P. Hima Varshini
CCE-23031

Data Base Management System

Fraud Detection in Banking Transactions Using SQL

Abstract

Fraudulent activities in banking systems contribute to significant financial losses each year. This project focuses on detecting suspicious banking transactions through the use of SQL queries and a structured relational database. By analyzing transaction attributes such as high-value transfers, frequency, and patterns of unusual activity, the system identifies potential anomalies for further investigation. The implementation of this fraud detection system demonstrates the efficacy of SQL in recognizing patterns indicative of fraudulent behavior, providing a foundation for a more secure banking environment.

Introduction

Fraudulent activities in the banking sector have become a significant threat, impacting financial institutions, customers, and even the global economy. As digital banking continues to grow, the sophistication of fraudsters has also increased, making traditional fraud detection methods less effective. This project aims to address these challenges by creating an automated fraud detection system using SQL within a relational database structure.

The primary goal of this system is to identify suspicious patterns in banking transactions that may point to fraudulent behavior. By analyzing transaction data—such as amounts, frequencies, and account activity—the system can flag anomalies in real-time. This allows banks to detect potential fraud early and prevent substantial financial losses.

This report outlines the design and implementation of the fraud detection system, including the methodologies, database structure, and key SQL queries used to identify suspicious activity. It also presents the results of the system's performance and explores how this approach can be applied in real-world banking environments. The scope of this project includes identifying transaction anomalies like unusually high-value transactions, frequent transfers, and irregular activity patterns.

By utilizing this fraud detection system, banks can minimize the risk of fraudulent transactions and improve the security of their digital platforms, ensuring a safer experience for their customers.

Requirements Definition

The fraud detection system must efficiently store, analyze, and flag suspicious transactions while ensuring **security, performance, and scalability**.

Functional Requirements

1. User Management

- Store user details (**ID, account number, email, phone, normal locations**).

2. Transaction Management

- Record transactions (**ID, user ID, amount, timestamp, location, device/IP**).
- Allow querying based on transaction attributes (**amount, location, time**).

3. Fraud Detection Logic

- Flag suspicious transactions based on:
  - **High value** (> $5000).
  - **High frequency** (> 5 transactions in 24 hours).
  - **Location mismatches** (outside normal user locations).
  - **Unknown devices/IPs**.
- Store flagged transactions with **detection reasons**.

4. Data Storage

- Use a **relational database (SQL-based)** for structured storage.
- Maintain a **Flags Table** to log suspicious transactions.

Non-Functional Requirements

1. Performance

- Fraud detection queries must execute **in <2 seconds**.
- Handle **10,000+ transactions per hour** with optimized indexing.

2. Security

- **Role-Based Access Control (RBAC)** restricts unauthorized access.
- **AES-256 encryption** secures sensitive data.
- **Multi-Factor Authentication (MFA)** for admin access.

3. Scalability

- Support **large datasets & real-time fraud detection integration**.
- **Cloud-compatible** for future expansion.

4. Reliability & Data Integrity

- Ensure **ACID compliance** for transaction consistency.
- Use **foreign key constraints & automated backups** to prevent data loss.

5. Maintainability & Usability

- **Clear documentation** for fraud detection logic & database structure.
- Queries should be **modular & easily updatable**.

- Include **dashboard/reporting tools** for reviewing flagged transactions.

