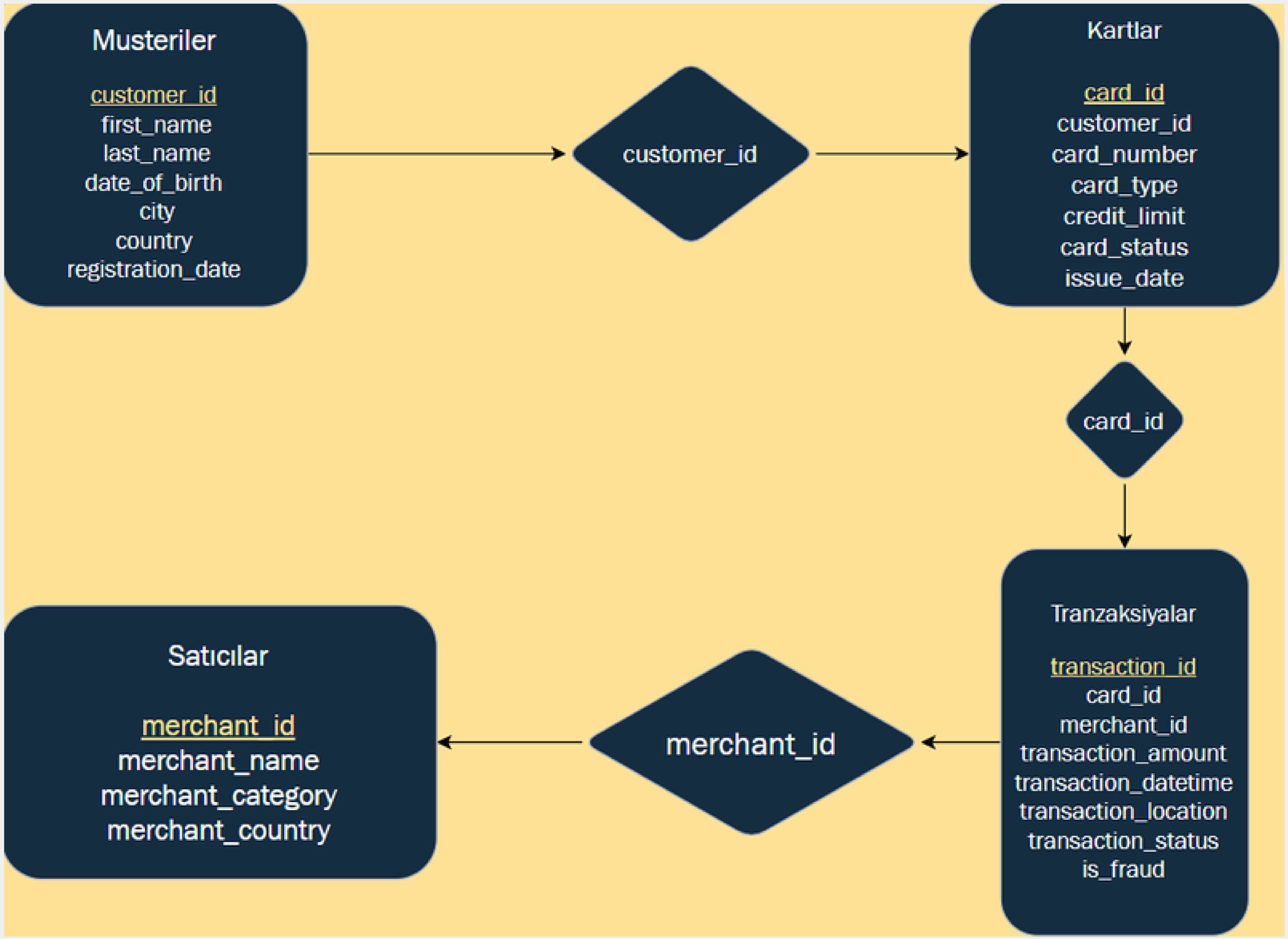
Fraud Risk Analysis

Dataset for the Azerbaijani Banking System

Table of Contents

Dataset


Musteriler
customer_id
first_name
last_name
date_of_birth
city
country
registration_date
customer_id
Kartlar
card_id
customer_id
card_number
card_type
credit_limit
card_status
issue_date
card_id
Tranzaksiyalar
transaction_id
card_id
merchant_id
transaction_amount
transaction_datetime
transaction_location
transaction_status
is_fraud
merchant_id
Satıcılar
merchant_id
merchant_name
merchant_category
merchant_country

Customer Segmentation and Risk Analysis

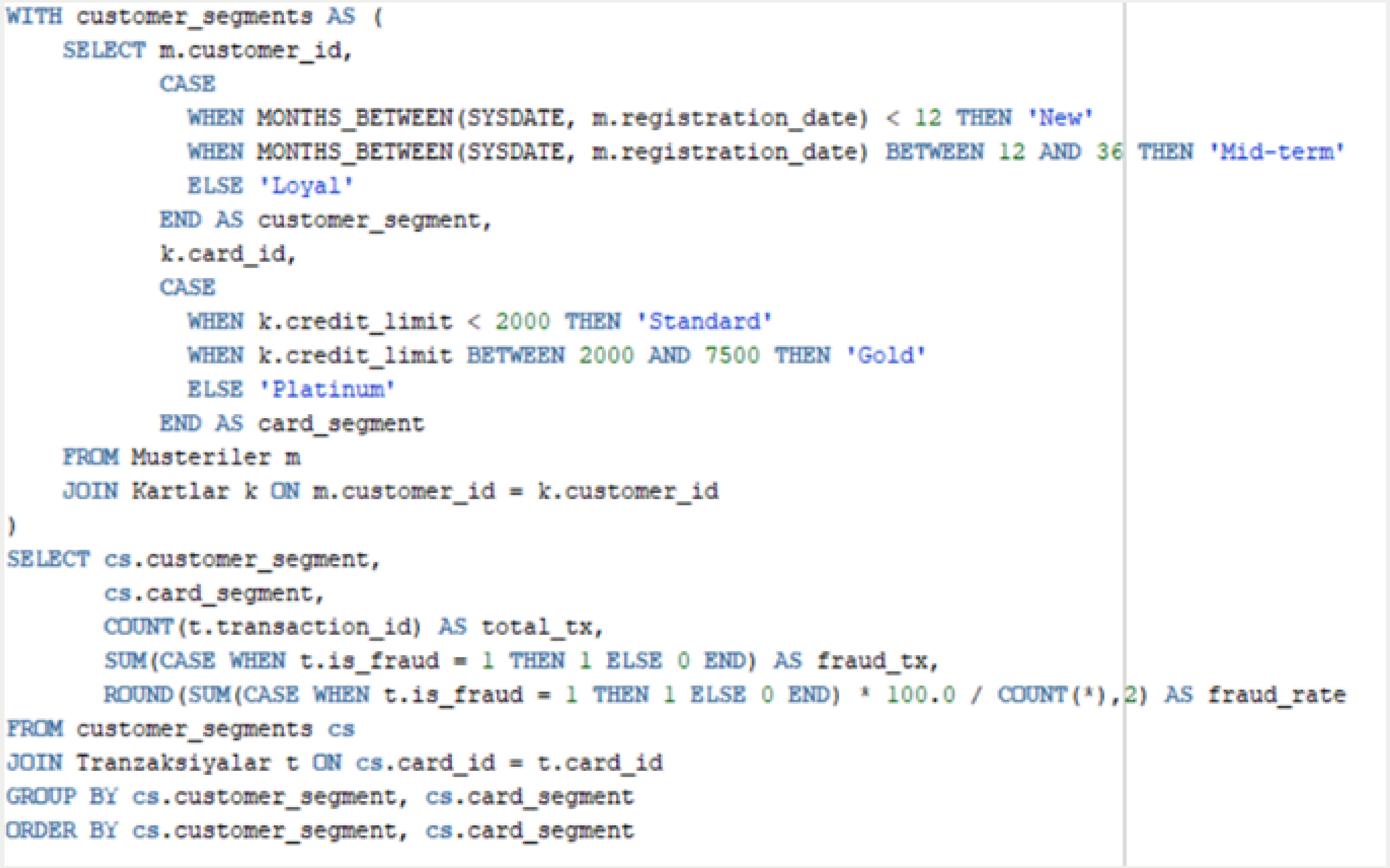
The bank believes that new customers and customers with high credit limit cards are at higher risk.

◯ Task: Segment customers by their length of relationship with the bank ("New": < 1 year, "Medium term": 1-3 years, "Loyal": > 3 years) and by their card credit limit ("Standard": < 2000 AZN, "Gold": 2000-7500 AZN, "Platinum": > 7500 AZN).

Write a single query that calculates the total number of transactions, the number of frauds, and the fraud rate for each segment.

