--- Identify customers with low credit scores and high-risk loans to predict potential defaults and prioritize risk mitigation strategies.

SELECT

high\_risk.customer\_id,

high\_risk.name,

high\_risk.credit\_score,

COUNT(DISTINCT high\_risk.loan\_id) AS total\_loans,

SUM(high\_risk.loan\_amount) AS total\_debt,

COUNT(CASE WHEN high\_risk.transaction\_type = 'Penalty' OR high\_risk.status = 'Failed' THEN 1 END) AS missed\_payments,

MAX(high\_risk.default\_risk\_numeric) AS max\_default\_risk,

ROUND(

((600 - COALESCE(high\_risk.credit\_score, 600)) / 600) +

MAX(high\_risk.default\_risk\_numeric) +

(COUNT(CASE WHEN high\_risk.transaction\_type = 'Penalty' OR high\_risk.status = 'Failed' THEN 1 END) \* 0.5),

2) AS risk\_score

FROM (

SELECT

c.customer\_id,

c.name,

c.credit\_score,

l.loan\_id,

l.loan\_amount,

CASE

WHEN l.default\_risk = 'High' THEN 1.0

WHEN l.default\_risk = 'Medium' THEN 0.5

WHEN l.default\_risk = 'Low' THEN 0.2

ELSE 0

END AS default\_risk\_numeric,

COALESCE(t.transaction\_type, 'No Transactions') AS transaction\_type,

COALESCE(t.status, 'No Transactions') AS status

FROM cross\_river\_bank.customer\_table c

JOIN cross\_river\_bank.loan\_table l ON c.customer\_id = l.customer\_id

LEFT JOIN cross\_river\_bank.transaction\_table t ON l.loan\_id = t.loan\_id

WHERE c.credit\_score < 600

AND (l.default\_risk = 'High' OR CAST(l.default\_risk AS DECIMAL(3,2)) > 0.7)

) AS high\_risk

GROUP BY high\_risk.customer\_id, high\_risk.name, high\_risk.credit\_score

ORDER BY risk\_score DESC

LIMIT 1000;

SELECT

customer\_id,

name,

credit\_score,

risk\_category,

active\_loans

FROM

cross\_river\_bank.customer\_table

WHERE

credit\_score < 600

AND risk\_category = 'High';

--- Determine the most popular loan purposes and their associated revenues to align financial products with customer demands

SELECT

l.loan\_purpose,

COUNT(l.loan\_id) AS loan\_count,

COALESCE(SUM(CASE

WHEN t.transaction\_type = 'EMI Payment' AND t.status = 'Successful'

THEN t.transaction\_amount

ELSE 0

END), 0) AS total\_revenue,

ROUND(AVG(c.income), 2) AS avg\_income,

ROUND(AVG(c.credit\_score), 2) AS avg\_credit\_score

FROM cross\_river\_bank.loan\_table l

JOIN cross\_river\_bank.customer\_table c ON l.customer\_id = c.customer\_id

LEFT JOIN cross\_river\_bank.transaction\_table t ON l.loan\_id = t.loan\_id

GROUP BY l.loan\_purpose

ORDER BY loan\_count DESC, total\_revenue DESC;

SELECT

loan\_purpose,

COUNT(loan\_id) AS number\_of\_loans,

SUM(loan\_amount) AS total\_revenue

FROM

cross\_river\_bank.loan\_table

GROUP BY

loan\_purpose

ORDER BY

total\_revenue DESC;

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

SELECT

t.transaction\_id,

t.loan\_id,

t.customer\_id,

t.transaction\_date,

t.transaction\_amount,

l.loan\_amount,

ROUND((t.transaction\_amount / NULLIF(l.loan\_amount, 0)) \* 100, 2) AS percentage\_of\_loan,

t.transaction\_type,

t.status,

COALESCE(c.risk\_category, 'Unknown') AS risk\_category,

CASE

WHEN t.transaction\_amount > (l.loan\_amount \* 0.5) THEN 'HIGH FRAUD RISK'

