📘 Financial Transaction Analysis- using SQL

🎯 Project Goal

To utilize **MySQL** for a comprehensive analysis of financial transaction data. The aim is to uncover spending patterns, identify potential anomalies and fraud indicators, and evaluate average transaction metrics over time. This project highlights how structured SQL queries can provide critical business and security insights from raw financial data.

📁 Dataset Summary

Each record represents a financial transaction. It encompasses various transaction dimensions such as timestamps, account activity, customer demographics, and usage contexts like devices, channels, and merchant interactions.

https://www.kaggle.com/datasets/valakhorasani/bank-transaction-dataset-for-fraud-detection

📌 Notable Fields:

- **TransactionID**: Unique identifier for each transaction
- **AccountID**: Identifier linking each transaction to a customer account
- **TransactionAmount**: Monetary value of the transaction
- **TransactionDate**: Date and time of transaction execution
- **TransactionType**: Either 'Credit' or 'Debit'
- **Location**: U.S. city where transaction occurred
- **DeviceID**: Device used for transaction
- **IP Address**: IPv4 address involved
- **MerchantID**: Vendor identifier
- **AccountBalance**: Post-transaction account balance
- **PreviousTransactionDate**: Time of the account's last transaction
- **Channel**: Access medium (Online, ATM, or Branch)
- **CustomerAge**, **CustomerOccupation**: Demographic features
- **TransactionDuration**: Transaction execution time (in seconds)
- **LoginAttempts**: Number of login attempts prior to transaction

---

🧹 Step 1: Data Loading & Preparation

Initial cleanup focuses on ensuring that transaction values and timestamps are properly formatted, and that records with missing or inconsistent entries are excluded from further analysis.

```sql
-- 1 Data Cleaning:- Checking for missing values, editing the data types and

CREATE TABLE prepared_transactions AS SELECT TransactionID,
    AccountID,
    ROUND(TransactionAmount, 2) AS TransactionAmount,
    DATE(TransactionDate) AS TransactionDate,
    TransactionType,
    Location,
    DeviceID,
    MerchantID,
    ROUND(AccountBalance, 2) AS AccountBalance,
    LoginAttempts,
    DATE(PreviousTransactionDate) AS PreviousTransactionDate,
    Channel,
    CustomerAge,
    CustomerOccupation,
    TransactionDuration FROM
    bank_transactions_data
WHERE
    TransactionAmount IS NOT NULL
        AND TransactionDate IS NOT NULL;
```

📊 Step 2: Identifying Transaction Trends

A. Month-by-Month and Quarter by-Quarter Transaction Summary

```
35      -- 2 Trend Analysis
36      -- a. Month-by-Month Transaction Summary
37  •   SELECT
38          DATE_FORMAT(TransactionDate, '%Y-%m') AS Month,
39          COUNT(*) AS TotalTransactions,
40          SUM(TransactionAmount) AS TotalSpent
41      FROM
42          prepared_transactions
43      GROUP BY Month
44      ORDER BY Month;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Month	TotalTransactions	TotalSpent
2023-01	207	63899.04000000004
2023-02	218	57516.10000000003
2023-03	197	61036.12000000002
2023-04	161	41003.840000000004
2023-05	220	62868.01000000002
2023-06	212	61559.57999999998
2023-07	195	58861.390000000014
2023-08	224	71437.75999999998
2023-09	214	72832.25
2023-10	226	64705.61999999995
2023-11	221	66051.12999999998
2023-12	204	63719.429999999986
2024-01	13	2065.2999999999997

```
46      -- Quater-By-Quater Transaction Summary
47  •   SELECT
48          CONCAT(YEAR(TransactionDate), '-Q', QUARTER(TransactionDate)) AS Quarter,
49          COUNT(*) AS TotalTransactions,
50          SUM(TransactionAmount) AS TotalSpent
51      FROM
52          prepared_transactions
53      GROUP BY Quarter
54      ORDER BY Quarter;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Quarter	TotalTransactions	TotalSpent
2023-Q1	622	182451.25999999983
2023-Q2	593	165431.42999999996
2023-Q3	633	203131.40000000005
2023-Q4	651	194476.1800000002
2024-Q1	13	2065.2999999999997

This query retrieves the monthly and quarterly transaction trends.

