Explorer Bank

INFO 364 Group Project
By: Chris Combs, Juwan Jackson, Kevon Mahban, Chris Copeland, Emmanuel Afriyie

Business Scenario

- Explorer Banking serves a broad spectrum of clients, ranging from individuals to large businesses.
- Services offered include basic checking accounts to advanced investment solutions.
- Responding to rapid advancements in financial technology, the bank is launching a new database system.
- The new database system aims to streamline operations, accelerate transactions, and ensure regulatory compliance.
- Its implementation is geared towards enhancing customer service and supporting future growth in the digital era.

Database requirement analysis

1. Loan
2. Application
3. Branch
4. Loan Payment
5. Transfer
6. Transaction
7. Card (Credit,Debit)
8. Representative
9. Credit Transaction
10. Customer
11. Account

To implement a new database system, granting easier handle of data, boost in transaction speed, and pave the way for new services and optimal improvements as the bank continues to grow and evolve in the digital era.

CUSTOMER

This entity stores information related to a customer in the banking system.

Attributes:

- Customer’s ID
- Representative ID (Foreign Key)
- Password
- First name
- Middle name
- Last name
- Street address
- City
- State

ACCOUNT

This entity contains bank accounts relating to an individual customer.

Attributes:

- Account ID
- Customer ID (Foreign)
- Account Type
- Balance
- Interest Rate
- Account Status

CARD

This supertype entity stores information related to a customer in the banking system.

Attributes:

- Card ID
- Account ID(Foreign)
- Card Number
- Expiration Date
- Security Code
- Card Type

DEBIT

This subtype entity Debit Card information

Attributes:

- PIN Number

CREDIT

This subtype entity stores Credit Card information

Attributes:

- Credit Limit
- Payment Due Date

TRANSACTION

The Transaction entity keeps track of payments and other transaction information.

Attributes:

- Transaction ID
- Account ID(Foreign)
- Card ID(Foreign)
- Amount
- Transaction Date
- Transaction Type

CREDIT TRANSACTION

The Credit Transaction entity is where information about transactions related to a credit account is stored.

Attributes:

- Credit TransactionID
- Card ID (Foreign)
- Transaction ID (Foreign)
- Amount
- Statement Balance

LOAN

The loan entity contains loans linked to a specific account, meaning an account could have zero loans or multiple loans linked to it.

Attributes:

- Loan ID
- Loan Type
- Principal Amount
- Interest Rate
- Start Date
- End Date

LOAN PAYMENT

The entity contains the payment date and the amount paid. This stores data about which accounts have made a payment on a loan.

Attributes:

- Loan Payment ID
- Loan ID (Foreign)
- Account ID (Foreign)
- Payment Date
- Amount Paid

APPLICATION

This entity contains the status of the application. Associative entity between the Customer and Loan entities.

Attributes:

- Application ID
- Loan ID (Foreign)
- Customer ID (Foreign)
- Status

REPRESENTATIVE

This entity stores information about the customer representative who is associated with aiding the customer.

Attributes:

- Representative ID
- Branch ID (Foreign)
- Name
- Email
- Phone Number

BRANCH

This entity stores information about the Branch that Customer Representatives are assigned to.

Attributes:

- Branch ID
- Branch Name
- Street Address
- City
- State
- Zip Code
- Manager Name
- Phone Number

