Department of Computer Science & Engineering

University of Dhaka

Project Report

Course Code: CSE-2211

Course Tittle: Database Management System-I Lab

Project Tittle:

"Banking Management System"

Submitted To

Dr. Md. Mustafizur Rahman	Dr. Muhammad Ibrahim
Professor	Assistant Professor
Department of CSE	Department of CSE
University of Dhaka	University of Dhaka

Submitted BY

Asef Sami Chowdhury
Roll: 53
Year-2, Semester-2
Department of Computer Science & Engineering
University of Dhaka

A Banking Management System

Project Area Major: Finance.

Project Area Minor: Banking Management.

Brief Description of the Database: The Banking Management System database is designed to efficiently manage and organize critical financial information within a banking institution. The primary objectives of this database include the effective management of customer accounts, tracking financial transactions, and overseeing loan-related activities. The system aims to streamline banking operations, enhance customer service, and ensure accurate and secure record-keeping.

The database encompasses various entities such as customers, accounts, transactions, and loans. It facilitates the recording and retrieval of customer details, including personal information and account-related data. Additionally, the system tracks financial transactions in real-time, maintaining a comprehensive record of deposits, withdrawals, and transfers. The inclusion of loan management functionalities allows the system to handle loan applications, approvals, and repayments efficiently.

Overall, the Banking Management System database is tailored to meet the diverse needs of a modern banking institution, promoting data integrity, security, and operational efficiency.

Detailed Description of the Database: The banking database serves as a critical component in managing financial data, customer information, and operational processes within our banking system. This comprehensive database plays a pivotal role in ensuring the efficiency, security, and regulatory compliance of our financial services.

Database Architecture: Our banking database operates on a robust relational database management system (RDBMS) architecture. It comprises well-defined tables, relationships, and specialized components designed to handle the complexities of banking data.

Data Model: The data model is crafted to represent key banking entities, including Customers, Accounts, Transactions, and Employees. Normalization processes have been applied to ensure optimal data organization and integrity.

Entities and Attributes:

- **Customers:** Unique identifiers, personal details, account information.
- **Accounts:** Account type, balance, transaction history.
- **Transactions:** Transaction type, amount, timestamp.
- **Employees:** Employee details, roles, access levels.

Security Measures: The database employs robust security measures, including role-based access controls, encryption of sensitive data, and strict authentication protocols. Regular security audits and monitoring are conducted to identify and address potential vulnerabilities.

Transaction Processing: The database ensures the integrity and consistency of financial data through a transaction processing system that adheres to ACID principles, guaranteeing Atomicity, Consistency, Isolation, and Durability.

Concurrency Control: Mechanisms are in place to manage concurrent transactions, preventing conflicts and maintaining data accuracy during simultaneous operations.

Data Integrity and Validation: Constraints, triggers, and validation rules are implemented to maintain data integrity. Regular checks are conducted to ensure the accuracy and reliability of stored information.

Backup and Recovery: Regular backups are performed to mitigate the risk of data loss or corruption. Robust recovery procedures are in place to swiftly restore the database in the event of system failures.

Scalability: The database is designed to scale efficiently, accommodating an increasing volume of data and user transactions. Scalability features are implemented to support the growing demands of our banking operations.

Performance Optimization: Indexing, query optimization, and caching mechanisms are employed to enhance database performance, ensuring swift and efficient data retrieval.

Regulatory Compliance: The database adheres to all relevant banking regulations and data protection laws. Auditing features are implemented to facilitate compliance monitoring and reporting.

User Interfaces and Integration: User interfaces and applications interact seamlessly with the database, providing a user-friendly experience. Integration points with other banking systems and services are carefully managed for a cohesive banking infrastructure.

Maintenance and Monitoring: Routine maintenance tasks and continuous monitoring processes are in place to uphold the performance, security, and regulatory compliance of the database.

Future Development: Ongoing efforts focus on enhancements, upgrades, and future developments to ensure the database remains aligned with the evolving needs of our banking services.

The Banking Management System comprises several interconnected tables to efficiently manage and organize crucial financial information. Below is an in-depth description of each table, highlighting their respective scopes:

1. **Customers Table:**
    - **Attributes:** CustomerID, FirstName, LastName, Address, Email, Phone.
    - **Scope:** Records customer details, including personal information and contact details. It serves as the primary entity for establishing relationships with other tables.

2. **Accounts Table:**
    - **Attributes:** AccountID, CustomerID (Foreign Key), AccountType, Balance, OpenDate, Status.
    - **Scope:** Manages information related to bank accounts, including the account type, balance, and status. The CustomerID establishes a relationship with the Customers table, linking each account to a specific customer.

3. **Transactions Table:**
    - **Attributes:** TransactionID, AccountID (Foreign Key), TransactionType, Amount, TransactionDate, Description.
    - **Scope:** Tracks financial transactions, including deposits, withdrawals, and transfers. The AccountID links transactions to specific accounts, ensuring a comprehensive record of financial activities.

4. **Branches Table:**
    - **Attributes:** BranchID, BranchName, Location, ManagerID.
    - **Scope:** Manages information about different bank branches, including their names, locations, and the manager overseeing each branch. The ManagerID establishes a link to the Employees table.

5. **Employees Table:**
    - **Attributes:** EmployeeID, FirstName, LastName, Position, BranchID.
    - **Scope:** Stores details about bank employees, including their positions and the branch they are associated with. The BranchID establishes a link to the Branches table.
6. **Loans Table:**
    - **Attributes:** LoanID, CustomerID (Foreign Key), LoanType, Amount, InterestRate, Term, ApprovalDate, Status.
    - **Scope:** Manages information related to loans, including types, amounts, interest rates, and approval status. The CustomerID establishes a relationship with the Customers table.

7. **CreditCards Table:**
    - **Attributes:** CardID, CustomerID (Foreign Key), CardType, CreditLimit, ExpiryDate, OutstandingBalance.
    - **Scope:** Stores information about credit cards issued to customers, including credit limits and outstanding balances. The CustomerID establishes a relationship with the Customers table.

