Name	**SHANTANU KAUSHIK [ABADS BATCH 16]**
Email	**shankshk@gmail.com**
Phone	**9887779477**
Project Overview:	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.
Task 1	**Customer Risk Analysis:** Identify customers with low credit scores and high-risk loans to predict potential defaults and prioritize risk mitigation strategies.
Solution 1	SELECT c.customer_id, c.name, c.credit_score, l.loan_id, l.default_risk FROM cross_river_bank.customer_table as c JOIN cross_river_bank.loan_table as l ON c.customer_id = l.customer_id WHERE c.credit_score < 600 & l.default_risk = 'High' GROUP BY c.customer_id, c.name, c.credit_score, l.loan_id, l.loan_amount, l.default_risk ORDER BY c.credit_score ASC;

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

customer_id	name	credit_score	loan_id	default_risk
72	Jacob Myers	300	1042	High
72	Jacob Myers	300	356	Low
413	Lynn Juarez	300	1368	Medium
413	Lynn Juarez	300	764	Low
533	William Bishop	301	959	Medium
384	Jennifer Kennedy	301	2646	Medium
533	William Bishop	301	1305	Medium

Task 2	**Loan Purpose Insights:** Determine the most popular loan purposes and their associated revenues to align financial products with customer demands

Solution 2

```
SELECT
   l.loan_purpose,
   COUNT(l.loan_id) AS total_loans,
   SUM(l.loan_amount) AS total_revenue
FROM
   cross_river_bank.loan_table AS l
WHERE
   l.loan_status IN ('Approved', 'Closed')
GROUP BY
   l.loan_purpose
ORDER BY
   total_loans DESC,
   total_revenue DESC;
```

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

loan_purpose	total_loans	total_revenue
Business Expansion	407	101616176
Education	392	97899697
Medical	389	97723847
Home Renovation	373	91775511

Task 3

High-Value Transactions: Detect transactions that exceed 30% of their respective loan amounts to flag potential fraudulent activities

Solution 3

```
SELECT
   t.transaction_id,
   t.customer_id,
   t.loan_id,
   t.transaction_date,
   t.transaction_amount,
   l.loan_amount,
   (t.transaction_amount / l.loan_amount) * 100 AS transaction_percentage
```

```
FROM
   cross_river_bank.transaction_table AS t
JOIN
   cross_river_bank.loan_table AS l ON t.loan_id = l.loan_id
WHERE
   (t.transaction_amount / l.loan_amount) > 0.30
ORDER BY
   transaction_percentage DESC;
```

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

transaction_id	customer_id	loan_id	transaction_date	transaction_amount	loan_amount	transaction_percentage
487	247	2998	12/17/2021 8:00	45399	6375	712.1412
4279	989	328	10/26/2021 11:16	44032	6280	701.1465
2757	427	1022	6/9/2020 18:59	49909	7232	690.1134
1022	275	379	5/2/2020 4:40	40986	6085	673.5579
2949	259	597	9/4/2020 5:46	43017	6677	644.2564
4299	126	597	3/2/2022 19:01	42407	6677	635.1206
3350	152	2970	11/22/2022 8:15	42041	6627	634.3896

Result 3

Task 4

Missed EMI Count: Analyze the number of missed EMIs per loan to identify loans at risk of default and suggest intervention strategies

Solution 4

```
SELECT
   l.loan_id,
   l.customer_id,
   COUNT(t.transaction_id) AS total_missed_emis,
   l.loan_amount,
   l.loan_status,
   c.name AS customer_name,
   c.credit_score
FROM
   cross_river_bank.transaction_table AS t
JOIN
   cross_river_bank.loan_table AS l ON t.loan_id = l.loan_id
JOIN
```

```
    cross_river_bank.customer_table AS c ON l.customer_id = c.customer_id
WHERE
    t.transaction_type = 'EMI Payment'
    AND t.status = 'Failed'
GROUP BY
    l.loan_id, l.customer_id, l.loan_amount, l.loan_status, c.name, c.credit_score
HAVING
    total_missed_emis > 0
ORDER BY
    total_missed_emis DESC;
```

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

loan_id	customer_id	total_missed_emis	loan_amount	loan_status	customer_name	credit_score
2888	731	3	48938	Closed	Eric Mcdonald	800
378	85	2	60428	Rejected	Megan Leonard	749
1958	128	2	411667	Approved	Justin Turner	650
814	176	2	189623	Approved	Miranda Rodriguez	482
1799	272	2	450556	Rejected	Katelyn Simpson	753
621	378	2	189860	Defaulted	Evan Jones	385
2748	467	2	151355	Closed	Jonathon Johnson	394

Result 5

Task 5

Regional Loan Distribution: Examine the geographical distribution of loan disbursements to assess regional trends and business opportunities.

