

SQL

A. Customer Analytics

1. Total Customers

Counts how many customers you have.

Query:

```
SELECT COUNT(*) AS TotalCustomers  
FROM Customers;
```

Results		Messages
	TotalCustomers	
1	5000	

2. New Customers by Year

Shows how many new customers joined each year

Query:

```
SELECT YEAR(JoinDate) AS JoinYear, COUNT(*) AS NewCustomers  
FROM Customers  
GROUP BY YEAR(JoinDate)  
ORDER BY JoinYear;
```

Results			Messages
	JoinYear	NewCustomers	
1	2015	346	
2	2016	493	
3	2017	488	
4	2018	542	
5	2019	509	
6	2020	505	
7	2021	504	
8	2022	476	
9	2023	506	
10	2024	474	
11	2025	157	

3. Active/Inactive Customers

Checks which customers use accounts, loans, or cards and which don't.

Query:

SELECT

c.CustomerID,

c.FirstName + ' ' + c.LastName **AS** FullName,

CASE

WHEN a.CustomerID **IS NOT NULL** OR l.CustomerID **IS NOT NULL** OR
ca.CustomerID **IS NOT NULL** **THEN** 'Active'

ELSE 'Inactive'

END AS Status

FROM Customers c

LEFT JOIN Accounts a **ON** c.CustomerID = a.CustomerID

LEFT JOIN Loans l **ON** c.CustomerID = l.CustomerID

LEFT JOIN Cards ca **ON** c.CustomerID = ca.CustomerID;

	CustomerID	FullName	Status
1	1	Dustin Diaz	Active
2	1	Dustin Diaz	Active
3	2	Jessica Anderson	Active
4	3	Jeremy Wagner	Active
5	3	Jeremy Wagner	Active
6	4	Crystal Roberts	Active
7	5	Anna Bryant	Active
8	5	Anna Bryant	Active
9	6	Stephanie Anderson	Active

4. Total Active Customers (last 12 months)

Counts customers who made transactions in the last year.

Query:

SELECT COUNT(DISTINCT a.CustomerID) **AS** Total_Active_Customers

FROM Transactions t

JOIN Accounts a **ON** t.AccountID = a.AccountID

WHERE t.TransactionDate >= DATEADD(MONTH, -12, '2025-05-17');

Results		Messages	
	Total_Active_Customers		
1	2610		

5. Churn Risks (no transactions in last 6 months)

Finds customers who have not done anything in 6 months, at risk of leaving.

Query:

```
SELECT COUNT(DISTINCT c.CustomerID) AS Churn_Risk_Customers
FROM Customers c
LEFT JOIN Accounts a ON c.CustomerID = a.CustomerID
LEFT JOIN Transactions t ON a.AccountID = t.AccountID
WHERE t.TransactionID IS NULL
OR t.TransactionDate < DATEADD(MONTH, -6, '2025-05-17');
```

	Churn_Risk_Customers
1	4960

6. Customers with Accounts

Counts how many customers have accounts.

Query:

```
SELECT COUNT(DISTINCT CustomerID) AS CustomersWithAccounts
FROM Accounts;
```

	CustomersWithAccounts
1	3171

7. Average Accounts per Customer

Shows the average number of accounts each customer has.

Query:

```
SELECT CAST(COUNT(a.AccountID) AS FLOAT) / COUNT(DISTINCT a.CustomerID)
AS Avg_Accounts_Per_Customer
FROM Accounts a;
```

	Avg_Accounts_Per_Customer
1	1.57678965625985

B. Account & Balance Analysis

1. Total Accounts

Counts all accounts.

Query:

```
SELECT COUNT(*) AS TotalAccounts
FROM Accounts;
```

	TotalAccounts
1	5000

2. Account Type Distribution

Shows how many accounts of each type (like savings or checking) exist.

Query:

```
SELECT AccountType, COUNT(*) AS CountPerType
FROM Accounts
GROUP BY AccountType;
```

	AccountType	CountPerType
1	Savings	1605
2	Business	1757
3	Checking	1638

3. Total Balance by Account Type

Adds up the money in each account type.