8. **CreditCardTransactions Table:**
    - **Attributes:** CreditCardTransactionID, CardID (Foreign Key), TransactionType, Amount, TransactionDate, Merchant, Description.
    - **Scope:** Records transactions specific to credit cards, including details about merchants and transaction types. The CardID links transactions to specific credit cards.

9. **Payments Table:**
    - **Attributes:** PaymentID, LoanID (Foreign Key), Amount, PaymentDate, Method.
    - **Scope:** Manages information about loan payments, including amounts, payment dates, and payment methods. The LoanID establishes a relationship with the Loans table.

This interconnected structure enables the Banking Management System to effectively manage customer accounts, transactions, loans, and credit card-related activities, providing a comprehensive solution for the banking institution's diverse needs.

Expected query outputs from the project:

This section is written as a story:

Once upon a time in the bustling world of finance, our hero, a leading bank, dreamt of a future where data wasn't just stored but used to its fullest potential. With a commitment to efficiency, customer satisfaction, and informed decision-making, the bank embarked on a journey to optimize its database system. Driven by the vision of delivering unparalleled financial services, the bank outlined key objectives.

They aimed to aggregate crucial customer information, understand transaction patterns, and assess the performance of various financial products. Additionally, the bank was keen on gaining insights into employee distribution across branches and ensuring seamless integration of credit-related data.

To achieve these ambitious goals, the bank recognized the power of structured querying and data analysis. They identified ten pivotal queries, like secret codes for their computer system, that would help them uncover hidden treasures within their data. Once upon a time in the busy world of banking, there was a bank that dreamt of using its data not just as numbers on a computer but as a powerful tool for making things better. This bank was on a mission to be super-efficient, make customers super happy, and make super smart decisions.

They wanted to gather important info about their customers, understand how money moved around, and check how well their different money services were doing. Plus, they were curious about where their employees were working and wanted to make sure all the credit-related stuff was nicely connected. To make these dreams come true, the bank knew they needed to ask their data some smart questions. They came up with ten special questions, kind of like secret codes for their computer system.

These questions would help them figure out everything from how much money people had to how often they were making transactions. And here's the cool part – they found a way to connect all these pieces of information like a puzzle. It was like magic! They used something called outer joins to combine different sets of data, giving them a big picture of their customers, loans, and credit card adventures. And so, with these ten special questions and the magic of outer joins, our bank set off on a journey to make their financial world even more awesome.

They believed that understanding their data better would help them make decisions that would make their customers smile and their bank shine in the world of finance. And they all lived happily ever after, making banking better one query at a time.

List of Tables with schema: Here is list of tables for bank management system with schema:

1. **Customers Table**
   - Customers = (CustomerID, FirstName, LastName, Address, Email, Phone)

2. **Accounts Table**
   - Accounts = (AccountID, CustomerID, AccountType, Balanc, OpenDate, Status)

3. **Transactions Table**
   - Transactions = (TransactionID, AccountID, TransactionType, Amount, TransactionDate, Description)

4. **Branches Table**
   - Branches = (BranchID, BranchName, Location, ManagerID)

5. **Employees Table**
   - Employees = (EmployeeID, FirstName, LastName, Position, BranchID)

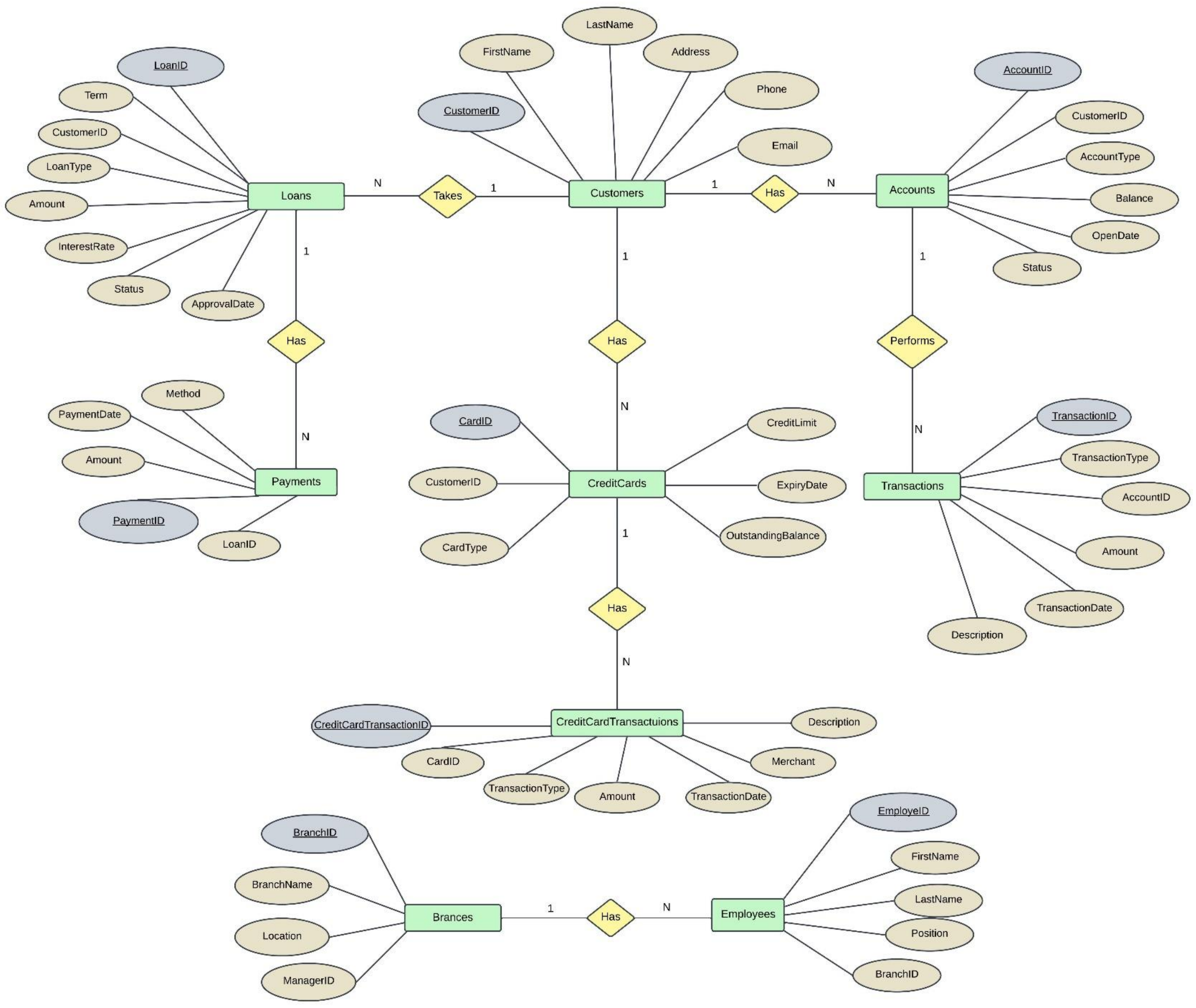
6. **Loans Table**
   - Loans = (LoanID, CustomerID, LoanType, Amount, InterestRate, Term, ApprovalDate, Status)