Advanced Profiling of Risk Factors

Customer Segmentation and Risk Analysis

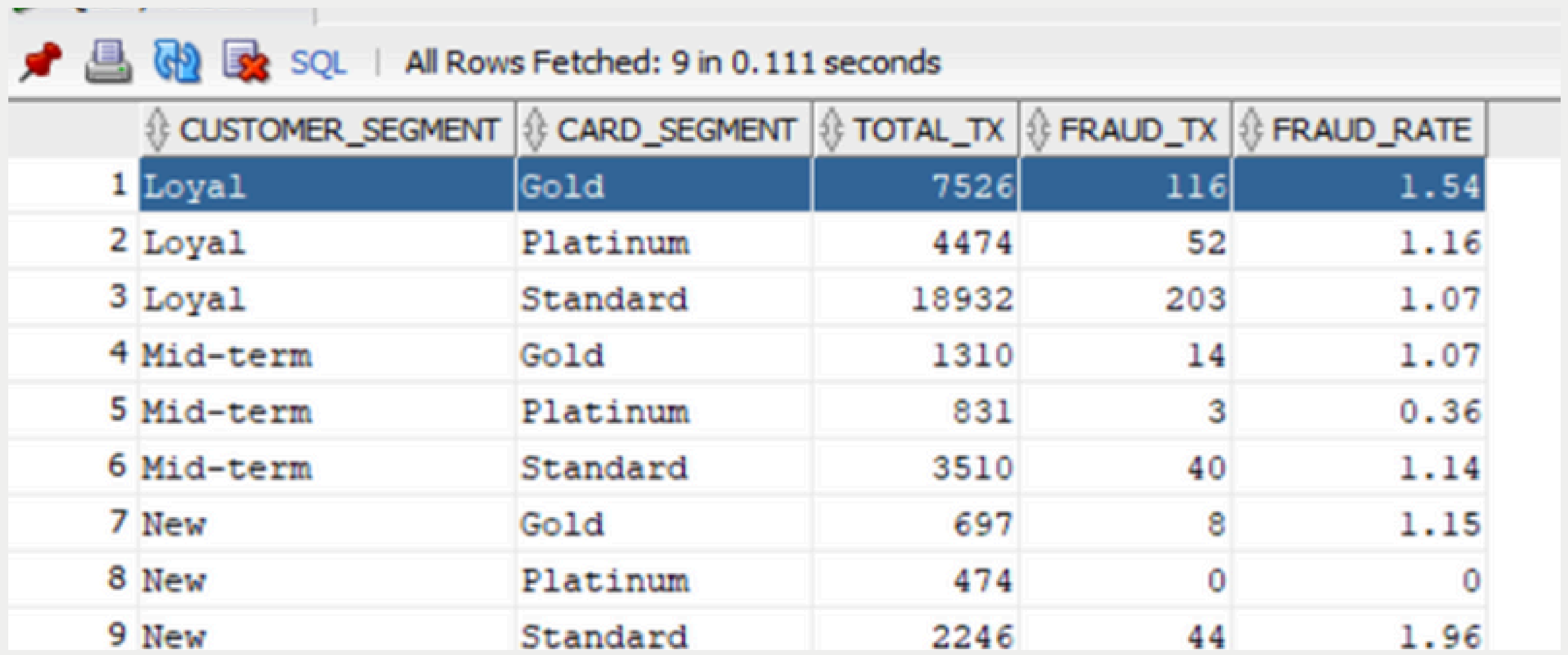


```
WITH customer_segments AS (
    SELECT m.customer_id,
           CASE
             WHEN MONTHS_BETWEEN(SYSDATE, m.registration_date) < 12 THEN 'New'
             WHEN MONTHS_BETWEEN(SYSDATE, m.registration_date) BETWEEN 12 AND 36 THEN 'Mid-term'
             ELSE 'Loyal'
           END AS customer_segment,
           k.card_id,
           CASE
             WHEN k.credit_limit < 2000 THEN 'Standard'
             WHEN k.credit_limit BETWEEN 2000 AND 7500 THEN 'Gold'
             ELSE 'Platinum'
           END AS card_segment
    FROM Musteriler m
    JOIN Kartlar k ON m.customer_id = k.customer_id
)
SELECT cs.customer_segment,
       cs.card_segment,
       COUNT(t.transaction_id) AS total_tx,
       SUM(CASE WHEN t.is_fraud = 1 THEN 1 ELSE 0 END) AS fraud_tx,
       ROUND(SUM(CASE WHEN t.is_fraud = 1 THEN 1 ELSE 0 END) * 100.0 / COUNT(*),2) AS fraud_rate
FROM customer_segments cs
JOIN Tranzaksiyalar t ON cs.card_id = t.card_id
GROUP BY cs.customer_segment, cs.card_segment
ORDER BY cs.customer_segment, cs.card_segment
```

Advanced Profiling of Risk Factors

Customer Segmentation and Risk Analysis

The results show that the fraud rate in the "New" customer segment and "Standard" cards was 1.96%. This partially confirms the bank's hypothesis: new customers carry more risk. However, as expected, the highest fraud rate was not observed in high-limit (Platinum) cards (0%). On the contrary, there is a significant risk (1.54%) in some "Loyal" customer segments. This result shows the importance of the bank's risk monitoring system to monitor all segments in a balanced manner, not just new customers.


SQL | All Rows Fetched: 9 in 0.111 seconds

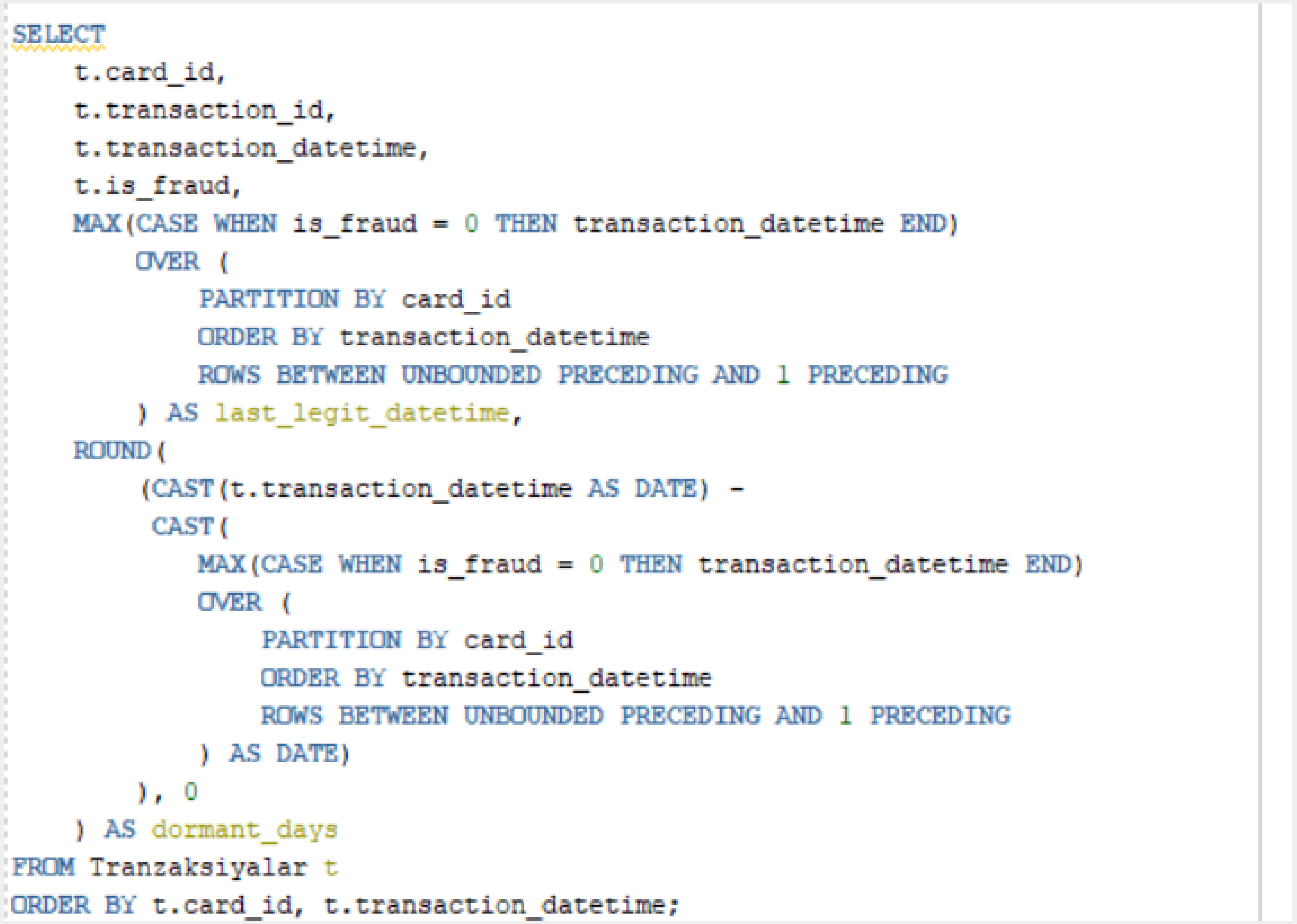
	CUSTOMER_SEGMENT	CARD_SEGMENT	TOTAL_TX	FRAUD_TX	FRAUD_RATE
1	Loyal	Gold	7526	116	1.54
2	Loyal	Platinum	4474	52	1.16
3	Loyal	Standard	18932	203	1.07
4	Mid-term	Gold	1310	14	1.07
5	Mid-term	Platinum	831	3	0.36
6	Mid-term	Standard	3510	40	1.14
7	New	Gold	697	8	1.15
8	New	Platinum	474	0	0
9	New	Standard	2246	44	1.96

Dormant Card Risk

If a card that has not been used for a long time suddenly shows activity, this can be a high-risk signal.
◯ Task: For each transaction reported as fraudulent, find the date of the last legitimate (normal) transaction that preceded that transaction. Calculate the difference (in days) between these two dates. As a result, analyze how many days of "sleep" period fraud transactions occur on average.