Query:

```
SELECT a.AccountType, SUM(a.Balance) AS Total_Balance
FROM Accounts a
GROUP BY a.AccountType;
```

	AccountType	Total_Balance
1	Savings	79055168
2	Business	87161577
3	Checking	83154050

4. Average Balance by Account Type

Shows the average amount of money in each account type.

Query:

```
SELECT AccountType, AVG(Balance) AS AvgBalance
FROM Accounts
GROUP BY AccountType;
```

	AccountType	AvgBalance
1	Savings	49255
2	Business	49608
3	Checking	50765

5. Account Creation by Year

Tracks how many accounts were opened each year.

Query:

```
SELECT YEAR(CreatedDate) AS YearCreated, COUNT(*) AS AccountsCreated
FROM Accounts
GROUP BY YEAR(CreatedDate)
ORDER BY YearCreated;
```

	YearCreated	AccountsCreated
1	2015	319
2	2016	534
3	2017	498
4	2018	489
5	2019	496
6	2020	461
7	2021	494
8	2022	510
9	2023	500
10	2024	502
11	2025	197

6. Customer-Account Link

Lists which customers own, which accounts and their details.

Query:

```
SELECT
    a.AccountID,
    c.FirstName + ' ' + c.LastName AS CustomerName,
    a.AccountType,
    a.Balance,
    a.CreatedDate
FROM Accounts a
JOIN Customers c ON a.CustomerID = c.CustomerID;
```

	AccountID	CustomerName	AccountType	Balance	CreatedDate
1	2982	Dustin Diaz	Business	25463	2022-07-28
2	1064	Jessica Anderson	Business	20529	2019-07-28
3	3143	Jeremy Wagner	Savings	49912	2016-05-29
4	4693	Jeremy Wagner	Checking	94620	2023-06-20
5	3616	Crystal Roberts	Business	52596	2020-03-24
6	4198	Anna Bryant	Business	41127	2023-12-20
7	848	Anna Bryant	Savings	10610	2015-12-26
8	954	Charles Leach	Savings	78420	2024-09-07
9	2705	Charles Leach	Savings	61210	2024-04-05

7. Accounts with No Transaction

Finds accounts with no activity.

Query:

```
SELECT a.*  
FROM Accounts a  
LEFT JOIN Transactions t ON a.AccountID = t.AccountID  
WHERE t.TransactionID IS NULL;
```

	AccountID	CustomerID	AccountType	Balance	CreatedDate
1	6	735	Business	62579	2022-08-20
2	140	1552	Business	83569	2016-07-16
3	204	2075	Business	93200	2017-08-09
4	227	2327	Business	63666	2021-12-13
5	249	3646	Savings	35224	2024-05-25
6	265	1612	Checking	75467	2016-09-23

8. Top 5 Accounts by Balance:

Shows the 5 accounts with the most money.

Query:

```
SELECT TOP 5 *  
FROM Accounts  
ORDER BY Balance DESC;
```

	AccountID	CustomerID	AccountType	Balance	CreatedDate
1	3722	781	Checking	99968	2024-12-02
2	4611	4950	Business	99874	2017-03-17
3	2261	3855	Checking	99842	2017-12-13
4	1347	3768	Business	99834	2025-04-14
5	154	4109	Checking	99820	2024-08-23

9. Accounts per Customer:

Counts how many accounts each customer has.

Query:

```
SELECT  
    c.CustomerID,  
    c.FirstName + ' ' + c.LastName AS CustomerName,  
    COUNT(a.AccountID) AS NumberOfAccounts  
FROM Customers c  
LEFT JOIN Accounts a ON c.CustomerID = a.CustomerID  
GROUP BY c.CustomerID, c.FirstName, c.LastName;
```

	CustomerID	CustomerName	NumberOfAccounts
1	1	Dustin Diaz	1
2	2	Jessica Anderson	1
3	3	Jeremy Wagner	2
4	4	Crystal Roberts	1

10. Dormant Accounts (no transactions in last 6 months)

Finds accounts with no activity in the last 6 months.

