme values are added to simulate outliers.

2. Outlier Detection

- Z-score is calculated for trade volumes.
- Trades with Z-score > 2 or < -2 are marked as outliers.

3. Graph Construction

- Each trade is represented as a node.
- Consecutive trades are linked using directed edges.
- Nodes are color-coded:
  - **Red** for outliers
  - **Green** for normal trades

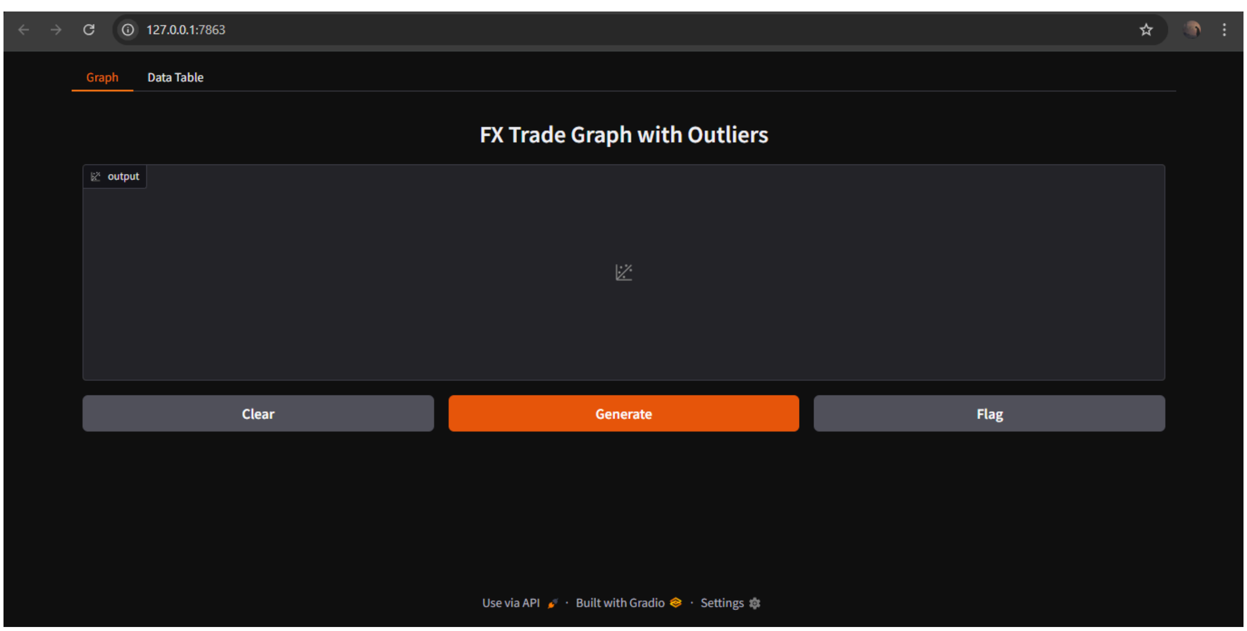
4. Visualization & UI

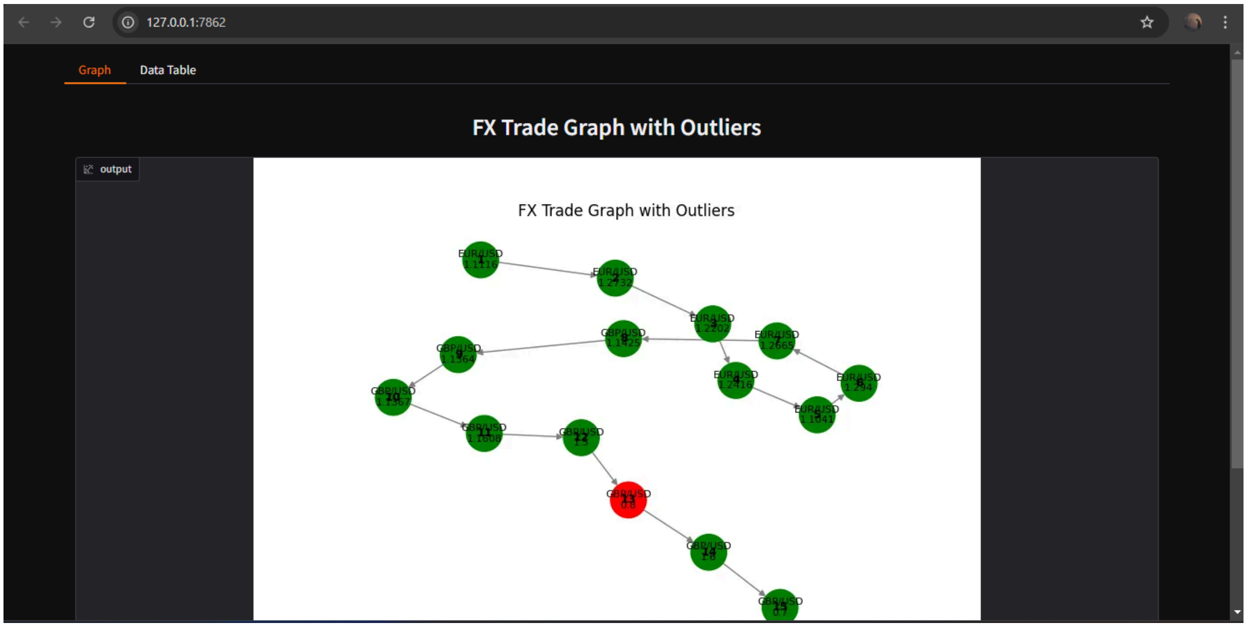
- **Trade Graph:** Displays relationships between trades, highlighting anomalies.
- **Data Table:** Shows raw trade data with outlier information.

