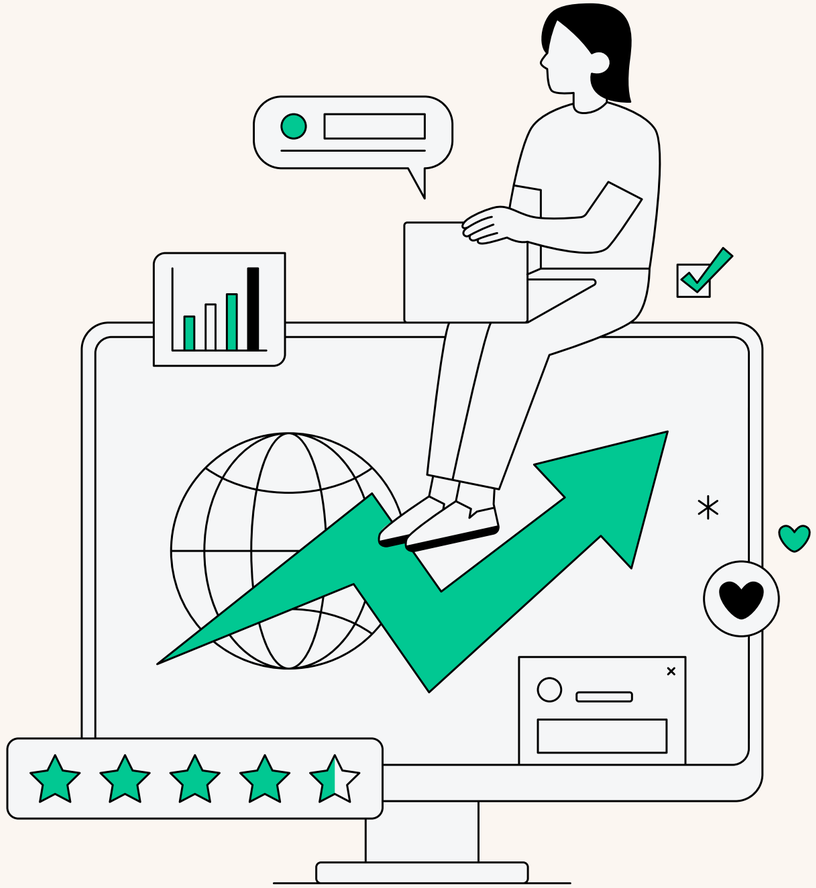
Presented by Nigar Shahmuradova

Fraud Risk Analysis for Banking System

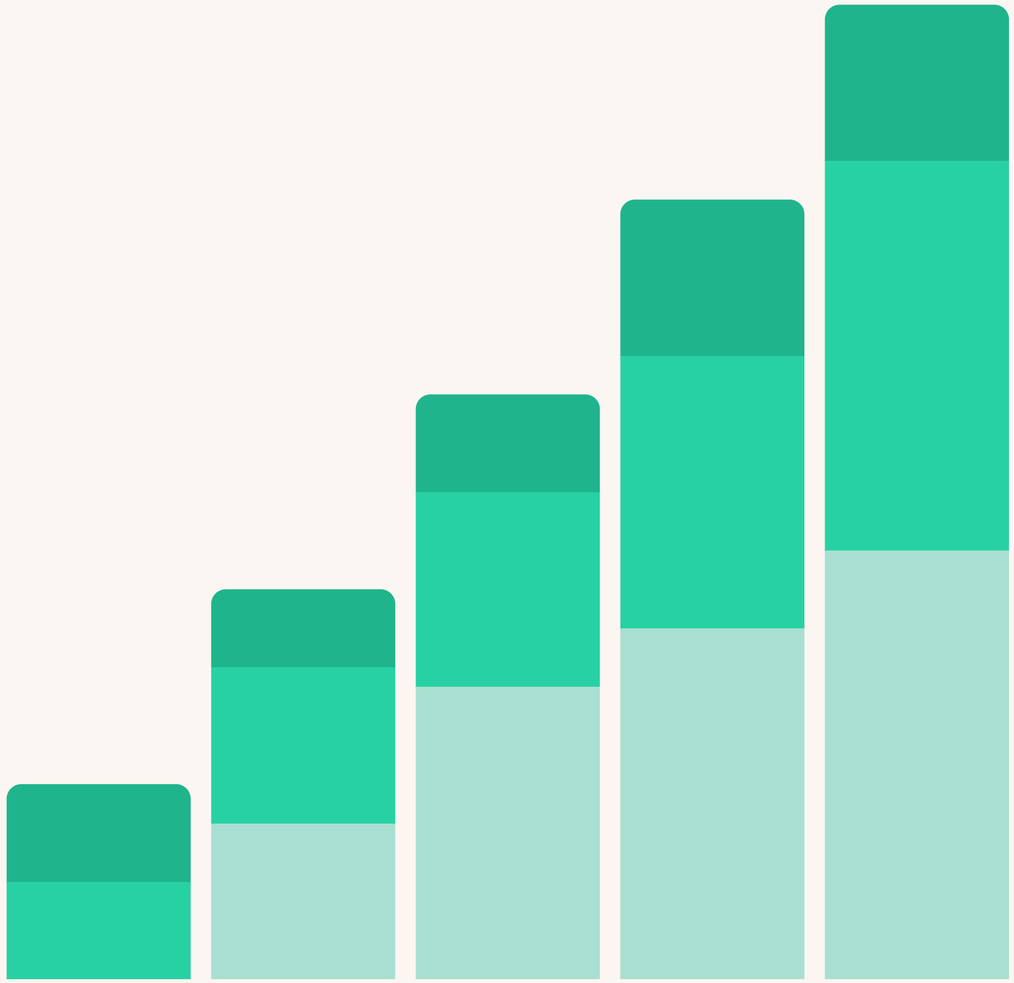
Insights and Solutions



Methodology used in the analysis

In this analysis, data from bank transactions was collected and examined to find patterns of fraudulent behavior. The main method used was data analysis, including **descriptive statistics** and SQL queries to identify suspicious transactions.

