```
--витрина кредитного скоринга
 --определение ежемесячного платежа по внешним кредитам
WITH ExternalRemainingPayments AS (
SELECT
        ech.Client_ClientID,
        ech.BankName,
        ech.LoanAmount,
        ech.TotalPaidAmount,
        ech.RemainingBalance,
        ech.StartDate,
        ech.EndDate,
        MONTHS_BETWEEN(ech.ENDDATE, SYSDATE ) AS RemainingMonths,
        CASE
                WHEN MONTHS_BETWEEN(ech.ENDDATE, SYSDATE ) > 0
            THEN ech.RemainingBalance /
                 MONTHS_BETWEEN(ech.ENDDATE, SYSDATE )
                WHEN MONTHS_BETWEEN(ech.ENDDATE, SYSDATE ) < 0
            THEN ech.RemainingBalance
                ELSE 0
        END AS MonthlyPayment
FROM
        ExternalCreditHistory ech
WHERE
        ech.CurrentStatus = 'Активный'
),
InternalMonthlyPayments AS(
SELECT
        (
        SELECT
                c.ClientID
        FROM
                AccountDeal ad
        JOIN Account a ON
                a.ACCOUNTID = ad.ACCOUNT_ACCOUNTID
        JOIN Client c ON
                c.ClientID = a.CLIENT_CLIENTID
        WHERE
                ad.DEAL_DEAL_ID = d.DealID
     FETCH FIRST 1 ROWS ONLY) AS ClientID,
        -- определяем клиента через подзапрос
        d.DealID,
        (SUM(ps.PrincpalPayment + ps.InterestPayment) / MONTHS_BETWEEN(d.ENDDATE, d.STARTDATE)) AS TotalMonthlyPayment
FROM
        PaymentSchedule ps
JOIN
    Deal d ON
        ps.Deal_DealID = d.DealID
WHERE
        d.ENDDATE > SYSDATE
GROUP BY
        d.DealID, MONTHS_BETWEEN(d.ENDDATE, d.STARTDATE)
),
DTI_Calculation AS (
SELECT
        c.ClientID,
        c.MonthlyIncome,
        COALESCE(SUM(emp.MonthlyPayment), 0) AS TotalExternalMonthlyPayments,
        -- внешние платежи
        COALESCE(imp.TotalMonthlyPayment, 0) AS TotalInternalMonthlyPayments
        -- внутренние платежи
FROM
        Client c
LEFT JOIN
        ExternalRemainingPayments emp ON
        c.ClientID = emp.Client_ClientID
LEFT JOIN
        InternalMonthlyPayments imp ON
        c.ClientID = imp.ClientID
GROUP BY
        c.ClientID,
        c.MonthlyIncome,
        TotalMonthlyPayment
),
DTI AS (SELECT
        dti.ClientID,
        dti.MonthlyIncome,
        dti.TotalExternalMonthlyPayments,
        dti.TotalInternalMonthlyPayments,
        (dti.TotalExternalMonthlyPayments + dti.TotalInternalMonthlyPayments) AS TotalDebtPayments,
        (dti.TotalExternalMonthlyPayments + dti.TotalInternalMonthlyPayments) / dti.MonthlyIncome * 100 AS DTI
        -- расчет DTI в процентах
FROM
        DTI_Calculation dti
ORDER BY dti.ClientID),
DTIScoring AS (SELECT d.ClientID, d.DTI,
CASE
                WHEN d.DTI BETWEEN 0 AND 20 THEN 5
                WHEN d.DTI BETWEEN 20 AND 40 THEN 4
                WHEN d.DTI BETWEEN 40 AND 60 THEN 3
                WHEN d.DTI BETWEEN 60 AND 80 THEN 2
