```sql
-- Creating a primary key for a customer table that store uniques ID
ALTER TABLE [Customers$]
ADD CONSTRAINT PK_Customers PRIMARY KEY (customer_id);
GO

-- Ensuring matching data types
ALTER TABLE [Disbursements$]
ALTER COLUMN customer_id NVARCHAR(255) NOT NULL;
GO

-- Creating Foreign Key relationship
ALTER TABLE [Disbursements$]
ADD CONSTRAINT FK_Disbursements_Customers
FOREIGN KEY (customer_id) REFERENCES [Customers$](customer_id);
GO

-- Ensuring matching data types first
ALTER TABLE [Repayments$]
ALTER COLUMN customer_id NVARCHAR(255) NOT NULL;
GO

-- Creating Foreign Key relationship
ALTER TABLE [Repayments$]
ADD CONSTRAINT FK_Repayments_Customers
FOREIGN KEY (customer_id) REFERENCES [Customers$](customer_id);
GO

-- Verifying thr relationships
EXEC sp_helpconstraint '[Disbursements$]';
EXEC sp_helpconstraint '[Repayments$]';
GO

-- Inspecting your data structure to document key features
SELECT DISTINCT tenure,
                MIN(loan_amount) AS Min_Loan,
                MAX(loan_amount) AS Max_Loan,
                AVG(loan_amount) AS Avg_Loan,
                AVG(loan_fee) AS Avg_Fee
FROM [Disbursements$]
GROUP BY tenure;
GO


-- Monthly Loan Disbursement Trends
SELECT
    YEAR(disb_date) AS Year,
    MONTH(disb_date) AS Month,
    COUNT(*) AS Loan_Count,
    SUM(loan_amount) AS Total_Disbursement
FROM [Disbursements$]
GROUP BY YEAR(disb_date), MONTH(disb_date)
ORDER BY Year, Month;
```

```sql
GO


-- Monthly Repayment Trends
SELECT
    LEFT(rep_month,4) AS Year,
    RIGHT(rep_month,2) AS Month,
    COUNT(*) AS Repayment_Count,
    SUM(amount) AS Total_Repayment
FROM [Repayments$]
GROUP BY LEFT(rep_month,4), RIGHT(rep_month,2)
ORDER BY Year, Month;
GO



-- Current Credit Exposure and Risk Management:
-- Current Outstanding Credit Exposure (Loan amount disbursed minus total repayment 
  per customer)
SELECT
    d.customer_id,
    SUM(d.loan_amount) AS Total_Loan_Amount,
    ISNULL(SUM(r.amount),0) AS Total_Repaid,
    (SUM(d.loan_amount) - ISNULL(SUM(r.amount),0)) AS Outstanding_Exposure
FROM [Disbursements$] d
LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
GROUP BY d.customer_id
ORDER BY Outstanding_Exposure DESC;
GO



-- Default Rate Calculation (for risk analysis)
SELECT
    COUNT(DISTINCT d.customer_id) AS Total_Customers,
    COUNT(DISTINCT CASE WHEN ISNULL(r.amount,0) < d.loan_amount THEN d.customer_id 
      END) AS Defaulting_Customers,
    (COUNT(DISTINCT CASE WHEN ISNULL(r.amount,0) < d.loan_amount THEN d.customer_id 
      END) * 100.0 / COUNT(DISTINCT d.customer_id)) AS Default_Rate_Percent
FROM [Disbursements$] d
LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id;
GO



-- Monthly Default Rate Query
SELECT
    YEAR(d.disb_date) AS Year,
    MONTH(d.disb_date) AS Month,
    COUNT(DISTINCT CASE WHEN ISNULL(r.amount,0) < d.loan_amount THEN d.customer_id 
      END)*100.0/COUNT(DISTINCT d.customer_id) AS Default_Rate
FROM [Disbursements$] d
```

```sql
LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
GROUP BY YEAR(d.disb_date), MONTH(d.disb_date);
GO


-- default rates by segment
SELECT
    d.tenure,
    COUNT(DISTINCT CASE WHEN ISNULL(r.amount,0) < d.loan_amount THEN d.customer_id
      END)*100.0/COUNT(DISTINCT d.customer_id) AS Default_Rate_Percent
FROM [Disbursements$] d
LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
GROUP BY d.tenure;
GO

-- Segmenting by Loan Amount
SELECT
  Loan_Amount_Range,
  COUNT(DISTINCT customer_id) AS Num_Customers,
  COUNT(DISTINCT CASE WHEN ISNULL(total_repaid,0) < loan_amount THEN customer_id END)
    AS Num_Defaulted_Customers,