Solution 5

```
SELECT
    c.address,
    COUNT(l.loan_id) AS total_loans_disbursed,
    SUM(l.loan_amount) AS total_loan_amount,
    AVG(l.loan_amount) AS average_loan_amount
FROM
    cross_river_bank.loan_table AS l
JOIN
    cross_river_bank.customer_table AS c ON l.customer_id = c.customer_id
WHERE
    l.loan_status IN ('Approved', 'Closed')
```

```
GROUP BY
   c.address
ORDER BY
   total_loan_amount DESC;
```

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

address	total_loans_disbursed	total_loan_amount	average_loan_amount
PSC 5810, Box 0062, APO AA 90331	6	1801040	300173.3333
7772 Brandon Spring Suite 742, Luisfurt, ME 75...	6	1722817	287136.1667
22879 Anna Mount Suite 571, Derektown, NM 6...	5	1698191	339638.2000
274 Karen Meadows, New Jessica, AK 21632	6	1656534	276089.0000
4269 Maurice Fords, South Christopher, MA 90...	5	1635470	327094.0000
1238 Maxwell Common, West Kathleen, TN 15476	5	1564919	312983.8000
USCGC Henry, FPO AE 51819	4	1548493	387123.2500

Result 6

Task 6

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.

Solution 6

```
SELECT
   c.customer_id,
   c.name,
   c.customer_since,
   TIMESTAMPDIFF(YEAR, STR_TO_DATE(c.customer_since, '%m/%d/%Y'), CURDATE()) AS years_with_bank,
   COUNT(l.loan_id) AS total_loans,
   SUM(l.loan_amount) AS total_loan_amount,
   AVG(l.loan_amount) AS average_loan_amount
FROM
   cross_river_bank.customer_table AS c
LEFT JOIN
   cross_river_bank.loan_table AS l ON c.customer_id = l.customer_id
WHERE
   TIMESTAMPDIFF(YEAR, STR_TO_DATE(c.customer_since, '%m/%d/%Y'), CURDATE()) > 5
GROUP BY
```

```
    c.customer_id, c.name, c.customer_since
ORDER BY
    years_with_bank DESC, total_loan_amount DESC;
```

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

customer_id	name	customer_since	years_with_bank	total_loans	total_loan_amount	average_loan_amount
241	Holly Molina	12/4/2014	10	5	1677468	335493.6000
228	Lindsay Jenkins	12/16/2014	10	6	1453074	242179.0000
684	Nicholas Skinner	12/15/2014	10	6	1378769	229794.8333
616	Robin Carter	12/4/2014	10	4	1065219	266304.7500
387	Tanya Mann	12/17/2014	10	2	616022	308011.0000
917	Keith Armstrong	12/10/2014	10	3	612637	204212.3333
99	James Washington	12/2/2014	10	3	500771	166923.6667

Result 8

Output

Task 7

High-Performing Loans: Identify loans with excellent repayment histories to refine lending policies and highlight successful products.

Solution 7

```
SELECT
    l.loan_id,
    l.loan_purpose,
    sum(l.loan_amount),
    avg(l.repayment_history)
FROM
    cross_river_bank.loan_table AS l
JOIN
    cross_river_bank.customer_table AS c ON l.customer_id = c.customer_id
WHERE
    l.repayment_history >= 9
    AND l.loan_status = 'Closed'
group by l.loan_purpose, l.loan_id
```

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

loan_id	loan_purpose	sum(l.loan_amount)	avg(l.repayment_history)
852	Education	447708	10.0000
2824	Home Renovation	185327	9.0000
2464	Medical	122199	9.0000
97	Home Renovation	319264	10.0000
13	Home Renovation	174410	9.0000
2060	Education	466370	9.0000
236	Business Expansion	332705	10.0000

Task 8

Age-Based Loan Analysis: Analyze loan amounts disbursed to customers of different age groups to design targeted financial products.

Solution 8

```
SELECT
   CASE
      WHEN c.age < 25 THEN 'Under 25'
      WHEN c.age BETWEEN 25 AND 34 THEN '25-34'
      WHEN c.age BETWEEN 35 AND 44 THEN '35-44'
      WHEN c.age BETWEEN 45 AND 54 THEN '45-54'
      WHEN c.age BETWEEN 55 AND 64 THEN '55-64'
      ELSE '65 and above'
   END AS age_group,
   COUNT(l.loan_id) AS total_loans,
   SUM(l.loan_amount) AS total_loan_amount,
   AVG(l.loan_amount) AS average_loan_amount
FROM
   cross_river_bank.loan_table AS l
JOIN
   cross_river_bank.customer_table AS c ON l.customer_id = c.customer_id
GROUP BY
   age_group
ORDER BY
   total_loan_amount DESC;
```

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

age_group	total_loans	total_loan_amount	average_loan_amount
55-64	659	165148589	250604.8392
45-54	596	154934904	259957.8926
35-44	535	135604961	253467.2168
25-34	500	130399920	260799.8400
Under 25	358	86196808	240773.2067
65 and above	352	82594510	234643.4943

Result 10

Task 9

Seasonal Transaction Trends: Examine transaction patterns over years and months to identify seasonal trends in loan repayments.