Query:

```
SELECT COUNT(a.AccountID) AS Dormant_Accounts
FROM Accounts a
LEFT JOIN Transactions t ON a.AccountID = t.AccountID
WHERE t.TransactionID IS NULL
OR t.TransactionDate < DATEADD(MONTH, -6, '2025-05-17');
```

	Dormant_Accounts
1	16879

11. Average Balance per Customer:

Shows the average money each customer has across their accounts.

Query:

```
SELECT CAST(SUM(a.Balance) AS FLOAT) / COUNT(DISTINCT a.CustomerID) AS
Avg_Balance_Per_Customer
FROM Accounts a;
```

	Avg_Balance_Per_Customer
1	78641.0580258594

C. Transaction Analytics

1. Total Transactions

Counts all transactions.

Query:

```
SELECT COUNT(*) AS TotalTransactions
FROM Transactions;
```

	TotalTransactions
1	20000

2. Transaction Type Distribution:

Shows how many transactions are deposits, withdrawals, etc.

Query:

```
SELECT TransactionType, COUNT(*) AS CountPerType
FROM Transactions
GROUP BY TransactionType;
```

	TransactionType	CountPerType
1	Payment	5056
2	Transfer	5055
3	Withdrawal	4919
4	Deposit	4970

3. Total and Average Amount by Transaction Type:

Adds up and averages the money for each transaction type.

Query:

```
SELECT
    TransactionType,
    COUNT(*) AS NumTransactions,
    SUM(Amount) AS TotalAmount,
    AVG(Amount) AS AvgAmount
FROM Transactions
GROUP BY TransactionType;
```

	TransactionType	NumTransactions	TotalAmount	AvgAmount
1	Payment	5056	25432001.4021139	5030.06356845607
2	Transfer	5055	25312679.808589	5007.45396806904
3	Withdrawal	4919	24384102.4509029	4957.12593025065
4	Deposit	4970	24980106.4753008	5026.17836525167

4. Transaction Analysis by Month:

Tracks how many transactions and how much money moved each month.

Query:

```
SELECT
    MONTH(TransactionDate) AS Month,
    FORMAT(TransactionDate, 'yyyy-MM') AS YearMonth,
    COUNT(*) AS TransactionsCount,
    SUM(Amount) AS TotalAmount
FROM Transactions
GROUP BY MONTH(TransactionDate), FORMAT(TransactionDate, 'yyyy-MM')
ORDER BY YearMonth ;
```


	Month	YearMonth	TransactionsCount	TotalAmount
1	5	2022-05	397	2052223.4846344
2	6	2022-06	541	2736146.55495453
3	7	2022-07	530	2529131.14809132
4	8	2022-08	579	2783344.88460922
5	9	2022-09	550	2824568.12779808

5. Transaction-Customer Link:

Shows which customers made which transactions.

Query:

SELECT

t.TransactionID,
t.TransactionType,
t.Amount,
t.TransactionDate,
a.AccountID,
c.CustomerID,
c.FirstName + ' ' + c.LastName AS CustomerName

FROM Transactions t

JOIN Accounts a ON t.AccountID = a.AccountID

JOIN Customers c ON a.CustomerID = c.CustomerID;

	TransactionID	TransactionType	Amount	TransactionDate	AccountID	CustomerID	CustomerName
1	1	Transfer	3150.1201171875	2023-09-24	3913	13	Melissa Santiago
2	2	Transfer	6212.1201171875	2022-06-07	2591	1811	Sarah Reed
3	3	Transfer	451.720001220703	2024-11-24	3277	2274	Jean Vazquez
4	4	Deposit	8525.2802734375	2023-04-06	3404	1981	Matthew Mckee
5	5	Deposit	7306.169921875	2025-01-21	4345	2842	Gilbert Todd

6. Top 5 Transactions by Amount:

Lists the 5 biggest transactions.

Query:

SELECT TOP 5 *

FROM Transactions

ORDER BY Amount DESC;

	TransactionID	AccountID	TransactionType	Amount	TransactionDate
1	7443	1463	Deposit	9999.8896484375	2025-04-04
2	17110	4432	Deposit	9999.4599609375	2023-09-26
3	6456	2207	Deposit	9999.25	2024-05-22
4	10101	3155	Deposit	9998.990234375	2024-07-07
5	3555	804	Deposit	9997.4697265625	2022-10-29

7. Average Transaction Value by Account Type:

Shows the average transaction size for each account type.

