MGMT 590: ADVANCED DATABASE

SPRING 2024

REPORT

FINANCIAL FRAUD DETECTION USING SQL

BY
TEAM 2

AVISHEK DASGUPTA, SANTOSH PAWAN KURADA,
SHIVAM MISHRA, SIDDHANT TREASURE,
SOHAM AGARWAL, VARUN ANNAPAREDDY

BACKGROUND

Client Details and Status

Our client, a prominent figure in the financial landscape, renowned for its extensive banking services and comprehensive credit card operations, serves a diverse customer base and maintains a wide network of merchant partnerships. This position underscores their pivotal role in facilitating economic transactions and growth. However, the digital era has introduced sophisticated financial frauds, posing significant risks to customer trust and the institution's stability. The urgency to combat these threats has become paramount, necessitating innovative and robust security measures.

In response, our client has embarked on a strategic initiative to develop a state-of-the-art, SQL-based real-time fraud detection system. This project is aimed at harnessing the power of real-time data analytics, to identify and neutralize fraudulent activities efficiently. The commitment to this project reflects the client's dedication to upholding security, restoring customer confidence, and establishing new benchmarks for fraud prevention in the financial sector.

Dataset Description

To support the development of this cutting-edge fraud detection system, the client has provided a comprehensive dataset, detailed as follows:

- **Transaction Records:** This dataset includes detailed logs of customer transactions, capturing the date, time, amount, and merchant details for each transaction. It forms the backbone of our analysis, allowing us to identify patterns and anomalies indicative of fraudulent activity.
- **Customer Profiles:** Containing rich demographic and behavioral data for each customer, this dataset enables personalized fraud detection strategies, taking into account individual customer risk profiles and transaction habits.
- **Merchant Information:** This dataset provides insights into the merchants involved in transactions, including business type, location, and historical transaction volumes. It is crucial for identifying risky merchants or unusual transaction patterns.
- **Transaction Amounts and Anomaly Scores:** These datasets include transaction amounts and computed anomaly scores, which are instrumental in pinpointing transactions that deviate significantly from established patterns.
- **Transaction Metadata and Category Labels:** Offering additional context for each transaction, such as the method of transaction and categorization of the spending, these datasets enrich the analysis and help in fine-tuning fraud detection algorithms.

As we will discuss later in this report, we also generate an anomaly scoring system based on a collection of the above features.

INTRODUCTION

Project Question

In the context of increasing sophistication in financial fraud, there is a critical need for a dynamic and efficient fraud detection system for a banking or credit card company. This system must leverage a comprehensive dataset encompassing transaction records, customer profiles, indicators of fraudulent patterns, transaction amounts, and merchant information. The primary objective is to develop an SQL-based real-time fraud detection system that can adaptively learn from and identify new fraudulent patterns in transactional data. Our team aims to develop a system capable of processing this information in real-time, identifying anomalies and potential fraud, and continuously updating its understanding of fraudulent patterns to ensure robust, up-to-date security measures against financial crimes.