Usage

1. Run the script to launch the Gradio interface.
2. Navigate between **Graph** and **Data Table** tabs.
3. Analyze outliers and trade patterns interactively.

OUTPUT:



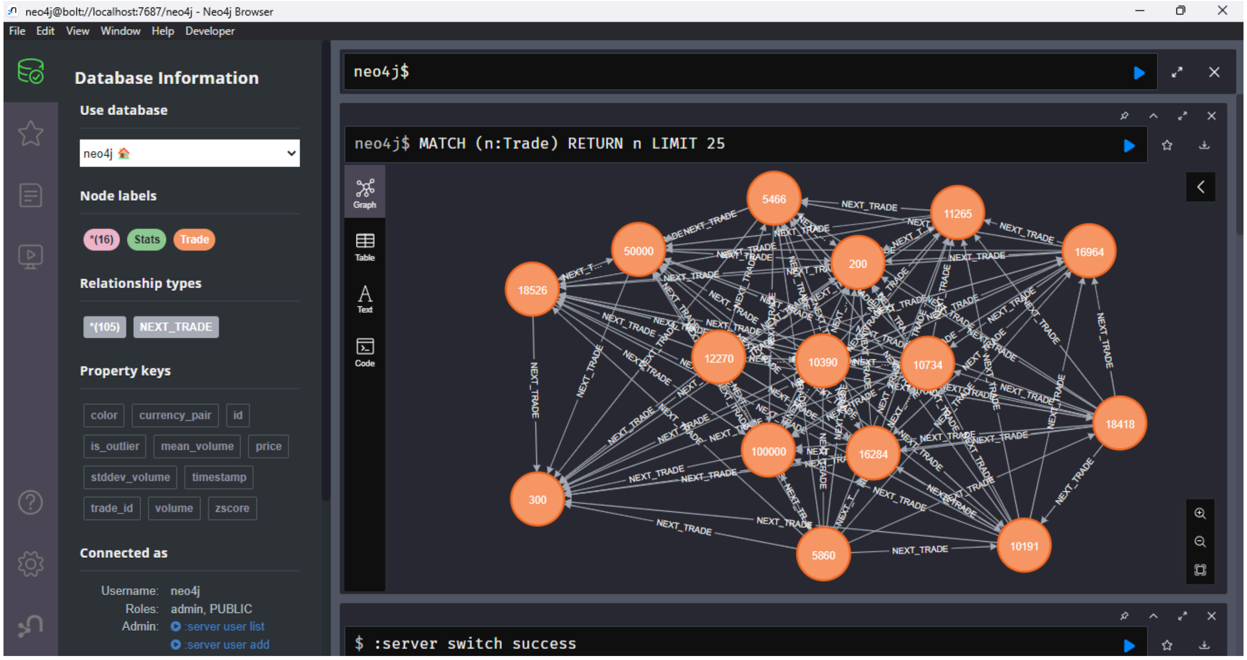


Graph Data Table

Trade Data Table

output

trade_id	currency_pair	volume	price	timestamp	volume_zscore	is_outlier
1	EUR/USD	12270	1.1116	2024-02-01 00:00:00	-0.27158662163316655	false
2	EUR/USD	5860	1.2732	2024-02-01 01:00:00	-0.5255601833638845	false
3	EUR/USD	10390	1.2202	2024-02-01 02:00:00	-0.3460749673513802	false
4	EUR/USD	18418	1.2416	2024-02-01 03:00:00	-0.027993882550412352	false
5	EUR/USD	10191	1.1041	2024-02-01 04:00:00	-0.35395963798857416	false
6	EUR/USD	16964	1.294	2024-02-01 05:00:00	-0.08560348610056275	false
7	EUR/USD	16284	1.2665	2024-02-01 06:00:00	-0.11254607923268259	false
8	GBP/USD	10734	1.1425	2024-02-01 07:00:00	-0.33244518494336667	false
9	GBP/USD	11265	1.1364	2024-02-01 08:00:00	-0.31140618942402015	false
10	GBP/USD	5466	1.1367	2024-02-01 09:00:00	-0.5411710387963187	false
11	GBP/USD	18526	1.1608	2024-02-01 10:00:00	-0.023714764817663905	false
12	GBP/USD	50000	1.5	2024-02-01 11:00:00	1.223331082594601	false



Conclusion

This project provides a visual and statistical approach to detecting anomalies in FX trades. It can be extended with real-time data integration and machine learning-based anomaly detection for enhanced accuracy.

2.Stock Trade Gap Analysis using Graph Database

Introduction