Solution 9

```
SELECT
   YEAR(STR_TO_DATE(t.transaction_date, '%m/%d/%Y')) AS transaction_year,
   MONTH(STR_TO_DATE(t.transaction_date, '%m/%d/%Y')) AS transaction_month,
   COUNT(t.transaction_id) AS total_transactions,
   SUM(t.transaction_amount) AS total_transaction_amount,
   AVG(t.transaction_amount) AS average_transaction_amount
FROM
   cross_river_bank.transaction_table AS t
GROUP BY
   transaction_year, transaction_month
ORDER BY
   transaction_year ASC, transaction_month ASC;
```

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

transaction_year	transaction_month	total_transactions	total_transaction_amount	average_transaction_amount
2019	11	3	61133	20377.6667
2019	12	84	2130715	25365.6548
2020	1	91	2235589	24566.9121
2020	2	82	2188899	26693.8902
2020	3	98	2283677	23302.8265
2020	4	87	2249080	25851.4943
2020	5	71	1833222	25820.0282

Result 11

Task 10

Fraud Detection: Highlight potential fraud by identifying mismatches between customer address locations and transaction IP locations.

Solution 10

The CSV file did not have IP Location data

But if it had the following code will be working (Considering ip_location as the column):

```
SELECT
   t.transaction_id,
   t.customer_id,
   c.address AS customer_address,
   t.ip_location AS transaction_ip_location,
   t.transaction_date,
   t.transaction_amount
FROM
   cross_river_bank.transaction_table AS t
JOIN
   cross_river_bank.customer_table AS c ON t.customer_id = c.customer_id
WHERE
   c.address NOT LIKE CONCAT('%', t.ip_location, '%')
ORDER BY
   t.transaction_date DESC;
```

Task 11

Repayment History Analysis: Rank loans by repayment performance using window functions.

Solution 11

```
SELECT
   l.loan_id,
   l.customer_id,
   l.loan_amount,
   l.repayment_history,
   l.loan_status,
   RANK() OVER (ORDER BY l.repayment_history DESC, l.loan_amount DESC) AS repayment_rank
FROM
   cross_river_bank.loan_table AS l
WHERE
   l.loan_status IN ('Closed', 'Approved')
ORDER BY
   repayment_rank ASC;
```

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

loan_id	customer_id	loan_amount	repayment_history	loan_status	repayment_rank
2490	614	497811	10	Closed	1
2697	70	496637	10	Approved	2
1473	540	494274	10	Approved	3
153	683	493919	10	Closed	4
1027	916	482021	10	Closed	5
371	972	480186	10	Approved	6
1274	665	476971	10	Closed	7

Result 12

Task 12

Credit Score vs. Loan Amount: Compare average loan amounts for different credit score ranges.

Solution 12

```
SELECT
   CASE
      WHEN c.credit_score < 500 THEN 'Very Poor'
      WHEN c.credit_score BETWEEN 500 AND 599 THEN 'Poor'
      WHEN c.credit_score BETWEEN 600 AND 699 THEN 'Fair'
      WHEN c.credit_score BETWEEN 700 AND 799 THEN 'Good'
      ELSE 'Excellent'
```

```
    END AS credit_score_range,
    COUNT(l.loan_id) AS total_loans,
    AVG(l.loan_amount) AS average_loan_amount
FROM
    cross_river_bank.loan_table AS l
JOIN
    cross_river_bank.customer_table AS c ON l.customer_id = c.customer_id
GROUP BY
    credit_score_range
ORDER BY
    average_loan_amount DESC;
```

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

credit_score_range	total_loans	average_loan_amount
Excellent	289	266335.3910
Good	545	255410.2972
Very Poor	1093	251763.9460
Poor	556	248571.4514
Fair	517	242410.8936

Result 13

Task 13	**Top Borrowing Regions:** Identify regions with the highest total loan disbursements.
Solution 13	

```
SELECT
    c.address AS region,
    COUNT(l.loan_id) AS total_loans,
    SUM(l.loan_amount) AS total_loan_disbursements,
    AVG(l.loan_amount) AS average_loan_amount
FROM
    cross_river_bank.loan_table AS l
JOIN
    cross_river_bank.customer_table AS c ON l.customer_id = c.customer_id
GROUP BY
```

```
    c.address
ORDER BY
    total_loan_disbursements DESC;
```

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

region	total_loans	total_loan_disbursements	average_loan_amount
PSC 5754, Box 7187, APO AA 12064	8	2974541	371817.6250
532 Davis Fords, West Josephport, MD 83729	11	2498952	227177.4545
1238 Maxwell Common, West Kathleen, TN 15476	8	2458984	307373.0000
3725 Anna Island Apt. 366, Lambton, MO 09810	7	2426611	346658.7143
4691 Taylor Ford, East Bryanfurt, NE 03803	9	2418197	268688.5556
03641 Kayla Pines Suite 497, Kennethtown, MN...	8	2300192	287524.0000
336 Santana Tunnel, Laurieside, DC 60997	7	2291441	327348.7143

Result 14

Task 14

Early Repayment Patterns: Detect loans with frequent early repayments and their impact on revenue.