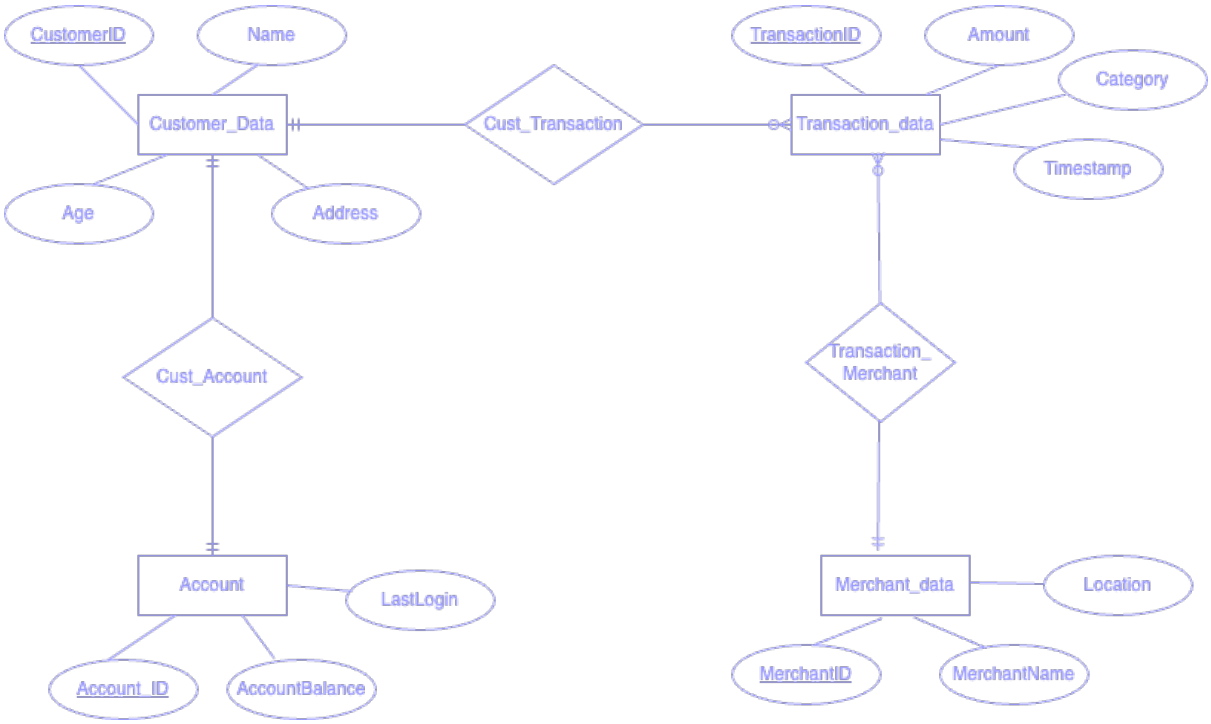
Objectives

The primary objective of this project is to leverage the provided datasets to develop an SQL-based real-time fraud detection system capable of adaptively learning and identifying new fraudulent patterns. Our approach involves integrating these datasets to create a comprehensive analysis framework that allows for real-time processing of transactions, identification of anomalies, and continuous learning from new data. This system aims to significantly enhance the client's ability to pre-emptively detect and mitigate financial fraud, thereby safeguarding customer assets and maintaining the integrity of the financial system.

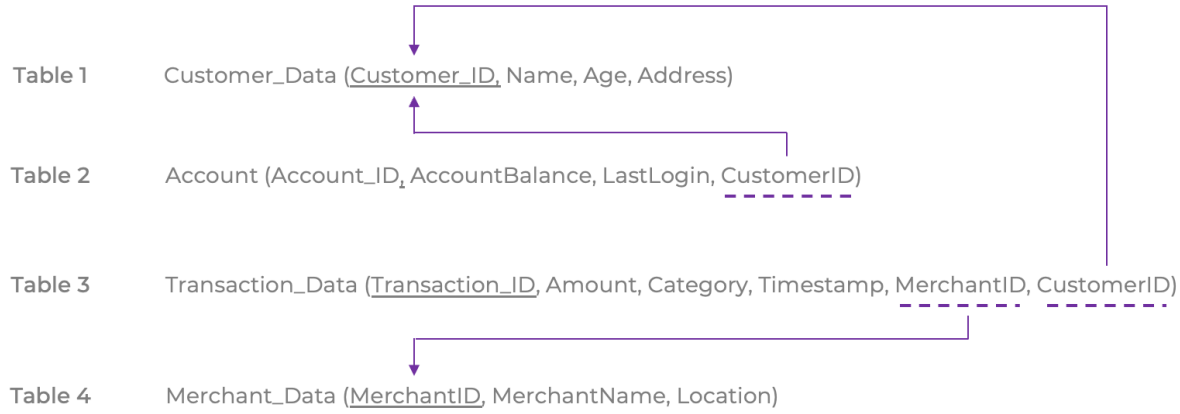
Our specific goals for the project are as follows:

- **Enhanced Fraud Detection Accuracy:** Implement advanced algorithms and SQL queries to accurately identify fraudulent transactions with higher precision, reducing false positives and false negatives.
- **Dynamic Pattern Recognition:** Develop the system's ability to dynamically learn and adapt to new and evolving fraudulent patterns, ensuring the detection mechanism remains effective against sophisticated fraud techniques.
- **Customer Trust and Retention:** Increase customer trust and confidence in the financial institution's security measures, contributing to higher customer retention and satisfaction rates by ensuring their financial assets are well-protected.
- **Operational Efficiency:** Streamline fraud detection operations, reducing the need for manual reviews and investigations, thereby saving time and resources for the financial institution.
- **Regulatory Compliance:** Ensure the system adheres to financial regulations and standards related to fraud detection and data protection, avoiding potential legal penalties and reputational damage.
- **Data-Driven Decision Making:** Leverage the comprehensive datasets to inform strategic decisions related to fraud prevention, risk management, and customer service improvements.
- **Merchant Risk Assessment:** Implement mechanisms for evaluating and monitoring merchant risk, identifying and mitigating risks associated with high-risk merchants or transaction types.