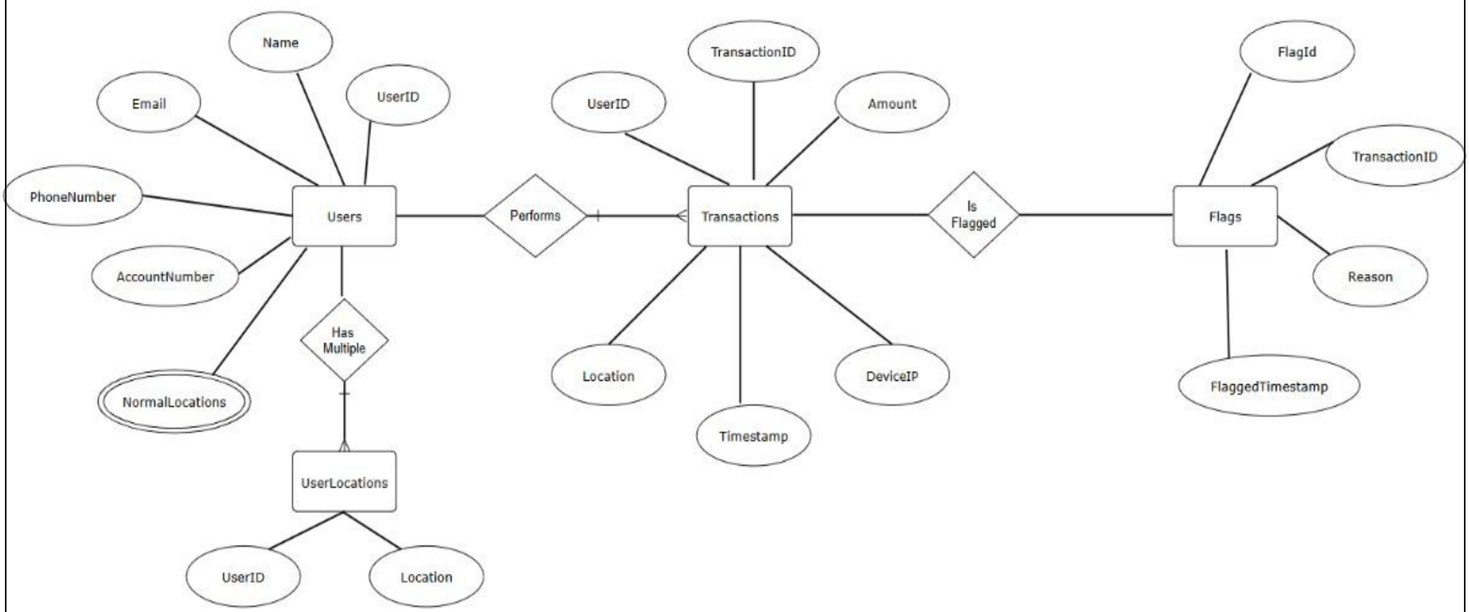
Database Design and Schema

To effectively detect fraudulent transactions, a structured relational database was designed. The database, named FraudDetectionDB, consists of three key tables:

1. Users Table – Stores user details such as name, email, phone number, and address.
2. Transactions Table – Contains transaction records, including the amount, date, type (debit/credit), and the associated merchant.
3. Flags Table – Maintains flagged transactions, recording suspicious activities along with the reason and timestamp.

Each table is linked using foreign key constraints to ensure data integrity. The Users table connects to the Transactions table via user_id, and the Transactions table links to the Flags table via transaction_id.

ER Diagram:



Schema:

<u>UserID</u>	Name	AccountNumber	Email	PhoneNumber	NormalLocations

TransactionsID	UserID	Amount	Location	Timestamp	DeviceIP

TransactionsID	FlagID	Reason	FlaggedTimestamp

UserID	Location

Data Insertion and Sample Records

To test the fraud detection queries, sample data was inserted into the **Users** and **Transactions** tables. The dataset includes multiple users with unique details such as name, email, phone number, and address, along with various transaction records containing transaction amounts, types (debit/credit), timestamps, and merchant names.

Below is a screenshot showing the inserted sample data in the **Users** table:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

user_id	name	email	phone	address
1	Hima	hima@gmail.com	9812345670	No. 12, 1st Cross, Anna Nagar, Chennai, Tamil ...
2	Kavya Priya	kavyapriya@gmail.com	9908765432	Plot 34, Kothapet, Hyderabad, Telangana
3	Thanmayee	thanmayee@gmail.com	9976543210	Sector 19, HUDA, Gurgaon, Haryana
4	Pranathi	pranathi@gmail.com	9345678901	Block A, Sushant Lok, Gurgaon, Haryana
5	Kasyap Sharma	kasyapsharma@gmail.com	9834671230	Street 45, Mansarovar, Jaipur, Rajasthan
6	Abhilash Sharma	abhilashsharma@gmail.com	9865432109	House 5, Vishnu Nagar, Lucknow, Uttar Pradesh
7	Sudhesha	sudhesha@gmail.com	9556784321	Main Road, Palarivattom, Kochi, Kerala
8	Aadhya Sharma	aadhyasharma@gmail.com	9476543210	Flat 302, Andheri West, Mumbai, Maharashtra

Similarly, transaction records were added to the **Transactions** table. These records were used to execute fraud detection queries and identify suspicious activity.