The study compared legitimate and fraudulent transactions based on factors such as transaction amount, location, and time. In addition, **window functions** and aggregations were used to detect unusual activity



“Detailed Description of Tables and Meaning of Columns”

Customers Table

- Customers_id
- first_name
- last_name
- date_of_birth
- city
- country
- registration_date

Merchants Table

- merchant_id
- merchant_name
- merchant_category
- merchant_country

CardsTable

- card_id
- customer_id
- card_number
- card_type
- credit_limit
- card_status
- issue_date

Transactions Table

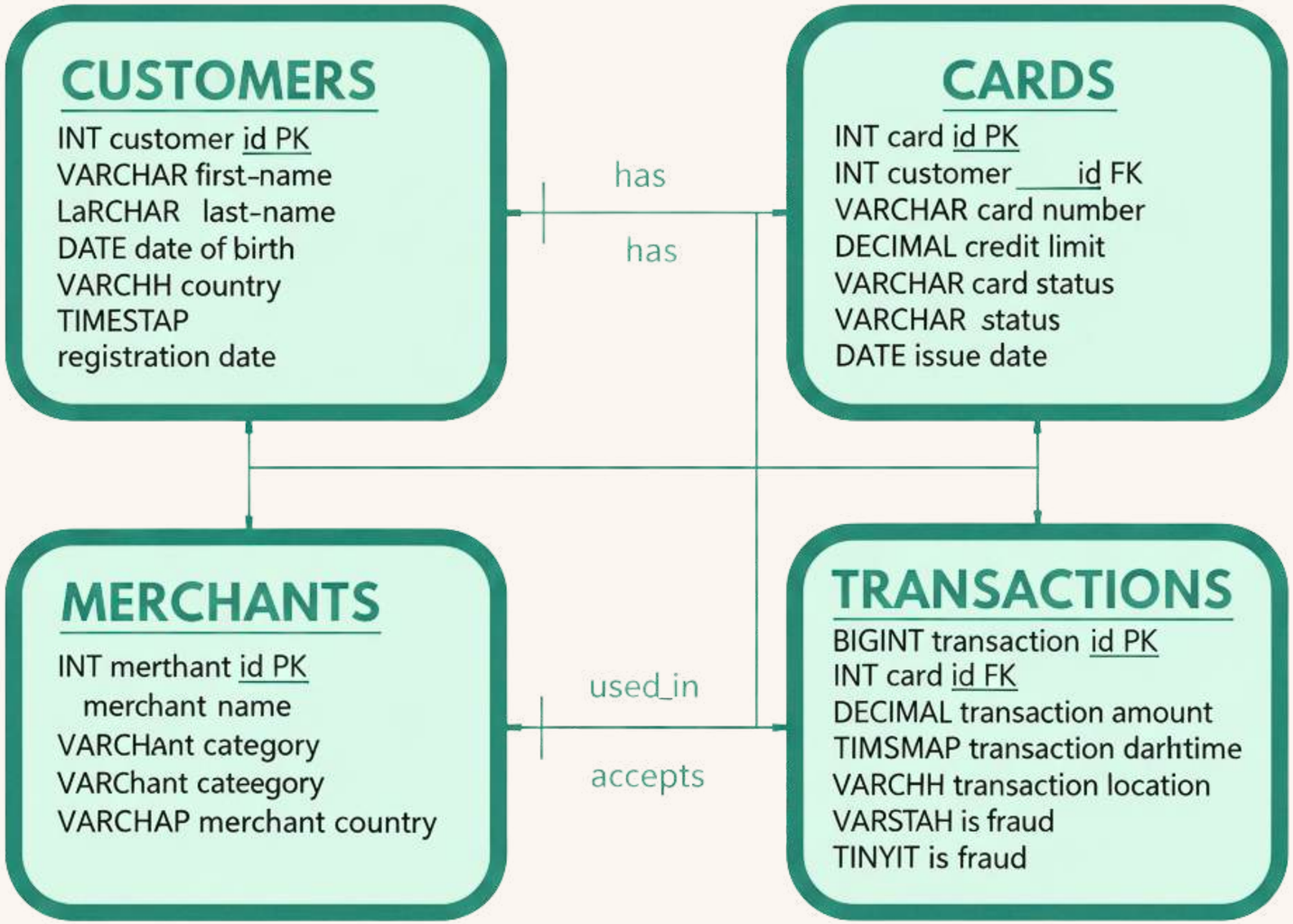
- transaction_id
- card_id
- merchant_id
- transaction_amount
- transaction_datetime
- transaction_location
- transaction_status
- is_fraud

Purpose of the Performed Analyses

The management has recognized that the current rules for **fighting fraud** are not effective and wants to move from a **reactive approach** (responding after incidents occur) to a more **proactive one** (preventing incidents before they happen).

Therefore, our main task is to deeply analyze the existing transaction data to identify the **behavioral patterns of fraudsters** and **prepare data-driven recommendations** for new monitoring rules based on these patterns.