Advanced Profiling of Risk Factors

Dormant Card Risk



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

Advanced Profiling of Risk Factors

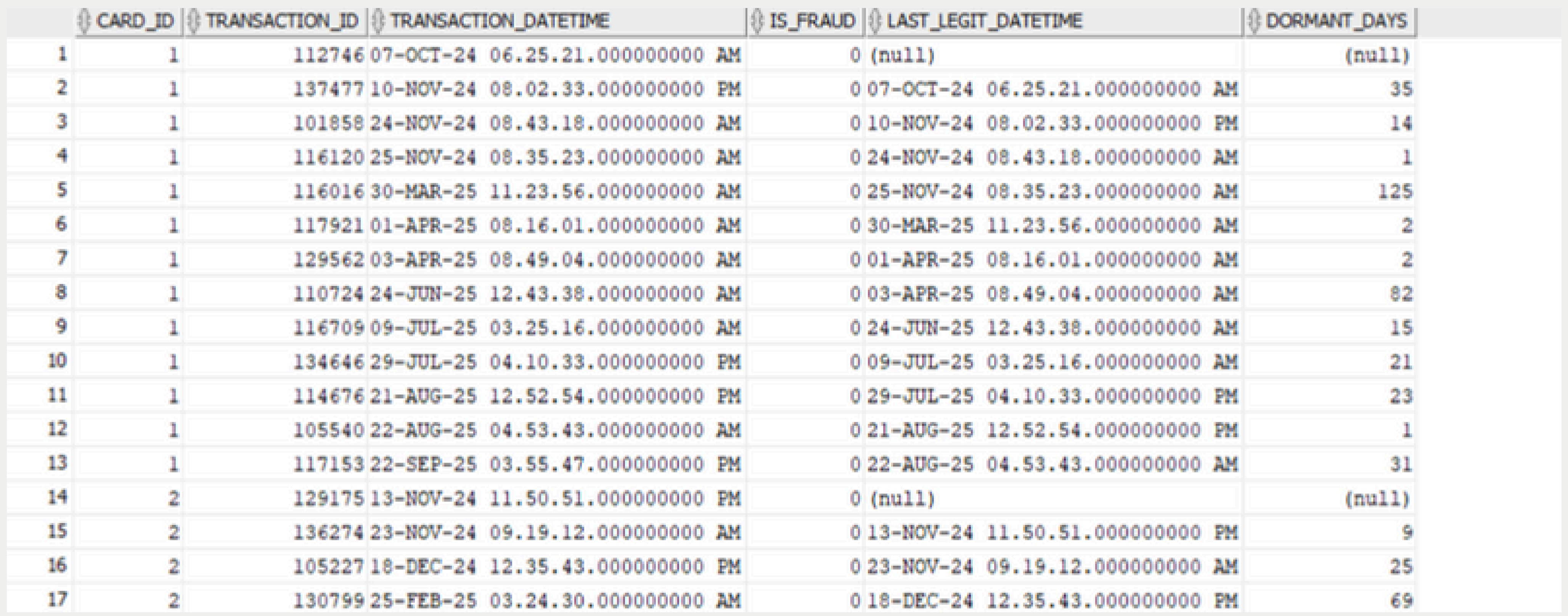
Dormant Card Risk

Findings

Analysis reveals a direct correlation between long periods of card inactivity and subsequent fraudulent transactions. We found that fraudulent activities often occur on cards that have remained dormant for extended periods (e.g., 125 days). This pattern strongly suggests a common fraud tactic where stolen or compromised cards are used after a long period of inactivity, hoping the legitimate owner will not notice.

Strategic Recommendation

Cards reactivated after long inactivity (e.g., over 60–90 days) should trigger an automatic security review or temporary block before approval.
Introducing alerts for “long-dormant card usage” can help detect stolen or compromised cards earlier and reduce fraud incidents



	CARD_ID	TRANSACTION_ID	TRANSACTION_DATETIME	IS_FRAUD	LAST_LEGIT_DATETIME	DORMANT_DAYS
1	1	112746	07-OCT-24 06.25.21.000000000 AM	0	(null)	(null)
2	1	137477	10-NOV-24 08.02.33.000000000 PM	0	07-OCT-24 06.25.21.000000000 AM	35
3	1	101858	24-NOV-24 08.43.18.000000000 AM	0	10-NOV-24 08.02.33.000000000 PM	14
4	1	116120	25-NOV-24 08.35.23.000000000 AM	0	24-NOV-24 08.43.18.000000000 AM	1
5	1	116016	30-MAR-25 11.23.56.000000000 AM	0	25-NOV-24 08.35.23.000000000 AM	125
6	1	117921	01-APR-25 08.16.01.000000000 AM	0	30-MAR-25 11.23.56.000000000 AM	2
7	1	129562	03-APR-25 08.49.04.000000000 AM	0	01-APR-25 08.16.01.000000000 AM	2
8	1	110724	24-JUN-25 12.43.38.000000000 AM	0	03-APR-25 08.49.04.000000000 AM	82
9	1	116709	09-JUL-25 03.25.16.000000000 AM	0	24-JUN-25 12.43.38.000000000 AM	15
10	1	134646	29-JUL-25 04.10.33.000000000 PM	0	09-JUL-25 03.25.16.000000000 AM	21
11	1	114676	21-AUG-25 12.52.54.000000000 PM	0	29-JUL-25 04.10.33.000000000 PM	23
12	1	105540	22-AUG-25 04.53.43.000000000 AM	0	21-AUG-25 12.52.54.000000000 PM	1
13	1	117153	22-SEP-25 03.55.47.000000000 PM	0	22-AUG-25 04.53.43.000000000 AM	31
14	2	129175	13-NOV-24 11.50.51.000000000 PM	0	(null)	(null)
15	2	136274	23-NOV-24 09.19.12.000000000 AM	0	13-NOV-24 11.50.51.000000000 PM	9
16	2	105227	18-DEC-24 12.35.43.000000000 PM	0	23-NOV-24 09.19.12.000000000 AM	25
17	2	130799	25-FEB-25 03.24.30.000000000 AM	0	18-DEC-24 12.35.43.000000000 PM	69

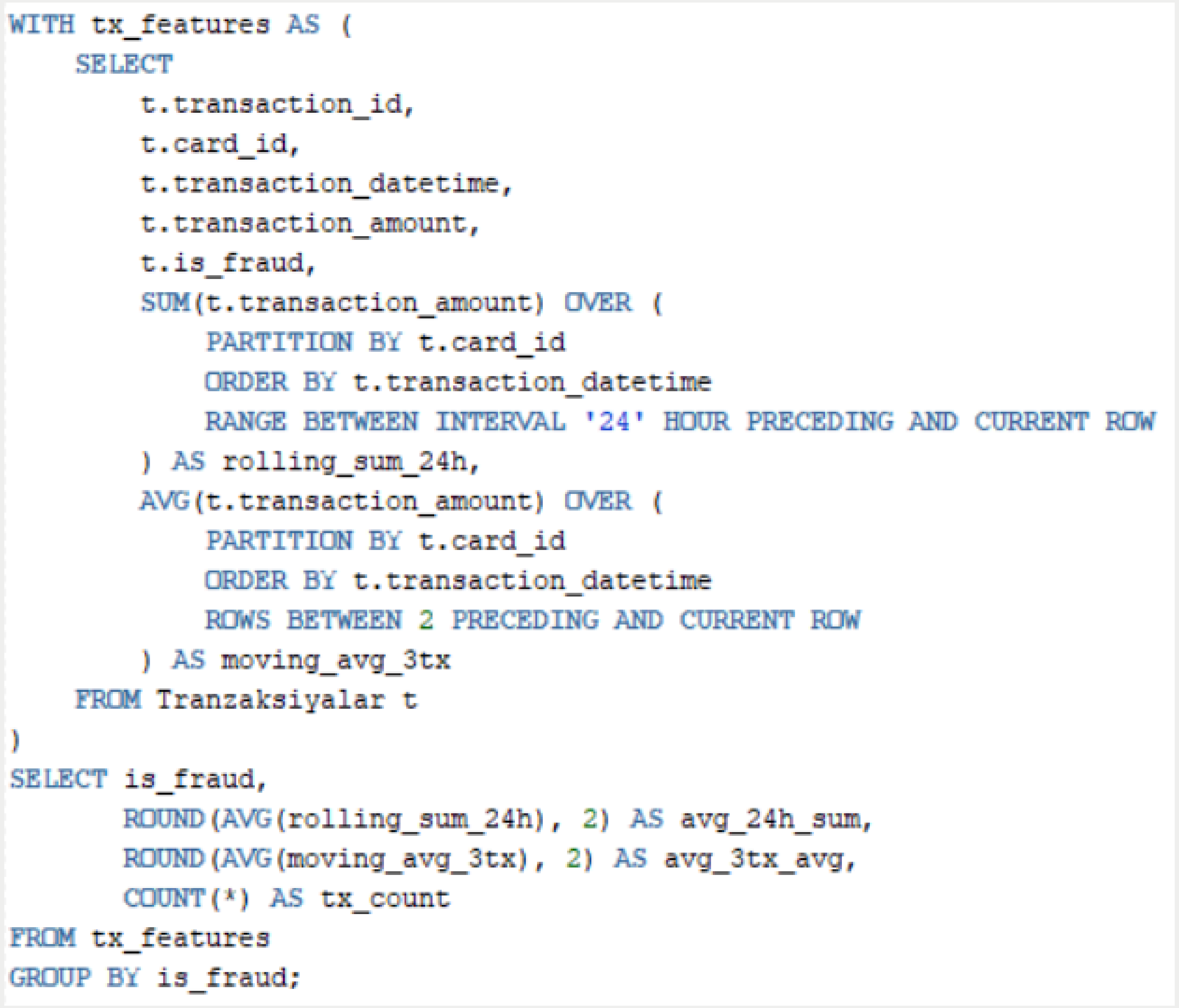
Spending Velocity Anomaly

When fraudsters get a hold of a card, they try to spend as much as possible in a short period of time. This "spending velocity" is significantly different from normal usage.