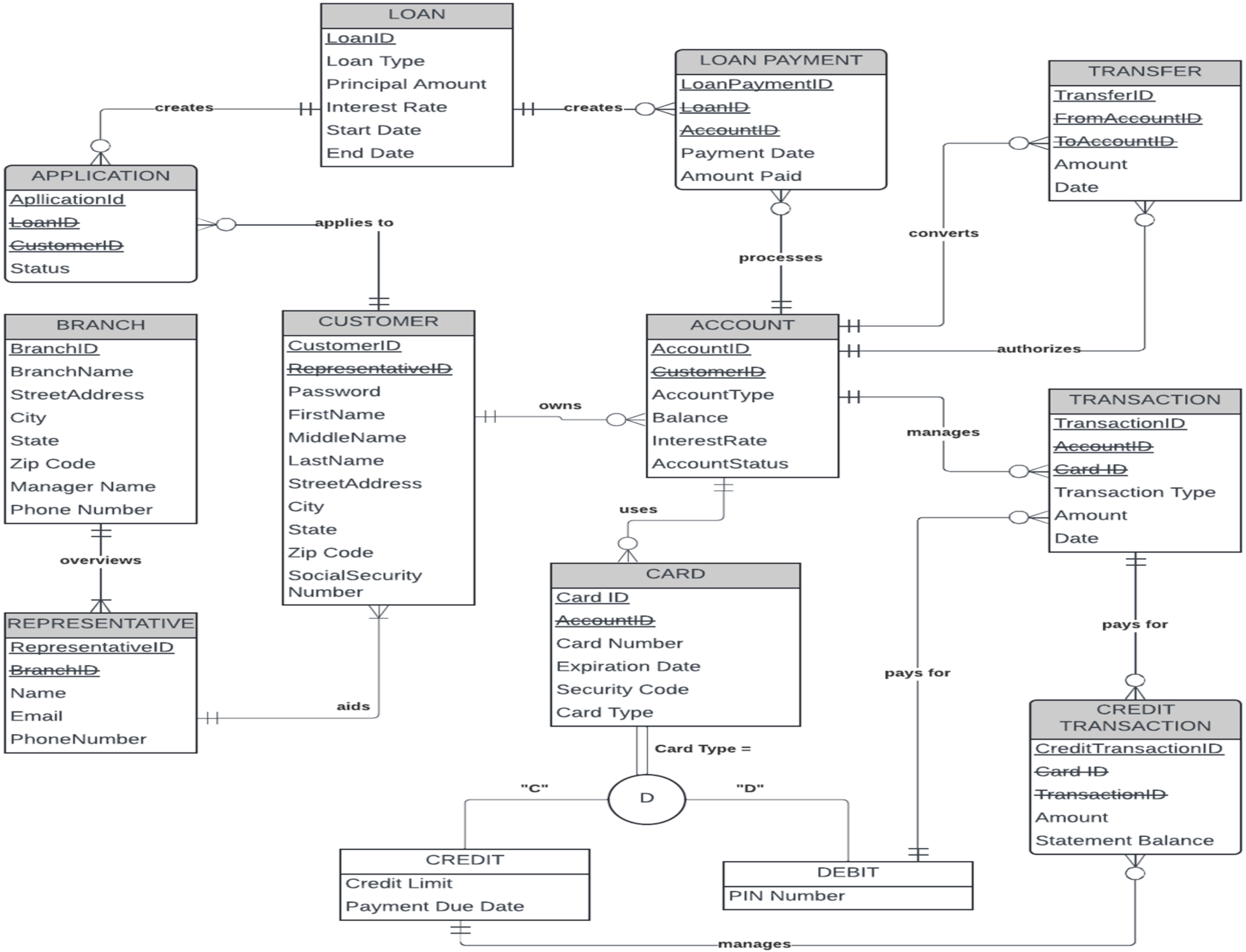
TRANSFER

This entity stores information about Account transfers to different accounts.

Attributes:

- Transfer ID
- FromAccountID (Foreign)
- ToAccountID (Foreign)
- Amount
- Transfer Date

ERD Diagram



LOAN
LoanID
Loan Type
Principal Amount
Interest Rate
Start Date
End Date

LOAN PAYMENT
LoanPaymentID
~~LoanID~~
~~AccountID~~
Payment Date
Amount Paid

TRANSFER
TransferID
~~FromAccountID~~
~~ToAccountID~~
Amount
Date

APPLICATION
AplicationId
~~LoanID~~
~~CustomerID~~
Status

BRANCH
BranchID
BranchName
StreetAddress
City
State
Zip Code
Manager Name
Phone Number

CUSTOMER
CustomerID
~~RepresentativeID~~
Password
FirstName
MiddleName
LastName
StreetAddress
City
State
Zip Code
SocialSecurity Number