CONCEPTUAL DATA MODELLING



RELATIONAL SCHEMA



NORMALIZATION

- **Validation for 1NF:** The data available in all the tables were validated for multi-valued or composite attributes. There are no multi-valued or composite attributes. Hence, data in all the tables are already in 1NF form.
- **Validation for 2NF:** The data available in all the tables were validated for partial dependencies. It was observed that the data is already in 2 NF, since no partial dependencies were observed.
- **Validation for 3 NF:** After validating the Schema for 2 NF, the data available in all the tables were validated for transitive dependencies. It was observed that the data is already in 3 NF, since no transitive dependencies were observed.

DATA CLEANING

1. **Missing Values:** These queries identifies and filters records from the dataset containing null or missing values which help to ensure data integrity and quality. This step is crucial in preparing the data for further analysis or reporting.

```sql
-- 1. Missing Values
SELECT * FROM transaction_data
WHERE TransactionID IS NULL
OR Timestamp IS NULL
OR Amount IS NULL
OR CustomerID IS NULL
OR MerchantID IS NULL
OR Category IS NULL;

SELECT * FROM merchant_data
WHERE MerchantID IS NULL
OR MerchantName IS NULL
OR Location IS NULL;

SELECT * FROM customer_data
WHERE CustomerID IS NULL
OR Name IS NULL
OR Age IS NULL
OR Address IS NULL;
```

2. **Duplicate Records:** This query is designed to detect duplicate records which is valuable for identifying and subsequently removing or resolving duplicate transactions, which helps maintain data accuracy and consistency in the dataset.

```sql
-- 2. Duplicate Records
SELECT TransactionID, COUNT(*)
FROM transaction_data
GROUP BY TransactionID
HAVING COUNT(*) > 1;
```

3. **Valid Ranges:** These queries are designed to detect and mitigate data inconsistencies, contributing to the overall reliability and quality of the datasets for subsequent analysis or processing. Such discrepancies may stem from data entry errors, other financial irregularities or potential fraudulent activities

```sql
-- 3. Valid Ranges
SELECT * FROM transaction_data
WHERE Amount < 0;

SELECT * FROM customer_data
WHERE Age NOT BETWEEN 18 AND 100;

SELECT * FROM account
WHERE AccountBalance < 0;
```

4. **Spelling inconsistencies or variations in names or column dates:** This query aims to identify spelling inconsistencies or variations within the "Category" column in the "transaction_data" dataset. This information is essential for standardizing and harmonizing category names, enhancing data consistency, and facilitating accurate analysis and reporting.

```sql
-- 4. Spelling inconsistencies or variations in names or column dates
SELECT Category, COUNT(*)
FROM transaction_data
GROUP BY Category;
```

5. **Table Data type check:** These queries conduct a comprehensive examination of the data types present within key tables of the database. This analysis is crucial for verifying the consistency and compatibility of data types across tables, ensuring data integrity, and facilitating smooth data processing and analysis tasks.

```sql
-- 5. Table Data type check
SELECT column_name, data_type from information_schema.columns
where table_name = 'transaction_data';

SELECT column_name, data_type from information_schema.columns
where table_name = 'merchant_data';

SELECT column_name, data_type from information_schema.columns
where table_name = 'customer_data';

SELECT column_name, data_type from information_schema.columns
where table_name = 'account';
```

6. **Date - Time Correction: Text to Timestamp:** These alterations ensure that date and time information within the respective tables is correctly formatted, facilitating accurate data analysis and processing tasks.

```sql
-- 6. Date – Time Correction: Text to Timestamp
ALTER TABLE account ADD COLUMN TempLastLogin DATE;
UPDATE account
SET TempLastLogin = STR_TO_DATE(`LastLogin`, '%d-%m-%Y');
ALTER TABLE account DROP COLUMN `LastLogin`;
ALTER TABLE account CHANGE COLUMN TempLastLogin `LastLogin` DATE;

ALTER TABLE transaction_data ADD COLUMN TempTimestamp TIMESTAMP;
UPDATE transaction_data
SET TempTimestamp = STR_TO_DATE(`Timestamp`, '%d-%m-%Y %H:%i');
ALTER TABLE transaction_data DROP COLUMN `Timestamp`;
ALTER TABLE transaction_data CHANGE COLUMN TempTimestamp `Timestamp` TIMESTAMP;
```

ANOMALY DETECTION CALCULATION (Appendix 1)

Our fraud detection system is designed with a multifaceted approach, utilizing a suite of key metrics to enhance the identification and prevention of fraudulent transactions. These metrics include the Amount Score, Time Score, Repeated Transactions, Account Inactivity, Uncommon Merchant-Category Combinations metric and the Account Transaction Volume.

