



**BANK**

# **Pioneer Trust Bank**

## **SQL-Based Fraud Detection in Banking**

**Osaroh Ekhonoragbon**

**May 19, 2025**

# Business Overview

- Pioneer Trust Bank is a financial institution facing a surge in fraudulent activities across customer transactions. The existing fraud detection system is inadequate for real-time analysis, leading to financial loss and reduced customer trust.
- This project applies SQL-based analytics on historical transaction data to identify high-risk activities and deliver strategic insights for fraud prevention.
- This demonstrates the potential of structured SQL analysis in driving actionable risk mitigation strategies for the banking industry.

# High Transaction Amounts

The screenshot displays the pgAdmin 4 web interface. On the left, the Object Explorer shows the database structure for 'PioneerTrust\_db', including Schemas (public), Tables (accounts, customers, transaction), and various other database objects. The main panel shows a SQL query being executed against the 'transaction' table. The query is as follows:

```
--understanding the questions
--understanding the tables that you need
--understanding the columns that you Need
--understanding the syntax like aggregation, order by, filter(where, having clause),limit
--to be used

--What are the top 5 highest transaction amounts this month?
--transaction TABLE
--transaction date and amount
--limit and order BY

select amount, transaction_date from transaction
order by amount DESC
limit 5;
```

The results of the query are displayed in the Data Output tab, showing 5 rows of data:

	amount integer	transaction_date date
1	15000	2024-03-02
2	15000	2024-03-02
3	15000	2024-03-03
4	15000	2024-03-02
5	15000	2024-03-03

The status bar at the bottom indicates 'Total rows: 5' and 'Query complete 00:00:00.141'. The current cursor position is at Line 14, Column 9.



# High Transaction Amounts

The screenshot shows the pgAdmin 4 interface with the following components:

- Object Explorer:** Displays the database structure. The 'PioneerTrust\_db' database is selected, showing its schemas (public) and tables (accounts).
- Query Editor:** Contains the following SQL query:

```
--What is the average transaction amount for all transactions this month?
--transaction TABLE
--amount
--average

select round(avg(amount),2) as avgtransaction from transaction;

--Are there transactions above the average limit?
--transaction TABLE
--amount, transactiondate, customerid
--subquery using greather than 6650.00
```
- Data Output:** Shows the result of the query as a table with one row:

avgtransaction
6650.00
- Messages:** Displays the status of the query execution.
- Notifications:** Displays any system notifications.

## A screenshot of a web browser interface. At the top, there's a dark header bar with a square icon and an 'X' button. Below this is a light blue toolbar containing three icons: a vertical ellipsis, a circular arrow (refresh), and a double-headed arrow (full screen). The main area of the browser is white and currently blank. At the bottom, there's a navigation bar with back, forward, and home buttons, followed by a scrollbar on the right side. The status bar at the very bottom shows "32, Col 79".



# Insight: High Transaction Amounts

**Objectives:** Identify unusually large transactions to detect potential fraudulent behavior.

## **Findings:**

- ❖ The top 5 transactions, each valued at ₦15,000, revealed repeated high-value activities.
- ❖ The average transaction was ₦6,650 – transactions above this were flagged.
- ❖ Several transactions exceeded the average transaction limit, indicating high-value outliers.

## **Recommendation:**

- ❖ Set automated alerts for transactions exceeding the 2x average threshold.
- ❖ These thresholds can be tailored to customer type and account profile.

pgAdmin 4

FileObjectToolsEditViewWindowHelp

Object Explorer

Servers (1)

PostgreSQL 17

Databases (3)

Greenspace DB

PioneerTrust\_db

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (3)

accounts

customers

transaction

DashboardPropertiesSQLStatisticsDependenciesDependentsProcessesPioneerTrust\_db case study.sql\*

PioneerTrust\_db/postgres@PostgreSQL 17

Query Query History

```
--Are there customers that made more than 1 flagged transaction?...
--which customer has the highest flagged transactions?
--customers and transaction TABLE
--name, customerid, fraudulentflag
--join
--on customerid

select t.customer_id, name, fraudulent_flag, count(fraudulent_flag) from transaction t
join customers c on t.customer_id = c.customer_id
where fraudulent_flag = 'true'
group by t.customer_id, name, fraudulent_flag
having count(fraudulent_flag) > 1
order by count(fraudulent_flag) desc;
```