ACCOUNT
AccountID
~~CustomerID~~
AccountType
Balance
InterestRate
AccountStatus

TRANSACTION
TransactionID
~~AccountID~~
~~Card ID~~
Transaction Type
Amount
Date

REPRESENTATIVE
RepresentativeID
~~BranchID~~
Name
Email
PhoneNumber

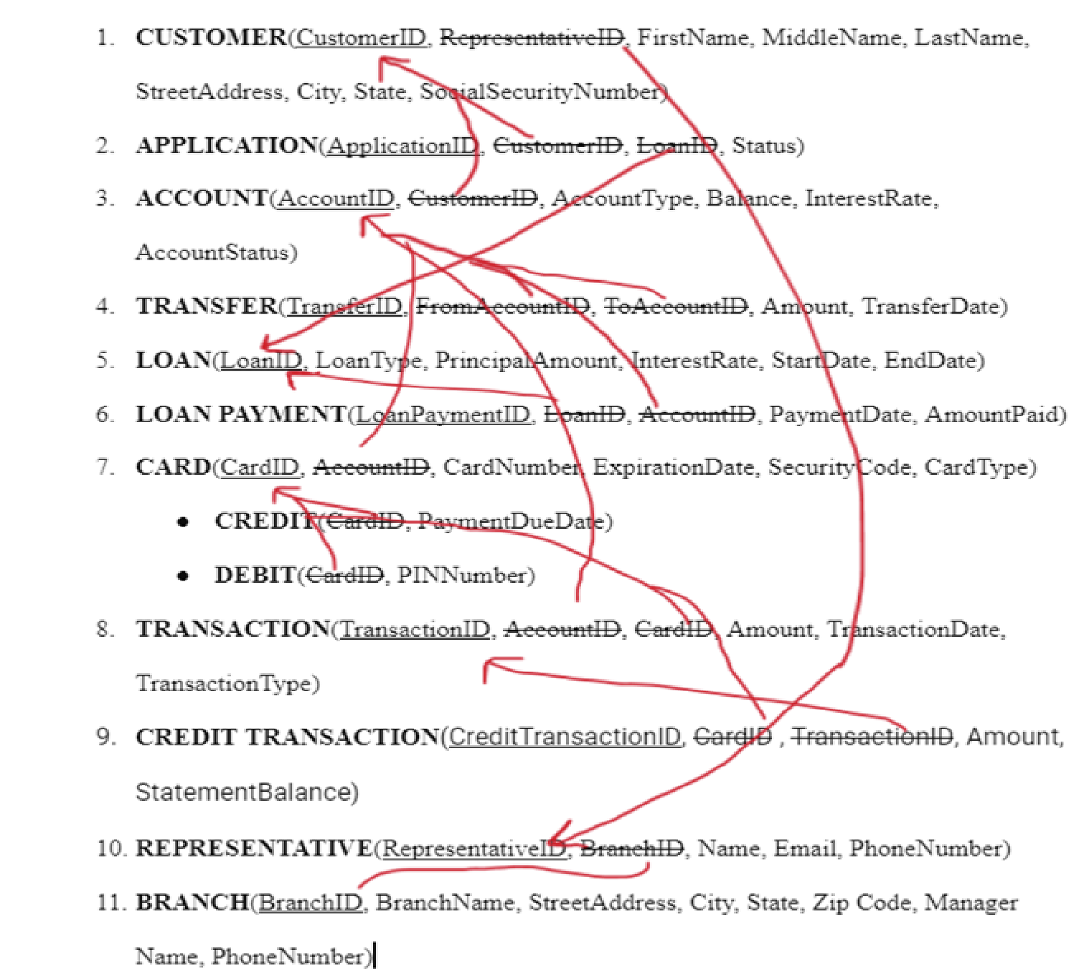
CARD
Card ID
~~AccountID~~
Card Number
Expiration Date
Security Code
Card Type

CREDIT TRANSACTION
CreditTransactionID
~~Card ID~~
~~TransactionID~~
Amount
Statement Balance