Query:

```
SELECT a.AccountType, AVG(t.Amount) AS Avg_Transaction_Value
FROM Transactions t
JOIN Accounts a ON t.AccountID = a.AccountID
GROUP BY a.AccountType;
```

	AccountType	Avg_Transaction_Value
1	Savings	5018.91191542289
2	Business	4981.94449105329
3	Checking	5016.78286683096

8. Fraud/Unusual Transactions:

Finds transactions that are unusually large, possibly fraudulent.

Query:

```
WITH Stats AS (
    SELECT AVG(Amount) AS Mean, STDEV(Amount) AS StdDev
    FROM Transactions
)
SELECT t.TransactionID, t.Amount
FROM Transactions t, Stats
WHERE t.Amount > (Stats.Mean + 3 * StdDev);
```

TransactionID	Amount

D. Loan Portfolio

1. Total Loans:

Counts all loans.

Query:

```
SELECT COUNT(*) AS TotalLoans
FROM Loans;
```

	TotalLoans
1	2500

2. Loan Type Distribution:

Shows how many loans are mortgages, personal loans, etc.

Query:

```
SELECT LoanType, COUNT(*) AS CountPerType
FROM Loans
GROUP BY LoanType;
```

	LoanType	CountPerType
1	Education	624
2	Personal	617
3	Car	633
4	Home	626

3. Total Loan Amount by Type:

Adds up the money lent for each loan type.

Query:

```
SELECT l.LoanType, SUM(l.LoanAmount) AS Total_Loan_Amount
FROM Loans l
GROUP BY l.LoanType;
```

	LoanType	Total_Loan_Amount
1	Education	154218864
2	Personal	154925690
3	Car	153848761
4	Home	153665578

4. Total and Average Loan Amount by Type:

Shows the number and average size of loans by type.

Query:

```
SELECT
    LoanType,
    COUNT(*) AS NumberOfLoans,
    SUM(LoanAmount) AS TotalLoanAmount,
    AVG(LoanAmount) AS AvgLoanAmount
FROM Loans
GROUP BY LoanType;
```

	LoanType	NumberOfLoans	TotalLoanAmount	AvgLoanAmount
1	Education	624	154218864	247145
2	Personal	617	154925690	251095
3	Car	633	153848761	243047
4	Home	626	153665578	245472

5. Average Interest Rate per Loan Type:

Shows the average interest rate for each loan type.

Query:

```
SELECT l.LoanType, AVG(l.InterestRate) AS Avg_Interest_Rate
FROM Loans l
GROUP BY l.LoanType;
```

	LoanType	Avg_Interest_Rate
1	Education	7.37951923219057
2	Personal	7.6685413240034
3	Car	7.36938388765706
4	Home	7.50738018313155

6. Interest Rate Analysis:

Tracks loan amounts and highest/lowest interest rates by loan type.

Query:

```
SELECT
-- Loan Type and Date Information
l.LoanType,
MONTH((SELECT TOP 1 l2.LoanStartDate
FROM Loans l2
WHERE l2.LoanType = l.LoanType
AND l2.InterestRate = (SELECT MAX(InterestRate) FROM Loans WHERE
LoanType = l.LoanType))) AS MaxInterestMonth,
YEAR((SELECT TOP 1 l2.LoanStartDate
FROM Loans l2
WHERE l2.LoanType = l.LoanType
AND l2.InterestRate = (SELECT MAX(InterestRate) FROM Loans WHERE
LoanType = l.LoanType))) AS MaxInterestYear,
MONTH((SELECT TOP 1 l2.LoanStartDate
FROM Loans l2
WHERE l2.LoanType = l.LoanType
AND l2.InterestRate = (SELECT MIN(InterestRate) FROM Loans WHERE
LoanType = l.LoanType))) AS MinInterestMonth,
YEAR((SELECT TOP 1 l2.LoanStartDate
FROM Loans l2
WHERE l2.LoanType = l.LoanType
AND l2.InterestRate = (SELECT MIN(InterestRate) FROM Loans WHERE
LoanType = l.LoanType))) AS MinInterestYear,
```