P. Hima Varshini
CCE-23031

transaction_id	user_id	amount	transaction_date	transaction_type	merchant_name
1	1	5000.00	2025-01-10 08:30:00	credit	Amazon
2	1	2000.00	2025-01-15 22:45:00	debit	Flipkart
3	2	1500.00	2025-01-12 14:15:00	credit	Snapdeal
4	2	3000.00	2025-01-18 20:30:00	debit	Myntra
5	3	2000.00	2025-01-11 23:50:00	debit	Big Bazaar
6	3	2500.00	2025-01-14 12:05:00	credit	Zappos
7	4	7000.00	2025-01-10 11:45:00	debit	BookMyShow
8	4	1000.00	2025-01-17 00:30:00	credit	Swiggy
9	5	4000.00	2025-01-13 19:10:00	credit	Jabong
10	5	1200.00	2025-01-16 21:20:00	debit	BookMyShow
11	6	3500.00	2025-01-14 07:55:00	credit	Snapdeal
12	6	4500.00	2025-01-19 23:10:00	debit	Amazon
13	7	5500.00	2025-01-10 17:00:00	credit	Flipkart
14	7	1800.00	2025-01-13 18:40:00	debit	Myntra
15	8	3000.00	2025-01-11 10:20:00	credit	Zappos
16	8	2500.00	2025-01-15 01:10:00	debit	Big Bazaar
NULL	NULL	NULL	NULL	NULL	NULL

To identify suspicious transactions, flagged records were inserted into the **Flags** table. Each flagged transaction is assigned a **flag_reason** based on predefined fraud detection criteria, such as **high transaction amounts, frequent small debits, or unusual activity patterns**.

flag_id	transaction_id	flag_reason	flag_date
1	1	High transaction amount	2025-01-11 00:00:00
2	2	Frequent transaction pattern	2025-01-16 00:00:00
3	5	Frequent debit transactions	2025-01-16 00:00:00
4	6	Large debit amount	2025-01-20 00:00:00
5	7	Multiple high-value transactions	2025-01-15 00:00:00
6	9	Suspicious merchant	2025-01-17 00:00:00
7	12	Unusual transaction activity	2025-01-20 00:00:00
8	13	Frequent small debits	2025-01-18 00:00:00

System Architecture

The fraud detection system is a **SQL-based analytical system** that processes banking transactions to identify suspicious activities using **MySQL Workbench**. It consists of three key components:

1. **Database (Storage Layer)**
   - A **relational database (MySQL)** stores transaction data.

  - Three tables: **Users, Transactions, Flags** (for suspicious transactions).

2. **Query Execution (Processing Layer)**
  - Fraud detection is performed by **executing SQL queries** in **MySQL Workbench**.
  - Queries analyze patterns like **high-value transactions, unusual locations, and frequent transactions**.
3. **Manual Review (Analysis Layer)**
  - Flagged transactions are reviewed for fraud assessment.
  - Results help refine fraud detection rules.

Future Enhancements

- Automate fraud detection with **scheduled SQL queries**.
- Integrate Python for **real-time alerts**.
- Use **ML models** for improved fraud detection.