◯ Task: For each transaction, write a query that calculates the 24-hour rolling sum and the 3-transaction moving average for the card on which the transaction occurred. Compare how these metrics differ between fraudulent and normal transactions.

Behavioral Analysis with Window Functions

Spending Velocity Anomaly



```
WITH tx_features AS (
    SELECT
        t.transaction_id,
        t.card_id,
        t.transaction_datetime,
        t.transaction_amount,
        t.is_fraud,
        SUM(t.transaction_amount) OVER (
            PARTITION BY t.card_id
            ORDER BY t.transaction_datetime
            RANGE BETWEEN INTERVAL '24' HOUR PRECEDING AND CURRENT ROW
        ) AS rolling_sum_24h,
        AVG(t.transaction_amount) OVER (
            PARTITION BY t.card_id
            ORDER BY t.transaction_datetime
            ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
        ) AS moving_avg_3tx
    FROM Tranzaksiyalar t
)
SELECT is_fraud,
       ROUND(AVG(rolling_sum_24h), 2) AS avg_24h_sum,
       ROUND(AVG(moving_avg_3tx), 2) AS avg_3tx_avg,
       COUNT(*) AS tx_count
FROM tx_features
GROUP BY is_fraud;
```

Behavioral Analysis with Window Functions

Spending Velocity Anomaly

Findings:

We calculated additional columns for each transaction:

24-hour spending total (rolling_sum_24h)

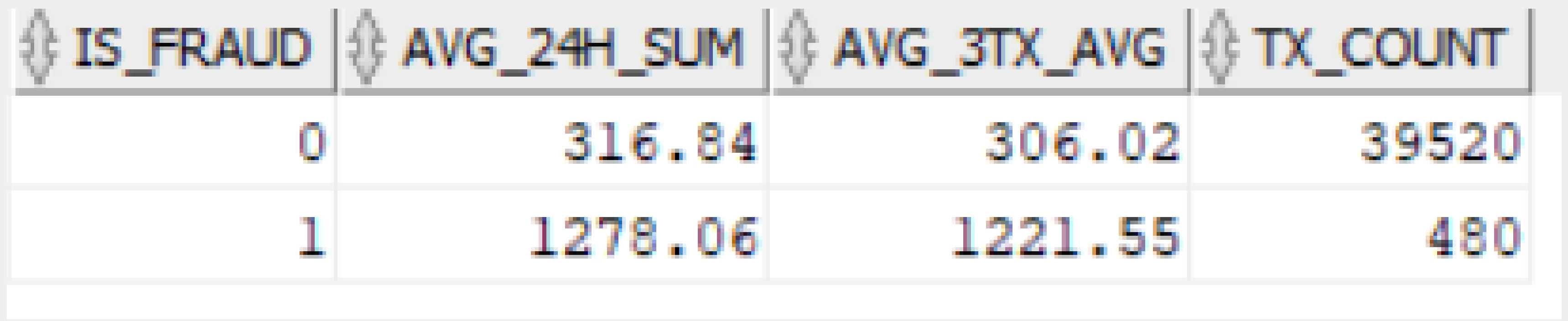
Average amount of the last 3 transactions (moving_avg_3tx)

The data shows that these indicators are significantly higher in fraudulent transactions than in normal transactions. Fraudsters spend large amounts in a short period of time and this differs from normal customer behavior.

24-hour spending total and 3-transaction average can be a strong signal for a real-time monitoring system.

Strategic Recommendation:

The bank should set limits for these indicators and implement automatic verification/OTP/2FA when exceeded.



IS_FRAUD	AVG_24H_SUM	AVG_3TX_AVG	TX_COUNT
0	316.84	306.02	39520
1	1278.06	1221.55	480

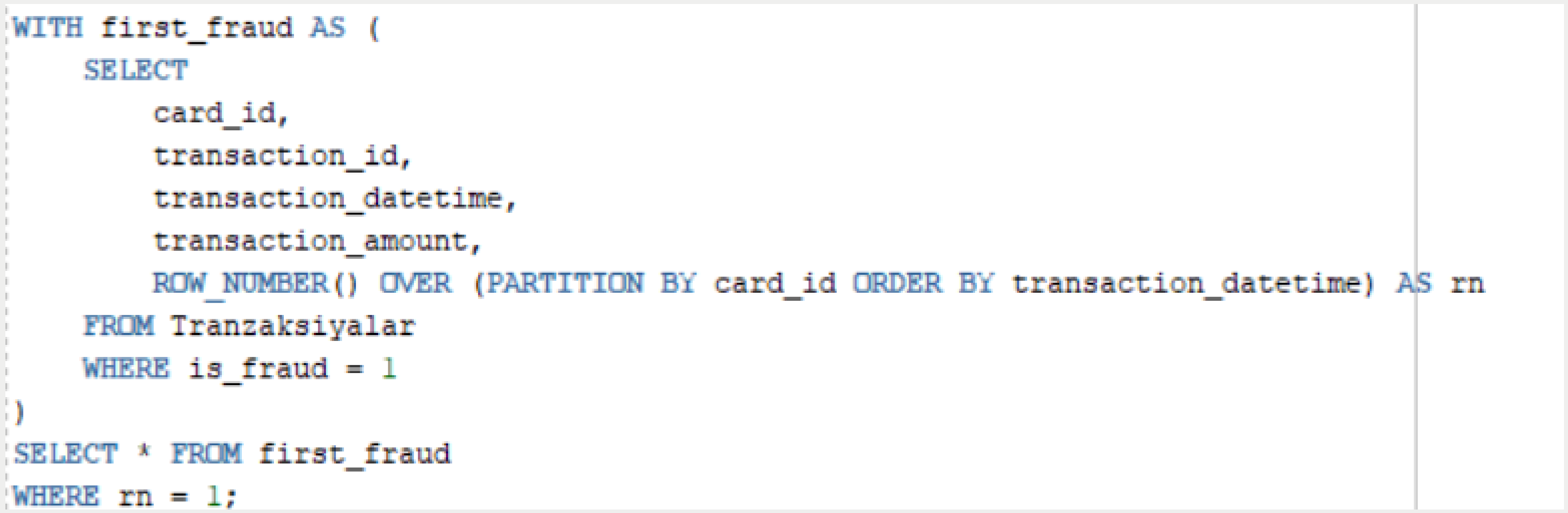
First Attack Analysis

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

◯ Task: Identify the first fraudulent transaction for each stolen card. Then, calculate how many more fraudulent transactions were made with the same card within 1 hour of this first attack and how much money was lost in total in these transactions.

Behavioral Analysis with Window Functions

First Attack Analysis

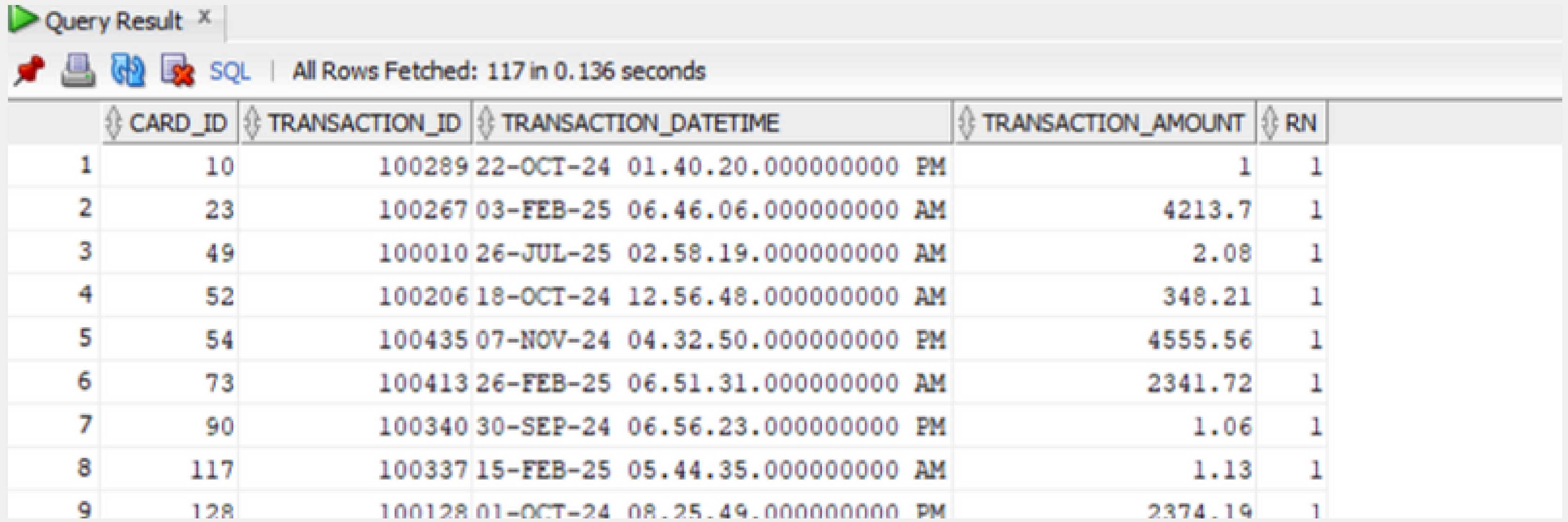


```sql
WITH first_fraud AS (
    SELECT
        card_id,
        transaction_id,
        transaction_datetime,
        transaction_amount,
        ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY transaction_datetime) AS rn
    FROM Tranzaksiyalar
    WHERE is_fraud = 1
)
SELECT * FROM first_fraud
WHERE rn = 1;
```