This project aims to analyze stock trade data, identify gaps, and visualize trade relationships using a graph database (Neo4j). It detects missing trades, unusual price changes, and inconsistencies by leveraging statistical analysis and graph structures.

Objectives

- **Stock Trade Gap Analysis** – Detect anomalies such as missing trades, unusual price jumps, and inconsistencies.
- **Graph-Based Trade Model** – Represent trades as nodes and relationships as edges in Neo4j.
- **Trade Gap Detection** – Identify missing trade volumes and sudden price changes.
- **Comparison with Expected Patterns** – Highlight deviations from normal trading behavior.
- **Data Visualization** – Generate interactive trade graphs and tables using Gradio.

Technologies Used

- **Python** – Core programming language
- **Neo4j** – Graph database for trade storage
- **NetworkX** – For trade graph visualization
- **Matplotlib** – Plotting graphs
- **Gradio** – UI framework for visualization
- **Pandas & NumPy** – Data analysis

Implementation Details

1. Data Generation & Insertion

- Created **20 sample stock trades** for AAPL and MSFT.
- Added missing values and sudden price jumps to simulate trade gaps.
- Inserted trade data as nodes in **Neo4j**.
- Established **NEXT_TRADE** relationships between consecutive trades.

2. Gap Analysis

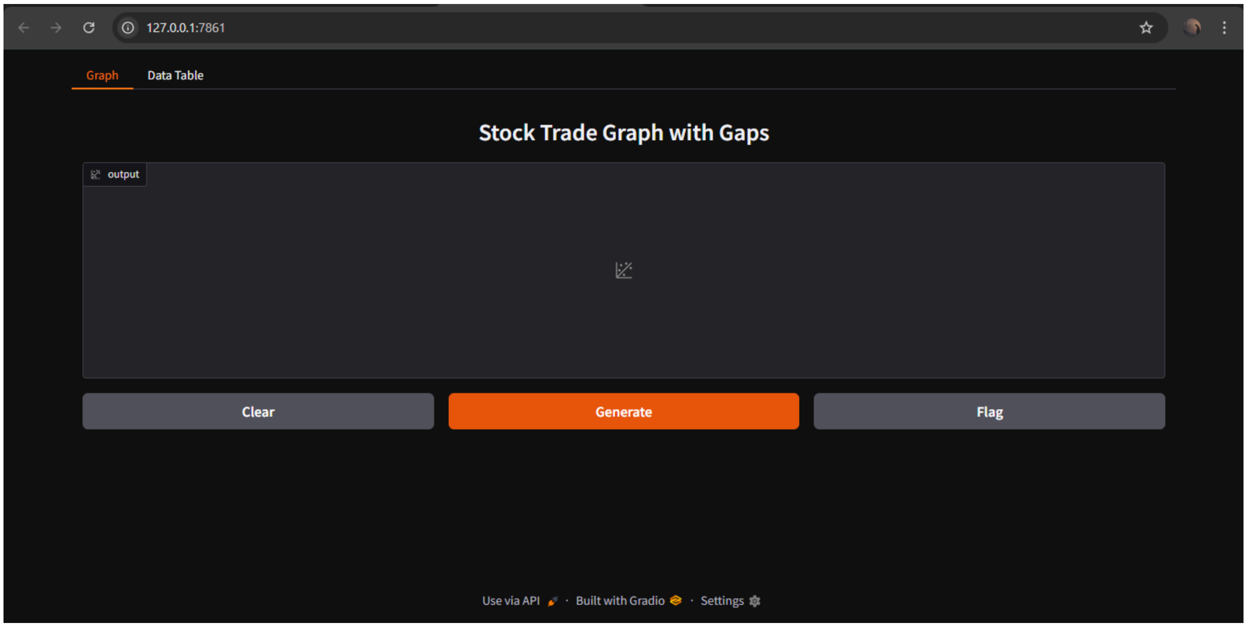
- Marked trades with **missing volume (0)** or **missing price (0)** as gaps.
- Detected **sudden price jumps (>20 points change)**.
- Flagged trades with anomalies in the database.