                WHEN d.DTI BETWEEN 80 AND 100 THEN 1
                ELSE 0
        END AS DTI_SCORE
FROM DTI d),
AgeScoring AS(
SELECT
        c.ClientID,
        TRUNC(MONTHS_BETWEEN(SYSDATE, BirthDate) / 12) AGE,
        CASE
                WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, BirthDate) / 12) BETWEEN 18 AND 25 THEN 3
                WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, BirthDate) / 12) BETWEEN 26 AND 35 THEN 5
                WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, BirthDate) / 12) BETWEEN 36 AND 55 THEN 4
                WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, BirthDate) / 12) BETWEEN 56 AND 67 THEN 3
                WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, BirthDate) / 12) >= 68 AND c.GENDER = 'М' THEN 2
        WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, BirthDate) / 12) >= 78 AND c.GENDER = 'Ж' THEN 2
                ELSE 0
        END AS AGE_SCORE
FROM
        CLIENT c),
MaritalScoring AS(
SELECT
        c.CLIENTID,
        c.MARITALSTATUS,
        CASE
                WHEN c.MARITALSTATUS = 'Одинок' THEN 2
                WHEN c.MARITALSTATUS = 'В браке' THEN 5
                ELSE 0
        END AS MARITAL_SCORE
FROM
        CLIENT c
),
MonthlyIncomeScoring AS(
SELECT
        c.CLIENTID,
        c.MonthlyIncome,
        CASE
                WHEN c.MONTHLYINCOME >= 100000 THEN 5
                WHEN c.MONTHLYINCOME BETWEEN 80000 AND 100000 THEN 4
                WHEN c.MONTHLYINCOME BETWEEN 60000 AND 80000 THEN 3
                WHEN c.MONTHLYINCOME BETWEEN 40000 AND 60000 THEN 2
                WHEN c.MONTHLYINCOME BETWEEN 20000 AND 40000 THEN 1
                ELSE 0
        END AS INCOME_SCORE
FROM
        CLIENT c),
CountInternalActiveLoans AS(
SELECT
        c.CLIENTID,
        count(DISTINCT d.dealid) COUNT_ACTIVE_LOAN
FROM
        CLIENT c
JOIN ACCOUNT a
ON
        c.CLIENTID = a.CLIENT_CLIENTID
JOIN ACCOUNTDEAL AD
ON
        a.ACCOUNTID = ad.ACCOUNT_ACCOUNTID
JOIN DEAL d
ON
        ad.DEAL_DEAL_ID = d.DEALID
WHERE
        d.ENDDATE > SYSDATE
        AND d.dealtype = 'Кредитные операции'
GROUP BY
        c.CLIENTID
),
CountExternalActiveLoans AS(
SELECT
        c.CLIENTID,
        count(DISTINCT e.EXTERNALCREDITID) COUNT_ACTIVE_LOAN
FROM
        CLIENT c
JOIN EXTERNALCREDITHISTORY e
ON
        c.CLIENTID = e.CLIENT_CLIENTID
WHERE
        e.CURRENTSTATUS = 'Активный'
GROUP BY
        c.CLIENTID),
TotalActiveCredits AS(SELECT pod.CLIENTID, sum(pod.COUNT_ACTIVE_LOAN) sum_active_loans
FROM(SELECT CIA.CLIENTID, CIA.COUNT_ACTIVE_LOAN
FROM CountInternalActiveLoans CIA
UNION ALL
SELECT CEA.CLIENTID,CEA.COUNT_ACTIVE_LOAN
FROM CountExternalActiveLoans CEA) POD
GROUP BY pod.CLIENTID),
ActiveCreditsScoring AS (
SELECT
        TAC.CLIENTID,