CREDIT
Credit Limit
Payment Due Date

DEBIT
PIN Number

List of Normalized Relations

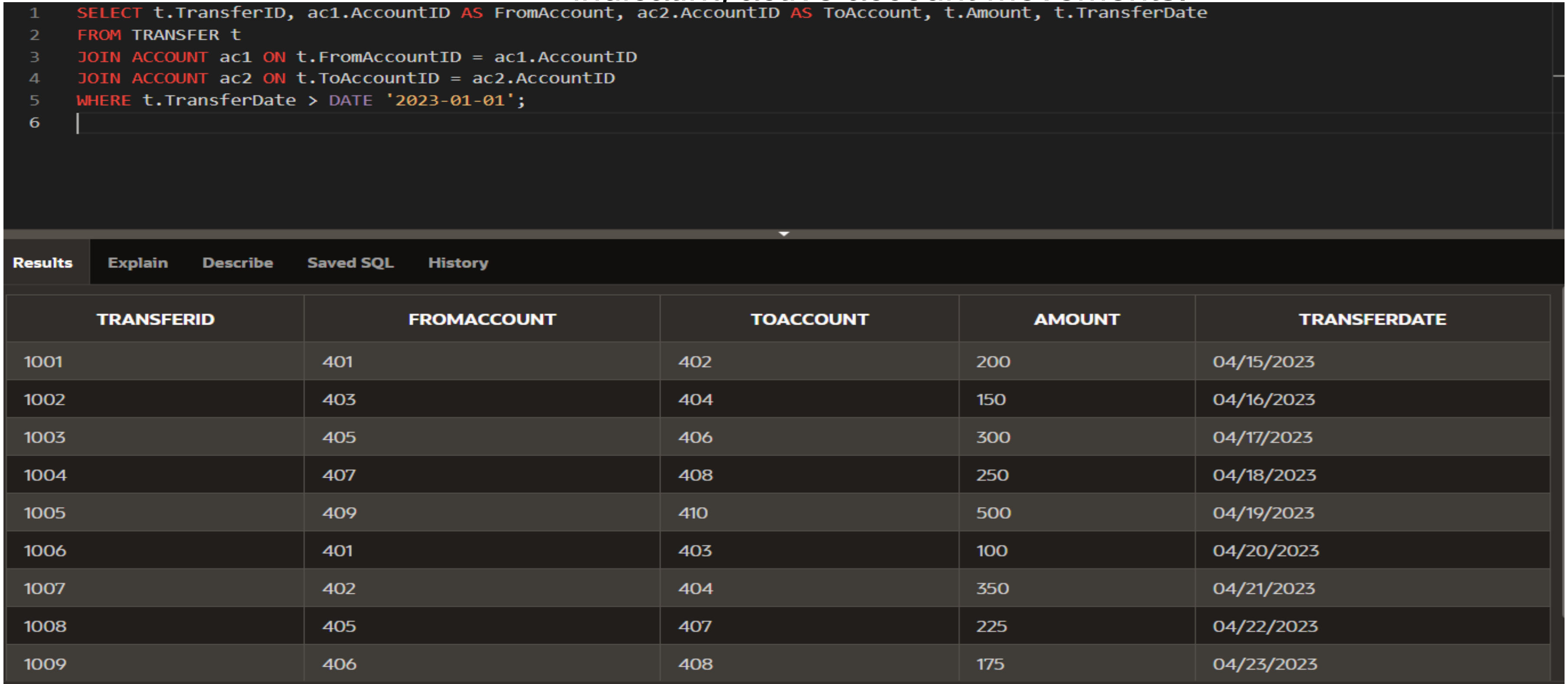


1. **CUSTOMER**(CustomerID, ~~RepresentativeID~~, FirstName, MiddleName, LastName, StreetAddress, City, State, SocialSecurityNumber)
2. **APPLICATION**(ApplicationID, ~~CustomerID~~, ~~LoanID~~, Status)
3. **ACCOUNT**(AccountID, ~~CustomerID~~, AccountType, Balance, InterestRate, AccountStatus)
4. **TRANSFER**(TransferID, ~~FromAccountID~~, ~~ToAccountID~~, Amount, TransferDate)
5. **LOAN**(LoanID, LoanType, PrincipalAmount, InterestRate, StartDate, EndDate)
6. **LOAN PAYMENT**(LoanPaymentID, ~~LoanID~~, ~~AccountID~~, PaymentDate, AmountPaid)
7. **CARD**(CardID, ~~AccountID~~, CardNumber, ExpirationDate, SecurityCode, CardType)
    - **CREDIT**(~~CardID~~, PaymentDueDate)
    - **DEBIT**(~~CardID~~, PINNumber)
8. **TRANSACTION**(TransactionID, ~~AccountID~~, ~~CardID~~, Amount, TransactionDate, TransactionType)
9. **CREDIT TRANSACTION**(CreditTransactionID, ~~CardID~~ , ~~TransactionID~~, Amount, StatementBalance)
10. **REPRESENTATIVE**(RepresentativeID, ~~BranchID~~, Name, Email, PhoneNumber)
11. **BRANCH**(BranchID, BranchName, StreetAddress, City, State, Zip Code, Manager Name, PhoneNumber)

Demo of Data Queries

Inter-account Transfers

This query helps monitor the transfers between accounts, indicating active account movements.

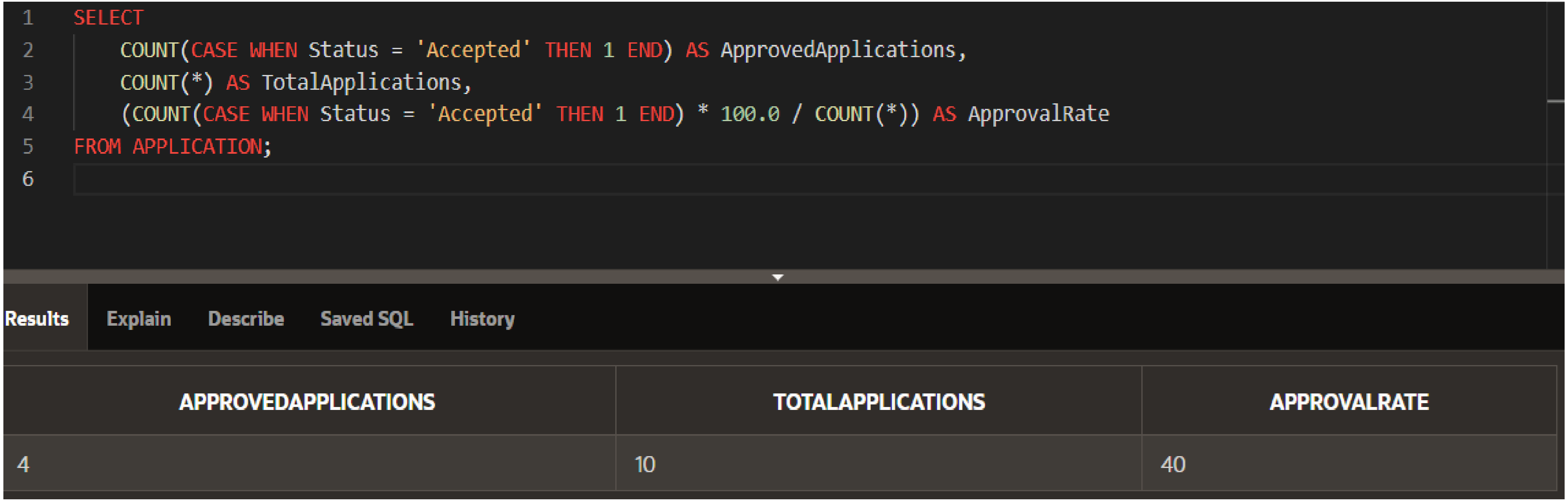


```sql
SELECT t.TransferID, ac1.AccountID AS FromAccount, ac2.AccountID AS ToAccount, t.Amount, t.TransferDate
FROM TRANSFER t
JOIN ACCOUNT ac1 ON t.FromAccountID = ac1.AccountID
JOIN ACCOUNT ac2 ON t.ToAccountID = ac2.AccountID
WHERE t.TransferDate > DATE '2023-01-01';
```

TRANSFERID	FROMACCOUNT	TOACCOUNT	AMOUNT	TRANSFERDATE
1001	401	402	200	04/15/2023
1002	403	404	150	04/16/2023
1003	405	406	300	04/17/2023
1004	407	408	250	04/18/2023
1005	409	410	500	04/19/2023
1006	401	403	100	04/20/2023
1007	402	404	350	04/21/2023
1008	405	407	225	04/22/2023
1009	406	408	175	04/23/2023

Demo of Data Queries

Loan Approval Rate - Calculate the percentage of approved loan applications compared to the total number of applications received.