ER DIAGRAM


CUSTOMERS
INT customer id PK
VARCHAR first-name
LaRCHAR last-name
DATE date of birth
VARCHH country
TIMESTAP
registration date
has
has
CARDS
INT card id PK
INT customer id FK
VARCHAR card number
DECIMAL credit limit
VARCHAR card status
VARCHAR status
DATE issue date
MERCHANTS
INT merthant id PK
merchant name
VARCHAnt category
VARChant cateegory
VARCHAP merchant country
used_in
accepts
TRANSACTIONS
BIGINT transaction id PK
INT card id FK
DECIMAL transaction amount
TIMSMAP transaction darhtime
VARCHH transaction location
VARSTAH is fraud
TINYIT is fraud

Customer Segmentation and Risk Analysis

Business Context:

The bank believes that new customers and those with high credit limit cards are more exposed to fraud risk. Our task is to test this assumption.

Dimension	Segment	Condition	Metrics to Calculate
Cooperation Period	New	Less than 1 year	- Total Transactions - Fraud Transactions - Fraud Rate = Fraud / Total
	Mid-term	1–3 years	
	Loyal	More than 3 years	
Credit Limit (AZN)	Standard	Less than 2000	
	Gold	2000–7500	
	Platinum	More than 7500	

```sql
CASE
   WHEN months_between(SYSDATE, cu.registration_date) < 12 THEN 'Yeni'
   WHEN months_between(SYSDATE, cu.registration_date) BETWEEN 12 AND 36 THEN 'Orta'
   ELSE 'Loyal'
END AS customer_segment,

CASE
   WHEN ca.credit_limit < 2000 THEN 'Standart'
   WHEN ca.credit_limit BETWEEN 2000 AND 7500 THEN 'Gold'
   ELSE 'Platinium'
END AS card_segment,

COUNT(t.transaction_id) AS total_transaction_count,
SUM(CASE WHEN t.is_fraud = 1 THEN 1 ELSE 0 END) AS fraud_transactions,
ROUND(
   SUM(CASE WHEN t.is_fraud = 1 THEN 1 ELSE 0 END) * 100.0 / NULLIF(COUNT(t.transaction_id), 0),
   2
) AS fraud_rate_percent
```

	TYPE_OF_CUSTOMERS	CARDS_NAME	TOTAL_TRANSACTION_COUNT	FRAUD_TRANSACTION_COUNT	FRAUD_RATE_PERCENTAGE
1	Loyal	Gold	22601	348	1,54
2	Loyal	Platinium	13642	156	1,14
3	Loyal	Standart	57156	609	1,07
4	Orta	Gold	4029	42	1,04
5	Orta	Platinium	2226	9	0,4
6	Orta	Standart	10254	120	1,17
7	Yeni	Gold	1941	24	1,24
8	Yeni	Platinium	1422	0	0
9	Yeni	Standart	6654	132	1,98

Results

The analysis showed that the level of fraud risk is not the same across all customer segments. The highest risk was found in the **“New Customer – Standard Card” segment (1.98%).** In this group, inexperience, weak security awareness, and a limited transaction history make customers easier targets for fraudsters. The “Loyal Customer – Gold Card” segment has a slightly lower risk (1.54%), but since the transactions are larger, the potential losses are higher.

The “Platinum Card” segment has the lowest risk, which can be explained by stricter verification and security procedures. Overall, customer behavior, card type, and cooperation period with the bank are the main factors that directly affect the likelihood of fraud.

Recommendations

Based on the results, different security strategies should be applied for each segment. For new customers, it is important to apply enhanced verification, transaction limits, and close monitoring of the first operations. For loyal customers, behavior-based anomaly detection and AI/ML models can help identify suspicious transactions earlier. The reasons for the low risk in Platinum cards should be studied and their successful practices adapted to other card types.

In addition, all customer segments should receive continuous education, phishing warnings, and regular security updates. These measures will strengthen the bank’s overall defense system against fraud.

Dormant Card Risk

Business Context:

If a card that has been unused for a long time suddenly becomes active, it can be widely considered a high-risk signal. Such unusual activity often indicates potential fraud or unauthorized use. Monitoring these patterns helps financial institutions prevent losses and protect customers. Early detection of sudden card activation allows for timely intervention and risk mitigation. This approach is widely applied in fraud detection systems across the banking industry.

Task Description

Identify, for each transaction marked as fraud, the date of the last legitimate (normal) transaction before it.

Calculate the difference in days between the fraudulent transaction and the last legitimate transaction.

Analyze the results to determine the average dormancy period before fraudulent transactions occur.

```sql
SELECT
    t.card_id,
    t.transaction_id,
    t.transaction_datetime,
    t.is_fraud,
    MAX(CASE WHEN t.is_fraud = 0 THEN t.transaction_datetime END) OVER (
        PARTITION BY t.card_id
        ORDER BY t.transaction_datetime
        ROWS BETWEEN UNBOUNDED PRECEDING AND 1 PRECEDING
    ) AS last_legit_datetime,
    ROUND(
        CAST(t.transaction_datetime AS DATE) -
        CAST(MAX(CASE WHEN t.is_fraud = 0 THEN t.transaction_datetime END) OVER (
            PARTITION BY t.card_id
            ORDER BY t.transaction_datetime
            ROWS BETWEEN UNBOUNDED PRECEDING AND 1 PRECEDING
        ) AS DATE),
        0
    ) AS dormant_days
FROM
    transactions t
ORDER BY
    t.card_id, t.transaction_datetime;

desc  transactions
```