We observe a steady increase over the quarters, especially a spike in Q3, which may suggest seasonal influence or a business promotion. Tracking such patterns helps forecast future transaction load.

B. Summary by Transaction Type

```
46      -- b.Summary by Transaction Type
47 •    SELECT
48          TransactionType,
49          COUNT(*) AS Count,
50          SUM(TransactionAmount) AS Total,
51          AVG(TransactionAmount) AS Average
52      FROM
53          prepared_transactions
54      GROUP BY TransactionType;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

TransactionType	Count	Total	Average
Debit	1944	573463.000000001	294.99125514403346
Credit	568	174092.5700000001	306.5010035211269

This query gets count of transactions for different transaction type.

Most transactions are debit-based, indicating routine customers activity like bill payments and shopping. Credit transactions, while fewer in count, tend to have higher average values.

C. Summary by Transaction Channels

```
56      -- C.Getting Transaction detail of various Transaction channels
57 •    SELECT
58          Channel,
59          COUNT(*) AS Count,
60          SUM(TransactionAmount) AS Total_Transaction_Amt,
61          AVG(TransactionAmount) AS Average_Transaction_Amt
62      FROM
63          prepared_transactions
64      GROUP BY Channel;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Channel	Count	Total_Transaction_Amt	Average_Transaction_Amt
ATM	833	256331.43	307.7208043217287
Online	811	241041.1399999999	297.21472256473476
Branch	868	250183.0000000001	288.22926267281116

This query retrieves summary of different Transaction channels.

We observed that Branch dominate in transaction count. ATM-based transactions are frequent but not as much as Branch and are typically associated with larger or more sensitive transactions

D. Spending Behavior by Occupation and Channel

```sql
-- D.Spending Behavior by Occupation and Channel
SELECT
    CustomerOccupation,
    Channel,
    COUNT(*) AS NumberOfTransactions
FROM
    prepared_transactions
GROUP BY CustomerOccupation , channel
ORDER BY NumberOfTransactions DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerOccupation	Channel	NumberOfTransactions
Student	Branch	224
Student	ATM	222
Doctor	Branch	219
Engineer	Branch	218
Doctor	ATM	215
Engineer	Online	214
Student	Online	211
Retired	Branch	207
Retired	ATM	203
Doctor	Online	197
Engineer	ATM	193
Retired	Online	189

- **Students** are the most active users across all channels (top 2 spots).
- **Doctors and Engineers** also show strong activity in both Branch and ATM channels.
- **Retired customers** have lower transaction volumes comparatively.

E. Spending behavior by Age -group

Age groups 60+ and 46–60 demonstrate the highest spending behavior, making them key segments for premium financial services. Younger users transact less frequently but in bigger amounts.

```sql
-- E. Spending Behavior by AGE group
SELECT
    CASE
        WHEN CustomerAge BETWEEN 18 AND 25 THEN '18-25'
        WHEN CustomerAge BETWEEN 26 AND 35 THEN '26-35'
        WHEN CustomerAge BETWEEN 36 AND 45 THEN '36-45'
        WHEN CustomerAge BETWEEN 46 AND 60 THEN '46-60'
        WHEN CustomerAge > 60 THEN '60+'
        ELSE 'Unknown'
    END AS AgeGroup,
    COUNT(*) AS NumberOfTransactions,
    ROUND(AVG(TransactionAmount), 2) AS AverageTransactionAmount
FROM
    prepared_transactions
GROUP BY AgeGroup
ORDER BY AverageTransactionAmount DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AgeGroup	NumberOfTransactions	AverageTransactionAmount
18-25	475	315.9
26-35	486	308.13
60+	563	303.13
36-45	303	281.5
46-60	685	279.99

