Fraud Detection and Risk Analysis in Cross River Bank

Objective

To analyze structured and unstructured data independently to identify loan fraud, assess customer risk, and understand customer behavior. This case study focuses on the loan and transaction data of Cross River Bank, leveraging MySQL and MongoDB for a comprehensive analysis.

Problem Statement

Cross River Bank, a U.S.-based financial institution, faces challenges in detecting fraudulent activities and assessing customer risk effectively. With an increasing volume of loans and transactions, traditional manual methods are insufficient. The bank requires an automated system to analyze both structured data (e.g., customer, loan, and transaction data) and unstructured data (e.g., customer feedback, behavior logs) to identify patterns of fraud, optimize lending policies, and improve customer satisfaction.

Files Included for Analysis

The following files contain structured and unstructured data used for this case study:

1. Customer Table (CSV): Includes customer details such as ID, name, age, income, credit score, and address.
2. Loan Table (CSV): Contains loan information, including loan ID, customer ID, loan amount, purpose, and default risk.
3. Transaction Table (CSV): Details transactions related to loans, including transaction type, amount, and date.
4. Behavior Logs (JSON): Captures customer activities such as login, missed payments, and early repayments.
5. Customer Feedback (JSON): Includes feedback from customers, sentiment scores, escalation flags, and feedback categories.

Database Creation Details

MySQL (Structured Data):

1. Customer Table:
    a. Fields: customer_id (Primary Key), name, age, income, credit_score, address, customer_since.

b. File: `customer_table.csv`.
c. Purpose: Stores customer demographic and financial details.

2. Loan Table:
   a. Fields: loan_id (Primary Key), customer_id (Foreign Key), loan_amount, loan_purpose, default_risk.
   b. File: `loan_table.csv`.
   c. Purpose: Stores loan details linked to customers.
3. Transaction Table:
   a. Fields: transaction_id (Primary Key), loan_id (Foreign Key), customer_id (Foreign Key), transaction_amount, transaction_type, transaction_date.
   b. File: `transaction_table.csv`.
   c. Purpose: Records loan-related transactions.

MongoDB (Unstructured Data):

1. Behavior Logs Collection:
   a. Fields: customer_id, timestamp, action, device, ip_address, location.
   b. File: `shuffled_balanced_behavior_logs.json`.
   c. Purpose: Tracks customer behavior patterns.
2. Customer Feedback Collection:
   a. Fields: loan_id, customer_id, feedback_text, sentiment_score, feedback_category, escalation_flag, escalation_reason.
   b. File: `realistic_customer_feedback.json`.
   c. Purpose: Stores customer feedback and sentiment analysis.

Tasks with MySQL

- Customer Risk Analysis: Identify customers with low credit scores and high-risk loans to predict potential defaults and prioritize risk mitigation strategies. **BASIC**
- Loan Purpose Insights: Determine the most popular loan purposes and their associated revenues to align financial products with customer demands. **BASIC**
- High-Value Transactions: Detect transactions that exceed 30% of their respective loan amounts to flag potential fraudulent activities. **BASIC**
- Missed EMI Count: Analyze the number of missed EMIs per loan to identify loans at risk of default and suggest intervention strategies. **BASIC**
- Regional Loan Distribution: Examine the geographical distribution of loan disbursements to assess regional trends and business opportunities. **INTERMEDIATE**

- Loyal Customers: List customers who have been associated with Cross River Bank for over five years and evaluate their loan activity to design loyalty programs. **INTERMEDIATE**
- High-Performing Loans: Identify loans with excellent repayment histories to refine lending policies and highlight successful products. **INTERMEDIATE**
- Age-Based Loan Analysis: Analyze loan amounts disbursed to customers of different age groups to design targeted financial products. **INTERMEDIATE**
- Seasonal Transaction Trends: Examine transaction patterns over years and months to identify seasonal trends in loan repayments. **Advanced**
- Fraud Detection: Highlight potential fraud by identifying mismatches between customer address locations and transaction IP locations. **Advanced**
- Repayment History Analysis: Rank loans by repayment performance using window functions. **Advanced**
- Credit Score vs. Loan Amount: Compare average loan amounts for different credit score ranges. **Advanced**
- Top Borrowing Regions: Identify regions with the highest total loan disbursements. **Advanced**
- Early Repayment Patterns: Detect loans with frequent early repayments and their impact on revenue. **Advanced**
- Feedback Correlation: Correlate customer feedback sentiment scores with loan statuses. **Advanced**

Tasks with MongoDB

CRUD Operations

1. **Insert New Feedback. BASIC**

Input: Adding feedback for a customer and loan, capturing sentiments and details about the loan process.

Sample Input:

```
{
  "loan_id": 2001,
  "customer_id": 101,
  "feedback_text": "The loan approval process was seamless, but the interest rate was slightly higher than expected.",
```


```
    "sentiment_score": 0.7,
    "feedback_category": "Approval Process",
    "escalation_flag": false,
    "escalation_reason": null,
    "timestamp": "2024-11-15T10:30:00Z"
}
```


Expected Database Entry:


```
{
    "loan_id": 2001,
    "customer_id": 101,
    "feedback_text": "The loan approval process was seamless, but the interest rate was slightly higher than expected.",
    "sentiment_score": 0.7,
    "feedback_category": "Approval Process",
    "escalation_flag": false,
    "escalation_reason": null,
    "timestamp": "2024-11-15T10:30:00Z"
}
```


---

2. **Update Escalation Flags. BASIC**

Input: Update escalation flags in feedback to include specific reasons for unresolved customer complaints.