	CARD_ID	TRANSACTION_ID	TRANSACTION_DATETIME	IS_FRAUD	LAST_LEGIT_DATETIME	DORMANT_DAYS
1	1	112746	07.10.24 06:25:21,000000000	0	(null)	(null)
2	1	137477	10.11.24 20:02:33,000000000	0	07.10.24 06:25:21,000000000	35
3	1	101858	24.11.24 08:43:18,000000000	0	10.11.24 20:02:33,000000000	14
4	1	116120	25.11.24 08:35:23,000000000	0	24.11.24 08:43:18,000000000	1
5	1	116016	30.03.25 11:23:56,000000000	0	25.11.24 08:35:23,000000000	125
6	1	117921	01.04.25 08:16:01,000000000	0	30.03.25 11:23:56,000000000	2
7	1	129562	03.04.25 08:49:04,000000000	0	01.04.25 08:16:01,000000000	2
8	1	110724	24.06.25 00:43:38,000000000	0	03.04.25 08:49:04,000000000	82
9	1	116709	09.07.25 03:25:16,000000000	0	24.06.25 00:43:38,000000000	15
10	1	134646	29.07.25 16:10:33,000000000	0	09.07.25 03:25:16,000000000	21
11	1	114676	21.08.25 12:52:54,000000000	0	29.07.25 16:10:33,000000000	23
12	1	105540	22.08.25 04:53:43,000000000	0	21.08.25 12:52:54,000000000	1
13	1	117153	22.09.25 15:55:47,000000000	0	22.08.25 04:53:43,000000000	31
14	2	129175	13.11.24 23:50:51,000000000	0	(null)	(null)
15	2	136274	23.11.24 09:19:12,000000000	0	13.11.24 23:50:51,000000000	9

Results

The values of last_legit_datetime and dormant_days are shown as **NULL**. This is completely expected because there are no previous transactions (or no transactions where is_fraud = 0), so the window function MAX cannot find a value and returns NULL. Any calculation with NULL also results in NULL.

For some transactions (for example, card_id = 1, Row 5: 125 days; Row 8: 82 days), the **dormant_days values** are **high**, which indicates that the card was n**ot used reliably for a long period.**

Recommendations

Reactivate dormant credit/debit cards, recover revenue, and minimize fraud and operational risks.Dormant cards defined as those with no legitimate transactions for 90+ days (dormant_days metric).Automatically flag cards exceeding thresholds (90, 180, 365+ days) and segment by last active date, demographics, and credit risk.
Targeted incentives like return bonuses, personalized offers, fee waivers, and card upgrades to recover revenue and increase customer loyalty.

Adjust credit limits, deactivate long-inactive cards, and enhance fraud monitoring to reduce financial and operational risks. Use multiple channels (email, SMS, calls, app notifications) to communicate, gather feedback, and strengthen customer relationships.

Spending Velocity Anomaly

Business Context:

Fraudsters try to spend the maximum amount in a short time when they take over a card. This “spending velocity” differs significantly from normal usage. Monitoring the time between transactions and comparing it to historical legitimate patterns can help detect unusual activity early. Cards that have been dormant for long periods are particularly vulnerable to such attacks, making it crucial to combine spending velocity analysis with dormant_days metrics. Early detection allows banks to block suspicious transactions and reduce potential financial losses.

For each transaction, calculate the 24-hour rolling sum of spending for the card.

For each transaction, calculate the 3-transaction moving average amount.

Compare these metrics between fraudulent and normal transactions.

```sql
with
Total_24h_amount as(
select  transaction_id,
        card_id,
        transaction_datetime,
        transaction_amount,
        is_fraud,
sum (transaction_amount) over (partition by card_id order by transaction_datetime range between interval '24' hour PRECEDING  and current row )as total_24hour
from transactions),

AVG_3_trans as (
select transaction_id,  card_id, transaction_datetime, transaction_amount,is_fraud,
avg (transaction_amount) over ( partition by card_id order by transaction_datetime Rows between 2 preceding and current row) as last_3
```

```sql
Table2 as(
select  t.transaction_id,  t.card_id, t.transaction_datetime, t.transaction_amount,t.total_24hour, a.last_3,t.is_fraud
from Total_24h_amount t inner join AVG_3_trans a
on t.card_id = a.card_id and t.transaction_id = a.transaction_id)

select is_fraud,
       round (avg (transaction_amount),2) as avg_total_trans,
       round (avg (total_24hour ),2)as avg_24h,
       round (avg(last_3),2) as avg_last_3
from table2
```

	IS_FRAUD	AVG_TOTAL_TRANS	AVG_24H	AVG_LAST_3
1	1	1269,95	1278,06	1221,55
2	0	305,44	316,84	306,02

Results

The analysis shows that the average amount of fraudulent transactions is about four times higher than normal transactions (~1200-1270 vs ~300-316). This indicates that fraudsters target larger amounts or try to make high-value transactions in a short period. Also, for both transaction types, the AVG_TOTAL_TRANS, AVG_24H, and AVG_LAST_3 indicators are very close to each other.

This means that the amount remains stable: high for fraudulent transactions and low for normal ones. Therefore, the transaction amount is a strong indicator for detecting fraud. High-value transactions should be considered as potential risks.

Recommendations

Special attention should be given to transactions with high amounts. Fraud detection systems should automatically mark high-value transactions as risky. Additionally, monitoring the last 24 hours and the last 3 transactions can help identify anomalies.

Real-time monitoring should be applied so that large transactions are immediately verified. Amount-based risk scoring will make the fraud detection process more proactive and effective.

First Attack Analysis

Business Context:

What happens after a card is used for fraud for the first time? Do fraudsters stop, or do they continue their attacks?