        tac.sum_active_loans,
        CASE
                WHEN tac.sum_active_loans BETWEEN 0 AND 1 THEN 5
                WHEN tac.sum_active_loans = 2 THEN 4
                WHEN tac.sum_active_loans = 3 THEN 3
                WHEN tac.sum_active_loans = 4 THEN 2
                WHEN tac.sum_active_loans BETWEEN 5 AND 7 THEN 1
                ELSE 0
        END AS ACTIVE_LOANS_SCOR
FROM
        TotalActiveCredits TAC),
CollateralScoring AS (
    SELECT
        d.DealID,
        d.DealAmount,
        SUM(c.EstimatedValue) AS SUM_est,
        CASE
            WHEN d.DealAmount > 0 AND SUM(c.EstimatedValue) > 0 THEN (SUM(c.EstimatedValue) / d.DealAmount) * 100
            ELSE 0
        END AS CollateralPercentage,
        CASE
            WHEN (SUM(c.EstimatedValue) / d.DealAmount) * 100 < 5 THEN 1  -- 1 балл, если процент залога менее 5%
            WHEN (SUM(c.EstimatedValue) / d.DealAmount) * 100 BETWEEN 5 AND 20 THEN 2  -- 2 балла, если от 5% до 20%
            WHEN (SUM(c.EstimatedValue) / d.DealAmount) * 100 BETWEEN 21 AND 40 THEN 3  -- 3 балла, если от 21% до 40%
            WHEN (SUM(c.EstimatedValue) / d.DealAmount) * 100 BETWEEN 41 AND 60 THEN 4  -- 4 балла, если от 41% до 60%
            WHEN (SUM(c.EstimatedValue) / d.DealAmount) * 100 > 60 THEN 5  -- 5 баллов, если более 60%
            ELSE 0
        END AS CollateralScore,
        (SELECT c.ClientID
         FROM AccountDeal ad
         JOIN Account a ON ad.Account_AccountID = a.AccountID
         JOIN Client c ON a.Client_ClientID = c.ClientID
         WHERE ad.Deal_Deal_ID = d.DealID
         FETCH FIRST 1 ROWS ONLY) AS ClientID
    FROM
        Deal d
    JOIN
        Collateral c ON d.DealID = c.Deal_DealID
    WHERE
        d.DealType = 'Кредитные операции'
    GROUP BY
        d.DealID, d.DealAmount
),
```

```sql
ClientMissedPaymentsEx AS (
    SELECT
        c.ClientID,
        SUM(ech.MissedPaymentsCount) AS TotalMissedPayments
    FROM
        Client c
    JOIN
        ExternalCreditHistory ech ON c.ClientID = ech.Client_ClientID
    WHERE
        ech.CurrentStatus = 'Просрочен'
    GROUP BY
        c.ClientID
),
teast1 AS(SELECT DEAL_DEALID, SUM(p.PRINCPALPAYMENT) SUM_PRINCIPAL_PAYMENT, SUM(p.INTERESTPAYMENT) SUM_INTEREST_PAYMENT
FROM PAYMENTSCHEDULE p
GROUP BY p.DEAL_DEALID),
test2 AS(SELECT d.DEALID, ab.ACCOUNT_ACCOUNTID, ab.BALANCEAMOUNT, ab.BALANCEDATE, a.ACCOUNTTYPE
FROM DEAL d
JOIN ACCOUNTDEAL ad
ON d.DEALID = ad.DEAL_DEAL_ID
JOIN ACCOUNT a
ON ad.ACCOUNT_ACCOUNTID = a.ACCOUNTID
JOIN ACCOUNTBALANCE ab
ON a.ACCOUNTID = ab.ACCOUNT_ACCOUNTID
WHERE a.ACCOUNTTYPE = 'Счёт основного долга' OR a.ACCOUNTTYPE = 'Счёт процентов'),
PrincipalPaymentMade AS (SELECT
        t1.DEAL_DEALID,
        t2.ACCOUNT_ACCOUNTID,
        t2.BALANCEDATE,
        CASE