7. **CreditCards Table**
   - CreditCards = (CardID, CustomerID, CardType, CreditLimit, ExpiryDate, OutstandingBalance)

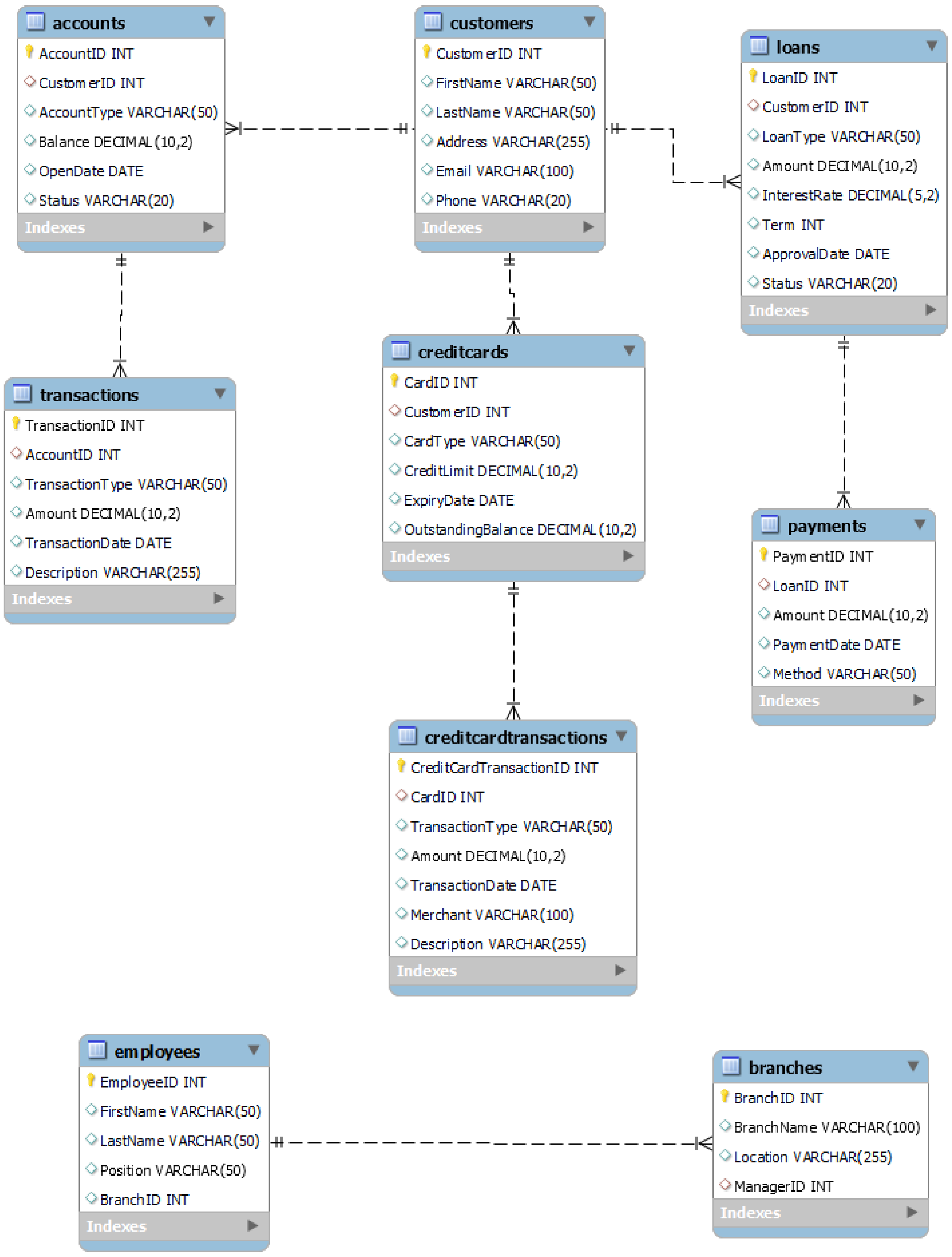
8. **CreditCardTransactions Table**
   - CreditCardTransactions = (CreditCardTransactionID, CardID, TransactionType, Amount,TransactionDate, Merchant,Description)

9. **Payments Table**
   - Payments = (PaymentID, LoanID, Amount, PaymentDate, Method)

ER Diagram:



LastName
FirstName
Address
Phone
Email
CustomerID
Customers
LoanID
Term
CustomerID
LoanType
Amount
InterestRate
Status
ApprovalDate
Loans
Takes
N
1
Has
1
N
Accounts
AccountID
CustomerID
AccountType
Balance
OpenDate
Status
1
Has
N
Payments
Method
PaymentDate
Amount
PaymentID
LoanID
1
Has
N
CreditCards
CardID
CustomerID
CardType
CreditLimit
ExpiryDate
OutstandingBalance
1
Performs
N
Transactions
TransactionID
TransactionType
AccountID
Amount
TransactionDate
Description
1
Has
N
CreditCardTransactuions
CreditCardTransactionID
CardID
TransactionType
Amount
TransactionDate
Merchant
Description
Brances
BranchID
BranchName
Location
ManagerID
1
Has
N
Employees
EmployeID
FirstName
LastName
Position
BranchID