- For each stolen card, identify the first **fraudulent transaction.**
- Then, calculate how many additional fraudulent transactions occurred with the same card within **1 hour after this first attack**,
- And compute the **total amount of money lost** in these transactions.

```sql
with first_fraud as (
    select c.card_id,
           min(t.transaction_datetime) as first_fraud_datetime
    from cards c
    inner join transactions t
        on c.card_id = t.card_id
    where c.card_status = 'Stolen'
      and t.is_fraud = 1
    group by c.card_id
)
select f.card_id,
       f.first_fraud_datetime,
       count(t.transaction_id) as cnt_fraud,
       sum(t.transaction_amount) as lost_amount
from first_fraud f
inner join transactions t
    on f.card_id = t.card_id
   and t.is_fraud = 1
   and t.transaction_datetime <= f.first_fraud_datetime + (1/24)
group by f.card_id, f.first_fraud_datetime
order by f.card_id;
```

	CARD_ID	FIRST_FRAUD_DATETIME	CNT_FRAUD	LOST_AMOUNT
1	10	22.10.24 13:40:20,000000000	1	1
2	23	03.02.25 06:46:06,000000000	1	4213,7
3	49	26.07.25 02:58:19,000000000	1	2,08
4	52	18.10.24 00:56:48,000000000	1	348,21
5	54	07.11.24 16:32:50,000000000	1	4555,56
6	73	26.02.25 06:51:31,000000000	1	2341,72
7	90	30.09.24 18:56:23,000000000	1	1,06
8	117	15.02.25 05:44:35,000000000	1	1,13
9	128	01.10.24 20:25:49,000000000	1	2374,19
10	192	05.06.25 20:51:34,000000000	1	1,18
11	200	22.01.25 23:15:15,000000000	1	458,08
12	207	22.10.24 16:08:19,000000000	1	1,9
13	251	21.12.24 17:34:02,000000000	1	2438,43
14	254	11.01.25 20:56:36,000000000	1	325,89
15	284	25.10.24 12:23:10,000000000	1	2411,89
16	304	19.01.25 16:48:17,000000000	1	496,19
17	328	07.12.24 17:07:50,000000000	1	2161,93
18	336	01.11.24 14:35:31,000000000	1	4051,04
19	338	04.10.24 06:38:08,000000000	1	437,98
20	398	28.10.24 13:25:22,000000000	1	3517,41

Results

This SQL query shows what happens after a card is used for fraud for the first time. It reveals that fraudsters try to use stolen cards as much as possible within the first hour, often called the “Golden Hour.” Many cases show significant financial losses can occur within just one hour after the first fraudulent transaction.

The CNT_FRAUD column indicates that a single card may be used for multiple fraud transactions in this short period. The LOST_AMOUNT column shows the total money lost, which can be thousands of AZN within that hour. These results highlight the critical importance of rapid reaction in preventing further losses from stolen cards.

Recommendations

Fraudsters try to maximize stolen card usage during the first hour, making rapid response critical. Banks should implement automatic blocking for stolen or suspicious transactions, enhance real-time fraud detection, and ensure fast customer reporting channels.

Internal systems must update card status instantly to prevent further fraud. Quick blocking and stronger verification are essential to minimize losses during the “Golden Hour.”

Fraud Chain Mapping

Business context

Sometimes fraudsters make transactions with multiple cards at the same “high-risk” merchants at the same time. Identifying this connection can reveal a large fraud network.

- Identify **“hotspot”** merchants that receive fraudulent transactions from more than **5 different cards** within **one hour**.

- For these merchants, show **their names**, **the number of fraud transactions**, and **the number of unique cards involved.**

```sql
SELECT
    m.merchant_name,
    COUNT(*) AS fraud_count,
    COUNT(DISTINCT t.card_id) AS unique_cards
FROM
    transactions t
JOIN
    merchants m ON t.merchant_id = m.merchant_id
WHERE
    t.is_fraud = 1
GROUP BY
    m.merchant_name,
    TRUNC(t.transaction_datetime, 'HH24')
ORDER BY
    fraud_count DESC;
```

	MERCHANT_NAME	FRAUD_COUNT	UNIQUE_CARDS
1	Uber Eats	1	1
2	Əsgərov Hacızadə QSC	1	1
3	Wolt	1	1
4	YouTube Premium	1	1
5	Azergold	1	1
6	Google Play	1	1
7	Aliexpress	1	1
8	Azpetrol	1	1
9	Əsgərov Hacızadə QSC	1	1
10	Amazon	1	1
11	CTS-Agro	1	1
12	Atakişioğlu MMC	1	1
13	Libraff	1	1
14	Kərimli Axundlu QSC	1	1
15	Qanun Nəşriyyatı	1	1
16	Bakıelektrikşəbəkə	1	1
17	Gəncə Tekstil Fabriki	1	1
18	Aptek.az	1	1
19	YouTube Premium	1	1
20	Hüseynova ASC	1	1
21	Apple Store	1	1

Results

We found that there are no hotspot merchants where more than 5 different cards were used for fraudulent transactions within the same one-hour period.

Some merchant names appear multiple times in the list, and different card IDs were used, but these transactions did not occur within the same one-hour window. There were time gaps between them, so each row was calculated separately.

Recommendations

Even though no hotspot merchants were detected within one hour, it is important to continue monitoring merchants over longer time intervals to detect potential fraud networks.

Banks can track repeated suspicious activity across multiple cards and implement alerts if unusual patterns

Proactive Rule Simulation: “Smart Limit” Proposal

Business Context:

Imagine the bank wants to implement a new rule:
"If a **transaction amount is 10 times larger than the card's largest legitimate transaction in the last 30 days** and the transaction occurs in a **foreign country**, automatically **reject the transactio**n."