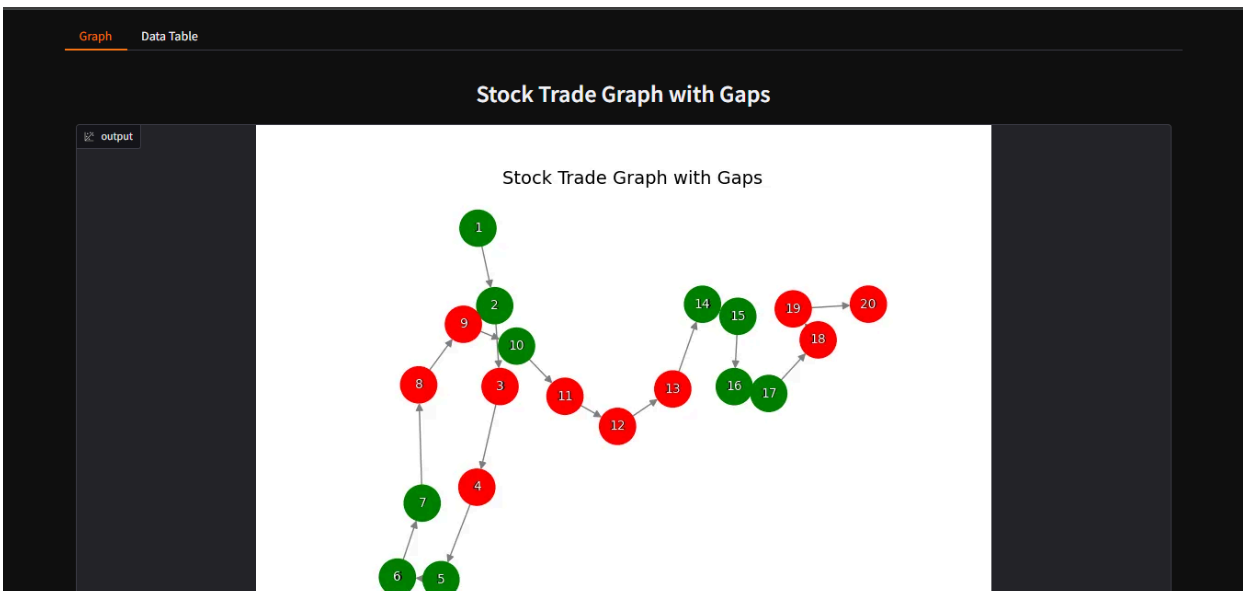
3. Visualization & UI

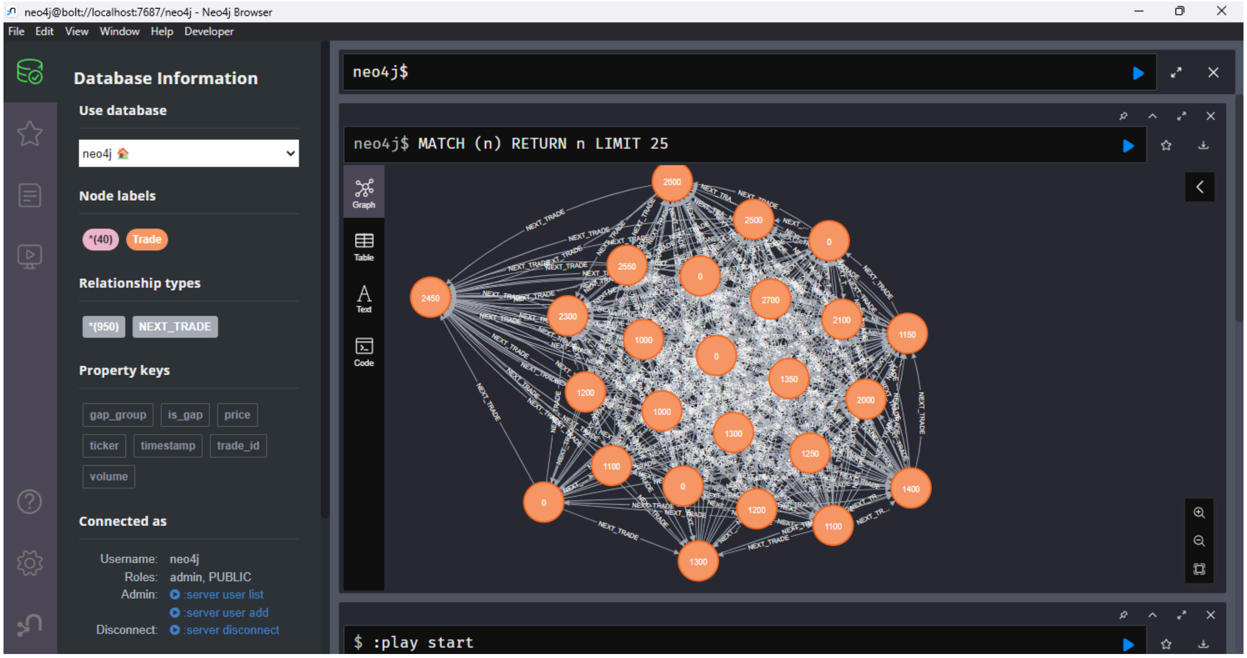
- **Trade Graph:** Nodes represent trades, with gaps marked in **red** and normal trades in **green**.
- **Data Table:** Displays trade records with gap indicators.

Usage Instructions

1. Run the script to establish a connection with **Neo4j**.
2. View the interactive **Stock Trade Graph** and **Data Table** in Gradio.
3. Identify **trade gaps** and **outliers** in the visual representation.


127.0.0.1:7861
Graph
Data Table
Stock Trade Graph with Gaps
output
Clear
Generate
Flag
Use via API
Built with Gradio
Settings


Graph
Data Table
Stock Trade Graph with Gaps
output
Stock Trade Graph with Gaps
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20



Conclusion

This system efficiently detects and visualizes stock trade anomalies, leveraging **graph databases** and **data analysis techniques**. Future improvements can include **real-time data processing** and **machine learning-based anomaly detection**.

3.Failure Mode and Effect Analysis (FMEA) for Financial Transactions

Introduction

Financial transactions involve risks such as fraud, duplicate payments, and unauthorized transfers. Failure Mode and Effects Analysis (FMEA) is a systematic approach to identifying and assessing potential failure points in financial transactions. This project implements FMEA using a graph database (Neo4j) and visualizes transaction anomalies using network graphs.

Objectives

- Develop an FMEA-based financial analysis system to identify risks in transactions.

- Construct a graph-based transaction model in Neo4j, representing transactions as nodes and relationships as edges.
- Identify and analyze failure modes, such as unusual transaction amounts and duplicate payments.
- Compare transactions against risk benchmarks to detect deviations.
- Visualize and report insights through interactive graphs and tables.

Tools & Technologies Used

- Python for data processing and transaction analysis.
- Neo4j for storing and managing financial transactions as a graph database.
- NetworkX and Matplotlib for visualizing financial transactions as graphs.
- Gradio for creating an interactive UI for transaction analysis.
- Pandas and NumPy for handling and analyzing transaction data.

Implementation Steps

1. Generate Sample Financial Transactions

- Created a dataset of 20 financial transactions with attributes like account details, amount, type, and timestamp.
- Introduced anomalies such as large transactions (>3000), zero amounts, and duplicate transactions.

2. Insert Transactions into Neo4j

- Established a connection to a local Neo4j database.
- Stored each transaction as a node with attributes.
- Created relationships between transactions based on timestamps.