accounts
AccountID INT
CustomerID INT
AccountType VARCHAR(50)
Balance DECIMAL(10,2)
OpenDate DATE
Status VARCHAR(20)
Indexes
customers
CustomerID INT
FirstName VARCHAR(50)
LastName VARCHAR(50)
Address VARCHAR(255)
Email VARCHAR(100)
Phone VARCHAR(20)
Indexes
loans
LoanID INT
CustomerID INT
LoanType VARCHAR(50)
Amount DECIMAL(10,2)
InterestRate DECIMAL(5,2)
Term INT
ApprovalDate DATE
Status VARCHAR(20)
Indexes
transactions
TransactionID INT
AccountID INT
TransactionType VARCHAR(50)
Amount DECIMAL(10,2)
TransactionDate DATE
Description VARCHAR(255)
Indexes
creditcards
CardID INT
CustomerID INT
CardType VARCHAR(50)
CreditLimit DECIMAL(10,2)
ExpiryDate DATE
OutstandingBalance DECIMAL(10,2)
Indexes
payments
PaymentID INT
LoanID INT
Amount DECIMAL(10,2)
PaymentDate DATE
Method VARCHAR(50)
Indexes
creditcardtransactions
CreditCardTransactionID INT
CardID INT
TransactionType VARCHAR(50)
Amount DECIMAL(10,2)
TransactionDate DATE
Merchant VARCHAR(100)
Description VARCHAR(255)
Indexes
employees
EmployeeID INT
FirstName VARCHAR(50)
LastName VARCHAR(50)
Position VARCHAR(50)
BranchID INT
Indexes
branches
BranchID INT
BranchName VARCHAR(100)
Location VARCHAR(255)
ManagerID INT
Indexes

List of Functional Dependencies: Functional dependencies describe the relationships between attributes in a table. Below are the functional dependencies for each table in my database:

1. **Customers Table:**
   - CustomerID → FirstName, LastName, Address, Email, Phone
     - The customer ID uniquely determines the first name, last name, address, email, and phone number of a customer.

2. **Accounts Table:**
   - AccountID → CustomerID, AccountType, Balance, OpenDate, Status
     - The account ID uniquely determines the associated customer ID, account type, balance, open date, and status.

   - CustomerID → AccountID
     - Each customer can have multiple accounts, but each account is associated with a single customer.

3. **Transactions Table:**
   - TransactionID → AccountID, TransactionType, Amount, TransactionDate, Description
     - The transaction ID uniquely determines the associated account ID, transaction type, amount, transaction date, and description.

   - AccountID → TransactionID
     - Each account can have multiple transactions, but each transaction is associated with a single account.

4. **Branches Table:**
   - BranchID → BranchName, Location, ManagerID
     - The branch ID uniquely determines the branch name, location, and manager ID.

   - ManagerID → BranchID
     - Each manager is associated with a single branch, but each branch has a unique manager.

5. **Employees Table:**
   - EmployeeID → FirstName, LastName, Position, BranchID
     - The employee ID uniquely determines the first name, last name, position, and associated branch ID.

- BranchID → ManagerID
    - Each branch is managed by a single employee, but each employee can be associated with multiple branches.

6. **Loans Table:**
    - LoanID → CustomerID, LoanType, Amount, InterestRate, Term, ApprovalDate, Status
        - The loan ID uniquely determines the associated customer ID, loan type, amount, interest rate, term, approval date, and status.
    - CustomerID → LoanID
        - Each customer can have multiple loans, but each loan is associated with a single customer.

7. **CreditCards Table:**
    - CardID → CustomerID, CardType, CreditLimit, ExpiryDate, OutstandingBalance
        - The card ID uniquely determines the associated customer ID, card type, credit limit, expiry date, and outstanding balance.
    - CustomerID → CardID
        - Each customer can have multiple credit cards, but each credit card is associated with a single customer.

8. **CreditCardTransactions Table:**
    - CreditCardTransactionID → CardID, TransactionType, Amount, TransactionDate, Merchant, Description
        - The credit card transaction ID uniquely determines the associated card ID, transaction type, amount, transaction date, merchant, and description.
    - CardID → CreditCardTransactionID
        - Each credit card can have multiple transactions, but each transaction is associated with a single credit card.

9. **Payments Table:**
    - PaymentID → LoanID, Amount, PaymentDate, Method
        - The payment ID uniquely determines the associated loan ID, payment amount, payment date, and method.
    - LoanID → PaymentID
        - Each loan can have multiple payments, but each payment is associated with a single loan.

These functional dependencies capture the relationships and dependencies within each table, ensuring data integrity and consistency in the database.

Tables Schema: Here is full table schema in grid format:

Customers

S/N	Attribute	Data Type	Constraint	Comments
1	CustomerID	INT	PK	Primary Key, Unique identifier for customer
2	FirstName	VARCHAR	-	First name of customer
3	LastName	VARCHAR	-	Last Name of customer
4	Address	VARCHAR	-	Address of customer
5	Email	VARCHAR	-	Email address of customer
6	Phone	VARCHAR	-	Phone number of customers
The Customers table stores information about banking customers, including their name, address, email, and phone number.				

Purpose of the Table: The Customers table serves as a repository for essential information about banking clients, encompassing their personal details such as names, addresses, email addresses, and contact numbers. This data is fundamental for establishing and maintaining customer relationships, enabling the bank to communicate effectively and provide personalized services.

Transactions

S/N	Attribute	Data Type	Constraint	Comments
1	TransactionID	INT	PK	Primary Key
2	AccountID	INT	FK	Foreign Key referencing Accounts
3	TransactionType	VARCHAR	-	Type of transaction (e.g., deposit, withdrawal)
4	Amount	DECIMAL	-	Transaction amount
5	TransactionDate	DATE	-	Date and time of the transaction
6	Description	VARCHAR	-	Description or notes about the transaction
The Transactions table stores details of financial transactions related to accounts.				

Purpose of the Table: The Transactions table functions as a ledger for recording intricate details of financial transactions associated with bank accounts. It encompasses transaction types, amounts, dates, and supplementary descriptions. This repository of transactional data aids in monitoring and analyzing financial movements, facilitating accurate and transparent record-keeping.