Normalization & Data Integrity

Users Table

Stores user details. **Primary Key: user_id**

Column Name	Data Type	Description
user_id	INT (PK)	Unique identifier for each user (Auto-incremented)
name	VARCHAR(100)	Full name of the user
email	VARCHAR(100) UNIQUE	Email address (ensures uniqueness)
phone	VARCHAR(15)	Contact number
address	TEXT	User's address

Transactions Table

Stores transaction details. **Primary Key: transaction_id**, **Foreign Key: user_id**

Column Name	**Data Type**	**Description**
transaction_id	INT (PK)	Unique identifier for each transaction (Auto-incremented)
user_id	INT (FK)	References **Users.user_id**
amount	DECIMAL(10,2)	Transaction amount
transaction_date	DATETIME	Date and time of transaction
transaction_type	ENUM('debit', 'credit')	Type of transaction
merchant_name	VARCHAR(100)	Merchant involved in the transaction

Flags Table

Stores flagged suspicious transactions. **Primary Key: flag_id**, **Foreign Key: transaction_id**

Column Name	**Data Type**	**Description**
flag_id	INT (PK)	Unique identifier for flagged transactions (Auto-incremented)
transaction_id	INT (FK)	References **Transactions.transaction_id**
flag_reason	TEXT	Reason for flagging the transaction
flag_date	DATETIME DEFAULT CURRENT_TIMESTAMP	Timestamp when the transaction was flagged

1NF (First Normal Form) - Ensured

- Each column has **atomic values** (no multi-valued attributes like multiple emails or phone numbers in a single field).
- Example: **NormalLocations** (if it existed) was not stored as a comma-separated list, avoiding redundancy.

2NF (Second Normal Form) - Ensured

- There are **no partial dependencies** (all non-key attributes fully depend on the primary key).

- Example: In **Transactions**, amount, transaction_type, and merchant_name depend entirely on transaction_id, not just user_id.

3NF (Third Normal Form) - Ensured

- There are **no transitive dependencies** (non-key attributes depend only on the primary key).
- Example: Instead of storing fraud detection logic inside the **Transactions Table**, a separate **Flags Table** is created to store flagged transactions.

Physical Database Design & Indexing Strategy

To ensure efficient query execution and optimized data retrieval, indexing is implemented on key columns. While the current dataset is synthetic and relatively small, these indexing strategies will be beneficial when handling large-scale banking transactions.

1. Primary and Foreign Key Indexing

- **Primary keys** (user_id, transaction_id, flag_id) are automatically indexed, ensuring fast lookups.
- **Foreign keys** (user_id in Transactions, transaction_id in Flags) maintain referential integrity and optimize **JOIN operations**.

2. Indexing for Query Optimization

Additional indexes are created to improve query performance:

Index	**Column(s)**	**Purpose**	**Optimized Query**
idx_amount	amount (Transactions)	Speeds up fraud detection for high-value transactions	WHERE amount > 5000
idx_transaction_date	transaction_date (Transactions)	Improves performance for time-based fraud analysis	WHERE transaction_date BETWEEN 'X' AND 'Y'
idx_merchant_type	merchant_name, transaction_type (Transactions)	Optimizes merchant-based transaction lookups	WHERE merchant_name = 'Amazon' AND transaction_type = 'debit'
idx_flag_reason	flag_reason (Flags)	Enhances flagged transaction searches	WHERE flag_reason = 'High-Value Transaction'

3. Storage Optimization Considerations

- **Use VARCHAR instead of TEXT** where possible (e.g., merchant_name VARCHAR(100)) to reduce storage space.
- **Archive old transactions periodically** to maintain query speed.

- **Partitioning large datasets** based on transaction_date could improve query performance when dealing with millions of records.

4. Testing Index Performance

Although **synthetic data is small**, indexing can be tested using:

EXPLAIN ANALYZE

SELECT * FROM Transactions WHERE amount > 5000;

This will show whether MySQL **uses the index (idx_amount)** for optimization.

<u>Key Fraud Indicators and Detection Criteria</u>

The system flags suspicious transactions based on key fraud indicators, which are defined through SQL queries:

Category 1: Anomaly Detection

Anomaly detection focuses on transactions that deviate significantly from normal patterns. These queries identify unusual or high-risk activities.

1. Detect High-Value Transactions This query identifies transactions with amounts greater than a specified threshold (e.g., $5,000) that could indicate fraudulent activities.