WHEN t.transaction\_amount > (l.loan\_amount \* 0.3) AND COALESCE(c.risk\_category, 'Unknown') = 'High'

THEN 'MEDIUM FRAUD RISK'

WHEN t.transaction\_amount > (l.loan\_amount \* 0.3) AND (t.transaction\_type = 'Prepayment' OR t.transaction\_type = 'Cash')

THEN 'SUSPICIOUS TRANSACTION'

ELSE 'NORMAL'

END AS fraud\_flag

FROM cross\_river\_bank.transaction\_table t

JOIN cross\_river\_bank.loan\_table l ON t.loan\_id = l.loan\_id

JOIN cross\_river\_bank.customer\_table c ON l.customer\_id = c.customer\_id

WHERE t.transaction\_amount > (l.loan\_amount \* 0.3)

ORDER BY fraud\_flag DESC, t.transaction\_amount DESC;

SELECT

t.transaction\_id,

t.loan\_id,

t.customer\_id,

t.transaction\_date,

t.transaction\_amount,

l.loan\_amount,

CASE

WHEN t.transaction\_amount > 0.3 \* l.loan\_amount THEN 'High-Value Transaction'

ELSE 'Normal Transaction'

END AS transaction\_flag

FROM

cross\_river\_bank.transaction\_table t

JOIN

cross\_river\_bank.loan\_table l ON t.loan\_id = l.loan\_id

WHERE

t.transaction\_amount > 0.3 \* l.loan\_amount;

--- Analyze the number of missed EMIs per loan to identify loans at risk of default and suggest intervention strategies

SELECT

l.loan\_id,

l.customer\_id,

l.loan\_amount,

l.loan\_status,

COUNT(CASE WHEN t.transaction\_type = 'Missed EMI' THEN 1 END) AS missed\_emi\_count,

c.credit\_score,

c.risk\_category,

CASE

WHEN COUNT(CASE WHEN t.transaction\_type = 'Missed EMI' THEN 1 END) >= 3

THEN 'HIGH RISK - Immediate Intervention Needed'

WHEN COUNT(CASE WHEN t.transaction\_type = 'Missed EMI' THEN 1 END) = 2

THEN 'MEDIUM RISK - Payment Reminder Required'

WHEN COUNT(CASE WHEN t.transaction\_type = 'Missed EMI' THEN 1 END) = 1

THEN 'LOW RISK - Soft Reminder'

ELSE 'ON TRACK'

END AS intervention\_strategy

FROM cross\_river\_bank.loan\_table l

JOIN cross\_river\_bank.customer\_table c ON l.customer\_id = c.customer\_id

LEFT JOIN cross\_river\_bank.transaction\_table t ON l.loan\_id = t.loan\_id

GROUP BY l.loan\_id, l.customer\_id, l.loan\_amount, l.loan\_status, c.credit\_score, c.risk\_category

ORDER BY missed\_emi\_count DESC;

--- Examine the geographical distribution of loan disbursements to assess regional trends and business opportunities.

SELECT

SUBSTRING\_INDEX(c.address, ',', -1) AS region,

COUNT(l.loan\_id) AS total\_loans,

SUM(l.loan\_amount) AS total\_loan\_amount,

ROUND(AVG(l.loan\_amount), 2) AS avg\_loan\_amount,

ROUND(AVG(c.income), 2) AS avg\_income,

ROUND(AVG(c.credit\_score), 2) AS avg\_credit\_score,

COUNT(CASE WHEN l.loan\_status = 'Defaulted' THEN 1 END) AS defaulted\_loans,

ROUND((COUNT(CASE WHEN l.loan\_status = 'Defaulted' THEN 1 END) / COUNT(l.loan\_id)) \* 100, 2) AS default\_rate,

CASE

WHEN COUNT(l.loan\_id) > 1000 AND ROUND((COUNT(CASE WHEN l.loan\_status = 'Defaulted' THEN 1 END) / COUNT(l.loan\_id)) \* 100, 2) < 5