F. Avg Transaction Duration per Channel

```
94    -- F Checking the mean of the transaction duration in each channel
95    SELECT
96        Channel, AVG(TransactionDuration)
97    FROM
98        prepared_transactions
99    GROUP BY Channel;
100
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Channel	AVG(TransactionDuration)
ATM	122.0912
Online	120.3058
Branch	116.6751

This query retrieves the average transaction duration for each transaction channel.

We observed that transactions done via ATM and Online take longer to process. Branch transactions are faster and reliable.

Step 3: Flagging Anomalies

A. Unusual Login Patterns

Devices and accounts with excessive login attempts should be flagged. This query helps to retrieve repeated access failures which are strong fraud indicators when followed by large transactions.

```
102    -- 3 Flagging Anomalies
103
104    --  A. Unusual Login Patterns
105    SELECT
106        *
107    FROM
108        prepared_transactions
109    WHERE
110        LoginAttempts >= 5;
111
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TransactionID	AccountID	TransactionAmount	TransactionDate	TransactionType	Location	DeviceID	MerchantID	AccountBalance
TX000027	AC00441	246.93	2023-04-17	Debit	Miami	D000046	M029	673.35
TX000148	AC00161	514.95	2023-04-13	Debit	New York	D000109	M056	421.93
TX000275	AC00454	1176.28	2023-12-20	Credit	Kansas City	D000476	M074	323.69
TX000395	AC00326	6.3	2023-12-14	Debit	Columbus	D000539	M017	7697.68
TX000415	AC00495	83.5	2023-05-15	Debit	Dallas	D000446	M100	1749.79
TX000464	AC00417	302.16	2023-10-18	Debit	Kansas City	D000123	M014	3876.61
TX000492	AC00318	505.19	2023-03-14	Debit	Kansas City	D000660	M023	5326.68
TX000509	AC00353	127	2023-09-15	Credit	Nashville	D000024	M015	431.16
TX000686	AC00071	119.3	2023-08-28	Debit	Mesa	D000597	M058	319.04
TX000692	AC00418	25.94	2023-10-16	Credit	San Jose	D000232	M003	4815.11
TX000694	AC00011	733.29	2023-03-15	Debit	Virginia Be...	D000618	M032	10427
TX000793	AC00468	253.55	2023-01-11	Debit	Jacksonville	D000112	M007	2797.26
TX000954	AC00187	119.95	2023-04-10	Credit	Philadelphia	D000292	M092	2596.61
TX000962	AC00174	353.59	2023-08-11	Credit	New York	D000311	M046	1822.34
TX001043	AC00246	180.65	2023-12-04	Debit	Charlotte	D000552	M012	816.01
TX001122	AC00426	303.97	2023-02-02	Debit	Fort Worth	D000215	M056	1590.88
TX001137	AC00032	341.67	2023-03-06	Debit	Omaha	D000160	M037	1442.29

B. Cities with Suspicious Logins

```
112    -- B. Filtering cities with high number of login
113    SELECT
114        Location, COUNT(*) AS Suspicious_Login
115    FROM
116        prepared_transactions
117    WHERE
118        LoginAttempts >= 5
119    GROUP BY Location
120    HAVING Suspicious_Login >= 2
121    ORDER BY Suspicious_Login DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Location	Suspicious_Login
Kansas City	3
Jacksonville	3
New York	2
Virginia Beach	2
Fort Worth	2
Los Angeles	2
San Diego	2

This query retrieves the with 2 or more suspicious login attempts.

We observed Kansas City and Jacksonville, appear frequently in high login attempts.

C. Standard Deviation Based Outliers

This query retrieves transactions outside the ±3 standard deviation range. These are flagged as anomalies. This technique is effective for isolating both very small and unusually large transactions for audit.