-- Financial Metrics

```
SUM(l.LoanAmount) AS TotalLoanAmount,  
(SUM(l.LoanAmount) * MAX(l.InterestRate) / 100) AS MaxInterestValue,  
(SUM(l.LoanAmount) * MIN(l.InterestRate) / 100) AS MinInterestValue
```

FROM Loans l

GROUP BY l.LoanType;

	LoanType	MaxInterestMonth	MaxInterestYear	MinInterestMonth	MinInterestYear	TotalLoanAmount	MaxInterestValue	MinInterestValue
1	Education	10	2020	12	2020	154218864	19200248.2738509	3855471.6
2	Personal	5	2022	4	2022	154925690	19365711.25	3904127.35845027
3	Car	6	2022	1	2021	153848761	19215709.8967681	3892373.60928352
4	Home	4	2022	6	2021	153665578	19208197.25	3841639.45

7. Loan Duration:

Shows how long each loan lasts in days.

Query:

```
SELECT
```

```
    LoanID,
```

```
    DATEDIFF(DAY, LoanStartDate, LoanEndDate) AS LoanDurationDays
```

```
FROM Loans;
```

	LoanID	LoanDurationDays
1	1	3308
2	2	1163
3	3	1196
4	4	1460
5	5	2531

8. Loan Issue by Year:

Counts loans given out each year.

Query:

```
SELECT
```

```
    YEAR(LoanStartDate) AS IssueYear,
```

```
    COUNT(*) AS LoansIssued
```

```
FROM Loans
```

```
GROUP BY YEAR(LoanStartDate)
```

```
ORDER BY IssueYear;
```

	IssueYear	LoansIssued
1	2020	371
2	2021	669
3	2022	636
4	2023	580
5	2024	244

9. Customer-Loan Link:

Lists which customers have, which loans and their details.

Query:

SELECT

l.LoanID,
c.FirstName + ' ' + c.LastName AS CustomerName,
l.LoanType,
l.LoanAmount,
l.InterestRate,
l.LoanStartDate,
l.LoanEndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID;

	LoanID	CustomerName	LoanType	LoanAmount	InterestRate	LoanStartDate	LoanEndDate
1	977	Dustin Diaz	Car	49722	6.05999994277954	2020-07-16	2028-07-19
2	2222	Jeremy Wagner	Education	376235	9.5	2024-05-03	2026-02-02
3	1982	Anna Bryant	Personal	427380	12.3400001525879	2021-05-16	2026-07-18
4	698	Stephanie Anderson	Car	207802	11.3599996566772	2022-01-27	2030-02-12
5	2414	Stephanie Anderson	Personal	27039	7.80999994277954	2023-06-09	2030-03-27
6	340	Christopher Baker	Home	130485	7.42999982833862	2022-08-26	2025-06-15

10. Top 5 Customers by Loan Amount:

Shows the 5 customers with the biggest loans.

Query:

SELECT TOP 5

c.CustomerID,
c.FirstName + ' ' + c.LastName AS CustomerName,
SUM(l.LoanAmount) AS TotalLoanAmount

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName

ORDER BY TotalLoanAmount DESC;

	CustomerID	CustomerName	TotalLoanAmount
1	1929	Paul Merritt	1612938
2	3668	Nicole Wilson	1483930
3	4054	Katrina Powell	1323490
4	3027	Stacy Todd	1276745
5	2156	Alexander Turner	1269439

11. Upcoming Loan Maturities:

Counts loans due by May 17, 2026.

Query:

```
SELECT COUNT(l.LoanID) AS Upcoming_Maturities
FROM Loans l
WHERE l.LoanEndDate BETWEEN '2025-05-17' AND DATEADD(YEAR, 1, '2025-05-17');
```

	Upcoming_Maturities
1	499

E. Card Issuance & Activity

1. Total Cards:

Counts all issued cards.

Query:

```
SELECT COUNT(*) AS TotalCards
FROM Cards;
```

	TotalCards
1	4000

2. Card Type Distribution:

Shows how many cards are credit, debit, etc.