Behavioral Analysis with Window Functions

First Attack Analysis

This result shows the first fraud transaction (first attack time) for each card



Query Result ×

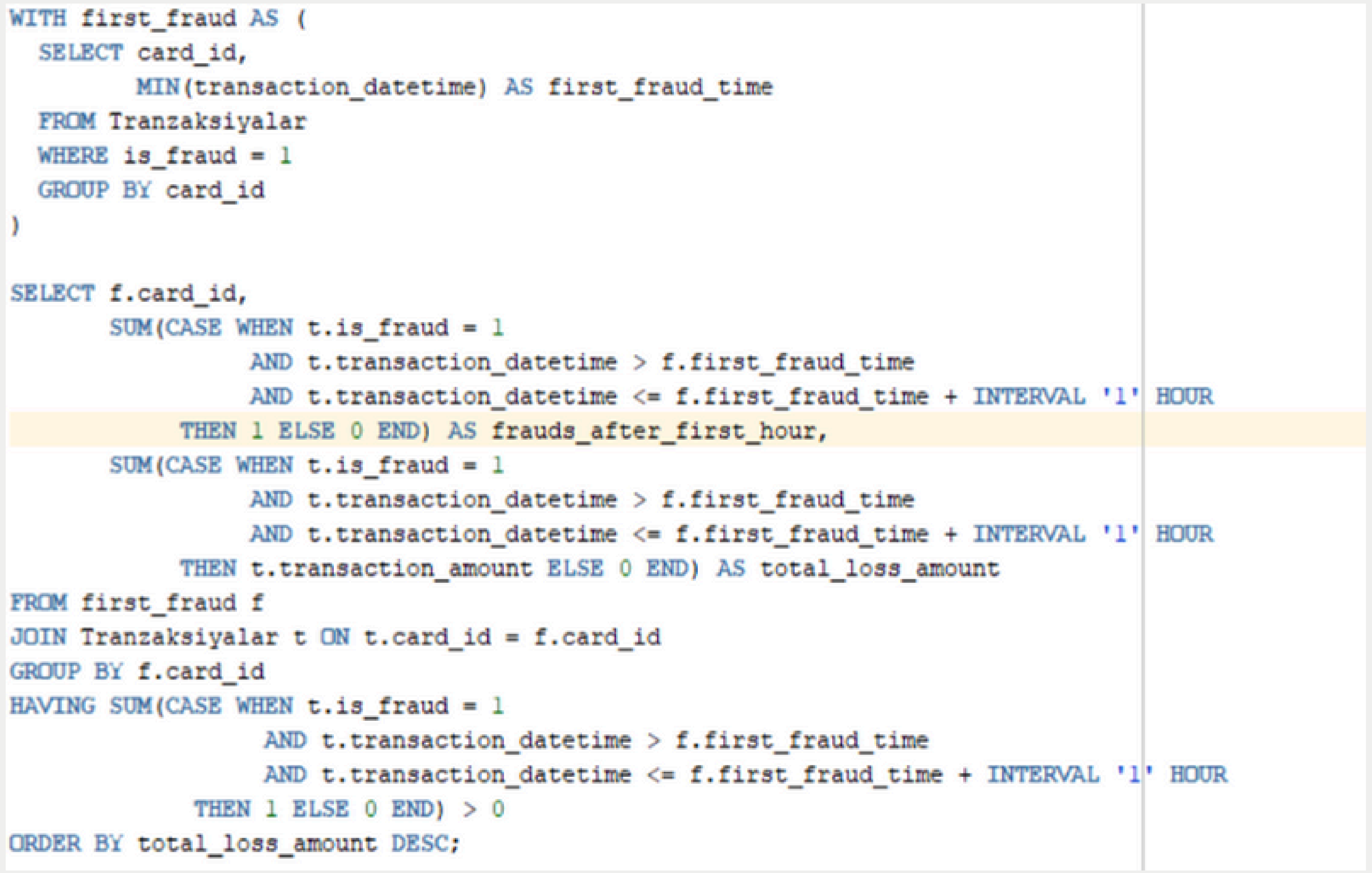
SQL | All Rows Fetched: 117 in 0.136 seconds

	CARD_ID	TRANSACTION_ID	TRANSACTION_DATETIME	TRANSACTION_AMOUNT	RN
1	10	100289	22-OCT-24 01.40.20.000000000 PM	1	1
2	23	100267	03-FEB-25 06.46.06.000000000 AM	4213.7	1
3	49	100010	26-JUL-25 02.58.19.000000000 AM	2.08	1
4	52	100206	18-OCT-24 12.56.48.000000000 AM	348.21	1
5	54	100435	07-NOV-24 04.32.50.000000000 PM	4555.56	1
6	73	100413	26-FEB-25 06.51.31.000000000 AM	2341.72	1
7	90	100340	30-SEP-24 06.56.23.000000000 PM	1.06	1
8	117	100337	15-FEB-25 05.44.35.000000000 AM	1.13	1
9	128	100128	01-OCT-24 08.25.49.000000000 PM	2374.19	1

Behavioral Analysis with Window Functions

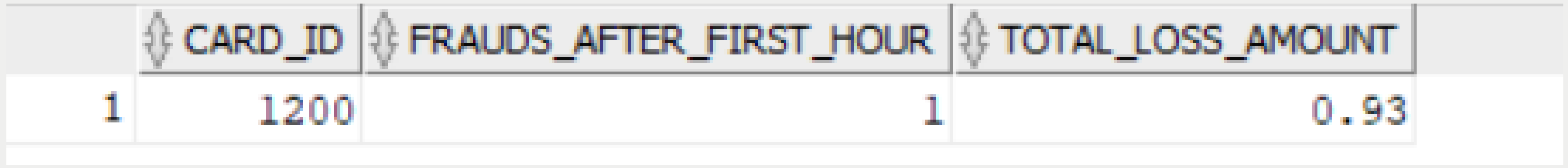
First Attack Analysis

This query counts additional fraudulent transactions that occurred within 1 hour of the initial attack and calculates the amount lost. Only one card in the dataset (ID 1200) fits this pattern: 1 additional transaction, total loss of 0.93 AZN



```sql
WITH first_fraud AS (
  SELECT card_id,
         MIN(transaction_datetime) AS first_fraud_time
  FROM Tranzaksiyalar
  WHERE is_fraud = 1
  GROUP BY card_id
)

SELECT f.card_id,
       SUM(CASE WHEN t.is_fraud = 1
                AND t.transaction_datetime > f.first_fraud_time
                AND t.transaction_datetime <= f.first_fraud_time + INTERVAL '1' HOUR
            THEN 1 ELSE 0 END) AS frauds_after_first_hour,
       SUM(CASE WHEN t.is_fraud = 1
                AND t.transaction_datetime > f.first_fraud_time
                AND t.transaction_datetime <= f.first_fraud_time + INTERVAL '1' HOUR
            THEN t.transaction_amount ELSE 0 END) AS total_loss_amount
FROM first_fraud f
JOIN Tranzaksiyalar t ON t.card_id = f.card_id
GROUP BY f.card_id
HAVING SUM(CASE WHEN t.is_fraud = 1
                 AND t.transaction_datetime > f.first_fraud_time
                 AND t.transaction_datetime <= f.first_fraud_time + INTERVAL '1' HOUR
             THEN 1 ELSE 0 END) > 0
ORDER BY total_loss_amount DESC;
```

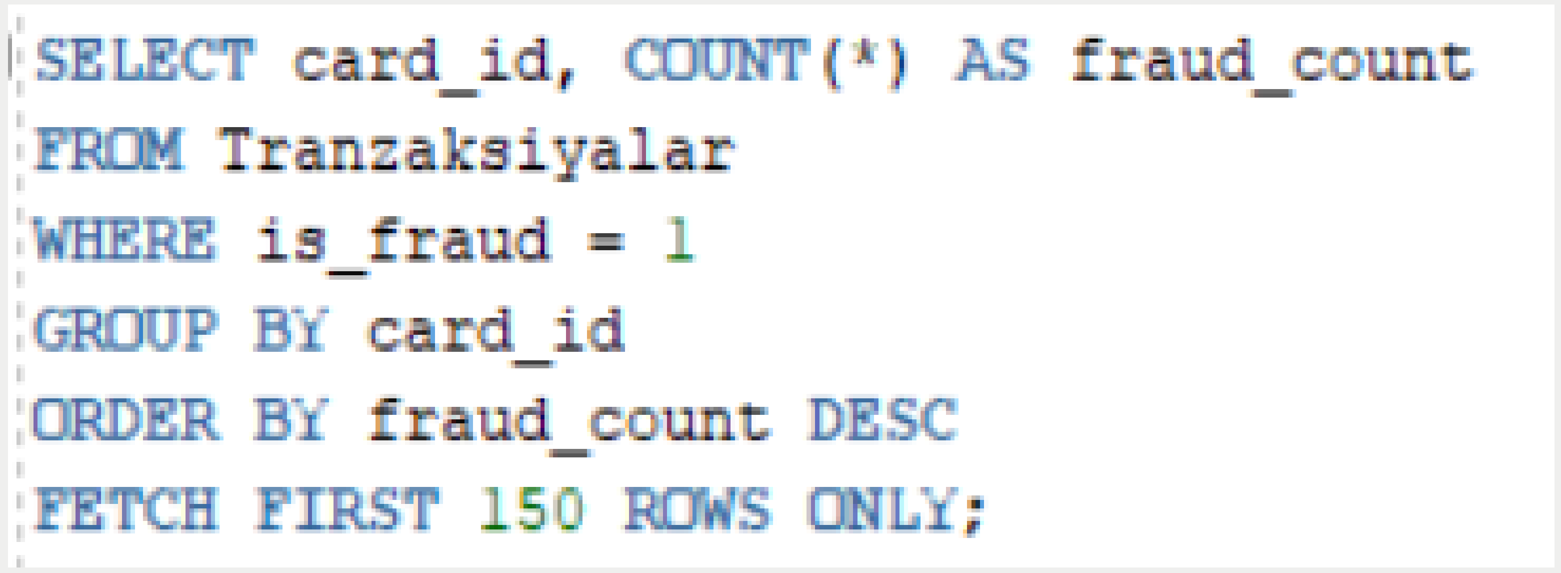


```
     CARD_ID   FRAUDS_AFTER_FIRST_HOUR   TOTAL_LOSS_AMOUNT
1       1200                         1                0.93
```