```
122    -- C.Outlier Transactions Based on Z-Scores
123    SELECT
124        AVG(TransactionAmount) AS avg_amt,
125        STDDEV(TransactionAmount) AS std_amt
126    INTO @avg_amt , @std_amt FROM
127        prepared_transactions;
128    -- Use those to identify potential outliers
129    SELECT
130        TransactionID,
131        AccountID,
132        TransactionAmount,
133        Channel,
134        Location,
135        LoginAttempts,
136        TransactionType,
137        CustomerAge
138    FROM
139        prepared_transactions
140    WHERE
141        TransactionAmount > @avg_amt + 3 * @std_amt
142            OR TransactionAmount < @avg_amt - 3 * @std_amt;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TransactionID	AccountID	TransactionAmount	Channel	Location	LoginAttempts	TransactionType	CustomerAge
TX000075	AC00265	1212.51	Branch	Indianapolis	1	Debit	20
TX000086	AC00098	1340.19	Online	Austin	1	Credit	54
TX000177	AC00363	1362.55	ATM	El Paso	1	Debit	29
TX000191	AC00396	1422.55	Branch	Washington	1	Debit	79
TX000275	AC00454	1176.28	ATM	Kansas City	5	Credit	54

D. Locations with Abnormal Transactions

```sql
-- D.Getting Locations with highest outlier transactions
SELECT
    Location, COUNT(*) AS Number_of_Outlier_Transaction
FROM
    (SELECT
        TransactionID,
            AccountID,
            TransactionAmount,
            Channel,
            Location,
            LoginAttempts,
            TransactionType,
            CustomerAge
    FROM
        prepared_transactions
    WHERE
        TransactionAmount > @avg_amt + 3 * @std_amt
            OR TransactionAmount < @avg_amt - 3 * @std_amt) AS Out_lierTable
GROUP BY Location
HAVING Number_of_Outlier_Transaction >= 2
ORDER BY Number_of_Outlier_Transaction DESC;
```

This query retrieves the cities with the abnormal transactions i.e +/-3 Standard deviation from normal transactions.

We observed that certain cities show disproportionate high-value transactions.

Location	Number_of_Outlier_Transaction
Austin	4
El Paso	3
Washington	2
San Antonio	2
Oklahoma City	2
Miami	2
Portland	2
San Francisco	2
Phoenix	2
Louisville	2
Columbus	2
Memphis	2
Colorado Springs	2
New York	2
Las Vegas	2

Step 4: Insight Generation & Summary Metrics

A. Average Spend by Channel

```
167     -- 4: Insight Generation & Summary Metrics
168     -- A. Average Spend by Channel
169 •   SELECT
170         Channel,
171         COUNT(*) AS Transactions,
172         AVG(TransactionAmount) AS AverageValue
173     FROM
174         prepared_transactions
175     GROUP BY Channel;
176
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

Channel	Transactions	AverageValue
ATM	833	307.7208043217287
Online	811	297.21472256473476
Branch	868	288.22926267281116

This query retrieves summary of different Transaction channels.

We observed that Branch dominate in transaction count. ATM-based transactions are frequent but not as much as Branch and are typically associated with larger or more sensitive transactions

.

B. Cities with High Transaction Frequency

```
177     -- B. Cities with Highest Transaction Activity
178 •   SELECT
179         Location,
180         COUNT(*) AS Frequency,
181         SUM(TransactionAmount) AS TotalSpend
182     FROM
183         prepared_transactions
184     GROUP BY Location
185     ORDER BY TotalSpend DESC
186     LIMIT 10;
187
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetc

Location	Frequency	TotalSpend
Austin	59	22740.9
Oklahoma City	68	21716.04000000001
Memphis	63	21170.53
Fort Worth	70	20776.74
Detroit	63	20609.76
Jacksonville	60	20519.47
Tucson	67	20459.75999999999
Colorado Springs	60	20344.629999999994
San Jose	59	20127.870000000006
Los Angeles	69	19675.75

This query retrieves the list of 10 cities with highest frequency of transactions and the total amount spend in each city.