Our methodology and the development of these metrics are heavily influenced by the pioneering research outlined in "*A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection*" by Zhaohui Zhang et al. The paper's insights into applying convolutional neural networks (CNN) for financial fraud detection, particularly the innovative use of a feature sequencing layer that rearranges raw transaction data into varied convolutional patterns, have been instrumental. This approach allows the CNN to efficiently identify fraudulent activities by leveraging low-dimensional and non-derivative data, without the need for complex data transformations. Zhang et al.'s model, which achieved notable precision and recall rates of approximately 91% and 94% respectively, underscores the effectiveness of feature sequencing in enhancing fraud detection. Inspired by these findings, we have integrated similar concepts into our fraud detection framework, optimizing feature arrangement and utilizing raw transaction data to elevate the accuracy and efficiency of our fraud detection efforts. This integration not only aligns with Zhang et al.'s demonstrated success in the field but also advances our own system's capabilities in combating online transaction fraud.

Our Metric	Feature in Paper	Description	Weightage
Amount Score	Amount	High-value transactions are scrutinized, highlighting the importance of the transaction amount in fraud detection.	30 %
Time Score	Time	Transactions during unusual hours may indicate fraud, aligning with the analysis of transaction time patterns.	20 %
Repeated Transactions	Behavioral Patterns	Frequent transactions with the same characteristics in a short timeframe can indicate fraud, similar to analyzing differences in behavior patterns.	25 %
Account Inactivity	Historical Behavior (e.g., LastLogin)	Sudden transactions from long-inactive accounts could be suspicious, relating to historical account activity analysis.	15 %
Uncommon Merchant - Category Combinations	Merchant & Category Features	Transactions with rare merchant-category pairings could be anomalies, akin to examining transaction types and patterns for irregularities.	3 %
Account Transaction Volume	Transaction Frequency	An exceptionally high volume of transactions may indicate potential fraud, relating to the analysis of account transaction behavior over time.	7 %

Reference: Zhang, Z., Zhou, X., Zhang, X., Wang, L., & Wang, P. (2018). A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection. Security and Communication Networks, 2018, Article ID 5680264.

FRAUD DETECTION (USING QUERIES)

1. **A query to identify the top 5 customers by total transaction amount within each category, including their rank based on the transaction amount. (Appendix 2.1)**
   Impact: The output ranks the top 5 spending customers in each category, providing insights into customer spending patterns and identifying key revenue segments for targeted marketing initiatives and strategic business planning.
   Utilizes: Aggregate functions, Common Table Expressions (CTEs), Window Functions.

CustomerID	Category	TotalAmount	Rank
1190	Food	193.46700488	1
1299	Food	176.75787443999997	2
1146	Food	174.66312682	3
1204	Food	168.95082366	4
1234	Food	167.66458351	5
1666	Online	187.73580209	1
1132	Online	165.53512122	2
1596	Online	158.92775942	3
1289	Online	144.76842734000002	4
1276	Online	144.10933243	5
1330	Other	206.53964358	1
1548	Other	169.82517114	2

2. **This query aims to identify each customer's most frequent transaction category, utilizing a combination of joins, string functions, and window functions to rank categories by frequency. (Appendix 2.2)**
   Impact: To determine which categories are most popular among individual customers, supporting personalized marketing efforts and inventory management providing a clear picture of spending habits for targeted engagement and service improvement.
   Utilizes: Joins, string functions, window functions, aggregate functions.

CustomerID	Category	TransactionsCount
1001	Food	1
1003	Travel	1
1004	Other	2
1005	Other	1
1007	Travel	1
1008	Retail	1
1009	Other	2
1012	Online	1
1012	Retail	1
1014	Retail	1
1016	Other	1
1017	Other	1

3. **When you execute this query, you're asking the database to provide you with a financial profile for each customer based on the data stored across different tables. (Appendix 2.3)**
   Impact: The view CustomerFinancialHealth provides each customer's latest balance, average transaction amount, and the most recent login date. This amalgamated financial snapshot can be crucial for identifying customers with high transaction activity or potential account irregularities, and it supports proactive customer engagement and account management.
   Utilizes: SQL Modes, Views, Window Functions, Aggregate Functions, DISTINCT Clause.