Scratch Pad

Data OutputMessagesNotifications

Showing rows: 1 to 9Page No: 1 of 1

	customer_id integer	name character varying (255)	fraudulent_flag boolean	count bigint
1	101	John Doe	true	5
2	104	Alice Brown	true	4
3	116	Hannah Brown	true	3
4	103	Mark Taylor	true	3
5	111	Jake Green	true	3
6	114	Jack Brown	true	3
7	113	Holly White	true	2
8	117	Sophia White	true	2

Total rows: 9Query complete 00:00:00.157CRLFLn 47, Col 38



# Repeat Offenders

The screenshot shows the pgAdmin 4 interface with the following components:

- Object Explorer:** Displays the database structure for 'PioneerTrust\_db', including Schemas (public), Tables (accounts, customers), and various database objects like Aggregates, Collations, Domains, etc.
- Query Editor:** Contains the following SQL query:

```
48
49
50 --How many total flagged transactions occurred this month?...
51 --what dates has the most flagged activity?
52 --transaction TABLE
53 --transaction_date, fraudulent_flag
54 --count
55 --order BY
56
57 select transaction_date, Fraudulent_flag, count(fraudulent_flag) from transaction
58 where fraudulent_flag = 'true'
59 group by transaction_date, Fraudulent_flag
60 order by count(fraudulent_flag) desc;
61
62
```
- Data Output:** Displays the results of the query in a table with 7 rows and 3 columns: transaction\_date, fraudulent\_flag, and count. The data is sorted by count in descending order.
- Status Bar:** Shows 'Total rows: 7' and 'Query complete 00:00:00.169'.

	transaction_date	fraudulent_flag	count
1	2024-03-02	true	7
2	2024-03-05	true	6
3	2024-03-07	true	5
4	2024-03-04	true	4
5	2024-03-03	true	3
6	2024-03-06	true	3
7	2024-03-01	true	2



# Repeat Offenders

The screenshot shows the pgAdmin 4 interface with the following components:

- Object Explorer:** Displays the database structure for 'PioneerTrust\_db', including Schemas (public), Tables (accounts, customers, transaction), and other database objects.
- Query Editor:** Contains a SQL query with line numbers 61 to 75. The query is as follows:

```
61
62
63 --Are flagged transactions more likely to be deposits or withdrawal
64 --transaction TABLE
65 --fraudulent_flag and transaction_type
66 --where, IN, count
67
68 select fraudulent_flag, transaction_type, count(fraudulent_flag) from transaction
69 where fraudulent_flag = 'true'
70 group by fraudulent_flag, transaction_type
71 order by count(fraudulent_flag) desc;
72
73
74 --Which customers have made transactions from more than 2 different locations in a single day
75 ---customer and transaction
```
- Data Output:** Displays the results of the query in a table with 3 columns: fraudulent\_flag, transaction\_type, and count. The results are as follows:

	fraudulent_flag boolean	transaction_type character varying (100)	count bigint
1	true	Deposit	18
2	true	Withdrawal	12
- Status Bar:** Shows 'Total rows: 2', 'Query complete 00:00:00.136', 'CRLF', and 'Ln 71, Col 38'.

# Insight: Repeat Offenders

**Objectives: Detect customers with multiple flagged transactions.**

## **Finding:**

**Multiple customers triggered repeated fraud flags. Notably, one customer (John Doe) had 5 flagged transactions, showing a pattern of repeated suspicious activity.**