USE FraudDetectionDB;

SELECT t.transaction_id, t.user_id, t.amount, t.transaction_date, t.transaction_type, t.merchant_name

FROM Transactions t

WHERE t.amount > 5000;

Result Grid | Filter Rows: | Edit: | Export/Import:

	transaction_id	user_id	amount	transaction_date	transaction_type	merchant_name
▶	7	4	7000.00	2025-01-10 00:00:00	debit	BookMyShow
	13	7	5500.00	2025-01-10 00:00:00	credit	Flipkart
*	NULL	NULL	NULL	NULL	NULL	NULL

2. Identify Transactions Outside Business Hours This query flags transactions that occur outside the usual business hours (9:00 AM to 9:00 PM), which could be unusual or indicative of suspicious behavior.

SELECT transaction_id, user_id, amount, transaction_date,

TIME(transaction_date) AS transaction_time, merchant_name

FROM Transactions

WHERE TIME(transaction_date) NOT BETWEEN '09:00:00' AND '21:00:00';

transaction_id	user_id	amount	transaction_date	transaction_time	merchant_name
1	1	5000.00	2025-01-10 08:30:00	08:30:00	Amazon
2	1	2000.00	2025-01-15 22:45:00	22:45:00	Flipkart
5	3	2000.00	2025-01-11 23:50:00	23:50:00	Big Bazaar
8	4	1000.00	2025-01-17 00:30:00	00:30:00	Swiggy
10	5	1200.00	2025-01-16 21:20:00	21:20:00	BookMyShow
11	6	3500.00	2025-01-14 07:55:00	07:55:00	Snapdeal
12	6	4500.00	2025-01-19 23:10:00	23:10:00	Amazon
16	8	2500.00	2025-01-15 01:10:00	01:10:00	Big Bazaar

Category 2: Behavioral Analysis

Behavioral analysis examines patterns in a user's transaction history to detect behaviors that are indicative of fraud.

3. Detect Frequent Transactions from the Same User This query identifies users who have made an unusually high number of transactions in the last 24 hours (e.g., more than 5 -- 3. Detect Frequent Transactions from the Same User

SELECT user_id, COUNT(transaction_id) AS transaction_count,

MAX(transaction_date) AS last_transaction

FROM Transactions

WHERE transaction_date > NOW() - INTERVAL 1 DAY

GROUP BY user_id

HAVING transaction_count > 5;

user_id	transaction_count	last_transaction

4. Identify Repeated Transactions to the Same Merchant This query detects users who make multiple transactions to the same merchant, which could indicate repeated fraudulent activity.

SELECT user_id, merchant_name, COUNT(transaction_id) AS transaction_count

FROM Transactions

GROUP BY user_id, merchant_name

HAVING transaction_count > 3;

user_id	merchant_name	transaction_count

Category 3: Pattern Recognition

Pattern recognition identifies specific patterns of activity that are indicative of fraud.

5. Find Suspicious Debit Transactions Exceeding the User's Average This query identifies debit transactions that are significantly higher than the user's average spending, which might indicate fraudulent activity.

SELECT T.user_id, T.transaction_id, T.amount, AVG(T2.amount) AS avg_amount

FROM Transactions T

JOIN Transactions T2 ON T.user_id = T2.user_id

WHERE T.transaction_type = 'debit'

GROUP BY T.user_id, T.transaction_id

*HAVING T.amount > 1.5 * AVG(T2.amount);*

Result Grid | Filter Rows:

user_id	transaction_id	amount	avg_amount
4	7	7000.00	4000.000000

6. Flag Users with Consecutive Declined Transactions This query identifies users who have had three or more consecutive declined transactions, which may indicate potential fraudulent attempts or a compromised account.

SELECT user_id, COUNT(transaction_id) AS declined_count

FROM Transactions

WHERE transaction_type = 'declined'

GROUP BY user_id

HAVING declined_count >= 3;

Result Grid | Filter Rows:

user_id	declined_count

Category 4: Statistical Insights

Statistical insights provide overall transaction data, which can help identify trends and large-scale fraudulent activities.