CustomerID	Cust_Name	LatestBalance	AverageTransactionAmount	LastLoginDate
1001	Customer 1001	9507.27206	33.67062621	01-01-2022
1003	Customer 1003	1715.321989	30.98023905	03-01-2022
1004	Customer 1004	3101.509134	28.508436175	04-01-2022
1005	Customer 1005	5405.766914	81.80865066	05-01-2022
1007	Customer 1007	2957.760054	41.21158342	07-01-2022
1008	Customer 1008	8303.773585	91.1696889	08-01-2022
1009	Customer 1009	5359.74781	65.8372969	09-01-2022
1012	Customer 1012	2706.148888	71.45122675	12-01-2022
1014	Customer 1014	9922.291973	42.60535295	14-01-2022
1016	Customer 1016	8612.857297	68.36571539	16-01-2022
1017	Customer 1017	5179.000965	62.83583260333333	17-01-2022
1018	Customer 1018	2727.830525	48.00091683	18-01-2022

4. **This query aims to detect accounts with irregular patterns by looking for anomalies in transaction amounts compared to the usual account activity. (Appendix 2.4)**
   Results and Purpose: The query identifies accounts where recent transaction amounts deviate from the norm, as indicated by a high standard deviation and an anomaly score above a set threshold. This helps in early detection of potential fraud, ensuring account security and maintaining customer trust. The results enable pinpointing accounts that may require further investigation or monitoring for irregular activity.
   Utilizes: Window Functions, Aggregate Functions, Joins.

Account_ID	CustomerID	AverageAmountSpent	StdDevAmount	RecentTransactionAmount	FinalAnomalyScore
590946	1946	78.12285327	0	78.12285327	0.33
590204	1204	84.47541183	2.4716722999999945	86.94708413	0.33
590424	1424	49.10060881666667	18.879452830184633	86.18094791	0.33
590302	1302	95.55400892	0	95.55400892	0.33
590700	1700	98.80425008	0	98.80425008	0.33
590321	1321	73.48557443499999	26.347060025	99.83263446	0.33
590117	1117	65.9679644375	31.104152215737667	99.1004274	0.33
590613	1613	71.62031277	13.799865419999998	85.42017819	0.33
590643	1643	84.05328459	0	84.05328459	0.33
590475	1475	92.61930298	0	92.61930298	0.33
590993	1993	77.72694417	0	77.72694417	0.33
590038	1038	80.53777563	0	80.53777563	0.33

5. **Check for merchants with an unusually high number of high-risk transactions. (Appendix 2.5)**
   Impact: The query screens for merchants with transaction counts significantly higher than the average within the high-risk category, as defined by the anomaly score. This can highlight merchants that may be involved in or subject to fraudulent activities, assisting in proactive risk management and merchant auditing. The outcome aids in the identification of potentially compromised or risky merchants that require closer monitoring or intervention.
   Utilizes: Subqueries, Aggregate Functions, Joins.

MerchantID	MerchantName	HighRiskTransactionCount
2220	Merchant 2220	3
2235	Merchant 2235	3

6. **This query detects customers whose transaction risk scores have significantly changed recently, indicating potential account compromise or a change in spending behavior. (Appendix 2.6)**
   Impact: The query screens for customers with rising risk scores between transactions, suggesting unusual activity. It serves as an early warning system, prompting further investigation to prevent fraud and protect account integrity. The results specifically list customers with a notable uptick in risk scores, which could be symptomatic of fraudulent behavior or shifts in financial habits.
   Utilizes: Window Functions, Conditional Aggregation.

CustomerID	TransactionsWithIncreasedRisk
1012	1
1023	1
1051	1
1073	1
1084	1
1108	1
1117	1
1123	1
1126	1
1132	1
1142	1
1144	1

7. **This query calculates the average anomaly score across different age groups, helping to identify if certain age demographics are more associated with transactions that have high anomaly scores. (Appendix 2.7)**
   Impact: The query sorts customers into age brackets to average their transaction anomaly scores, aiming to spot patterns or trends in fraud risk by age group. It highlights if certain age demographics exhibit higher-than-average anomaly scores, which may inform risk management strategies and customer education initiatives. The results show the distribution of anomaly scores across age groups, suggesting targeted approaches for fraud prevention may be required for specific demographics.
   Utilizes: Case statements, Aggregate Functions, Joins.