        -- Для первой строки: TOTAL_SUM_PRINCIPAL_PAYMENT - BALANCEAMOUNT
        WHEN ROW_NUMBER() OVER (PARTITION BY DEAL_DEALID ORDER BY BALANCEDATE) = 1 THEN
            t1.SUM_PRINCIPAL_PAYMENT - t2.BALANCEAMOUNT
        -- Для остальных строк: разница между предыдущим и текущим BALANCEAMOUNT
        ELSE LAG(t2.BALANCEAMOUNT) OVER (PARTITION BY t1.DEAL_DEALID ORDER BY t2.ACCOUNT_ACCOUNTID, t2.BALANCEDATE) - t2.BALANCEAMOUNT
    END AS PRINCIPAL_PAYMENT_MADE
FROM
        teast1 t1
JOIN test2 t2
ON
        t1.DEAL_DEALID = t2.DEALID
WHERE t2.ACCOUNTTYPE = 'Счёт основного долга'),
InterestPaymentMade AS (SELECT
        t1.DEAL_DEALID,
        t2.ACCOUNT_ACCOUNTID,
        t2.BALANCEDATE,
        CASE
        -- Для первой строки: TOTAL_SUM_INTEREST_PAYMENT - BALANCEAMOUNT
        WHEN ROW_NUMBER() OVER (PARTITION BY DEAL_DEALID ORDER BY BALANCEDATE) = 1 THEN
            t1.SUM_INTEREST_PAYMENT - t2.BALANCEAMOUNT
        -- Для остальных строк: разница между предыдущим и текущим BALANCEAMOUNT
        ELSE LAG(t2.BALANCEAMOUNT) OVER (PARTITION BY t1.DEAL_DEALID ORDER BY t2.ACCOUNT_ACCOUNTID, t2.BALANCEDATE) - t2.BALANCEAMOUNT
    END AS INTEREST_PAYMENT_MADE
FROM
        teast1 t1
JOIN test2 t2
ON
        t1.DEAL_DEALID = t2.DEALID
WHERE t2.ACCOUNTTYPE = 'Счёт процентов'),
WITH_TOTAL_PAYMENT_LOAN AS (SELECT
    p.DEAL_DEALID,
    p.ACCOUNT_ACCOUNTID AS PRINCIPAL_ACCOUNT,
    i.ACCOUNT_ACCOUNTID AS INTEREST_ACCOUNT,
    p.BALANCEDATE,
    p.PRINCIPAL_PAYMENT_MADE,
    i.INTEREST_PAYMENT_MADE
FROM
    PrincipalPaymentMade p
JOIN
    InterestPaymentMade i
ON
    p.DEAL_DEALID = i.DEAL_DEALID
    AND p.BALANCEDATE = i.BALANCEDATE
  ORDER BY p.DEAL_DEALID, PRINCIPAL_ACCOUNT, p.BALANCEDATE),
PSchedule AS (
SELECT
        p.DEAL_DEALID,
        p.DUEDATE,
        p.PRINCPALPAYMENT,
        p.INTERESTPAYMENT
FROM
        PAYMENTSCHEDULE p),
comparisonpayments AS(SELECT
        b.BRANCHID,
        ps.DEAL_DEALID,
        ps.DUEDATE,
        ps.PRINCPALPAYMENT,
        ps.INTERESTPAYMENT,
        COALESCE(wtp.PRINCIPAL_PAYMENT_MADE,0) PRINCIPAL_PAYMENT_MADE,
        COALESCE(wtp.INTEREST_PAYMENT_MADE,0) INTEREST_PAYMENT_MADE,
        CASE
                WHEN (ps.PRINCPALPAYMENT = wtp.PRINCIPAL_PAYMENT_MADE AND ps.INTERESTPAYMENT = wtp.INTEREST_PAYMENT_MADE AND ps.DUEDATE <= SYSDATE)
                OR (ps.DUEDATE > SYSDATE)
                THEN 1
                ELSE 0
        END AS comparison
FROM
        PSchedule PS
LEFT JOIN WITH_TOTAL_PAYMENT_LOAN WTP
ON
        ps.DEAL_DEALID = wtp.DEAL_DEALID
        AND ps.DUEDATE = wtp.BALANCEDATE
JOIN DEAL D
ON d.DEALID = ps.DEAL_DEALID
JOIN Branch B
ON b.BRANCHID = d.BRANCH_BRANCHID
ORDER BY
        ps.DEAL_DEALID, ps.DUEDATE),
CountOverduePayments AS(SELECT CP.BRANCHID, COUNT(CP.comparison) Count_Overdue_payments
FROM comparisonpayments cp