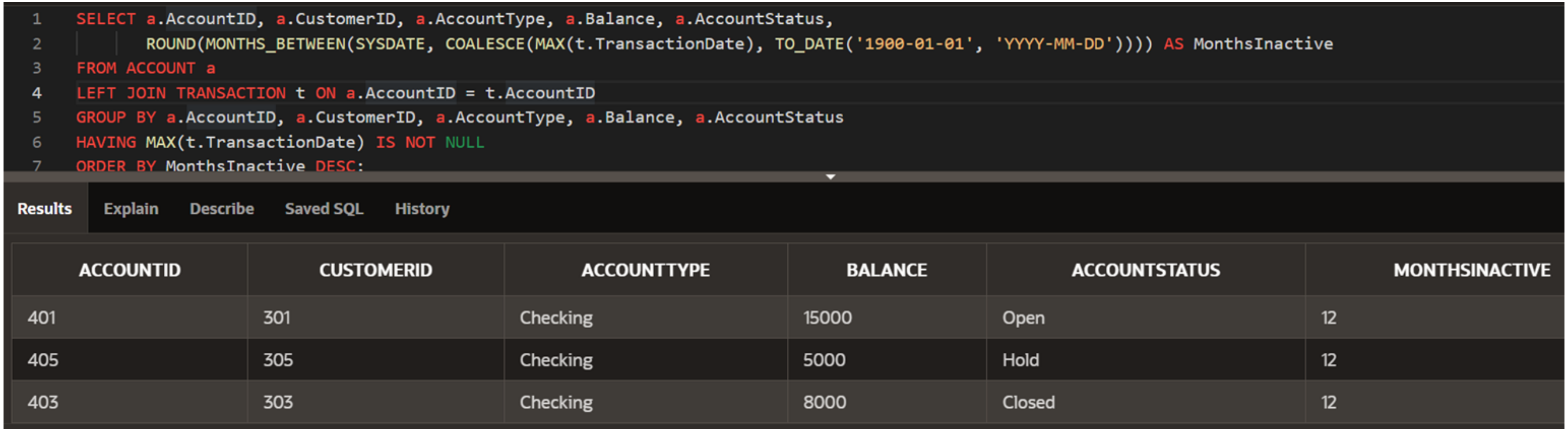


```
SELECT
    COUNT(CASE WHEN Status = 'Accepted' THEN 1 END) AS ApprovedApplications,
    COUNT(*) AS TotalApplications,
    (COUNT(CASE WHEN Status = 'Accepted' THEN 1 END) * 100.0 / COUNT(*)) AS ApprovalRate
FROM APPLICATION;
```

APPROVEDAPPLICATIONS	TOTALAPPLICATIONS	APPROVALRATE
4	10	40

Demo of Data Queries

Inactive Accounts - Identify accounts that have been inactive for a certain length, sorting by months inactive