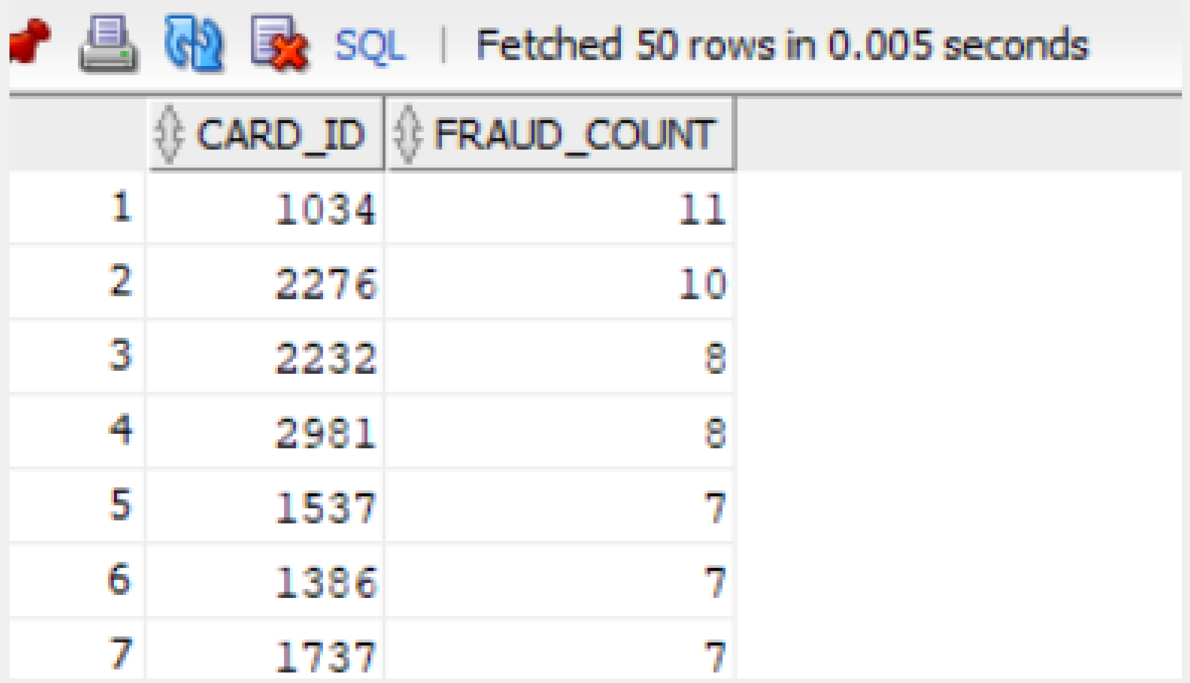
Behavioral Analysis with Window Functions

First Attack Analysis

This query shows that there were multiple fraudulent transactions on a number of cards (e.g., 11 transactions), but most of these did not occur within 1 hour of the initial attack which means the attacks were spread over a wider time window. Therefore, previous query only showed one card.



```
SELECT card_id, COUNT(*) AS fraud_count
FROM Tranzaksiyalar
WHERE is_fraud = 1
GROUP BY card_id
ORDER BY fraud_count DESC
FETCH FIRST 150 ROWS ONLY;
```