WHERE CP.comparison = 0
GROUP BY CP.BRANCHID),
SumOverduePayments AS(SELECT CP.BRANCHID, SUM(cp.PRINCPALPAYMENT + cp.INTERESTPAYMENT) sum_overdue_payments
FROM comparisonpayments cp
WHERE CP.comparison = 0
GROUP BY CP.BRANCHID),
InterestIncome as(SELECT CP.BRANCHID, SUM(cp.INTERESTPAYMENT) sum_interest_payments
FROM comparisonpayments cp
WHERE CP.comparison = 1 AND (cp.PRINCPALPAYMENT = cp.PRINCIPAL_PAYMENT_MADE AND cp.INTERESTPAYMENT = cp.INTEREST_PAYMENT_MADE AND cp.DUEDATE <= SYSDATE)
GROUP BY CP.BRANCHID
),
CountOverdueLoans AS (SELECT CP.BRANCHID, COUNT(DISTINCT CP.DEAL_DEALID) Count_Overdue_Loans,
(SELECT c.ClientID
         FROM AccountDeal ad
         JOIN Account a ON ad.Account_AccountID = a.AccountID
         JOIN Client c ON a.CLIENT_CLIENTID = c.ClientID
         WHERE ad.Deal_Deal_ID = cp.DEAL_DEALID
         FETCH FIRST 1 ROWS ONLY) AS ClientID
FROM comparisonpayments cp
WHERE CP.comparison = 0
GROUP BY CP.BRANCHID, cp.DEAL_DEALID),
TotalMissedPayments AS (
    SELECT
        c.ClientID,
        COALESCE(cmp.TotalMissedPayments, 0) + COALESCE(col.Count_Overdue_Loans, 0) AS TotalMissedPayments
    FROM
        Client c
    LEFT JOIN
        ClientMissedPaymentsEx cmp ON c.ClientID = cmp.ClientID
    LEFT JOIN
        CountOverdueLoans col ON c.ClientID = col.ClientID),
OverdueScoring AS (SELECT
    ClientID,
    CASE
        WHEN TotalMissedPayments > 0 THEN 0  -- Если есть хотя бы одна просрочка
        ELSE 5  -- Если просрочек нет
    END AS PaymentHistoryScore
    FROM
    TotalMissedPayments),
InternalLoanHistory AS (
SELECT
        DISTINCT
        c.ClientID,
        d.DealID,
        d.DealAmount
FROM
                Client c
JOIN
        Account a ON
                c.ClientID = a.Client_ClientID
JOIN
        AccountDeal ad ON
                a.AccountID = ad.Account_AccountID
JOIN
        Deal d ON
                ad.Deal_Deal_ID = d.DealID
LEFT JOIN
        AccountBalance ab ON
                ab.Account_AccountID = a.AccountID
WHERE
                d.DealType = 'Кредитные операции'
        AND (a.ACCOUNTTYPE = 'Счёт основного долга'
                OR a.ACCOUNTTYPE = 'Счёт процентов')
        AND ab.BALANCEAMOUNT LIKE 0
),
ExternalLoanHistory AS (
SELECT
        c.ClientID,
        ech.LoanAmount
FROM
        Client c
JOIN
        ExternalCreditHistory ech ON
        c.ClientID = ech.Client_ClientID
WHERE
        ech.CurrentStatus = 'Закрыт'
),
TotalHistory AS (SELECT pod.clientid, SUM(pod.dealamount) total_sum_history FROM(SELECT
        ilh.clientid,
        ilh.dealamount
FROM
        InternalLoanHistory ilh
UNION ALL
SELECT elh.clientid, elh.loanamount
FROM ExternalLoanHistory elh) POD
GROUP BY pod.clientid),
RepaymentScoring  AS (SELECT th.clientid, th.total_sum_history,
CASE
        WHEN th.total_sum_history >= 1000000 THEN 5