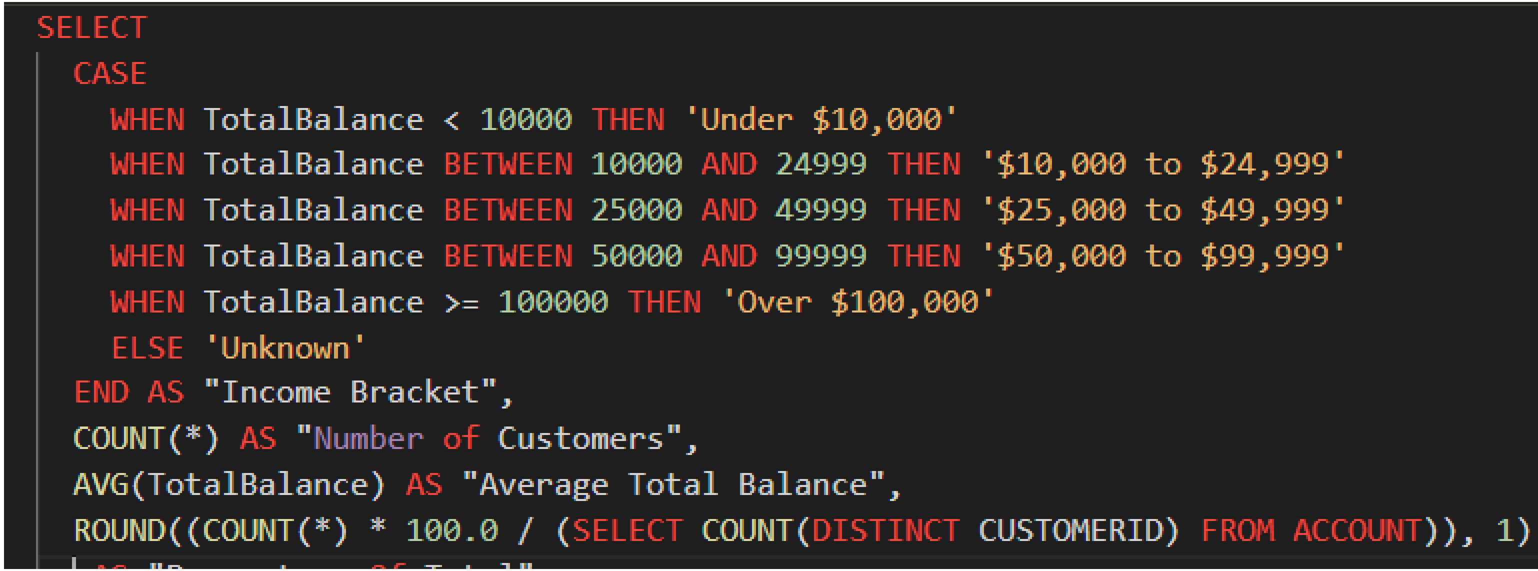


```
SELECT a.AccountID, a.CustomerID, a.AccountType, a.Balance, a.AccountStatus,
       ROUND(MONTHS_BETWEEN(SYSDATE, COALESCE(MAX(t.TransactionDate), TO_DATE('1900-01-01', 'YYYY-MM-DD')))) AS MonthsInactive
FROM ACCOUNT a
LEFT JOIN TRANSACTION t ON a.AccountID = t.AccountID
GROUP BY a.AccountID, a.CustomerID, a.AccountType, a.Balance, a.AccountStatus
HAVING MAX(t.TransactionDate) IS NOT NULL
ORDER BY MonthsInactive DESC;
```

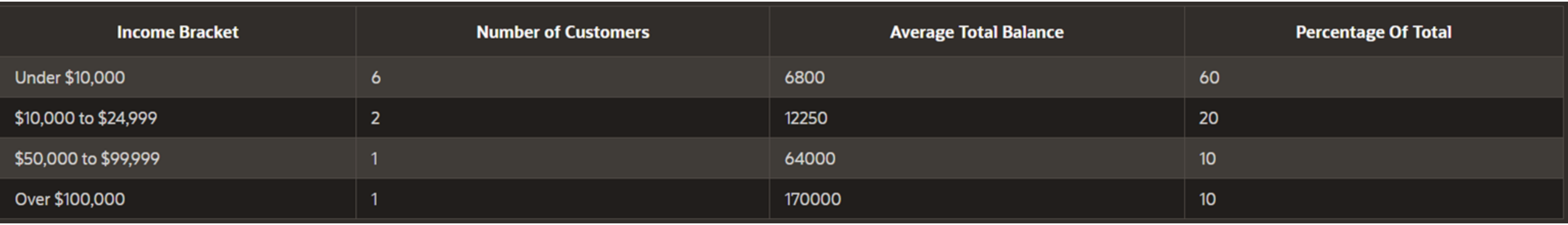
ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	ACCOUNTSTATUS	MONTHSINACTIVE
401	301	Checking	15000	Open	12
405	305	Checking	5000	Hold	12
403	303	Checking	8000	Closed	12

Demo of Data Queries

Classify Customer Accounts - Puts customers into categories to get an idea of the customer’s income levels.