Branches

S/N	Attribute	Data Type	Constraint	Comments
1	BranchID	INT	PK	Primary Key
2	BranchName	VARCHAR	-	Name of the branch
3	Location	VARCHAR	-	Location of the branch
4	ManagerID	INT	FK	Foreign Key referencing Employees
The Branches table stores information about different branches of the bank, including their name, location, and manager.				

Purpose of the Table: The Branches table serves as a repository for vital information about different branches of the bank, including names, locations, and the respective manager's identifiers. This information is crucial for organizational management, facilitating efficient oversight and coordination of activities across various branches.

Employees

S/N	Attribute	Data Type	Constraint	Comments
1	EmployeeID	INT	PK	Primary Key
2	FirstName	VARCHAR	-	Employee's first name
3	LastName	VARCHAR	-	Employee's last name
4	Position	VARCHAR	-	Employee's position in the bank
5	BranchID	INT	-	Foreign Key referencing Branches
The Employees table stores information about bank employees, including their name, position, and the branch they work in.				

Purpose of the Table: The Employees table is dedicated to storing pertinent details about bank personnel, encompassing their names, positions, and the branch in which they operate. This information is pivotal for human resource management and organizational structuring, ensuring a clear understanding of the workforce's composition and roles.

Loans

S/N	Attribute	Data Type	Constraint	Comments
1	LoanID	INT	PK	Primary Key
2	CustomerID	INT	FK	Foreign Key referencing Customers

3	LoanType	VARCHAR	-	Type of loan (e.g., personal, home, auto)
4	Amount	DECIMAL	-	Loan amount
5	InterestRate	DECIMAL	-	Interest rate on the loan
6	Term	INT	-	Term of the loan (in months)
7	ApprovalDate	DATE	-	Date when the loan was approved
8	Status	VARCHAR	-	Status of the loan (e.g., approved, rejected)
The Loans table stores information about loans provided by the bank, including type, amount, and approval status.				

Purpose of the Table: The Loans table functions as a repository for comprehensive information regarding loans provided by the bank. This includes details such as loan types, amounts, interest rates, terms, approval dates, and statuses. The table is linked to the Customers table through the CustomerID foreign key, establishing a connection between loan records and customer profiles.

CreditCards

S/N	Attribute	Data Type	Constraint	Comments
1	CardID	INT	PK	Primary Key
2	CustomerID	INT	FK	Foreign Key referencing Customers
3	CardType	VARCHAR	-	Type of credit card (e.g., gold, platinum)
4	CreditLimit	DECIMAL	-	Credit limit on the card
5	ExpiryDate	DATE	-	Expiry date of the credit card
6	OutstandingBalance	DECIMAL	-	Outstanding balance on the credit card
The CreditCards table stores information about credit cards issued by the bank, including type, credit limit, and outstanding balance.				

Purpose of the Table: The CreditCards table captures essential information about credit cards issued by the bank, including types, credit limits, expiry dates, and outstanding balances. This repository is vital for managing credit card portfolios and understanding the financial exposure associated with each cardholder.

CreditCardTransactions

S/N	Attribute	Data Type	Constraint	Comments
1	CreditCardTransactionID	INT	PK	Primary Key
2	CardID	INT	FK	Foreign Key referencing CreditCards
3	TransactionType	VARCHAR	-	Type of credit card transaction (e.g., purchase, payment)
4	Amount	DECIMAL	-	Transaction amount
5	TransactionDate	DATE	-	Date and time of the credit card transaction
6	Merchant	VARCHAR	-	Merchant involved in the transaction
7	Description	VARCHAR	-	Description or notes about the transaction
The CreditCardTransactions table stores details of transactions related to credit cards.				

Purpose of the Table: The CreditCardTransactions table records intricate details of transactions related to credit cards. This includes transaction types, amounts, dates, merchants involved, and additional descriptions. The data is instrumental for analyzing cardholder spending patterns and facilitating fraud detection.

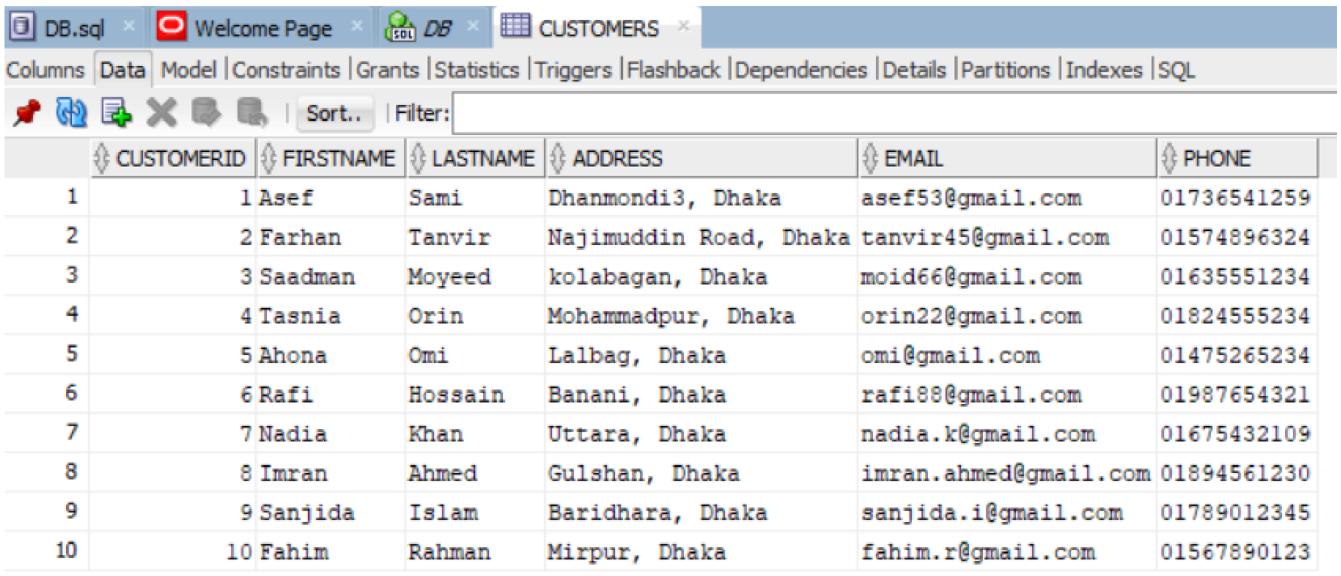
Payments

S/N	Attribute	Data Type	Constraint	Comments
1	PaymentID	INT	PK	Primary Key
2	LoanID	INT	FK	Foreign Key referencing Loans
3	Amount	DECIMAL	-	Payment amount
4	PaymentDate	DATE	-	Date and time of the payment
5	Method	VARCHAR	-	Payment method (e.g., cash, check)
The Payments table stores information about payments made towards loans, including the amount, date, and payment method.				