Solution 14

```
SELECT
    l.loan_id,
    l.customer_id,
    l.loan_amount,
    COUNT(t.transaction_id) AS early_repayments_count,
    SUM(t.transaction_amount) AS total_early_repayments,
    (SUM(t.transaction_amount) / l.loan_amount) * 100 AS early_repayment_percentage
FROM
    cross_river_bank.loan_table AS l
JOIN
    cross_river_bank.transaction_table AS t ON l.loan_id = t.loan_id
WHERE
    t.transaction_type = 'Prepayment'
GROUP BY
    l.loan_id, l.customer_id, l.loan_amount
HAVING
    early_repayment_percentage > 10
```

```
ORDER BY
   early_repayments_count DESC, total_early_repayments DESC;
```

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

loan_id	customer_id	loan_amount	early_repayments_count	total_early_repayments	early_repayment_percentage
2640	560	431161	4	140012	32.4733
1862	896	298663	3	124144	41.5666
1205	880	209412	3	112697	53.8159
963	189	155142	3	96650	62.2978
1614	284	193997	3	86256	44.4625
2088	329	124415	3	82773	66.5298
2004	371	211155	3	79221	37.5179

Result 15

Task 15

Feedback Correlation: Correlate customer feedback sentiment scores with loan statuses.

Solution 16

```
SELECT
   CASE
      WHEN t.remarks= 'Late penalty.' THEN 'Late penalty.'
      WHEN t.remarks= 'On-time payment.' THEN 'On-time payment.'
      WHEN t.remarks= 'Partial payment.' THEN 'Partial payment.'
      WHEN t.remarks= 'Payment missed.' THEN 'Payment missed.'
   END AS Feedback_sentiment,
   l.loan_status,
   SUM(l.loan_amount) AS total_loan_amount,
   AVG(l.loan_amount) AS average_loan_amount
FROM
   cross_river_bank.loan_table AS l
JOIN
   cross_river_bank.transaction_table AS t ON l.customer_id = t.customer_id
GROUP BY
   Feedback_sentiment , l.loan_status
ORDER BY
   total_loan_amount DESC;
```

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

Feedback_sentiment	loan_status	total_loan_amount	average_loan_amount
Late penalty.	Defaulted	263546456	256119.0049
On-time payment.	Defaulted	259850844	259591.2527
Late penalty.	Approved	255615240	243211.4558
Late penalty.	Closed	255253431	256536.1116
On-time payment.	Approved	250733432	246784.8740
On-time payment.	Closed	242226821	250493.0931
Partial payment.	Closed	242057158	261966.6212