AgeGroup	AverageAnomalyScore	TotalTransactions
18-25	0.118125	160
26-35	0.109923	259
36-45	0.105349	215
56-65	0.096234	154
46-55	0.095094	212

8. **Analyze transactions based on the merchant's location to identify regions with a higher occurrence of high-anomaly-score transactions, which could indicate regions more susceptible to fraudulent activities. (Appendix 2.8)**
   Impact: The query discerns patterns of fraud by location, flagging areas with a prevalence of high-risk transactions. It's pivotal for regional fraud monitoring and preemptive security measures. The results reveal hotspots for scrutiny, potentially guiding regional fraud prevention strategies.
   Utilizes: Joins, Conditional Aggregation, Case Statements.

Location	TotalTransactions	AverageAnomalyScore	HighRiskTransactions
Location 2220	3	0.330000	3
Location 2235	4	0.255000	3

CONCLUSION

Our analysis of transaction data has yielded significant insights into the detection and prevention of fraudulent activities, structured around several key dimensions:

- **Anomaly Indexing:** By applying anomaly indexing on a per-transaction basis, we've gained a granular perspective on potential fraud, enabling us to pinpoint irregularities with greater precision. This approach facilitates the early identification of unusual transaction patterns, enhancing our ability to respond proactively.
- **Customer Profiling:** Through detailed customer profiling based on transaction behaviors, we've identified distinct patterns that are essential for crafting personalized fraud detection strategies. This segmentation helps in understanding customer-specific risk factors and tailoring interventions accordingly.
- **Merchant Analysis:** Our investigation into merchant activities has uncovered certain vendors with an unusually high incidence of fraudulent transactions. This could point to compromised security measures or targeted fraud schemes, highlighting the need for targeted risk management strategies and merchant education on potential vulnerabilities.
- **Dynamic Scoring:** The detection of significant shifts in customers' transaction risk scores is a critical development, potentially indicative of account compromise or emerging fraud patterns. Our dynamic scoring system facilitates swift responses, allowing for the mitigation of risks in a timely manner.
- **Demographic Insights:** Analysis segmented by age groups has revealed that certain demographics are more prone to high-anomaly transactions. This suggests that targeted educational initiatives and fraud prevention strategies could be particularly effective for these groups.
- **Geographical Trends:** By analyzing transactions based on merchant locations, we've identified specific regions with elevated levels of high-anomaly transactions. These geographical insights are crucial for recognizing and addressing regional fraud trends, providing valuable information for law enforcement and regulatory bodies.

In conclusion, the depth and breadth of our analyses highlight the necessity of a comprehensive, multi-dimensional approach to fraud detection. Incorporating these insights into real-time fraud detection systems, enhancing educational efforts for customers and merchants, and fostering collaboration with authorities in high-risk areas are pivotal steps forward. Additionally, future research should consider the potential of advanced machine learning models to further refine the prediction and prevention of fraudulent transactions, thereby bolstering the overall effectiveness of fraud detection frameworks.