Purpose of the Table: The Payments table is dedicated to storing information about payments made towards loans. This encompasses payment amounts, dates, and methods. The data is crucial for tracking loan repayments and managing the overall financial health of the lending portfolio.

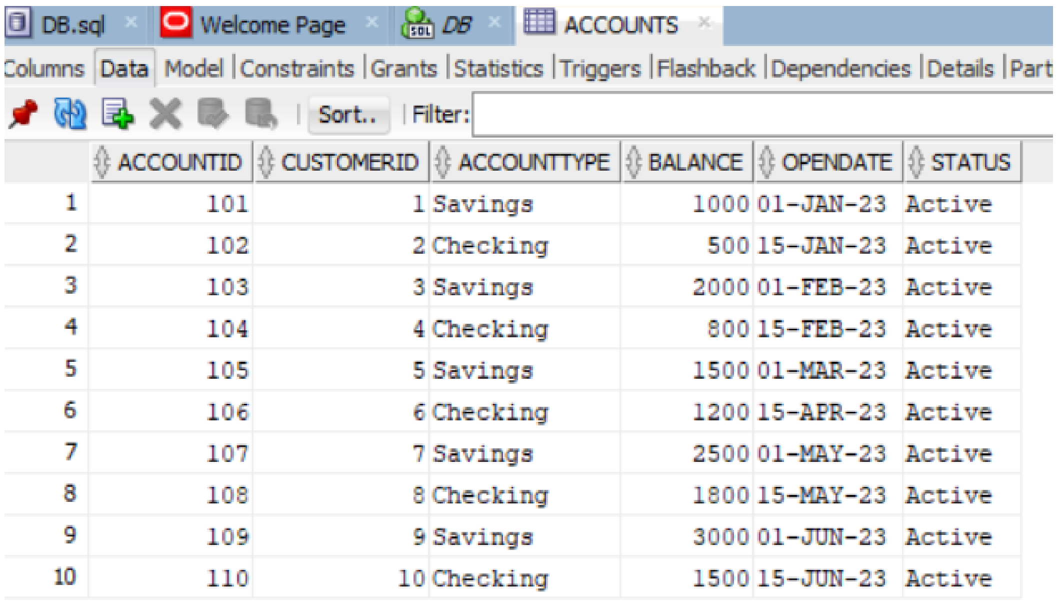
Example Data (Appendix): Some example plate data for different tables.

Customers Table:



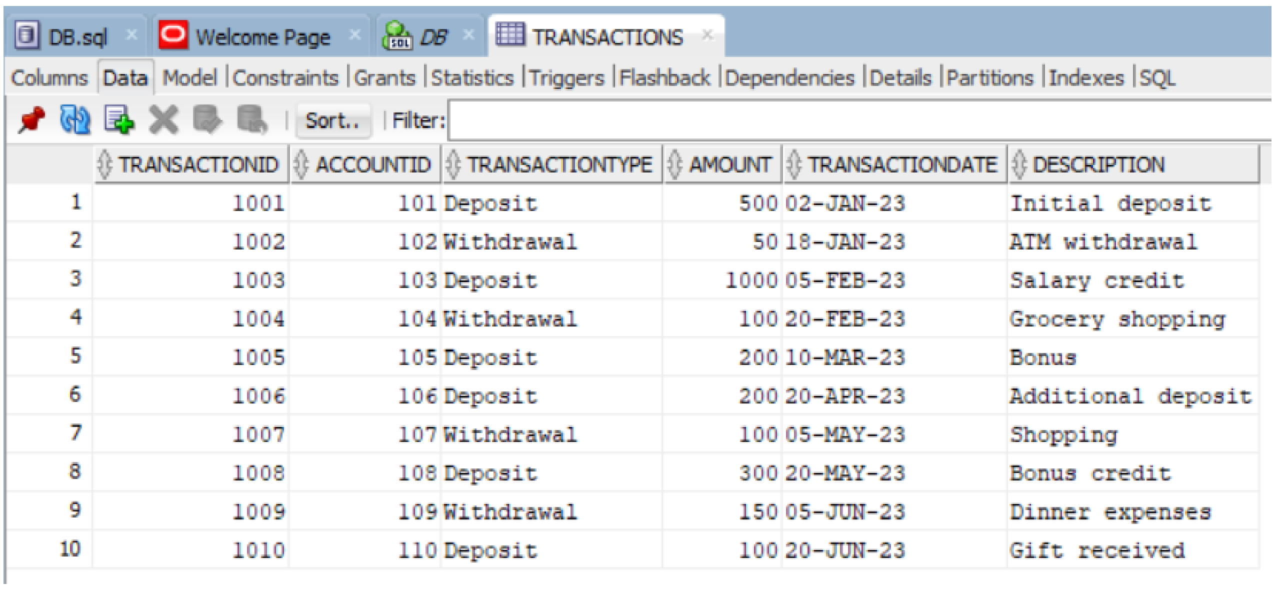
	CUSTOMERID	FIRSTNAME	LASTNAME	ADDRESS	EMAIL	PHONE
1	1	Asef	Sami	Dhanmondi3, Dhaka	asef53@gmail.com	01736541259
2	2	Farhan	Tanvir	Najimuddin Road, Dhaka	tanvir45@gmail.com	01574896324
3	3	Saadman	Moyeed	kolabagan, Dhaka	moid66@gmail.com	01635551234
4	4	Tasnia	Orin	Mohammadpur, Dhaka	orin22@gmail.com	01824555234
5	5	Ahona	Omi	Lalbag, Dhaka	omi@gmail.com	01475265234
6	6	Rafi	Hossain	Banani, Dhaka	rafi88@gmail.com	01987654321
7	7	Nadia	Khan	Uttara, Dhaka	nadia.k@gmail.com	01675432109
8	8	Imran	Ahmed	Gulshan, Dhaka	imran.ahmed@gmail.com	01894561230
9	9	Sanjida	Islam	Baridhara, Dhaka	sanjida.i@gmail.com	01789012345
10	10	Fahim	Rahman	Mirpur, Dhaka	fahim.r@gmail.com	01567890123