Result 16

Task 17

Insert New Feedback

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

Solution 17

Cross River Bank > cross_river_bank > customer_feedback

Open MongoDB shell

Documents 3.0K | Aggregations | Schema | Indexes 1 | Validation

Type a query: { field: 'value' } or Generate query

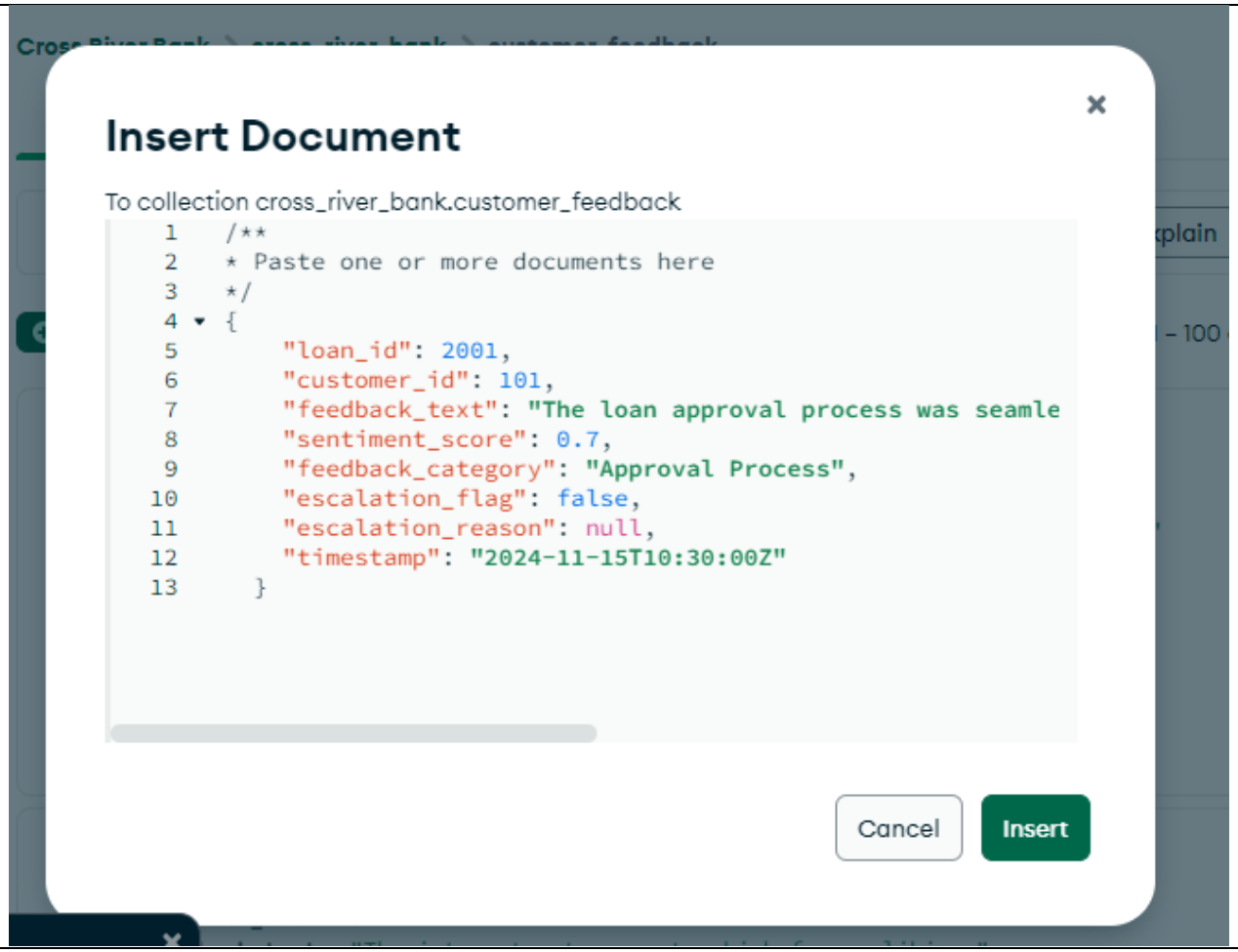
Explain | Reset | Find | Options

ADD DATA | EXPORT DATA | UPDATE | DELETE | 100 | 1 - 100 of 3000

Import JSON or CSV file

Insert document

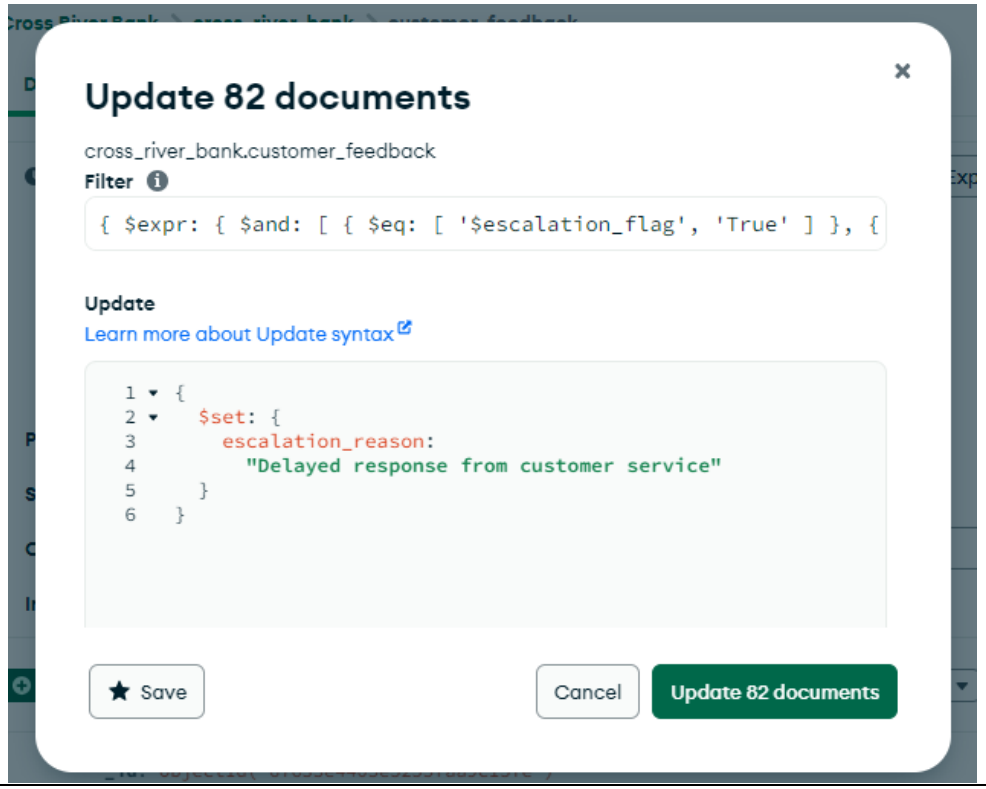
```
e4405e5255faa9c15c3')
interest rates are competitive compared to other banks."
sentiment_score : "-0.93"
language : "es"
timestamp : "2024-06-28T21:57:39"
feedback_category : "Interest Rates"
escalation_flag : "True"
escalation_reason : "None"
feedback_length : "59.0"

_id: ObjectId('67633e4405e5255faa9c15c4')
```



Task 18	**Update Escalation Flags** Input: Update escalation flags in feedback to include specific reasons for unresolved customer complaints.