3. Define Failure Modes and Identify Anomalies

- Defined conditions for failure modes: large transactions, zero amounts, and duplicate transactions.
- Labeled transactions based on failure mode criteria.

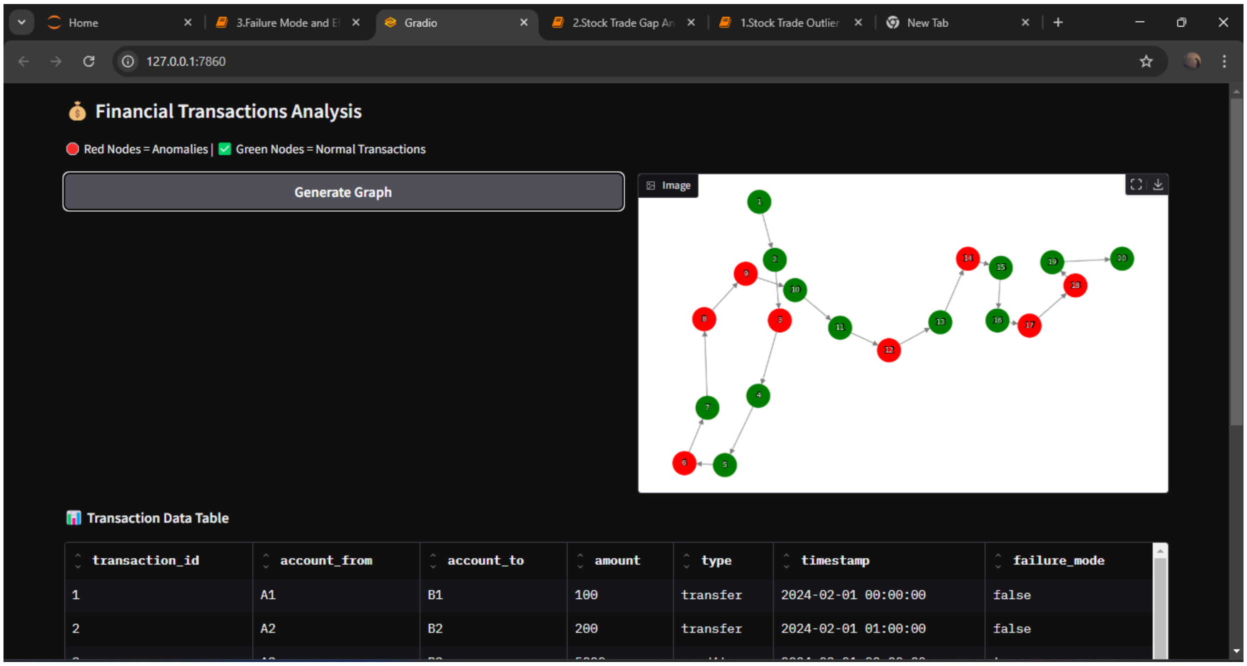
4. Visualize Financial Transactions

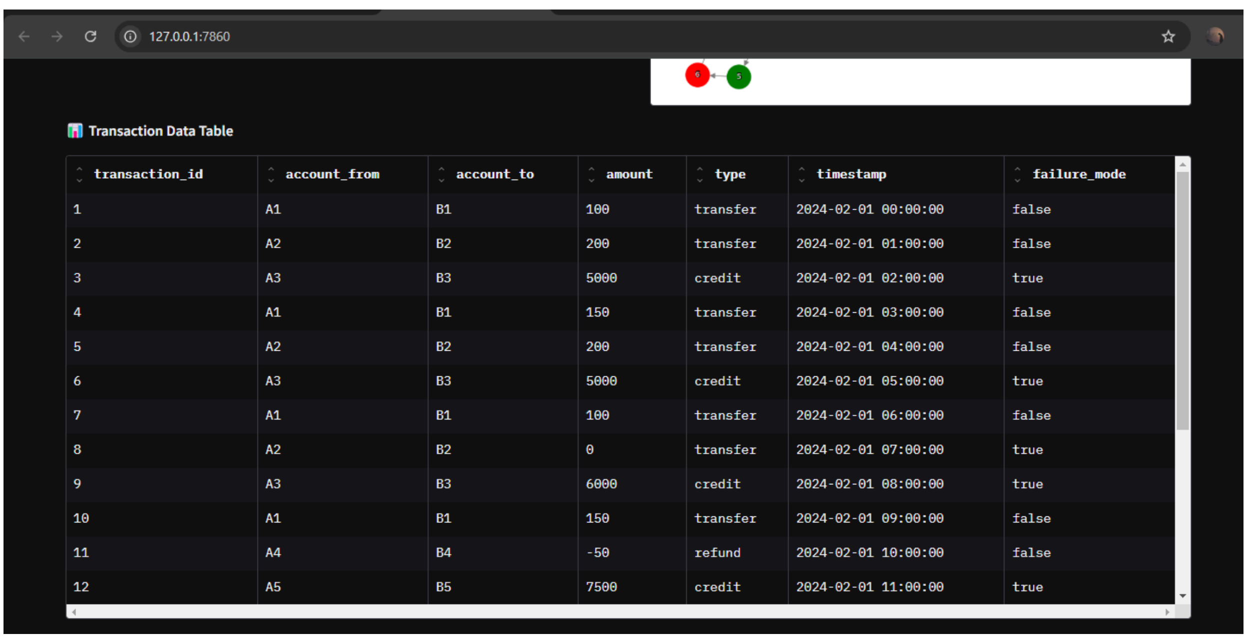
- Used NetworkX to build a transaction graph with anomalies highlighted.
- Generated an interactive UI using Gradio to display transaction data and graphs.