7. Find Users with the Highest Transaction Amounts This query identifies users who have spent the most, which could uncover high-risk accounts with large transactions.

SELECT user_id, SUM(amount) AS total_spent

FROM Transactions

GROUP BY user_id

ORDER BY total_spent DESC

LIMIT 5;

Result Grid | Filter Rows:

user_id	total_spent
4	8000.00
6	8000.00
7	7300.00
1	7000.00
8	5500.00

8. Merchant-Wise Total Transactions This query provides a summary of total transactions per merchant, which can be useful for identifying merchants with suspiciously high transaction volumes.

```
SELECT merchant_name, COUNT(transaction_id) AS transaction_count,
    SUM(amount) AS total_amount
FROM Transactions
GROUP BY merchant_name
ORDER BY total_amount DESC;
```

Result Grid | Filter Rows:

merchant_name	transaction_count	total_amount
Amazon	2	9500.00
BookMyShow	2	8200.00
Flipkart	2	7500.00
Zappos	2	5500.00
Snapdeal	2	5000.00
Myntra	2	4800.00
Big Bazaar	2	4500.00
Jabong	1	4000.00
Swiggy	1	1000.00

Analysis of Key Fraud Indicators and Detection Queries

Category 1: Anomaly Detection

1. Detect High-Value Transactions (Threshold: $5,000)

Flagged Transactions:

User	Transaction Amount	Merchant	Flag Reason
Pranathi (User 4)	$7,000	BookMyShow	High-value transaction
Sudhesha (User 7)	$5,500	Flipkart	Large single transaction

Observation:

- Pranathi had the highest transaction ($7,000), which is unusual for a single transaction.
- Sudhesha also had a large transaction of $5,500 on Flipkart, which is above the defined threshold.
- These transactions could be legitimate high-value purchases but should be monitored for potential fraud.

2. Transactions Outside Business Hours (9 AM - 9 PM)

User	Amount	Merchant	Transaction Time
Thanmayee	$2,000	Big Bazaar	23:50 (Late Night)
Pranathi	$1,000	Swiggy	00:30 (Midnight)
Abhilash Sharma	$4,500	Amazon	23:10 (Late Night)
Aadhya Sharma	$2,500	Big Bazaar	01:10 (Late Night)

- The transactions happening during late-night or early-morning hours could be flagged as suspicious since they fall outside of typical business hours.
- Transactions such as Pranathi's midnight transaction or Abhilash Sharma's late-night debit may be more prone to fraud, as fraudsters often attempt transactions during odd hours to avoid detection.
- Aadhya Sharma's late-night transaction at 01:10 AM could be indicative of unusual activity.

Category 2: Behavioral Analysis

3. Detect Frequent Transactions from the Same User (More than 5 in 24 Hours)

Query Output: *(No user exceeded 5 transactions in a single day in the dataset.)*

Observation:

- No immediate signs of rapid transaction attempts.
- However, monitoring real-time transaction logs could help identify cases where fraudsters rapidly attempt multiple transactions.

---

4. Identify Repeated Transactions to the Same Merchant (More than 3 Transactions)

Query Output: *(No user made more than 3 transactions at the same merchant.)*

Observation:

- No user exhibited excessive transactions to the same merchant, but this pattern could help detect **account takeover fraud** or **testing of stolen cards** in real-world scenarios.

Category 3: Pattern Recognition

5. Suspicious Debit Transactions Exceeding the User's Average

User	Transaction Amount	Merchant	Flag Reason
Pranathi (User 4)	$7,000	BookMyShow	Unusually high spending
Abhilash Sharma (User 6)	$4,500	Amazon	Higher than typical transactions

Observation:

- Pranathi's single large transaction is an anomaly and could be legitimate or a case of fraudulent ticket booking.
- Abhilash Sharma's $4,500 debit transaction is significantly larger than his usual spending, making it suspicious.

---

6. Flag Users with Consecutive Declined Transactions (3 or More)

Query Output: *(No declined transactions in the dataset.)*

Observation:

- No immediate fraudulent attempts found.
- This pattern is useful for detecting **stolen card testing**, where fraudsters attempt multiple unauthorized transactions.