Query:

```
SELECT CardType, COUNT(*) AS CountPerType
FROM Cards
GROUP BY CardType;
```

	CardType	CountPerType
1	Prepaid	1355
2	Credit	1282
3	Debit	1363

3. Card Issuance Trend (Monthly):

Tracks how many cards were issued each month.

Query:

```
SELECT YEAR(IssuedDate) AS IssueYear, MONTH(IssuedDate) AS IssueMonth,
COUNT(*) AS CardsIssued
FROM Cards
GROUP BY YEAR(IssuedDate), MONTH(IssuedDate)
ORDER BY IssueYear, IssueMonth;
```

	IssueYear	IssueMonth	CardsIssued
1	2020	5	41
2	2020	6	61
3	2020	7	70
4	2020	8	54
5	2020	9	89
6	2020	10	65

4. Cards Expiring by Year:

Counts cards expiring each year.

Query:

```
SELECT
YEAR(ExpirationDate) AS ExpiryYear,
COUNT(*) AS CardsExpiring
FROM Cards
GROUP BY YEAR(ExpirationDate)
ORDER BY ExpiryYear;
```

	ExpiryYear	CardsExpiring
1	2026	584
2	2027	1044
3	2028	1002
4	2029	987
5	2030	383

5. Customer-Card Link:

Shows which customers have which cards.

Query:

```
SELECT
c.CustomerID,
c.FirstName + ' ' + c.LastName AS CustomerName,
ca.CardID,
ca.CardType,
ca.IssuedDate,
ca.ExpirationDate
FROM Cards ca
JOIN Customers c ON ca.CustomerID = c.CustomerID;
```

	CustomerID	CustomerName	CardID	CardType	IssuedDate	ExpirationDate
1	1	Dustin Diaz	564	Prepaid	2023-06-29	2027-10-23
2	1	Dustin Diaz	2932	Prepaid	2022-08-17	2027-08-13
3	5	Anna Bryant	1135	Prepaid	2023-09-01	2029-10-19
4	7	Andrew Watson	3114	Credit	2021-12-03	2029-05-02
5	8	Charles Leach	2571	Debit	2024-03-27	2026-10-02

6. Expired Cards:

Lists cards that expired before May 17, 2025.

Query:

```
SELECT *  
FROM Cards  
WHERE ExpirationDate < '2025-05-17';
```

CardID	CustomerID	CardType	CardNumber	IssuedDate	ExpirationDate

7. Active vs Expired Cards:

Compares active cards to expired ones.

Query:

```
SELECT  
COUNT(CASE WHEN c.ExpirationDate > '2025-05-17' THEN c.CardID END) AS  
Active_Cards,  
COUNT(CASE WHEN c.ExpirationDate <= '2025-05-17' THEN c.CardID END) AS  
Expired_Cards  
FROM Cards c;
```

	Active_Cards	Expired_Cards
1	4000	0

8. Average Cards per Customer:

Shows the average number of cards per customer by card type

Query:

```
SELECT CardType, CAST(COUNT(CardID) AS FLOAT) / COUNT(DISTINCT  
CustomerID) AS Avg_Cards_Per_Customer  
FROM Cards  
GROUP BY CardType;
```

	CardType	Avg_Cards_Per_Customer
1	Credit	1.1295154185022
2	Debit	1.13868003341688
3	Prepaid	1.15811965811966

F. Customer Support Insights

1. Total Number of Support Calls:

Counts all support calls.

Query:

```
SELECT COUNT(sc.CallID) AS Total_Support_Calls
FROM SupportCalls sc;
```

	Total_Support_Calls
1	3000

2. Resolved vs Unresolved Calls:

Shows how many calls were solved vs. still open.

Query:

```
SELECT
    COUNT(CASE WHEN sc.Resolved = 1 THEN sc.CallID END) AS Resolved_Calls,
    COUNT(CASE WHEN sc.Resolved = 0 THEN sc.CallID END) AS Unresolved_Calls
FROM SupportCalls sc;
```

	Resolved_Calls	Unresolved_Calls
1	1479	1521

3. Top Issue Categories:

Lists the most common reasons customers call.