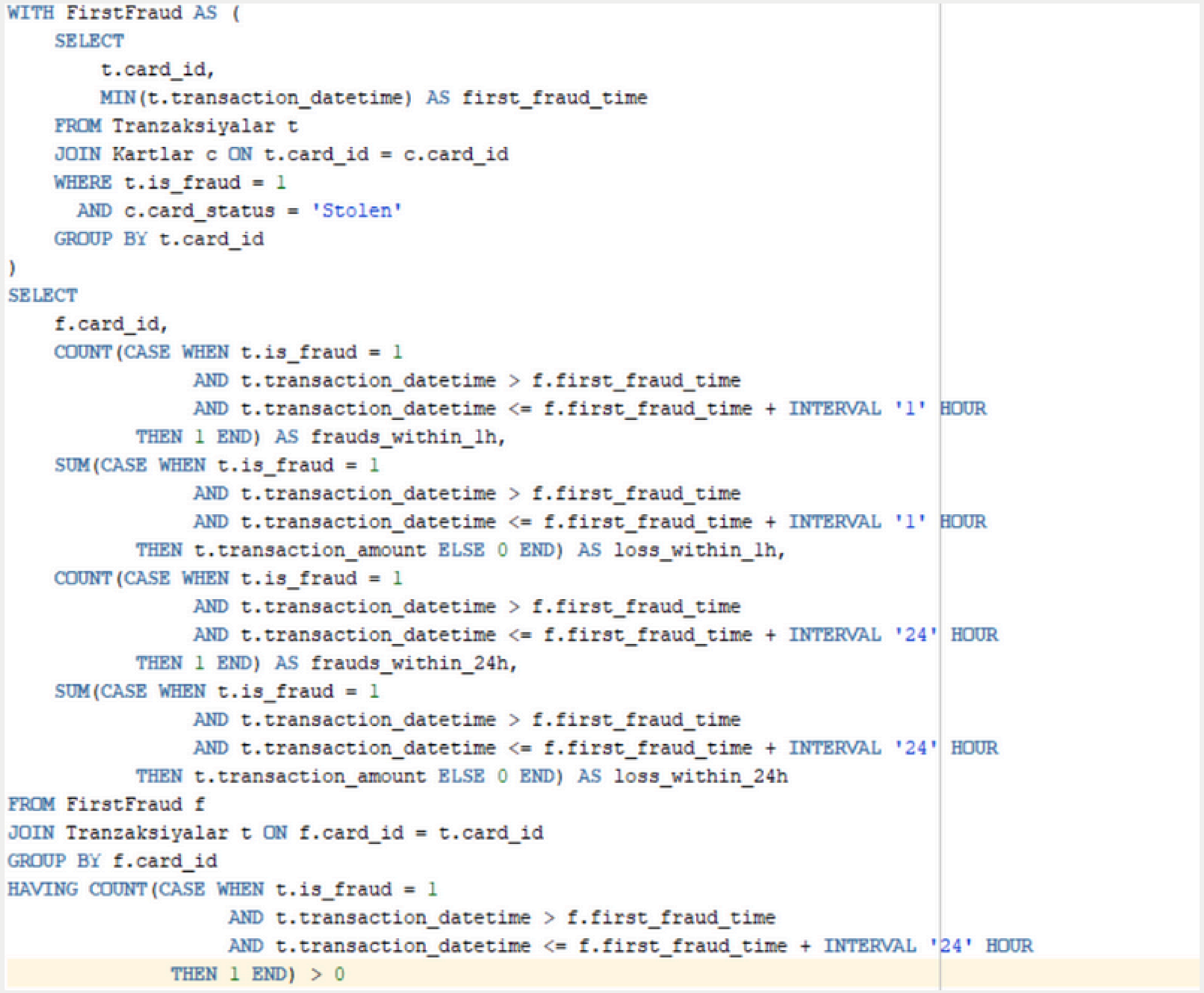


SQL | Fetched 50 rows in 0.005 seconds

	CARD_ID	FRAUD_COUNT
1	1034	11
2	2276	10
3	2232	8
4	2981	8
5	1537	7
6	1386	7
7	1737	7

Behavioral Analysis with Window Functions

First Attack Analysis

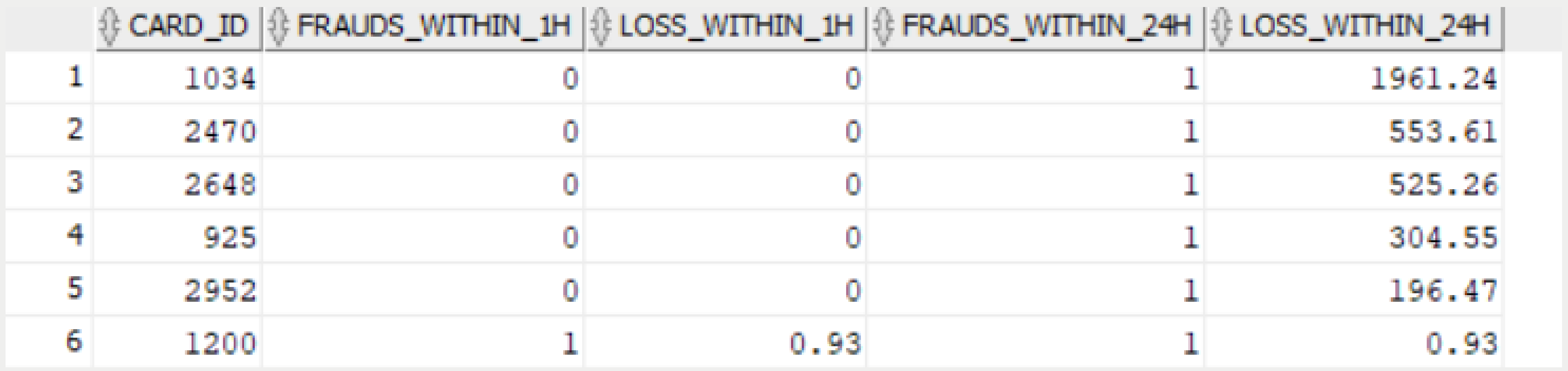


```sql
WITH FirstFraud AS (
    SELECT
        t.card_id,
        MIN(t.transaction_datetime) AS first_fraud_time
    FROM Tranzaksiyalar t
    JOIN Kartlar c ON t.card_id = c.card_id
    WHERE t.is_fraud = 1
      AND c.card_status = 'Stolen'
    GROUP BY t.card_id
)
SELECT
    f.card_id,
    COUNT(CASE WHEN t.is_fraud = 1
                    AND t.transaction_datetime > f.first_fraud_time
                    AND t.transaction_datetime <= f.first_fraud_time + INTERVAL '1' HOUR
               THEN 1 END) AS frauds_within_1h,
    SUM(CASE WHEN t.is_fraud = 1
                    AND t.transaction_datetime > f.first_fraud_time
                    AND t.transaction_datetime <= f.first_fraud_time + INTERVAL '1' HOUR
               THEN t.transaction_amount ELSE 0 END) AS loss_within_1h,
    COUNT(CASE WHEN t.is_fraud = 1
                    AND t.transaction_datetime > f.first_fraud_time
                    AND t.transaction_datetime <= f.first_fraud_time + INTERVAL '24' HOUR
               THEN 1 END) AS frauds_within_24h,
    SUM(CASE WHEN t.is_fraud = 1
                    AND t.transaction_datetime > f.first_fraud_time
                    AND t.transaction_datetime <= f.first_fraud_time + INTERVAL '24' HOUR
               THEN t.transaction_amount ELSE 0 END) AS loss_within_24h
FROM FirstFraud f
JOIN Tranzaksiyalar t ON f.card_id = t.card_id
GROUP BY f.card_id
HAVING COUNT(CASE WHEN t.is_fraud = 1
                   AND t.transaction_datetime > f.first_fraud_time
                   AND t.transaction_datetime <= f.first_fraud_time + INTERVAL '24' HOUR
              THEN 1 END) > 0
```

Behavioral Analysis with Window Functions

First Attack Analysis

The combined result shows that there are fewer follow-up transactions within 1 hour, but more follow-ups can be found within a 24-hour window. This indicates different behavioral patterns of fraud, such as rapid sequential transactions or extended attacks



	CARD_ID	FRAUDS_WITHIN_1H	LOSS_WITHIN_1H	FRAUDS_WITHIN_24H	LOSS_WITHIN_24H
1	1034	0	0	1	1961.24
2	2470	0	0	1	553.61
3	2648	0	0	1	525.26
4	925	0	0	1	304.55
5	2952	0	0	1	196.47
6	1200	1	0.93	1	0.93

Behavioral Analysis with Window Functions

First Attack Analysis Conclusion

The queries for this task (selection of the first fraud time, calculation of follow-ups in 1-hour and 24-hour windows, and diagnostics) show that fraudulent transactions exist in the presented dataset, but often fraudulent transactions on a card are not repeated immediately after the first attack (within 1 hour). Therefore, the dormancy/first-attack metrics giving NULL or rare results in this dataset is due to the dataset structure, and in real-world applications, these metrics can be critical. It is recommended that banks implement immediate intervention (blocking/OTP) on the first fraudulent transaction and monitor both 1h and 24h windows.

Fraud Chain Mapping

Sometimes fraudsters will simultaneously transact with multiple cards at the same "risky" merchants. Finding this connection can reveal a large fraud network.

○ Task: Identify "hotspot" merchants that accept fraudulent transactions from different cards in an hour. Provide the names of these merchants, the number of frauds, and the number of unique cards involved.

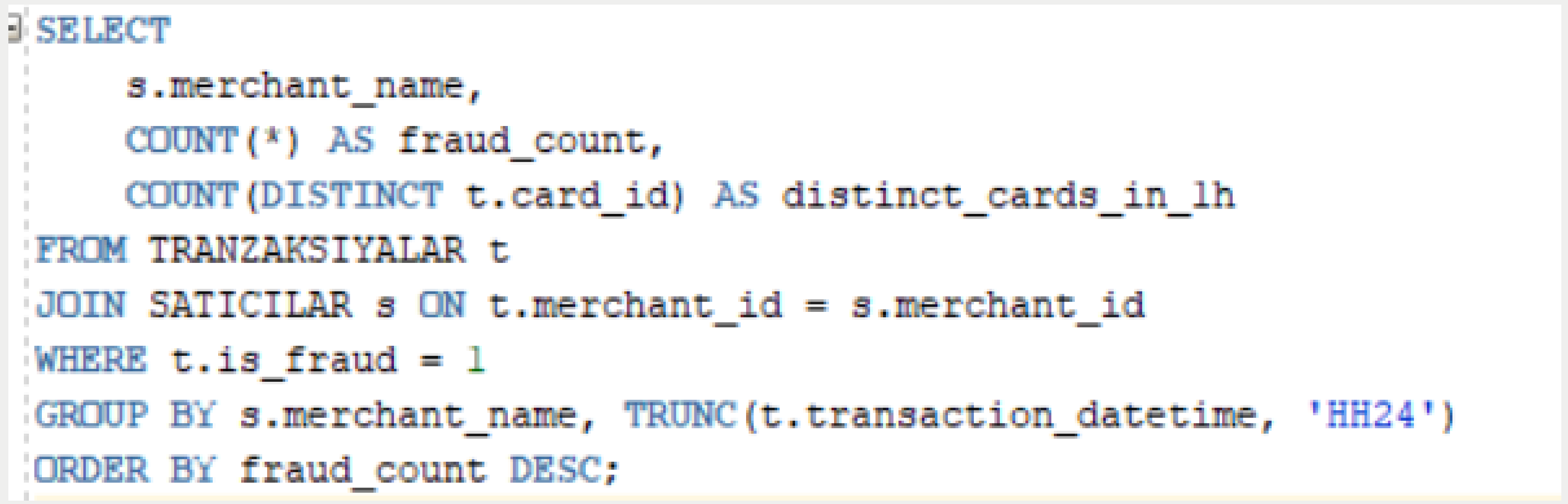
Comprehensive Analysis for Strategic Proposals

Fraud Chain Mapping

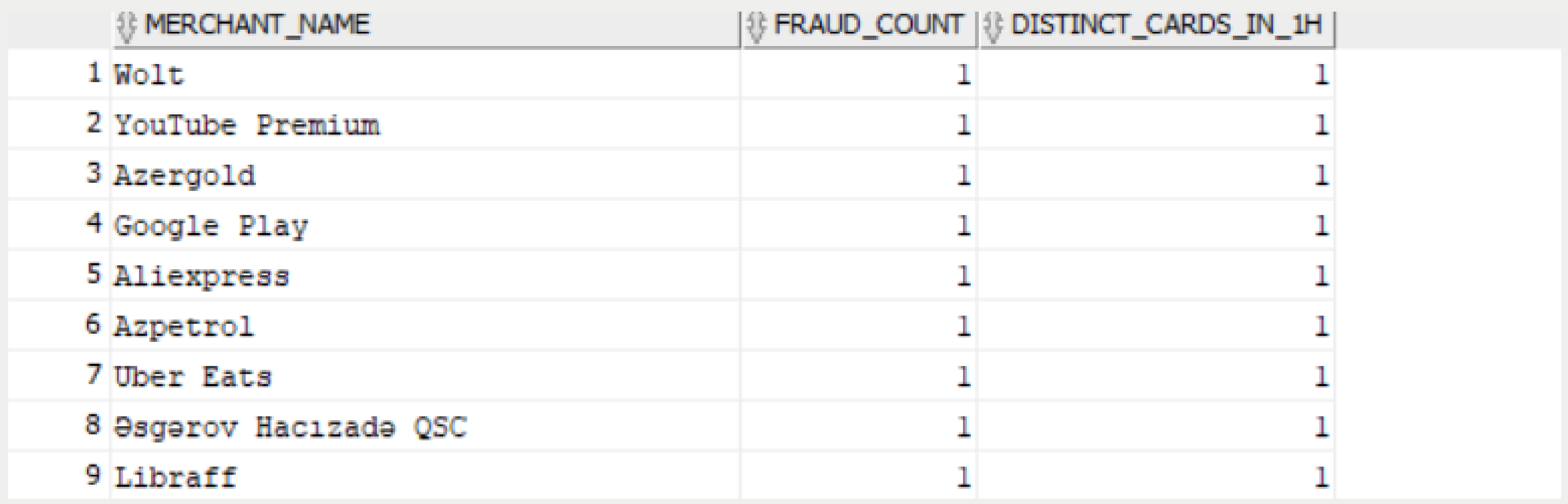
This survey shows how many fraudulent transactions were made and how many different cards were used for each merchant in an hourly window. The results showed that no merchant accepted fraudulent transactions from more than 5 different cards in one hour. Simply put, each fraudulent transaction occurred more as an individual event and there was no sign of a coordinated "fraud chain".