Solution 18	{
	$expr: {
	$and: [
	{ $eq: ["$escalation_flag", 'True'] },

```
      { $or: [
        { $eq: ["$escalation_reason", null] },
        { $eq: ["$escalation_reason", "None"] },
        { $eq: ["$escalation_reason", ""] }
      ] },
      { $eq: ["$feedback_category", "Customer Service"] }
    ]
  }
}
```

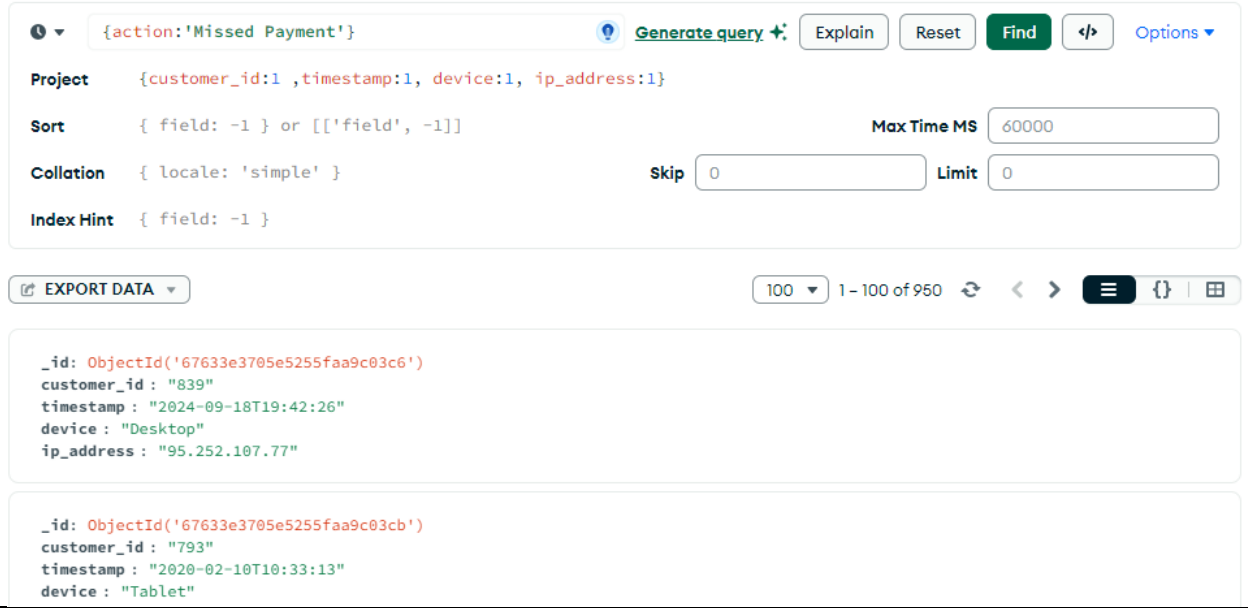


Task 19	**Remove Duplicate Behavior Logs** Input: Remove duplicate entries with the same customer_id and timestamp.

Solution 19	Stage 1 $group
	/**
	* _id: The id of the group.
	* fieldN: The first field name.
	*/
	{
	_id: {
	customer_id: "$customer_id",
	timestamp: "$timestamp"
	},
	ids: {
	$push: "$_id"
	},
	count: {
	$sum: 1
	}
	}
	Output after $group stage (Sample of 10 documents)
	_id: Object
	ids : Array (1)
	0: ObjectId('67633e4405e5255faa9c1ea0')
	count : 1
	_id: Object
	ids : Array (1
	count : 1
	Stage 2 $match
	/**
	* query: The query in MQL.
	*/
	{
	count: {
	$gt: 1
	}
	}
	Output after $match stage (Sample of 10 documents)
	_id: Object
	ids : Array (2)
	count : 2
	_id: Object
	ids : Array (2
	count : 2

	Stage 3 $project
	/**
	* specifications: The fields to
	* include or exclude.
	*/
	{
	_id: 0,
	idsToRemove: {
	$slice: [
	"$ids",
	1,
	{
	$size: "$ids"
	}
	]
	}
	}
	Output after $project stage (Sample of 10 documents)
	idsToRemove : Array (1)
	idsToRemove :

Task 20	**Retrieve Positive Feedback** Input: Retrieve feedback entries with sentiment scores greater than 0.5.
Solution 20	{sentiment_score: { $gt: 0.5 }} Generate query Explain Reset Find Options Project {loan_id:1,customer_id:1,feedback_text:1,sentiment_score:1} Sort { field: -1 } or [['field', -1]] Max Time MS 60000 Collation { locale: 'simple' } Skip 0 Limit 0 Index Hint { field: -1 } EXPORT DATA 100 1 - 1 of 1 _id: ObjectId('6765bd615beb418d4d08ad10') loan_id : 2001 customer_id : 101 feedback_text : "The loan approval process was seamless, but the interest rate was slig…" sentiment_score : 0.7
Task 21	**Fetch Logs for 'Missed Payment' Actions** Input: Retrieve behavior logs where the action is "Missed Payment."

Solution 21	
Task 22	Analyze customer behavior and feedback to identify high-risk customers and prioritize operational improvements for fraud detection and customer satisfaction Using an aggregate pipeline.
Solution 22	**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.