Query:

```
SELECT sc.IssueType, COUNT(sc.CallID) AS Issue_Count
FROM SupportCalls sc
GROUP BY sc.IssueType
ORDER BY Issue_Count DESC;
```

	IssueType	Issue_Count
1	Transaction Dispute	774
2	Account Access	768
3	Loan Query	729
4	Card Issue	729

4. Calls by Month:

Tracks how many support calls happen each month.

Query:

```
SELECT
    FORMAT(CallDate, 'yyyy-MM') AS YearMonth,
    COUNT(*) AS CallsCount
FROM SupportCalls
GROUP BY FORMAT(CallDate, 'yyyy-MM')
ORDER BY YearMonth;
```

	YearMonth	CallsCount
1	2024-05	192
2	2024-06	232
3	2024-07	253
4	2024-08	242
5	2024-09	237
6	2024-10	256
7	2024-11	270

5. Customer-Call Link:

Shows which customers made which support calls.

Query:

```
SELECT
    s.CallID,
    s.CallDate,
    s.IssueType,
    s.Resolved,
    c.FirstName + ' ' + c.LastName AS CustomerName
FROM SupportCalls s
JOIN Customers c ON s.CustomerID = c.CustomerID;
```

	CallID	CallDate	IssueType	Resolved	CustomerName
1	1885	2024-07-16	Loan Query	0	Jessica Anderson
2	1887	2025-01-19	Loan Query	0	Stephanie Anderson
3	2330	2024-08-18	Transaction Dispute	1	Andrew Watson
4	2438	2024-07-06	Card Issue	1	Andrew Watson
5	1868	2024-07-20	Account Access	1	Charles Leach
6	1137	2024-12-01	Transaction Dispute	0	Tracy Dominguez

Additional Queries

1. Customer Profitability Score:

Ranks the top 10 customers by their account balances, loans, and recent transactions.

Query:

```
SELECT TOP 10
  c.CustomerID,
  c.FirstName,
  c.LastName,
  COALESCE(SUM(a.Balance), 0) + COALESCE(SUM(l.LoanAmount), 0) +
  COALESCE(SUM(CASE WHEN t.TransactionDate >= DATEADD(MONTH, -12, '2025-
05-17') THEN t.Amount ELSE 0 END), 0) AS ProfitabilityScore
FROM Customers c
LEFT JOIN Accounts a ON c.CustomerID = a.CustomerID
LEFT JOIN Loans l ON c.CustomerID = l.CustomerID
LEFT JOIN Transactions t ON a.AccountID = t.AccountID
GROUP BY c.CustomerID, c.FirstName, c.LastName
ORDER BY ProfitabilityScore DESC;
```

	CustomerID	FirstName	LastName	ProfitabilityScore
1	1145	Denise	Nielsen	14819566.3398438
2	2773	Renee	Reyes	13414083.5400696
3	1956	Lindsey	Giles	12139910.4587402
4	3401	Brittany	Hampton	12076803.9587402
5	3970	Shane	Conrad	11926581.9599915
6	790	Ronald	Gibbs	11484303.7006226
7	2277	Ryan	Mathis	11270520.4801025
8	3653	Tara	Reid	11202033.3994141
9	1409	Brian	Payne	10726751.1196289
10	350	Katie	Alexander	10577859.7199707

2. Customer Risk Score:

Measures risk by comparing transaction count to loan amounts.

Query:

```
SELECT
  c.CustomerID,
  COUNT(t.TransactionID) AS TransactionsCount,
  COALESCE(SUM(l.LoanAmount), 0) AS TotalLoanAmount,
  CASE
    WHEN COALESCE(SUM(l.LoanAmount), 0) = 0 THEN 0
    ELSE COUNT(t.TransactionID) / (SUM(l.LoanAmount) / 1000)
```

```

    END AS RiskScore
FROM Customers c
LEFT JOIN Accounts a ON c.CustomerID = a.CustomerID
LEFT JOIN Transactions t ON a.AccountID = t.AccountID
LEFT JOIN Loans l ON c.CustomerID = l.CustomerID
GROUP BY c.CustomerID;

```

	CustomerID	TransactionsCount	TotalLoanAmount	RiskScore
1	2917	6	1894098	0
2	2253	2	0	0
3	3581	0	393916	0
4	1589	0	0	0
5	1566	11	0	0
6	902	7	0	0
7	2894	10	1033710	0

3. Loan Interest Rate Analysis:

Shows average and range of interest rates for each loan type.