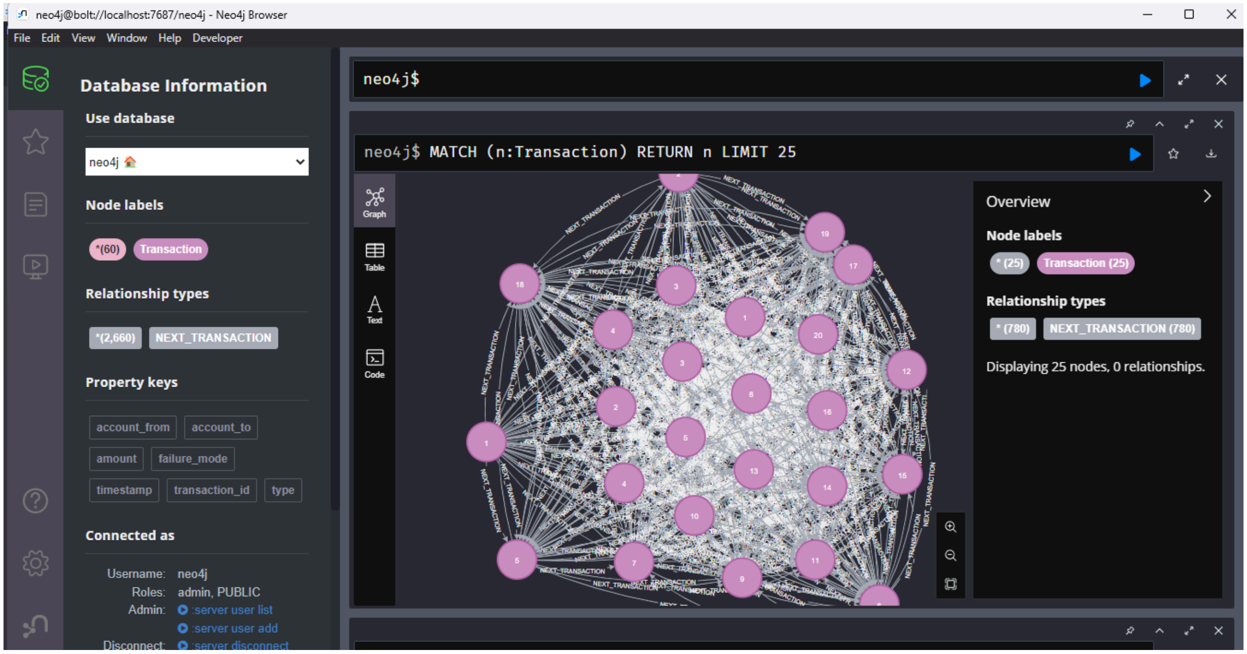
Usage

- Run the Python script to generate sample transaction data and insert it into Neo4j.
- Use Gradio UI to visualize the transaction graph and inspect anomalies.
- Analyze transactions using interactive graphs and tables.

OUTPUT







Conclusion

This project provides a structured approach to financial risk assessment using FMEA. By integrating a graph database and visualization tools, it enables effective detection of anomalies and improves financial transaction monitoring.