THEN 'High-Growth Market'

WHEN COUNT(l.loan\_id) > 500 AND ROUND((COUNT(CASE WHEN l.loan\_status = 'Defaulted' THEN 1 END) / COUNT(l.loan\_id)) \* 100, 2) BETWEEN 5 AND 10

THEN 'Moderate Risk - Potential Market'

ELSE 'High Risk - Limited Opportunity'

END AS business\_opportunity

FROM cross\_river\_bank.loan\_table l

JOIN cross\_river\_bank.customer\_table c ON l.customer\_id = c.customer\_id

GROUP BY region

ORDER BY total\_loan\_amount DESC, default\_rate ASC;

---- List customers who have been associated with Cross River Bank for over five years and evaluate their loan activity to design loyalty programs.

SELECT \*,

CASE

WHEN years\_with\_bank >= 10 AND successful\_repayment\_rate >= 90

THEN 'Platinum Loyalty - Premium Benefits'

WHEN years\_with\_bank >= 7 AND successful\_repayment\_rate >= 85

THEN 'Gold Loyalty - Exclusive Discounts'

WHEN years\_with\_bank >= 5

THEN 'Silver Loyalty - Special Offers'

ELSE 'Standard Customer'

END AS loyalty\_tier

FROM (

SELECT

c.customer\_id,

c.name,

c.customer\_since\_temp,

TIMESTAMPDIFF(YEAR, c.customer\_since\_temp, CURDATE()) AS years\_with\_bank,

COUNT(l.loan\_id) AS total\_loans,

COALESCE(SUM(l.loan\_amount), 0) AS total\_loan\_amount,

ROUND(COALESCE(AVG(l.loan\_amount), 0), 2) AS avg\_loan\_amount,

COUNT(CASE WHEN l.loan\_status = 'Closed' THEN 1 END) AS successfully\_closed\_loans,

COUNT(CASE WHEN l.loan\_status = 'Defaulted' THEN 1 END) AS defaulted\_loans,

ROUND(

(CASE

WHEN COUNT(l.loan\_id) > 0

THEN (COUNT(CASE WHEN l.loan\_status = 'Closed' THEN 1 END) / NULLIF(COUNT(l.loan\_id), 0)) \* 100

ELSE 0

END), 2) AS successful\_repayment\_rate

FROM cross\_river\_bank.customer\_table c

LEFT JOIN cross\_river\_bank.loan\_table l

ON c.customer\_id = l.customer\_id

GROUP BY c.customer\_id, c.name, c.customer\_since\_temp

) AS CustomerLoanStats

WHERE years\_with\_bank > 5

ORDER BY years\_with\_bank DESC, successful\_repayment\_rate DESC;

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

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,

SUM(l.loan\_amount) AS total\_loan\_amount,

COUNT(l.loan\_id) AS number\_of\_loans

FROM

cross\_river\_bank.customer\_table c

JOIN

cross\_river\_bank.loan\_table l ON c.customer\_id = l.customer\_id

GROUP BY

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

ORDER BY

age\_group;

---- Examine transaction patterns over years and months to identify seasonal trends in loan repayments.

SELECT

YEAR(STR\_TO\_DATE(t.transaction\_date, '%Y-%m-%d')) AS transaction\_year,

MONTH(STR\_TO\_DATE(t.transaction\_date, '%Y-%m-%d')) AS transaction\_month,

COUNT(t.transaction\_id) AS total\_transactions,

SUM(t.transaction\_amount) AS total\_repayments

FROM

cross\_river\_bank.transaction\_table t

LEFT JOIN

cross\_river\_bank.loan\_table l ON t.loan\_id = l.loan\_id

WHERE

t.transaction\_date IS NOT NULL

AND t.loan\_id IS NOT NULL

GROUP BY

YEAR(STR\_TO\_DATE(t.transaction\_date, '%Y-%m-%d')),

MONTH(STR\_TO\_DATE(t.transaction\_date, '%Y-%m-%d'))