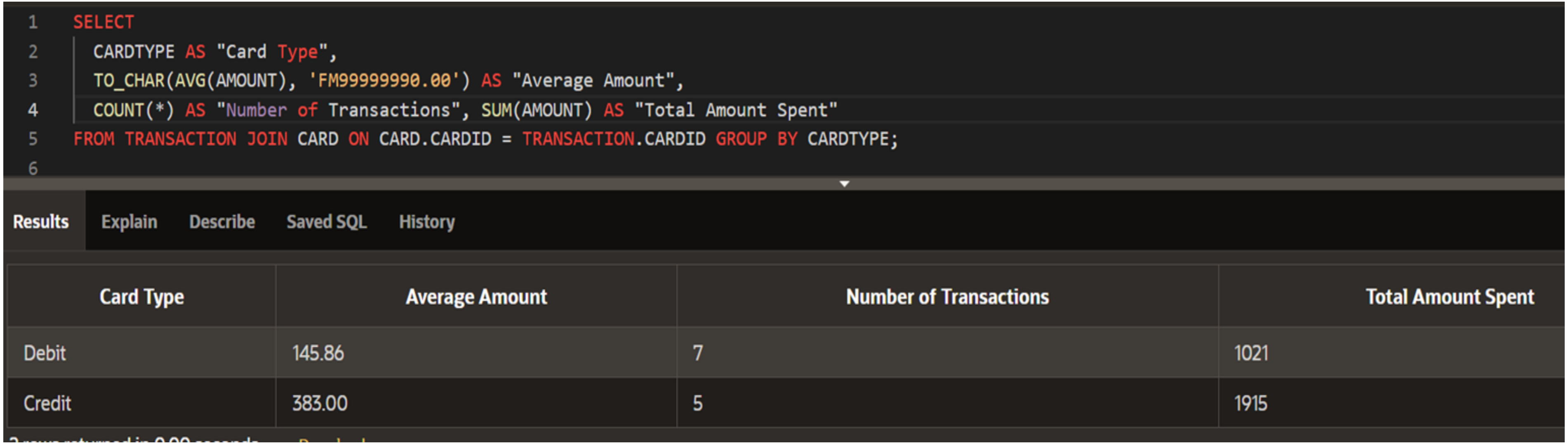
```
SELECT
  CASE
    WHEN TotalBalance < 10000 THEN 'Under $10,000'
    WHEN TotalBalance BETWEEN 10000 AND 24999 THEN '$10,000 to $24,999'
    WHEN TotalBalance BETWEEN 25000 AND 49999 THEN '$25,000 to $49,999'
    WHEN TotalBalance BETWEEN 50000 AND 99999 THEN '$50,000 to $99,999'
    WHEN TotalBalance >= 100000 THEN 'Over $100,000'
    ELSE 'Unknown'
  END AS "Income Bracket",
  COUNT(*) AS "Number of Customers",
  AVG(TotalBalance) AS "Average Total Balance",
  ROUND((COUNT(*) * 100.0 / (SELECT COUNT(DISTINCT CUSTOMERID) FROM ACCOUNT)), 1)
```



Income Bracket	Number of Customers	Average Total Balance	Percentage Of Total
Under $10,000	6	6800	60
$10,000 to $24,999	2	12250	20
$50,000 to $99,999	1	64000	10
Over $100,000	1	170000	10

Demo of Data Queries

Money Spent per Card Type - Identify how certain card types are utilized, and which are used more frequently.



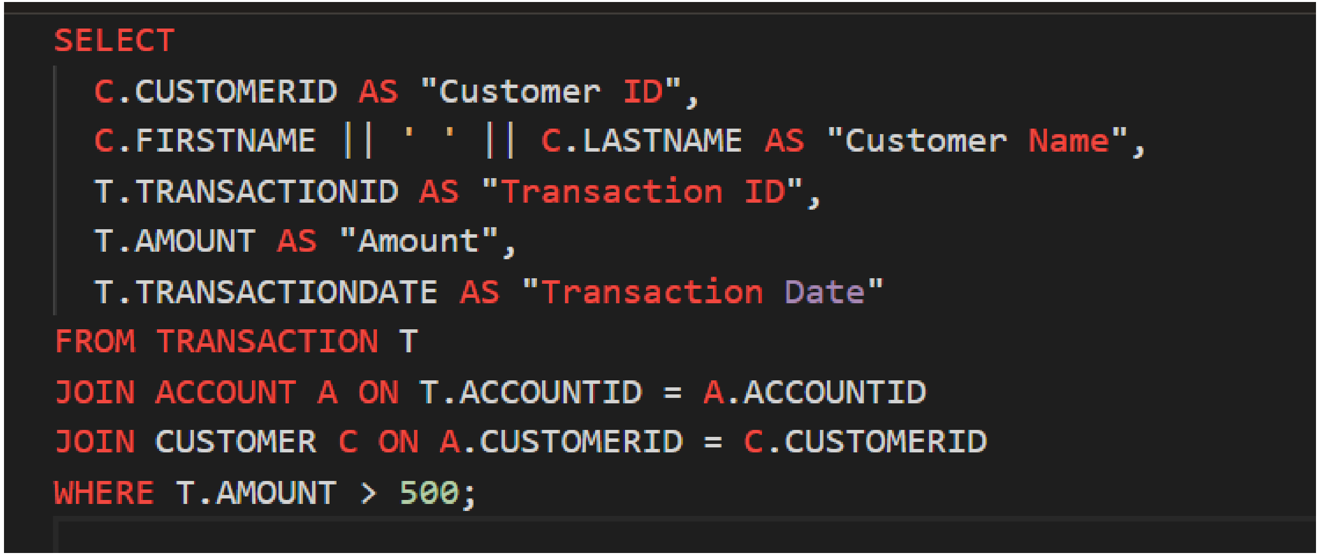
```
SELECT
  CARDTYPE AS "Card Type",
  TO_CHAR(AVG(AMOUNT), 'FM99999990.00') AS "Average Amount",
  COUNT(*) AS "Number of Transactions", SUM(AMOUNT) AS "Total Amount Spent"
FROM TRANSACTION JOIN CARD ON CARD.CARDID = TRANSACTION.CARDID GROUP BY CARDTYPE;
```

Card Type	Average Amount	Number of Transactions	Total Amount Spent
Debit	145.86	7	1021
Credit	383.00	5	1915

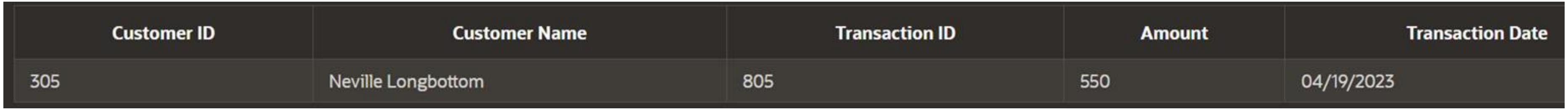
2 rows returned in 0.00 seconds   Download

Demo of Data Queries

Identify transactions over a certain amount of money, which can be used to identify fraudulent transactions.