Sample Feedback Before Update:

```
{
  "loan_id": 2002,
  "customer_id": 102,
  "feedback_text": "Customer service was unresponsive to my queries.",
  "sentiment_score": -0.6,
  "feedback_category": "Customer Service",
  "escalation_flag": true,
  "escalation_reason": null
}
```


Feedback After Update:


```
{
  "loan_id": 2002,
  "customer_id": 102,
  "feedback_text": "Customer service was unresponsive to my queries.",
  "sentiment_score": -0.6,
  "feedback_category": "Customer Service",
  "escalation_flag": true,
  "escalation_reason": "Delayed response from customer service"
}
```


---

3. **Remove Duplicate Behavior Logs. BASIC**

Input: Remove duplicate entries with the same customer_id and timestamp.

Sample Logs Before Removal:


```
[
  { "customer_id": 103, "timestamp": "2024-11-15T12:00:00Z", "action": "Login", "device": "Mobile" },
  { "customer_id": 103, "timestamp": "2024-11-15T12:00:00Z", "action": "Login", "device": "Mobile" }
]
```


Logs After Removal:


```
[
  { "customer_id": 103, "timestamp": "2024-11-15T12:00:00Z", "action": "Login", "device": "Mobile" }
]
```


---

4. **Retrieve Positive Feedback. BASIC**

Input: Retrieve feedback entries with sentiment scores greater than 0.5.

Sample Output:


```
[
  {
    "loan_id": 2003,
    "customer_id": 104,
    "feedback_text": "The repayment terms were flexible and suited my needs.",
    "sentiment_score": 0.9
  },
  {
    "loan_id": 2004,
```

```
    "customer_id": 105,
    "feedback_text": "Customer service was prompt and helpful during the loan application process.",
    "sentiment_score": 0.8
  }
]
```


---

5. Fetch Logs for 'Missed Payment' Actions. BASIC

Input: Retrieve behavior logs where the action is "Missed Payment."

Sample Output:


```
[
  {
    "customer_id": 106,
    "timestamp": "2024-10-10T08:45:00Z",
    "device": "Desktop",
    "ip_address": "192.168.1.101"
  },
  {
    "customer_id": 107,
    "timestamp": "2024-10-12T14:00:00Z",
    "device": "Mobile",
    "ip_address": "10.0.0.45"
  }
```

]

---

6. Analyze customer behavior and feedback to identify high-risk customers and prioritize operational improvements for fraud detection and customer satisfaction using an aggregate pipeline. ADVANCE (optional)

Steps in the Aggregate Pipeline:

1. Join Feedback and Behavior Logs
Use the $lookup stage to combine feedback from the customer_feedback collection with behavior logs from the behavior_logs collection, matching on customer_id.

Hint: In MongoDB, $lookup functions like a SQL LEFT JOIN. You can specify:

- from: The name of the second collection (behavior_logs).
- localField: The field in the first collection to match (customer_id).
- foreignField: The field in the second collection to match (customer_id).
- as: The name of the new field where the matched documents will be stored (e.g., behavior_data).

2. Flatten Joined Data
Apply $unwind to flatten the nested array created by $lookup. This ensures a clean structure for subsequent aggregation stages.

3. Aggregate Feedback and Behavioral Metrics
Group the combined data by customer_id to compute:

- Average sentiment scores.
- Total missed payments.
- Total session durations.
- Count of unresolved escalations.

4. Calculate Risk Scores
Add a composite risk score using $addFields, incorporating:

- Missed payments (weighted heavily as an indicator of risk).
- Negative sentiment (calculated as 1 - avgSentiment).
- Total session durations (scaled if necessary).

- Escalation counts.

5. Identify Device Usage Trends
Group behavior logs by device to detect preferences and anomalies in usage patterns, especially for high-risk actions like "Missed Payment."

6. Detect Session Outliers
Use $bucket to identify customers with session durations that fall into extreme ranges (e.g., above the 90th percentile or below the 10th percentile).

7. Escalation Analysis
Filter for customers with:

- Non-zero escalation counts.
- High composite risk scores.
  Highlight these customers for immediate follow-up and prioritization.

Outcome:
This revised pipeline leverages $lookup to seamlessly integrate datasets, providing a comprehensive view of customer behavior and risk, enabling actionable insights for fraud detection and customer service optimization**.**

Deliverables:

1. Insights from MySQL queries to identify high-risk customers, analyze loan performance, and detect fraudulent activities.
2. MongoDB CRUD and aggregation pipeline results highlighting customer behavior trends and feedback analysis.
3. A detailed report summarizing findings from both databases to assist Cross River Bank in optimizing its risk assessment and fraud detection mechanisms.

Submission Instructions:

To submit your assignment, please follow these guidelines:

- Ensure that your assignment is fully completed.
- Push your assignment to a GitHub repository.
- Share the repository link by including it in a text, Word, or PDF file.

Submit the file/text containing the repository link via Vlearn.