**Flagged transactions were more frequent on specific dates, notably March 2<sup>nd</sup>.**

**Deposits were more commonly flagged than withdrawals—highlighting where monitoring should be stricter.**

## **Recommendation:**

**Monitor high-frequency fraud customers with enhanced scrutiny.**

**Implement profiling of frequent offenders with real-time blocking triggers.**

**Review KYC data and transaction histories of flagged deposits and withdrawals.**

# Transactions from Multiple Locations

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure for 'PioneerTrust\_db', including Schemas (public), Tables (accounts, customers, transaction), and other database objects. The main pane shows a SQL query being executed on the 'PioneerTrust\_db/postgres@PostgreSQL 17' connection. The query is designed to find customers who have made transactions from more than 2 different locations in a single day, listing their names, transaction amounts, and the number of distinct locations.

```
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74 --Which customers have made transactions from more than 2 different locations in a single day
75 ---customer and transaction
76 --name, location, amount
77 --join
78 --customerid
79
80 select name, amount, count(distinct merchant_location) as transactionlocation from transaction
81 join customers c on t.customer_id=c.customer_id
82 group by name, amount
83 having count(distinct merchant_location) > 2
84 order by count(merchant_location) desc;
85
86 --Which Location has the most flagged transactions?
```

The Data Output pane shows the results of the query, displaying 3 rows. The columns are name, amount, and transactionlocation. The results are as follows:

	name	amount	transactionlocation
1	Jack Brown	15000	4
2	Mark Taylor	3000	4
3	Holly White	2000	3

Total rows: 3 Query complete 00:00:00.187 CRLF Ln 84, Col 40





# Transactions from Multiple Locations

The screenshot shows the pgAdmin 4 interface with the following components:

- Object Explorer:** Displays the database structure for 'PioneerTrust\_db', including Schemas (public), Tables (accounts, customers, transaction), and other objects like Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, and Schemas (1).
- Query Editor:** Contains a SQL query to find accounts with fraudulent transactions. The query is as follows:

```
93 order by count(fraudulent_flag) desc;
94
95 --Which account is most associated with flagged transactions
96 --customer and transaction table
97 --account_type, fraudulent_type
98 --join
99 --customer_id
100 --count
101
102 select account_type, Fraudulent_flag, count(t.fraudulent_flag) from transaction t
103 join customers c on t.customer_id = c.customer_id
104 where fraudulent_flag = 'true'
105 group by account_type, Fraudulent_flag;
106
107
```
- Data Output:** Shows the results of the query in a table with 2 rows and 3 columns: account\_type, fraudulent\_flag, and count. The results are:

	account_type	fraudulent_flag	count
1	Individual	true	26
2	Business	true	4
- Footer:** Displays 'Total rows: 2', 'Query complete 00:00:00.124', and 'Ln 105, Col 40'.

# Insight: Transactions from Multiple Locations

**Objectives:** Identify customers, account types, and transactions made from different geographical locations within short timeframes.

## **Finding:**

- ❖ Certain users (e.g., Jack Brown) made 4 transactions from different cities within one day.
- ❖ Locations like San Francisco and New York had the highest concentration of flagged transactions.
- ❖ Review KYC data and transaction histories of flagged Individuals.

## **Recommendation:**

- ❖ Introduce geo-fencing and IP-location consistency checks.
- ❖ Alert system should flag abnormal travel-based transaction patterns.
- ❖ Introduce two-factor authentication for transactions occurring from distant or new locations.





# Conclusion & Recommendations

This project revealed critical insights using SQL on banking transaction data:

- ❖ High-value transactions often exceed normal patterns and require threshold-based alerts.
- ❖ Repeat offenders signal the need for automated profile-based blocking rules.
- ❖ Multi-location transactions suggest account compromise and warrant geographic checks.
- ❖ By combining these strategies, Pioneer Trust Bank can improve fraud detection, reduce losses, and restore customer trust.

# Conclusion & Recommendations

The SQL-based analysis uncovered clear patterns of fraud risk:

- ❖ High-value and repeat transactions are key fraud indicators.
- ❖ Certain locations and individual account types are more fraud-prone.
- ❖ Time-based and behavior-based monitoring is crucial for proactive fraud detection.

# Next Steps for Management

- ❖ Integrate automated alert systems for high-risk behaviors.
- ❖ Enhance fraud prevention systems using geolocation and behavioral analytics.
- ❖ Invest in staff training for fraud pattern recognition and anomaly detection.





# FRAUD PREVENTION