Category 4: Statistical Insights

7. Users with the Highest Transaction Amounts

User	Total Transaction Amount
Pranathi (User 4)	$8,000
Abhilash Sharma (User 6)	$8,000
Sudhesha (User 7)	$7,300

Observation:

- Pranathi and Abhilash Sharma are the highest spenders, making them potential high-risk accounts.
- Sudhesha is also a high spender, but her transactions are more distributed.
- These users should be monitored closely for unusual activity.

8. Merchant-Wise Total Transactions

Merchant	Total Transaction Volume	Key Observations
Amazon	Highest total transactions	High activity
Flipkart	Multiple transactions	Consistently used
BookMyShow	Largest single transaction	Potential ticketing fraud

Observation:

- Amazon and Flipkart dominate in transaction volume, making them attractive targets for fraudsters.
- BookMyShow had the largest single transaction ($7,000), which could indicate ticketing fraud or misuse of stolen cards.

Test Plan

Since the dataset is **synthetic and small**, some fraud queries may not return meaningful results, but the test plan ensures correctness and scalability for real-world data.

1. Test Objectives

- Validate fraud detection logic.
- Identify **limitations of synthetic data** in query results.

2. Key Test Cases & Limitations

Test Case	Expected Output	Limitation Due to Synthetic Data
High-Value Transaction (amount > 5000)	Transaction flagged	Few transactions exceed threshold
Off-Hours Transaction (TIME(transaction_date) NOT BETWEEN '09:00:00' AND '21:00:00')	Flagged as suspicious	Few off-hours transactions exist
Frequent Transactions (More than 5 per user/day)	User flagged	Not enough repeated transactions
Merchant Fraud (Multiple transactions to the same merchant)	Merchant flagged	Limited merchant transaction variety
Spending Deviation (Debit amount 1.5x user's avg)	Transaction flagged	Small dataset may lack spending patterns

3. Future Enhancements

- Expand dataset for better fraud detection testing.
- Use **real-world data** or **simulate realistic transactions**.
- Optimize fraud queries for large-scale data processing.

Potential Limitations and Assumptions

While the analysis provides valuable insights, certain limitations and assumptions should be considered:

1. Limited Dataset: The dataset used for the analysis is small and may not fully represent broader fraud patterns. Real-world fraud detection systems require large datasets for more accurate and robust analysis.

2. Lack of Real-Time Data: Since the analysis was conducted on historical data, it does not account for real-time events. Fraud detection models need to be dynamic and responsive to live transaction data, which can lead to more immediate and accurate detection.

3. Assumed Thresholds: The thresholds used for fraud detection (such as $5,000 for high-value transactions) are arbitrary and may not apply universally. These values could vary based on industry standards, customer profiles, and risk tolerance.

4. Transaction and Merchant Patterns: The analysis assumes that user behavior and merchant transactions follow standard patterns. However, fraud can often involve sophisticated tactics that deviate from these norms, making some fraudulent activities harder to detect using simple rule-based methods.

Conclusions

The fraud detection analysis of banking transactions has highlighted key patterns in **high-value transactions, unusual timing, and spending behaviors** that may indicate fraudulent activity. Large transactions exceeding **$5,000**, such as those by Pranathi and Abhilash Sharma, require closer monitoring, as they could signify fraud or simply high spending habits. Transactions occurring **late at night or outside regular business hours** are more likely to be suspicious, making them a priority for further review. Behavioral anomalies, like **sudden spikes in spending**, also warrant investigation, as seen in cases of large debit transactions. Additionally, **high-traffic merchants** such as Amazon, Flipkart, and BookMyShow remain prime targets for fraudulent transactions, emphasizing the need for extra scrutiny.

While this analysis provides a **strong foundation for fraud detection**, integrating **real-time monitoring and machine learning** will improve accuracy and responsiveness. Advanced techniques, such as **behavior profiling and external data integration**, can enhance fraud detection systems, making them more adaptive to evolving threats. Continuous **refinement of detection rules and proactive monitoring** are essential to building a **robust and effective fraud prevention system**.