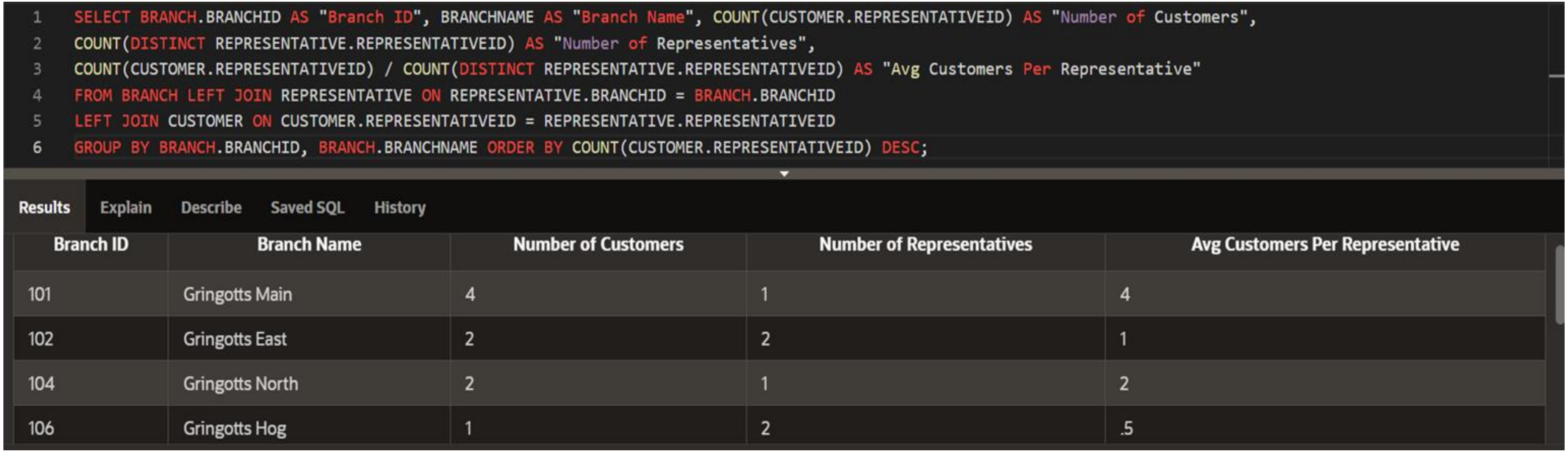
```
SELECT
    C.CUSTOMERID AS "Customer ID",
    C.FIRSTNAME || ' ' || C.LASTNAME AS "Customer Name",
    T.TRANSACTIONID AS "Transaction ID",
    T.AMOUNT AS "Amount",
    T.TRANSACTIONDATE AS "Transaction Date"
FROM TRANSACTION T
JOIN ACCOUNT A ON T.ACCOUNTID = A.ACCOUNTID
JOIN CUSTOMER C ON A.CUSTOMERID = C.CUSTOMERID
WHERE T.AMOUNT > 500;
```



Customer ID	Customer Name	Transaction ID	Amount	Transaction Date
305	Neville Longbottom	805	550	04/19/2023

Demo of Data Queries

Identify the number of customers per representative, which might indicate which branches might need more representatives.



```sql
SELECT BRANCH.BRANCHID AS "Branch ID", BRANCHNAME AS "Branch Name", COUNT(CUSTOMER.REPRESENTATIVEID) AS "Number of Customers",
COUNT(DISTINCT REPRESENTATIVE.REPRESENTATIVEID) AS "Number of Representatives",
COUNT(CUSTOMER.REPRESENTATIVEID) / COUNT(DISTINCT REPRESENTATIVE.REPRESENTATIVEID) AS "Avg Customers Per Representative"
FROM BRANCH LEFT JOIN REPRESENTATIVE ON REPRESENTATIVE.BRANCHID = BRANCH.BRANCHID
LEFT JOIN CUSTOMER ON CUSTOMER.REPRESENTATIVEID = REPRESENTATIVE.REPRESENTATIVEID
GROUP BY BRANCH.BRANCHID, BRANCH.BRANCHNAME ORDER BY COUNT(CUSTOMER.REPRESENTATIVEID) DESC;
```

Branch ID	Branch Name	Number of Customers	Number of Representatives	Avg Customers Per Representative
101	Gringotts Main	4	1	4
102	Gringotts East	2	2	1
104	Gringotts North	2	1	2
106	Gringotts Hog	1	2	.5

Thank You!