        WHEN th.total_sum_history BETWEEN 500000 AND 1000000 THEN 4
        WHEN th.total_sum_history BETWEEN 200000 AND 500000 THEN 3
        WHEN th.total_sum_history BETWEEN 100000 AND 200000 THEN 2
        WHEN th.total_sum_history < 100000 THEN 1
        ELSE 0
    END AS TotalLoanScore
FROM TotalHistory th)
SELECT DISTINCT c.clientid "Идентификатор клиента",
ages.age_score "Баллы возраста",
dtis.dti_score "Баллы DTI",
ms.marital_score "Баллы семейного положения",
COALESCE(cs.CollateralScore,0) "Баллы процента залога от суммы кредита",
os.PaymentHistoryScore "Баллы просрочек по кредитам",
COALESCE(rs.TotalLoanScore,0) "Баллы по сумме ранее взятых кредитов и их погашении",
COALESCE(acs.ACTIVE_LOANS_SCOR,0) "Баллы по количеству активных кредитов",
mis.INCOME_SCORE "Баллы ежемесячного дохода",
ages.age_score +
dtis.dti_score +
ms.marital_score +
COALESCE(cs.CollateralScore,0) +
os.PaymentHistoryScore +
COALESCE(rs.TotalLoanScore,0)  +
COALESCE(acs.ACTIVE_LOANS_SCOR,0)  +
mis.INCOME_SCORE "Сумма баллов",
CASE
```

```
        WHEN ages.age_score + dtis.dti_score + ms.marital_score + COALESCE(cs.CollateralScore,0) + os.PaymentHistoryScore + COALESCE(rs.TotalLoanScore,0)  + COALESC
        WHEN ages.age_score + dtis.dti_score + ms.marital_score + COALESCE(cs.CollateralScore,0) + os.PaymentHistoryScore + COALESCE(rs.TotalLoanScore,0)  + COALESCE
        WHEN ages.age_score + dtis.dti_score + ms.marital_score + COALESCE(cs.CollateralScore,0) + os.PaymentHistoryScore + COALESCE(rs.TotalLoanScore,0)  + COALESCE
        WHEN ages.age_score + dtis.dti_score + ms.marital_score + COALESCE(cs.CollateralScore,0) + os.PaymentHistoryScore + COALESCE(rs.TotalLoanScore,0)  + COALESCE
        WHEN ages.age_score + dtis.dti_score + ms.marital_score + COALESCE(cs.CollateralScore,0) + os.PaymentHistoryScore + COALESCE(rs.TotalLoanScore,0)  + COALESCE
END AS "Категория качества кредита"
FROM CLIENT c
LEFT JOIN AgeScoring ages
ON c.CLIENTID = ages.Clientid
LEFT JOIN DTIScoring dtis
ON dtis.clientid = c.clientid
LEFT JOIN MaritalScoring ms
ON ms.clientid = c.clientid
LEFT JOIN CollateralScoring cs
ON c.clientid = cs.ClientID
LEFT JOIN OverdueScoring os
ON c.clientid = os.ClientID
LEFT JOIN RepaymentScoring rs
ON c.clientid = rs.clientid
LEFT JOIN ActiveCreditsScoring  acs
ON c.clientid = acs.clientid
LEFT JOIN MonthlyIncomeScoring mis
ON c.clientid = mis.clientid
ORDER BY
C.CLIENTID
```

Идентификатор клиента	Баллы возраста	Баллы DTI	Баллы семейного положения	Баллы процента залога от суммы кредита	Баллы просрочек по кредитам	Баллы по сумме ранее взятых кредитов и их погашении	Баллы по кол активных кре
1	4	5	5	5	5	4	
2	5	0	2	0	5	0	
3	5	5	5	0	0	0	
4	4	4	2	0	5	0	
5	4	5	5	0	5	4	
6	4	1	2	0	5	0	
7	5	4	5	5	0	0	
8	4	4	2	0	5	0	
9	5	5	5	0	5	3	
10	4	4	2	5	0	0	