Query:

```

SELECT
    LoanType,
    AVG(InterestRate) AS AvgInterestRate,
    MAX(InterestRate) - MIN(InterestRate) AS InterestRateRange
FROM Loans
GROUP BY LoanType;

```

	LoanType	AvgInterestRate	InterestRateRange
1	Education	7.37951923219057	9.94999980926514
2	Personal	7.6685413240034	9.98000001907349
3	Car	7.36938388765706	9.95999979972839
4	Home	7.50738018313155	10

4. Support Risk Factor:

Identifies customers with many support calls compared to transactions, indicating potential issues.

Query:

```

SELECT
    c.CustomerID,
    COUNT(sc.CallID) AS SupportCallCount,
    COUNT(t.TransactionID) AS TransactionsCount,
    CASE
        WHEN COUNT(t.TransactionID) = 0 THEN NULL

```

```

ELSE COUNT(sc.CallID) / CAST(COUNT(t.TransactionID) AS FLOAT)
END AS SupportRiskFactor
FROM Customers c
LEFT JOIN Accounts a ON c.CustomerID = a.CustomerID
LEFT JOIN Transactions t ON a.AccountID = t.AccountID
LEFT JOIN SupportCalls sc ON c.CustomerID = sc.CustomerID
GROUP BY c.CustomerID;

```

	CustomerID	SupportCallCount	TransactionsCount	SupportRiskFactor
1	2917	0	6	0
2	2253	2	2	1
3	3581	0	0	NULL
4	1589	0	0	NULL
5	1566	33	33	1
6	902	0	7	0
7	2894	0	5	0

5. Clients Nearing Loan Repayment Deadlines

These are the clients whose loan repayment periods are nearing their end within 60 days, which suggests they might default. You can negotiate with them to renew, for example, as a business proposal or suggest a new loan.

Query:

```

SELECT
    L.CustomerID,
    L.LoanID,c.phone,c.Email,
    L.LoanAmount,
    L.LoanEndDate,
    DATEDIFF(DAY, GETDATE(), L.LoanEndDate) AS DaysToLoanEnd
FROM
    Loans L
    JOIN Customers AS c on L.CustomerID = C.CustomerID
WHERE
    L.LoanEndDate BETWEEN GETDATE() AND DATEADD(DAY, 60, GETDATE());

```

	CustomerID	LoanID	phone	Email	LoanAmount	LoanEndDate	DaysToLoanEnd
1	11	340	001-002-797-3791x10944	paige20@gmail.com	130485	2025-06-15 00:00:00.000	26
2	96	1948	418-164-6646x98839	kflores@smith.com	309534	2025-06-16 00:00:00.000	27
3	115	269	0092247502	daniel33@yahoo.com	61753	2025-06-13 00:00:00.000	24
4	117	1116	+1-543-837-1968	ybarnett@yahoo.com	119272	2025-07-06 00:00:00.000	47
5	208	1119	001-369-281-7992x112	wheelerbrett@gmail.com	171452	2025-06-14 00:00:00.000	25
6	223	210	001-087-426-1212x766	frenchkyle@jones-martinez.info	3968	2025-05-29 00:00:00.000	9
7	233	515	+1-796-300-4832	ryanturner@gmail.com	263793	2025-07-11 00:00:00.000	52
8	270	403	709-833-7345x0891	jorgewilson@luna.info	58928	2025-05-22 00:00:00.000	2
9	334	2463	001-199-897-4642x1869	glenduran@yahoo.com	49365	2025-06-05 00:00:00.000	16
10	419	2287	430.270.6465	arogers@gmail.com	394910	2025-07-12 00:00:00.000	53
11	448	1940	143.663.5718	juliaaguilar@hotmail.com	26669	2025-07-10 00:00:00.000	51