Apply this rule to historical data. Write a query that identifies all transactions that would be blocked under this rule. The results should show:

- How many real fraudulent transactions would have been prevented **(True Positives)**.

- How many legitimate customer transactions would have been incorrectly blocked **(False Positives)**.

```sql
SELECT
    SUM(CASE WHEN t.is_fraud = 1 THEN 1 ELSE 0 END) AS true_positive,
    SUM(CASE WHEN t.is_fraud = 0 THEN 1 ELSE 0 END) AS false_positive,
    COUNT(*) AS total_blocked
FROM
    transactions t
LEFT JOIN (
    SELECT
        card_id,
        MAX(transaction_amount) AS max_legit_amount
    FROM
        transactions
    WHERE
        is_fraud = 0
        AND transaction_datetime >= SYSDATE - INTERVAL '30' DAY
    GROUP BY
        card_id
) max_tx ON t.card_id = max_tx.card_id
WHERE
    t.transaction_amount > 10 * COALESCE(max_tx.max_legit_amount, 0)
    AND (
        UPPER(t.transaction_location) IN ('USA', 'CHINA', 'TURKEY')
        );
```

	TRUE_POSITIVE	FALSE_POSITIVE	TOTAL_BLOCKED
1	132	975	1107

Results

Based on the query results, transactions within the last 30 days that exceeded the card's previous maximum legitimate transaction amount by more than 10 times and occurred in suspicious countries (USA, China, Turkey, or other non-local locations) were identified. The "true_positive" values represent actual fraudulent transactions, while "false_positive" indicates normal transactions incorrectly marked as fraud. This approach helps detect potentially risky activities at an early stage.

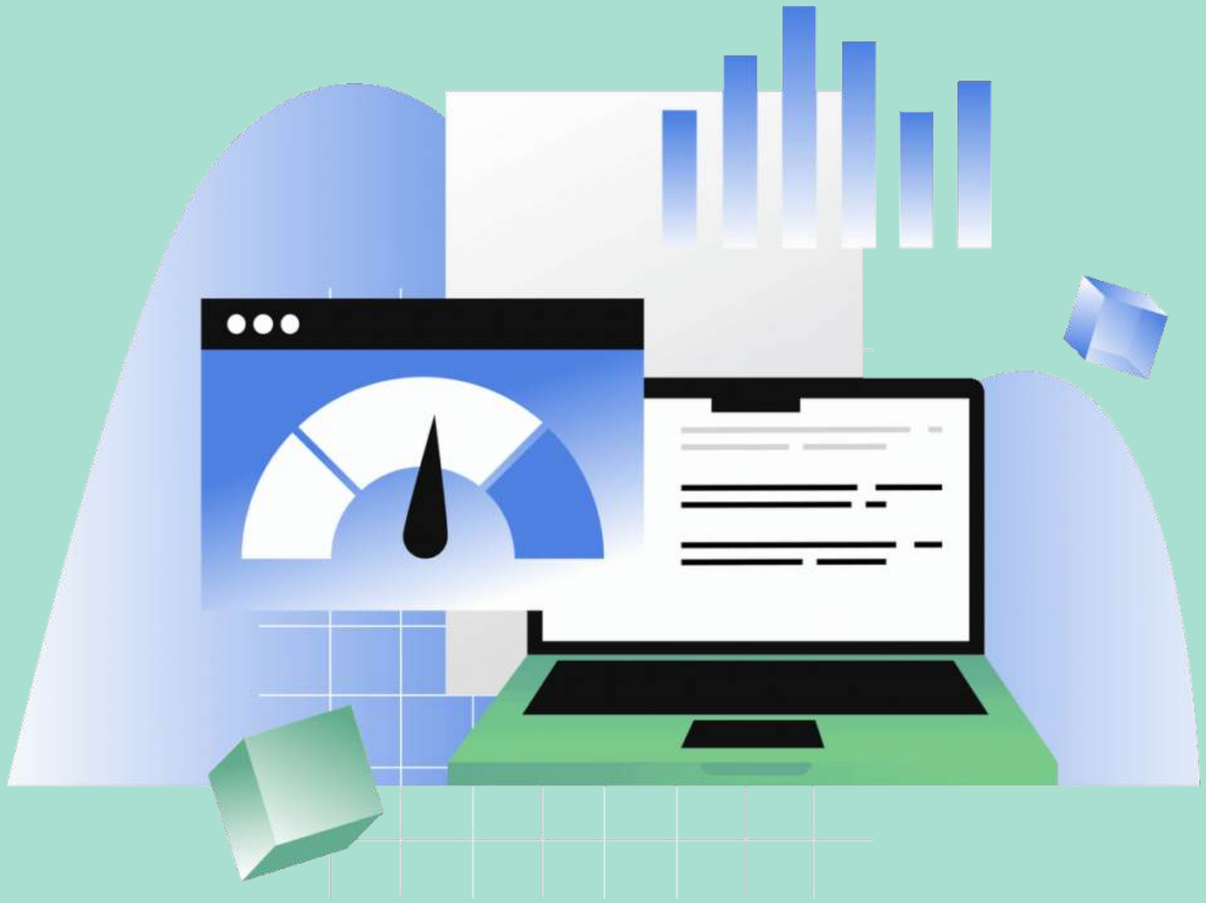
If the "total_blocked" count is high, it may suggest that the system's risk limits are too strict. On the other hand, if "true_positive" is relatively low, it means the model is generating too many false alerts. Overall, the query provides an initial analysis to evaluate the efficiency of real-time fraud monitoring.