C. High-Risk Devices

```
188     -- C.Frequent Devices in High-Risk Scenarios
189 •   SELECT
190         DeviceID, COUNT(*) AS SuspiciousLogins
191     FROM
192         prepared_transactions
193     WHERE
194         LoginAttempts >= 5
195     GROUP BY DeviceID
196     ORDER BY COUNT(*) DESC
197     LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

DeviceID	SuspiciousLogins
D000046	1
D000109	1
D000476	1
D000539	1
D000446	1

This query retrieves the devices appearing in multiple suspicious scenarios (high login attempts). These are security concerns. These should be blacklisted or flagged for manual review.

.

D. Top 5 Suspicious Login Devices

This query retrieves transaction records of the devices on which the suspicious logins were observed. The query was made to return top 5 transactions based on the transaction amount.

```
198     -- D. Details of the top 5 suspicious login from those Devices
199 •   WITH Compremised_Devices AS (
200         SELECT DeviceID
201         FROM prepared_transactions
202         WHERE LoginAttempts >= 5
203         GROUP BY DeviceID
204         ORDER BY COUNT(*) DESC
205         LIMIT 5
206     )
207
208     SELECT pt.*
209     FROM prepared_transactions pt
210     JOIN Compremised_Devices td ON pt.DeviceID = td.DeviceID
211     Where LoginAttempts>=5
212     Order by pt.TransactionAmount DESC
213     Limit 5 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TransactionID	AccountID	TransactionAmount	TransactionDate	TransactionType	Location	DeviceID	MerchantID	AccountBalance	PreviousTransactionDate	Chan
TX000275	AC00454	1176.28	2023-12-20	Credit	Kansas City	D000476	M074	323.69	2024-11-04	ATM
TX000148	AC00161	514.95	2023-04-13	Debit	New York	D000109	M056	421.93	2024-11-04	Online
TX000027	AC00441	246.93	2023-04-17	Debit	Miami	D000046	M029	673.35	2024-11-04	ATM
TX000415	AC00495	83.5	2023-05-15	Debit	Dallas	D000446	M100	1749.79	2024-11-04	Branc
TX000395	AC00326	6.3	2023-12-14	Debit	Columbus	D000539	M017	7697.68	2024-11-04	Branc

Summary & Recommendations

📌 Key Summary Points:

- **Branch channels** have the highest transaction count, possibly due to trust or habit in traditional banking.
- **Students** are the most active users across all channels, while **Doctors** and **Engineers** are high in transaction value.
- **Age groups 60+ and 46–60** demonstrate the highest overall spending behavior.
- **ATM and Online transactions** take longer, indicating potential bottlenecks or verification delays.
- **Cities like Kansas City and Jacksonville** show repeated login anomalies.
- **Several devices** appear frequently in suspicious login patterns and should be reviewed for fraud.

💡 Actionable Recommendations:

1. **Enhance fraud detection**:
    - Flag accounts with high login attempts before high-value transactions.
    - Monitor devices repeatedly linked with outlier behavior.
2. **Channel optimization**:
    - Investigate ATM and Online transaction delays to improve performance.
    - Consider expanding Branch self-service kiosks due to high footfall.
3. **Targeted financial offerings**:
    - Provide tailored services for high-value customer segments (60+ and working professionals).
    - Create student-friendly banking packages given their high transaction volume.
4. **Geo-based security**:
    - Deploy enhanced security for high-risk cities (e.g., rate-limiting, IP verification).
    - Use historical location data to build location consistency models.
5. **Continuous anomaly tracking**:
    - Regularly run deviation-based filters to detect outliers.
    - Cross-reference anomaly logs with device, channel, and demographic data for trend learning.