ORDER BY

transaction\_year,

transaction\_month;

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

WITH LoanPerformance AS (

SELECT

l.loan\_id,

l.customer\_id,

l.loan\_amount,

l.loan\_date,

l.loan\_status,

l.interest\_rate,

l.loan\_purpose,

l.collateral,

l.default\_risk,

COUNT(CASE WHEN t.transaction\_type = 'EMI Payment' AND t.status = 'Successful' THEN 1 END) AS successful\_payments,

COUNT(CASE WHEN t.transaction\_type = 'Missed EMI' THEN 1 END) AS missed\_payments,

ROUND(

(COUNT(CASE WHEN t.transaction\_type = 'EMI Payment' AND t.status = 'Successful' THEN 1 END)

/ NULLIF(COUNT(t.transaction\_id), 0)) \* 100, 2

) AS repayment\_success\_rate

FROM cross\_river\_bank.loan\_table l

LEFT JOIN cross\_river\_bank.transaction\_table t

ON l.loan\_id = t.loan\_id

GROUP BY l.loan\_id, l.customer\_id, l.loan\_amount, l.loan\_date, l.loan\_status,

l.interest\_rate, l.loan\_purpose, l.collateral, l.default\_risk

)

SELECT

loan\_id,

customer\_id,

loan\_amount,

loan\_date,

loan\_status,

interest\_rate,

loan\_purpose,

collateral,

default\_risk,

successful\_payments,

missed\_payments,

repayment\_success\_rate,

RANK() OVER (ORDER BY repayment\_success\_rate DESC) AS repayment\_rank

FROM LoanPerformance

ORDER BY repayment\_rank;

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

SELECT

CASE

WHEN c.credit\_score >= 800 THEN 'Excellent (800+)'

WHEN c.credit\_score BETWEEN 740 AND 799 THEN 'Very Good (740-799)'

WHEN c.credit\_score BETWEEN 670 AND 739 THEN 'Good (670-739)'

WHEN c.credit\_score BETWEEN 580 AND 669 THEN 'Fair (580-669)'

ELSE 'Poor (<580)'

END AS credit\_score\_category,

COUNT(l.loan\_id) AS total\_loans,

ROUND(AVG(l.loan\_amount), 2) AS avg\_loan\_amount

FROM cross\_river\_bank.customer\_table c

JOIN cross\_river\_bank.loan\_table l

ON c.customer\_id = l.customer\_id

GROUP BY credit\_score\_category

ORDER BY FIELD(credit\_score\_category, 'Excellent (800+)', 'Very Good (740-799)', 'Good (670-739)', 'Fair (580-669)', 'Poor (<580)');

--- Top Borrowing Regions: Identify regions with the highest total loan disbursements.

SELECT

c.address AS region,

COUNT(l.loan\_id) AS total\_loans,

SUM(l.loan\_amount) AS total\_loan\_disbursement

FROM cross\_river\_bank.customer\_table c

JOIN cross\_river\_bank.loan\_table l

ON c.customer\_id = l.customer\_id

GROUP BY c.address

ORDER BY total\_loan\_disbursement DESC

LIMIT 10;

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

SELECT

l.loan\_id,

l.loan\_amount,

COALESCE(SUM(t.transaction\_amount), 0) AS total\_repaid\_early\_payments,

COALESCE((SUM(t.transaction\_amount) \* 100.0) / NULLIF(l.loan\_amount, 0), 0) AS repayment\_percentage

FROM cross\_river\_bank.loan\_table l

LEFT JOIN cross\_river\_bank.transaction\_table t

ON l.loan\_id = t.loan\_id

AND t.transaction\_type = 'prepayment'

GROUP BY l.loan\_id, l.loan\_amount

HAVING total\_repaid\_early\_payments > 0

ORDER BY repayment\_percentage DESC

LIMIT 1000;