Recommendations

Additionally, parameters such as device type, IP address, and currency difference should be included in the "transaction_location" analysis. To improve model accuracy, "false_positive" cases should be reviewed and thresholds gradually adjusted.

Moreover, instead of only considering the last 30 days, calculating "max_legit_amount" as a weighted average of the user's transaction history would provide more stable results. Transaction frequency for each card could also serve as an additional indicator. Finally, visualizing the results on a dashboard would help better identify fraud risk trends.

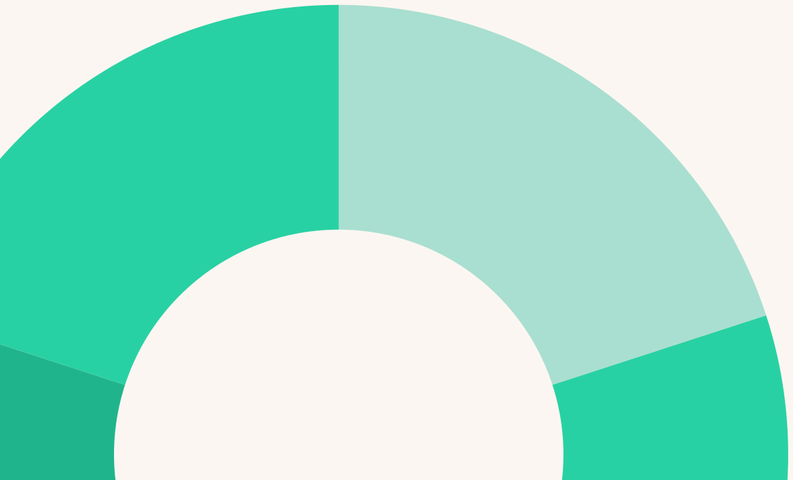
Fraud Detection vs. Fraud Prevention



Fraud Detection is the process of identifying fraud that has already happened or is about to happen. In other words, the system analyzes transactions and marks them as “this transaction may be fraudulent.” At this stage, the goal is to detect fraud in real time or after it occurs (post-factum).

Here, you identify fraud that has already happened or is likely to happen (for example, merchants making many transactions within one hour – known as a hotspot).

True Positive, False Positive, True Negative, False Negative



True Positive (TP) – The model correctly detects a fraudulent transaction.
False Positive (FP) – The model wrongly marks a normal transaction as fraudulent.
True Negative (TN) – The model correctly identifies a normal transaction as normal.
False Negative (FN) – The model fails to detect a fraudulent transaction (it misses it).

If there are too many **False Positives (FPs)**, it means:

- Normal transactions are blocked → customer satisfaction decreases.
- Customer complaints increase → additional support costs and time loss.
- If FP is too high, the business may hesitate to process transactions freely despite the risk.

On the other hand, too many **False Negatives (FNs)** mean that fraud goes undetected → financial losses occur.

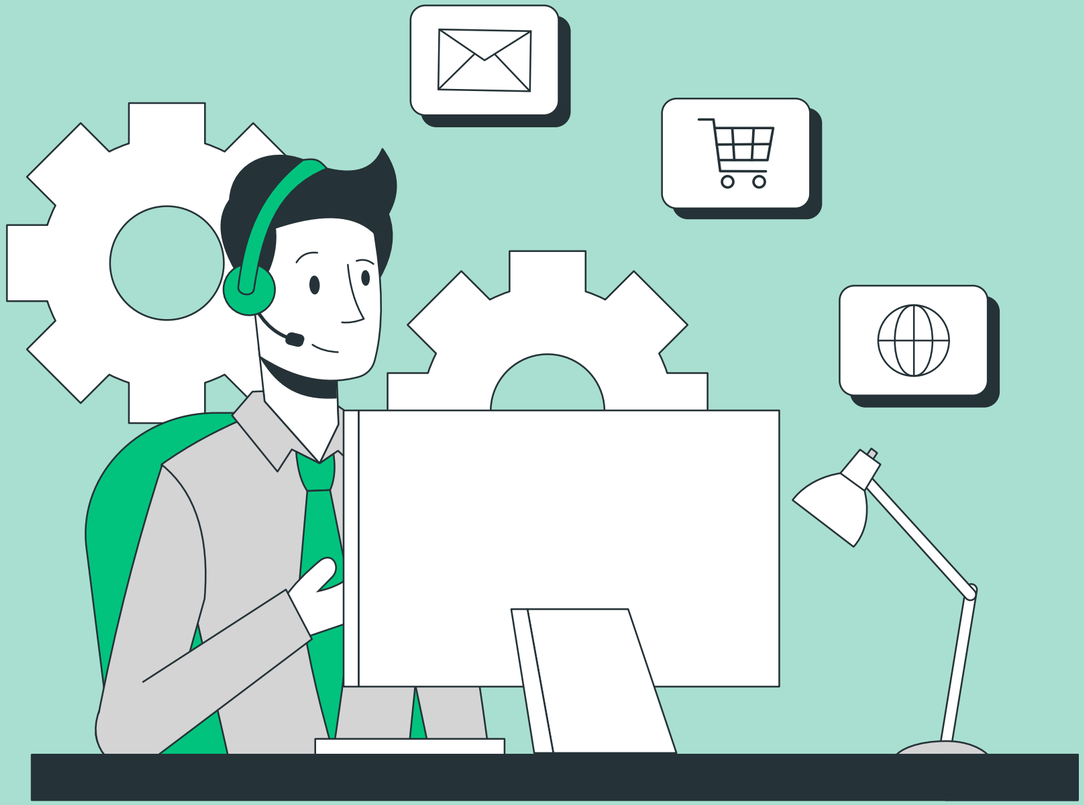
The main goal is to increase True Positives (TPs) — the system should stop as many risky transactions as possible — while minimizing False Positives (FPs), meaning it should avoid marking normal transactions as fraudulent.