  (COUNT(DISTINCT CASE WHEN ISNULL(total_repaid,0) < loan_amount THEN customer_id END)
     * 100.0 /
    COUNT(DISTINCT customer_id)) AS Default_Rate_Percent
FROM (
    SELECT
        d.customer_id,
        d.loan_amount,
        SUM(r.amount) AS total_repaid,
        CASE
            WHEN d.loan_amount BETWEEN 0 AND 1000 THEN '0-1K'
            WHEN d.loan_amount BETWEEN 1001 AND 3000 THEN '1K-3K'
            WHEN d.loan_amount BETWEEN 3001 AND 5000 THEN '3K-5K'
            ELSE '5K+'
        END AS Loan_Amount_Range
    FROM [Disbursements$] d
    LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
    GROUP BY d.customer_id, d.loan_amount
) loan_summary
GROUP BY Loan_Amount_Range
ORDER BY Loan_Amount_Range;
GO


-- Grouping by Exposure Range
SELECT
  CASE
    WHEN Outstanding_Exposure BETWEEN 0 AND 1000 THEN '0-1K'
    WHEN Outstanding_Exposure BETWEEN 1001 AND 5000 THEN '1K-5K'
    WHEN Outstanding_Exposure BETWEEN 5001 AND 10000 THEN '5K-10K'
    WHEN Outstanding_Exposure BETWEEN 10001 AND 20000 THEN '10K-20K'
    ELSE '20K+'
  END AS Exposure_Range,
  COUNT(customer_id) AS Num_Customers,
  SUM(Outstanding_Exposure) AS Total_Outstanding
FROM (
    SELECT
        d.customer_id,
        (SUM(d.loan_amount) - ISNULL(SUM(r.amount),0)) AS Outstanding_Exposure
    FROM [Disbursements$] d
    LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
    GROUP BY d.customer_id
) sub
GROUP BY
  CASE
    WHEN Outstanding_Exposure BETWEEN 0 AND 1000 THEN '0-1K'
    WHEN Outstanding_Exposure BETWEEN 1001 AND 5000 THEN '1K-5K'
    WHEN Outstanding_Exposure BETWEEN 5001 AND 10000 THEN '5K-10K'
    WHEN Outstanding_Exposure BETWEEN 10001 AND 20000 THEN '10K-20K'
    ELSE '20K+'
  END
ORDER BY Exposure_Range;
GO



-- Monthly Profit/Loss Estimation (for forecasting)
SELECT
    YEAR(d.disb_date) AS Year,
    MONTH(d.disb_date) AS Month,
    SUM(d.loan_fee) AS Total_Fees_Earned,
    SUM(ISNULL(r.amount,0)) AS Total_Repaid,
    (SUM(d.loan_fee) - (SUM(d.loan_amount) - SUM(ISNULL(r.amount,0)))) AS
      Estimated_Profit_Loss
FROM [Disbursements$] d
LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
GROUP BY YEAR(d.disb_date), MONTH(d.disb_date)
ORDER BY Year, Month;
GO



-- Recommendations for Provisioning/Write-off Thresholds
SELECT
    d.customer_id,
    d.disb_date,
    DATEADD(DAY, CAST(LEFT(d.tenure,2) AS INT), d.disb_date) AS Due_Date,
    SUM(d.loan_amount) AS Total_Loan,
    SUM(ISNULL(r.amount,0)) AS Total_Repaid,
    (SUM(d.loan_amount) - SUM(ISNULL(r.amount,0))) AS Outstanding_Amount,
    DATEDIFF(day, DATEADD(DAY, CAST(LEFT(d.tenure,2) AS INT), d.disb_date), GETDATE())
      AS Days_Overdue
FROM [Disbursements$] d
```

```
LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
GROUP BY d.customer_id, d.disb_date, d.tenure
HAVING (SUM(d.loan_amount) - SUM(ISNULL(r.amount,0))) > 0
ORDER BY Days_Overdue DESC;
GO



-- Portfolio Triggers & Alerts Recommendations
SELECT
    YEAR(d.disb_date) AS Year,
    MONTH(d.disb_date) AS Month,
    COUNT(DISTINCT CASE WHEN ISNULL(r.amount,0) < d.loan_amount THEN d.customer_id
      END)*100.0/COUNT(DISTINCT d.customer_id) AS Default_Rate
FROM [Disbursements$] d
LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
GROUP BY YEAR(d.disb_date), MONTH(d.disb_date)
ORDER BY Year, Month;
GO



-- Product Design & Feature Recommendations
SELECT
    d.tenure,
    COUNT(DISTINCT CASE WHEN ISNULL(r.amount,0) < d.loan_amount THEN d.customer_id
      END)*100.0/COUNT(DISTINCT d.customer_id) AS Default_Rate_Percent
FROM [Disbursements$] d
LEFT JOIN [Repayments$] r ON d.customer_id = r.customer_id
GROUP BY d.tenure;
GO
```