Stage 1 $lookup

```
{
  from: "behaviour_log",
  localField: "customer_id",
  foreignField: "customer_id",
  as: "behavior_data"
}
```

Output after $lookup stage (Sample of 10 documents)

```
_id: ObjectId('67633e4405e5255faa9c15c3')
loan_id : "1"
customer_id : "525"
feedback_text : "The interest rates are
                 competitive compared to other
                 banks."
sentiment_score : "-0.93"
language : "es"
timestamp : "2024-06-28T21:57:39"
```

```
_id: ObjectId(
loan_id : "2"
customer_id :
feedback_text

sentiment_scor
language : "en
timestamp : "2
feedback_categ
```

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

Stage 2 $unwind

```
{
  path: "$behavior_data",
  preserveNullAndEmptyArrays: true
}
```

Output after $unwind stage (Sample of 10 documents)

```
_id: ObjectId('67633e4405e5255faa9c15c3')
loan_id : "1"
customer_id : "525"
feedback_text : "The interest rates are
                 competitive compared to other
                 banks."
sentiment_score : "-0.93"
language : "es"
timestamp : "2024-06-28T21:57:39"
```

```
_id: ObjectId(
loan_id : "1"
customer_id :
feedback_text


sentiment_scor
language : "es
timestamp : "2
```

Group the combined data by customer_id to compute:

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

Stage 3 $group

```
{
  _id: "$customer_id",
  avgSentiment: {
    $avg: {
      $toDouble: "$sentiment_score"
    }
  },
  // Convert sentiment_score to numeric
  totalMissedPayments: {
    $sum: {
      $cond: [
        {
          $eq: [
            "$feedback_category",
            "Missed Payment"
          ]
        },
```

Output after $group stage (Sample of 10 documents)

```
_id: "342"
avgSentiment : NaN
totalMissedPayments : 0
totalSessionDuration : 16408
unresolvedEscalations : 10
```

```
_id: "691"
avgSentiment :
totalMissedPay
totalSessionDu
unresolvedEsca
```

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.**

Stage 4 $addFields

```
{
  riskScore: {
    $add: [
      {
        $multiply: ["$totalMissedPayments"
      },
      // Higher weight for missed payments
      {
        $multiply: [
          {
            $subtract: [1, "$avgSentiment"
          },
          3
        ]
      },
```

Output after $addFields stage (Sample of 10 documents)

```
_id: "316"
avgSentiment : NaN
totalMissedPayments : 0
totalSessionDuration : 6618
unresolvedEscalations : 5
riskScore : NaN
```

```
_id: "461"
avgSentiment :
totalMissedPay
totalSessionDu
unresolvedEsca
riskScore : 25
```

Stage 5 $lookup

```
1 {
2   from: "behaviour_log",
3   localField: "_id",
4   foreignField: "customer_id",
5   as: "device_usage"
6 }
```

Output after $lookup stage (Sample of 10 documents)

```
_id: "6"
avgSentiment : -0.726
totalMissedPayments : 0
totalSessionDuration : 13495
unresolvedEscalations : 20
riskScore : 38.673
▸ device_usage : Array (5)
```

```
_id: "542"
avgSentiment :
totalMissedPay
totalSessionDu
unresolvedEsca
riskScore : Nal
▸ device_usage :
```

Stage 6 $unwind

```
1 {
2   path: "$device_usage",
3   preserveNullAndEmptyArrays: true
4 }
```

Output after $unwind stage (Sample of 10 documents)

```
_id: "24"
avgSentiment : -0.58
totalMissedPayments : 0
totalSessionDuration : 6852
unresolvedEscalations : 5
riskScore : 16.592
▸ device_usage : Object
```

```
_id: "24"
avgSentiment :
totalMissedPay
totalSessionDu
unresolvedEsca
riskScore : 16
▸ device_usage :
```

Filter for customers with:

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

Stage 7 $match

```
/**
 * query: The query in MQL.
 */
{
  unresolvedEscalations: {
    $gt: 10
  },
  riskScore: {
    $gte: 10
  } // Threshold for high-risk customers
}
```

Output after $match stage (Sample of 10 documents)

```
_id: "354"
avgSentiment : -0.7183333333333334
totalMissedPayments : 0
totalSessionDuration : 8472
unresolvedEscalations : 20
riskScore : 33.626999999999995
device_usage : Object
```

```
_id: "354"
avgSentiment :
totalMissedPay
totalSessionDu
unresolvedEsca
riskScore : 33
device_usage :
```

Stage 8 $project

```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  _id: 1,
  avgSentiment: 1,
  totalMissedPayments: 1,
  totalSessionDuration: 1,
  unresolvedEscalations: 1,
  riskScore: 1,
  deviceTrends: 1
}
```

Output after $project stage (Sample of 10 documents)

```
_id: "955"
avgSentiment : -0.62
totalMissedPayments : 0
totalSessionDuration : 9070
unresolvedEscalations : 15
riskScore : 28.93
```

```
_id: "955"
avgSentiment :
totalMissedPay
totalSessionDu
unresolvedEsca
riskScore : 28
```

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

Stage 9 $group

```
1 {
2   _id: {
3     customer_id: "$_id",
4     device: "$device_usage.device"
5   },
6   count: {
7     $sum: 1
8   }
9 }
```

Output after $group stage (Sample of 10 documents)

```
▾ _id: Object
    customer_id: "266"
  count : 5
```

```
▸ _id: Object
  count : 5
```