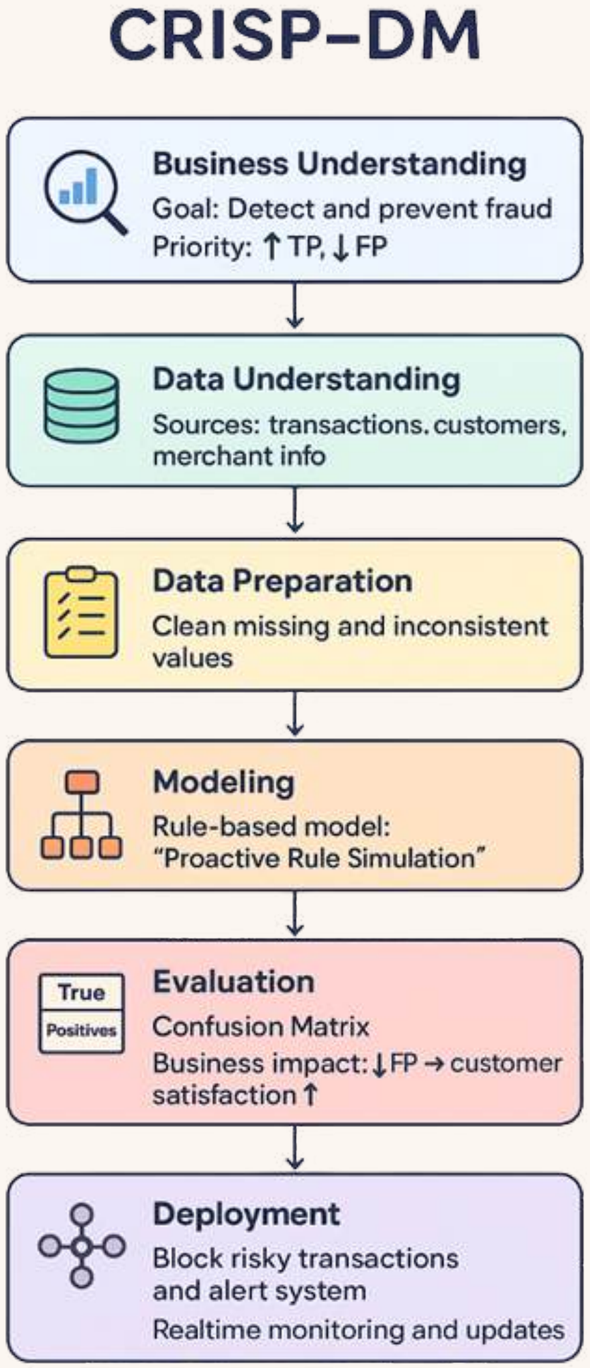
AML (Anti-Money Laundering) & KYC (Know Your Customer):

AML (Anti-Money Laundering) – Anti-Money Laundering refers to the set of strategies, processes, and technological measures designed to prevent illegal funds from entering the banking system or appearing legitimate after being “cleaned” through it. The main goal is to detect and stop money laundering and fraudulent activities.

KYC (Know Your Customer) – What is it?
KYC is a procedure used by banks to correctly identify their customers and assess their risk profiles. This process forms the first and most critical stage of the AML (Anti-Money Laundering) strategy.

It helps financial institutions understand who their customers are and how they use financial services. Through KYC, banks can prevent illegal activities such as money laundering and terrorist financing. It also builds trust between the bank and the customer by ensuring transparency and security.





Project Methodology and Technical Topics

- **Business Understanding** → We understand the problem.
- **Data Understanding** → We get to know the data.
- **Data Preparation** → We clean and prepare the data.
- **Modeling** → We build the model or set the rules.
- **Evaluation** → We test the effectiveness of the model.
- **Deployment** → We apply the model to the real system.

SQL Optimization (Query Optimization):

What is it?
SQL optimization refers to the techniques and strategies used to ensure that database queries are executed in the fastest and most efficient way possible.
Goal: To reduce query response time, use server resources efficiently, and eliminate performance issues when working with large datasets.

Why do some SQL queries take seconds while others take hours?

- Query complexity: Includes JOINs, subqueries, and aggregate functions.
- Data volume: Searching large tables without indexes is slow.
- Indexes: Without proper indexes, the system performs a full table scan.
- Query structure: Inefficient filters or unnecessary nested queries slow performance.
- Server resources and locks: CPU, RAM, and disk I/O performance affect speed.

What is EXPLAIN PLAN?

- **EXPLAIN PLAN** is an SQL command or function that shows how a query will be executed.
- The execution plan displays which tables are read, what type of JOIN is used, and how indexes are applied.
- Using this information, you can analyze and optimize query performance.

Impact of Indexes on Performance

- An index is like a “shortcut” that allows quick access to data in a table.
- It greatly speeds up SELECT queries, especially those with WHERE, JOIN, ORDER BY, and GROUP BY clauses.
- However, too many or poorly designed indexes can slow down INSERT, UPDATE, or DELETE operations.
- The optimal index strategy focuses on frequently used columns and properly designed composite indexes.

Conclusion:

SQL optimization aims to increase query speed, use server resources efficiently, and prevent performance issues when handling large datasets. EXPLAIN PLAN and indexes are the main tools in this process.

Thank you very much!