APPENDIX

1. Anomaly Score Query

```
-- Anomaly scoring query with CTE
CREATE VIEW score_view AS(
WITH MonthlyTransactionCounts AS (
    SELECT
        Account_ID,
        YEAR(Timestamp) AS Year,
        MONTH(Timestamp) AS Month,
        COUNT(*) AS NumTransactions
    FROM
        mastertable
    GROUP BY
        Account_ID, YEAR(Timestamp), MONTH(Timestamp)
),
TotalAccountsPerMonth AS (
    SELECT
        Year,
        Month,
        COUNT(DISTINCT Account_ID) AS TotalAccounts
    FROM
        MonthlyTransactionCounts
    GROUP BY
        Year, Month
),
Top5PercentThresholds AS (
    SELECT
        m.Year,
        m.Month,
        FLOOR(TotalAccounts * 0.05) AS Threshold
    FROM
        TotalAccountsPerMonth m
),
RankedTransactions AS (
    SELECT
        m.*,
        RANK() OVER (PARTITION BY m.Year, m.Month ORDER BY m.NumTransactions DESC) AS
TransactionRank
    FROM
        MonthlyTransactionCounts m
),
Top5PercentAccountsPerMonth AS (
    SELECT
        r.Account_ID,
        r.Year,
        r.Month
    FROM
        RankedTransactions r
    INNER JOIN Top5PercentThresholds t ON r.Year = t.Year AND r.Month = t.Month
    WHERE r.TransactionRank <= t.Threshold
),
SimplifiedCategories AS (
    SELECT
        TransactionID,
        Account_ID,
        Amount,
        Timestamp,
        MerchantID,
```

```
        CASE
            WHEN Category IN ('Online', 'Travel', 'Food', 'Retail') THEN 'GroupedOtherCategories'
            ELSE 'Others'
        END AS SimplifiedCategory,
        LastLogin
    FROM
        mastertable
),
AnomalyScores AS (
    SELECT
        sc.TransactionID,
        sc.Account_ID,
        sc.Amount,
        sc.Timestamp,
        sc.MerchantID,
        sc.SimplifiedCategory,
        sc.LastLogin,
        CASE
            WHEN sc.Amount > 75.86 THEN 1
            ELSE 0
        END AS AmountScore,
        CASE
            WHEN HOUR(sc.Timestamp) BETWEEN 0 AND 5 THEN 1
            ELSE 0
        END AS TimeScore,
        0 AS RepeatedTransactionScore,
        CASE
            WHEN DATEDIFF(sc.Timestamp, sc.LastLogin) > 180 THEN 1
            ELSE 0
        END AS InactivityScore,
        (
            SELECT
                CASE
                    WHEN COUNT(*) < 5 THEN 1
                    ELSE 0
                END
            FROM
                SimplifiedCategories sub
            WHERE
                sub.MerchantID = sc.MerchantID AND sub.SimplifiedCategory = sc.SimplifiedCategory
        ) AS UncommonMerchantCategoryScore
    FROM
        SimplifiedCategories sc
),
FinalScores AS (
    SELECT
        a.*,
        CASE
            WHEN EXISTS (
                SELECT 1
                FROM Top5PercentAccountsPerMonth tp
                WHERE tp.Account_ID = a.Account_ID AND tp.Year = YEAR(a.Timestamp) AND tp.Month =
MONTH(a.Timestamp)
            ) THEN 0.05
            ELSE 0
        END AS Top5PercentAccountRiskWeightage,
        (
            0.25 * a.RepeatedTransactionScore +
            0.3 * a.AmountScore +
            0.15 * a.InactivityScore +
```

```
            0.2 * a.TimeScore +
            0.03 * a.UncommonMerchantCategoryScore
        ) + CASE
              WHEN EXISTS (
                SELECT 1
                FROM Top5PercentAccountsPerMonth tp
                WHERE tp.Account_ID = a.Account_ID AND tp.Year = YEAR(a.Timestamp) AND tp.Month =
MONTH(a.Timestamp)
              ) THEN 0.07
              ELSE 0
            END AS FinalAnomalyScore
    FROM
        AnomalyScores a
)
SELECT TransactionId,FinalAnomalyScore FROM FinalScores);

SELECT *
FROM score_view;
```

2. Other Queries

2.1. Query 1

```
WITH TransactionSums AS (
  SELECT
    CustomerID,
    Category,
    SUM(Amount) AS TotalAmount
  FROM Transaction_data
  GROUP BY CustomerID, Category
), RankedTransactions AS (
  SELECT
    CustomerID,
    Category,
    TotalAmount,
    RANK() OVER (PARTITION BY Category ORDER BY TotalAmount DESC) AS `Rank`
  FROM TransactionSums
)
SELECT CustomerID, Category, TotalAmount, `Rank`
FROM RankedTransactions
WHERE `Rank` <= 5;
```

2.2. Query 2

```
 WITH CategoryFrequencies AS (
  SELECT
    t.CustomerID,
    t.Category,
    COUNT(*) AS TransactionsCount,
    RANK() OVER (PARTITION BY t.CustomerID ORDER BY COUNT(*) DESC) AS CategoryRank
  FROM Transaction_data t
  GROUP BY t.CustomerID, t.Category
)
SELECT
  cf.CustomerID,
  cf.Category,
  cf.TransactionsCount
FROM CategoryFrequencies cf
WHERE cf.CategoryRank = 1;
```

2.3. Query 3

```
SET sql_mode = 'STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION';
CREATE VIEW CustomerFinancialHealth AS
SELECT DISTINCT
  c.CustomerID,
  c.Name AS Cust_Name,
  FIRST_VALUE(a.AccountBalance) OVER (
    PARTITION BY c.CustomerID
    ORDER BY a.LastLogin DESC
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
  ) AS LatestBalance,
  AVG(t.Amount) AS AverageTransactionAmount,
  MAX(a.LastLogin) AS LastLoginDate
FROM customer_data c
JOIN account a ON c.CustomerID = a.CustomerID
JOIN transaction_data t ON c.CustomerID = t.CustomerID
GROUP BY c.CustomerID, c.Name;
SELECT *
FROM CustomerFinancialHealth;
```