Stage 10 $group

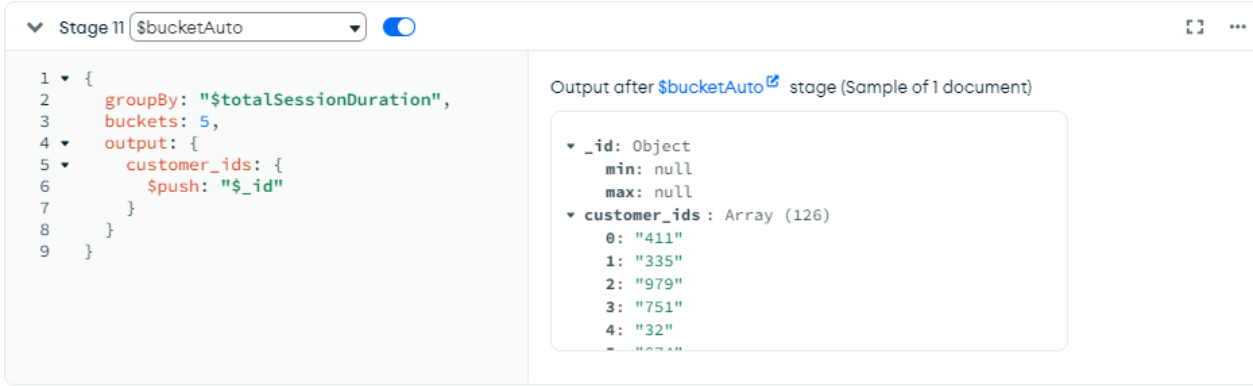
```
1 {
2   _id: "$_id.customer_id",
3   deviceTrends: {
4     $push: {
5       device: "$_id.device",
6       usageCount: "$count"
7     }
8   }
9 }
```

Output after $group stage (Sample of 10 documents)

```
  _id: "979"
▾ deviceTrends : Array (1)
  ▾ 0: Object
      usageCount : 5
```

```
  _id: "411"
▸ deviceTrends :
```

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



Aggregation Text :

```
[
 // Step 1: Join Feedback and Behavior Logs
 {
  $lookup: {
   from: "behaviour_log",
   localField: "customer_id",
   foreignField: "customer_id",
   as: "behavior_data"
  }
 },
 // Step 2: Flatten Joined Data
 {
  $unwind: {
   path: "$behavior_data",
   preserveNullAndEmptyArrays: true
  }
 },
 // Step 3: Aggregate Feedback and Behavioral Metrics
 {
```

```
$group: {
  _id: "$customer_id",
  avgSentiment: {
    $avg: {
      $toDouble: "$sentiment_score"
    }
  },
  // Convert sentiment_score to numeric
  totalMissedPayments: {
    $sum: {
      $cond: [
        {
          $eq: [
            "$feedback_category",
            "Missed Payment"
          ]
        },
        1,
        0
      ]
    }
  },
  totalSessionDuration: {
    $sum: {
      $toInt:
        "$behavior_data.session_duration"
    }
  },
  unresolvedEscalations: {
    $sum: {
      $cond: [
        {
          $eq: ["$escalation_flag", "True"]
```

```
          },
          1,
          // Increment count if escalation_flag is true
          0
        ]
      }
    }
  }
},
// Step 4: Calculate Risk Scores
{
  $addFields: {
    riskScore: {
      $add: [
        {
          $multiply: ["$totalMissedPayments", 5]
        },
        // Higher weight for missed payments
        {
          $multiply: [
            {
              $subtract: [1, "$avgSentiment"]
            },
            3
          ]
        },
        // Negative sentiment
        {
          $divide: [
            "$totalSessionDuration",
            1000
          ]
        },
```

```
        // Scale session durations
        "$unresolvedEscalations" // Direct addition for unresolved escalations
      ]
    }
  }
},
// Step 5: Identify Device Usage Trends
{
  $lookup: {
    from: "behaviour_log",
    localField: "_id",
    foreignField: "customer_id",
    as: "device_usage"
  }
},
{
  $unwind: {
    path: "$device_usage",
    preserveNullAndEmptyArrays: true
  }
},
{
  $match:
    /**
     * query: The query in MQL.
     */
    {
      unresolvedEscalations: {
        $gt: 10
      },
      riskScore: {
        $gte: 10
      } // Threshold for high-risk customers
```

```
    }
  },
  {
    $project:
      /**
       * specifications: The fields to
       *   include or exclude.
       */
      {
        _id: 1,
        avgSentiment: 1,
        totalMissedPayments: 1,
        totalSessionDuration: 1,
        unresolvedEscalations: 1,
        riskScore: 1,
        deviceTrends: 1
      }
  },
  {
    $group: {
      _id: {
        customer_id: "$_id",
        device: "$device_usage.device"
      },
      count: {
        $sum: 1
      }
    }
  },
  {
    $group: {
      _id: "$_id.customer_id",
      deviceTrends: {
```

	``
	$push: {
	device: "$_id.device",
	usageCount: "$count"
	}
	}
	}
	},
	// Step 6: Detect Session Outliers
	{
	$bucketAuto: {
	groupBy: "$totalSessionDuration",
	buckets: 5,
	output: {
	customer_ids: {
	$push: "$_id"
	}
	}
	}
	}
	]
	``

GITHUB Repository	https://github.com/Shantanuneo/Graded-Project-on-Fraud-Detection-and-Risk-Analysis.git