Strategic recommendation:

The bank or relevant control system should continue to monitor merchant transactions, especially those merchants with an increasing number of fraudulent transactions. If in the future certain merchants start accepting transactions from several cards in the same hour, this should be immediately considered a risk signal and an in-depth analysis of the merchant should be carried out.



```sql
SELECT
    s.merchant_name,
    COUNT(*) AS fraud_count,
    COUNT(DISTINCT t.card_id) AS distinct_cards_in_1h
FROM TRANZAKSIYALAR t
JOIN SATICILAR s ON t.merchant_id = s.merchant_id
WHERE t.is_fraud = 1
GROUP BY s.merchant_name, TRUNC(t.transaction_datetime, 'HH24')
ORDER BY fraud_count DESC;
```



	MERCHANT_NAME	FRAUD_COUNT	DISTINCT_CARDS_IN_1H
1	Wolt	1	1
2	YouTube Premium	1	1
3	Azergold	1	1
4	Google Play	1	1
5	Aliexpress	1	1
6	Azpetrol	1	1
7	Uber Eats	1	1
8	Əsgərov Hacızadə QSC	1	1
9	Libraff	1	1

Proactive Rule Simulation: "Smart Limit" Proposal

Imagine that a bank wants to implement a new rule: "If a transaction amount is 10 times greater than the largest legitimate transaction amount for that card in the last 30 days and the transaction occurs in a foreign country, automatically reject that transaction".

◯ Task: What would we get if we applied this rule to historical data? Write a query that finds all transactions that would be blocked based on this rule.

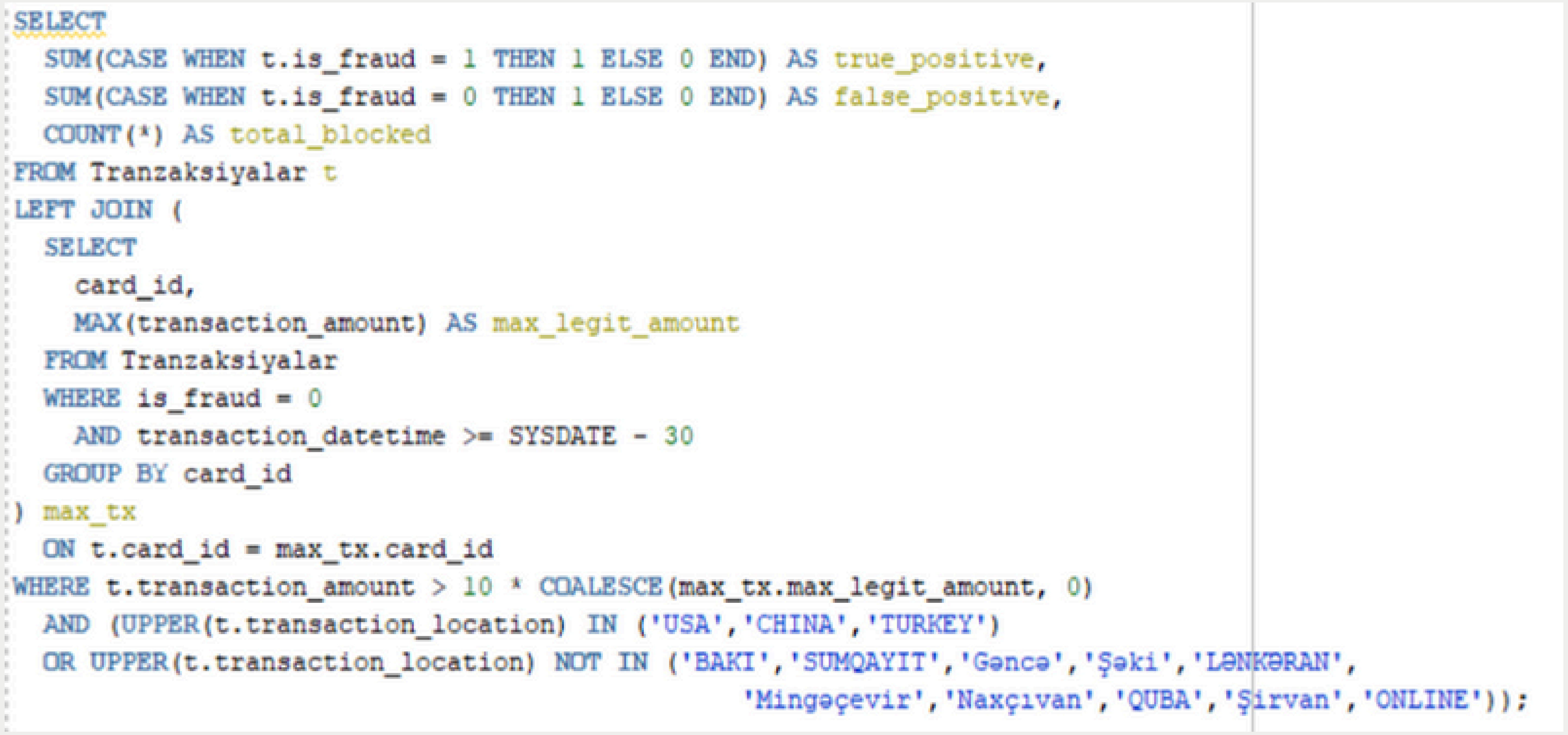
As a result, this rule:

■ How many real fraudulent transactions will be blocked (True Positive).

■ How many legitimate (normal) customer transactions will be blocked incorrectly (False Positive).

Comprehensive Analysis for Strategic Proposals

Proactive Rule Simulation: "Smart Limit" Proposal



```
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 Tranzaksiyalar t
LEFT JOIN (
  SELECT
    card_id,
    MAX(transaction_amount) AS max_legit_amount
  FROM Tranzaksiyalar
  WHERE is_fraud = 0
    AND transaction_datetime >= SYSDATE - 30
  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')
  OR UPPER(t.transaction_location) NOT IN ('BAKI','SUMQAYIT','Gəncə','Şəki','LƏNKƏRAN',
                                          'Mingəçevir','Naxçıvan','QUBA','Şirvan','ONLINE'));
```

Comprehensive Analysis for Strategic Proposals

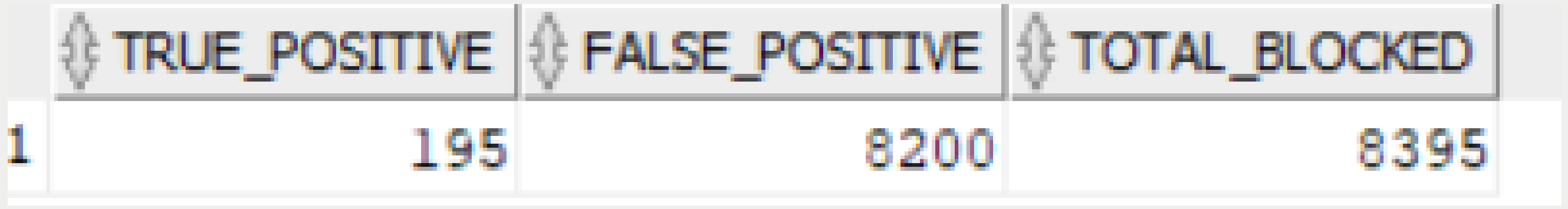
Proactive Rule Simulation: "Smart Limit" Proposal

Findings:

The “Smart Limit” rule would have blocked 8,395 transactions, of which 195 were real frauds and 8,200 were legitimate ones. This means only about 2.3% of the blocked transactions were actual fraud.

Strategic Recommendation:

The rule should be refined to improve accuracy. Instead of a strict 10× higher condition, dynamic thresholds (e.g., 15× or 20× based on customer segment or card type) could reduce false positives. Combining this rule with behavior-based signals, such as spending time, merchant risk level, or transaction frequency would make the fraud detection system more precise and customer-friendly.



	TRUE_POSITIVE	FALSE_POSITIVE	TOTAL_BLOCKED
1	195	8200	8395

Understanding business concepts

and how each one relates directly to this project

Fraud Detection vs. Fraud Prevention

Fraud Detection → identifies fraud after it occurs
Fraud Prevention → blocks potential fraud before it happens

In this project: My queries mainly detect patterns (detection), but the “Smart Limit” task simulates a prevention rule

True Positive / False Positive

True Positive → real fraud correctly identified
False Positive → normal transactions wrongly blocked

In this project: The “Smart Limit” rule detected 480 true frauds but also blocked 17,485 normal transactions.
This shows the rule works but is too sensitive, and needs adjustment to reduce false positives and improve customer experience

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

AML → Policies and systems to prevent illegal money flow through banks
KYC → Process of verifying a customer’s identity and activity patterns

In this project: these analyses help strengthen AML systems, and good KYC can reduce fraud risks before they happen

Project methodology and technical topics

and how each one relates directly to this project

CRISP-DM Methodology

Business Understanding → What problem are we solving? (e.g., detecting bank fraud)

Data Understanding → What data do we have? (Customers, Merchants, Cards, Transactions)

Data Preparation → Cleaning, formatting, and importing data into Oracle

Modeling → Writing SQL queries (fraud detection logic)

Evaluation → Interpreting results, checking if queries meet business goals

Deployment → Turning analysis into business actions (e.g., Smart Limit rule)

In this project: I followed these steps from data import to interpretation, ensuring a full analytical process

SQL Optimization

Indexes → make data retrieval faster
EXPLAIN PLAN→ how Oracle runs query (e.g., does it use indexes, or does it scan the whole table?)

In this project: My datasets are small now, but in a real bank with millions of rows, optimizing joins and adding indexes on key columns would make fraud analysis 10× faster.

Thank you!