2.4. Query 4

```
WITH AccountActivity AS (
  SELECT
    a.CustomerID,
    AVG(t.Amount) AS AverageAmountSpent,
    STDDEV(t.Amount) AS StdDevAmount
  FROM account a
  JOIN transaction_data t ON a.CustomerID = t.CustomerID
  GROUP BY a.CustomerID
)
SELECT
  a.Account_ID,
  a.CustomerID,
  aa.AverageAmountSpent,
  aa.StdDevAmount,
  t.Amount AS RecentTransactionAmount,
  sv.FinalAnomalyScore
FROM account a
JOIN AccountActivity aa ON a.CustomerID = aa.CustomerID
JOIN transaction_data t ON a.CustomerID = t.CustomerID
JOIN score_view sv ON t.TransactionID = sv.TransactionId
WHERE sv.FinalAnomalyScore > 0.3 -- Threshold for high risk
ORDER BY sv.FinalAnomalyScore DESC;
```

2.5. Query 5

```
SELECT
   m.MerchantID,
   m.MerchantName,
   COUNT(t.TransactionID) AS HighRiskTransactionCount
FROM merchant_data m
JOIN transaction_data t ON m.MerchantID = t.MerchantID
JOIN score_view sv ON t.TransactionID = sv.TransactionID
WHERE sv.FinalAnomalyScore > 0.3
GROUP BY m.MerchantID, m.MerchantName
HAVING COUNT(t.TransactionID) > (
  SELECT AVG(HighRiskTransactionCount) * 2 FROM (
    SELECT COUNT(t.TransactionID) AS HighRiskTransactionCount
    FROM transaction_data t
```

```
    JOIN score_view sv ON t.TransactionID = sv.TransactionID
    WHERE sv.FinalAnomalyScore > 0.3
    GROUP BY t.MerchantID
  ) AS SubQuery
);
```

2.6. Query 6

```
WITH RiskScoreChanges AS (
  SELECT
    t.CustomerID,
    LAG(sv.FinalAnomalyScore) OVER (PARTITION BY t.CustomerID ORDER BY t.Timestamp) AS PreviousScore,
    sv.FinalAnomalyScore,
    t.TransactionID
  FROM transaction_data t
  JOIN score_view sv ON t.TransactionID = sv.TransactionID
)
SELECT
  CustomerID,
  COUNT(TransactionID) AS TransactionsWithIncreasedRisk
FROM RiskScoreChanges
WHERE FinalAnomalyScore > PreviousScore + 0.25
GROUP BY CustomerID;
```

2.7. Query 7

```
SELECT
  CASE
    WHEN c.Age BETWEEN 18 AND 25 THEN '18-25'
    WHEN c.Age BETWEEN 26 AND 35 THEN '26-35'
    WHEN c.Age BETWEEN 36 AND 45 THEN '36-45'
    WHEN c.Age BETWEEN 46 AND 55 THEN '46-55'
    WHEN c.Age BETWEEN 56 AND 65 THEN '56-65'
    ELSE '65+'
  END AS AgeGroup,
  AVG(sv.FinalAnomalyScore) AS AverageAnomalyScore,
  COUNT(*) AS TotalTransactions
FROM customer_data c
INNER JOIN transaction_data t ON c.CustomerID = t.CustomerID
INNER JOIN score_view sv ON t.TransactionID = sv.TransactionID
GROUP BY AgeGroup
ORDER BY AverageAnomalyScore DESC;
```

2.8. Query 8

```
SELECT
  m.Location,
  COUNT(t.TransactionID) AS TotalTransactions,
  AVG(sv.FinalAnomalyScore) AS AverageAnomalyScore,
  SUM(CASE WHEN sv.FinalAnomalyScore > 0.3 THEN 1 ELSE 0 END) AS HighRiskTransactions
FROM merchant_data m
JOIN transaction_data t ON m.MerchantID = t.MerchantID
JOIN score_view sv ON t.TransactionID = sv.TransactionID
GROUP BY m.Location
HAVING HighRiskTransactions >= 3
ORDER BY HighRiskTransactions DESC, AverageAnomalyScore DESC;
```