Accounts Table:



	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	OPENDATE	STATUS
1	101	1	Savings	1000	01-JAN-23	Active
2	102	2	Checking	500	15-JAN-23	Active
3	103	3	Savings	2000	01-FEB-23	Active
4	104	4	Checking	800	15-FEB-23	Active
5	105	5	Savings	1500	01-MAR-23	Active
6	106	6	Checking	1200	15-APR-23	Active
7	107	7	Savings	2500	01-MAY-23	Active
8	108	8	Checking	1800	15-MAY-23	Active
9	109	9	Savings	3000	01-JUN-23	Active
10	110	10	Checking	1500	15-JUN-23	Active

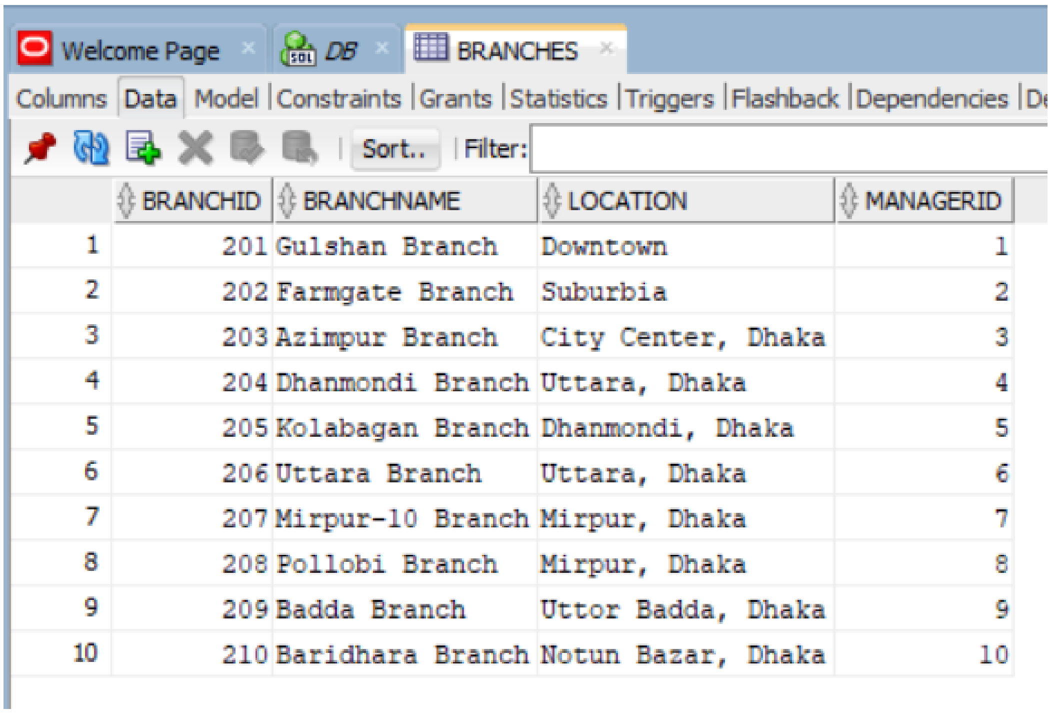
Transactions Table:



DB.sql
Welcome Page
DB
TRANSACTIONS
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

	TRANSACTIONID	ACCOUNTID	TRANSACTIONTYPE	AMOUNT	TRANSACTIONDATE	DESCRIPTION
1	1001	101	Deposit	500	02-JAN-23	Initial deposit
2	1002	102	Withdrawal	50	18-JAN-23	ATM withdrawal
3	1003	103	Deposit	1000	05-FEB-23	Salary credit
4	1004	104	Withdrawal	100	20-FEB-23	Grocery shopping
5	1005	105	Deposit	200	10-MAR-23	Bonus
6	1006	106	Deposit	200	20-APR-23	Additional deposit
7	1007	107	Withdrawal	100	05-MAY-23	Shopping
8	1008	108	Deposit	300	20-MAY-23	Bonus credit
9	1009	109	Withdrawal	150	05-JUN-23	Dinner expenses
10	1010	110	Deposit	100	20-JUN-23	Gift received

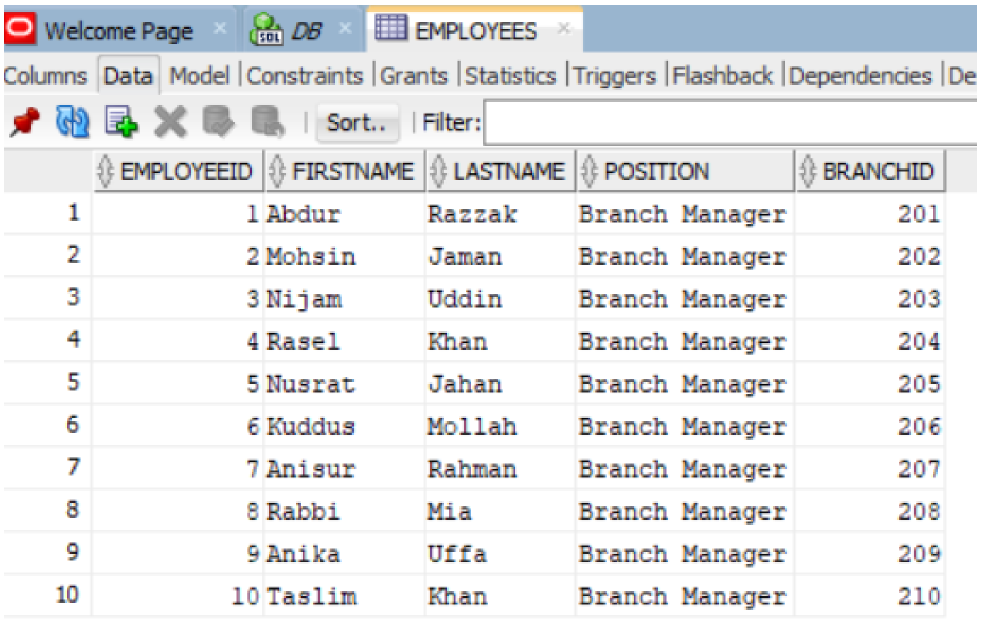
Branches Table:



Welcome Page
DB
BRANCHES
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | De
Sort..
Filter:

	BRANCHID	BRANCHNAME	LOCATION	MANAGERID
1	201	Gulshan Branch	Downtown	1
2	202	Farmgate Branch	Suburbia	2
3	203	Azimpur Branch	City Center, Dhaka	3
4	204	Dhanmondi Branch	Uttara, Dhaka	4
5	205	Kolabagan Branch	Dhanmondi, Dhaka	5
6	206	Uttara Branch	Uttara, Dhaka	6
7	207	Mirpur-10 Branch	Mirpur, Dhaka	7
8	208	Pollobi Branch	Mirpur, Dhaka	8
9	209	Badda Branch	Uttor Badda, Dhaka	9
10	210	Baridhara Branch	Notun Bazar, Dhaka	10

Employees Table:

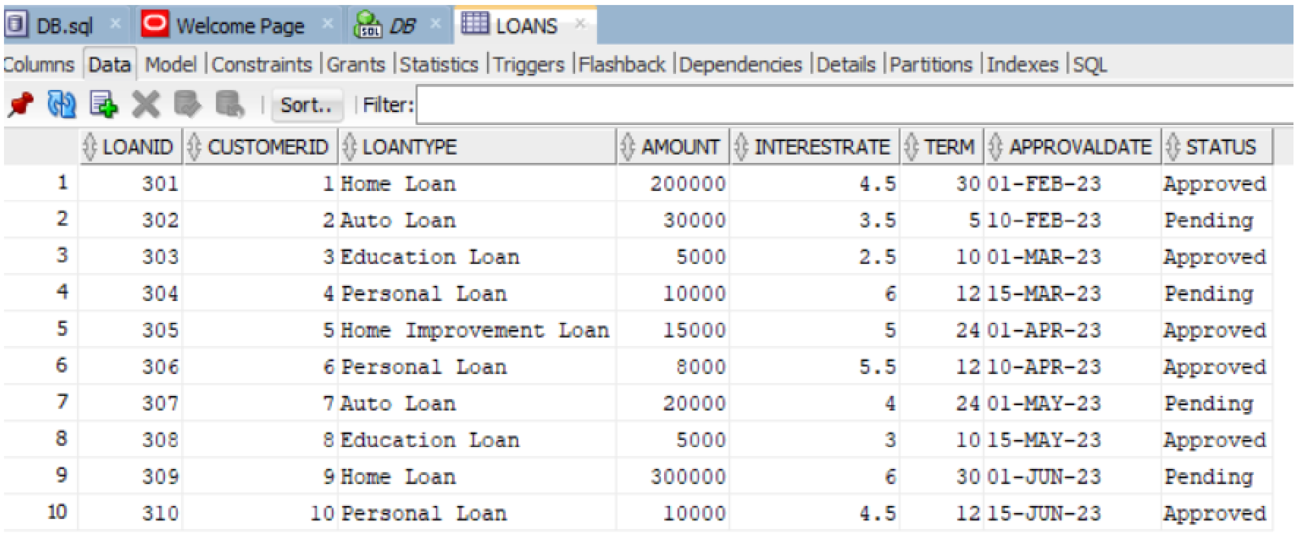

Welcome Page | DB | EMPLOYEES

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | De

Sort.. | Filter:

	EMPLOYEEID	FIRSTNAME	LASTNAME	POSITION	BRANCHID
1	1	Abdur	Razzak	Branch Manager	201
2	2	Mohsin	Jaman	Branch Manager	202
3	3	Nijam	Uddin	Branch Manager	203
4	4	Rasel	Khan	Branch Manager	204
5	5	Nusrat	Jahan	Branch Manager	205
6	6	Kuddus	Mollah	Branch Manager	206
7	7	Anisur	Rahman	Branch Manager	207
8	8	Rabbi	Mia	Branch Manager	208
9	9	Anika	Uffa	Branch Manager	209
10	10	Taslim	Khan	Branch Manager	210

Loans Table:

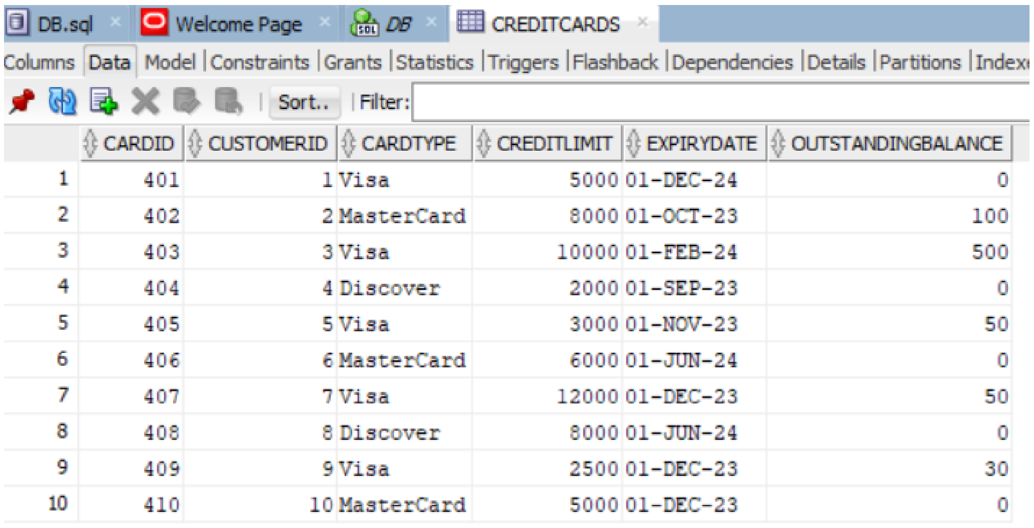

DB.sql | Welcome Page | DB | LOANS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

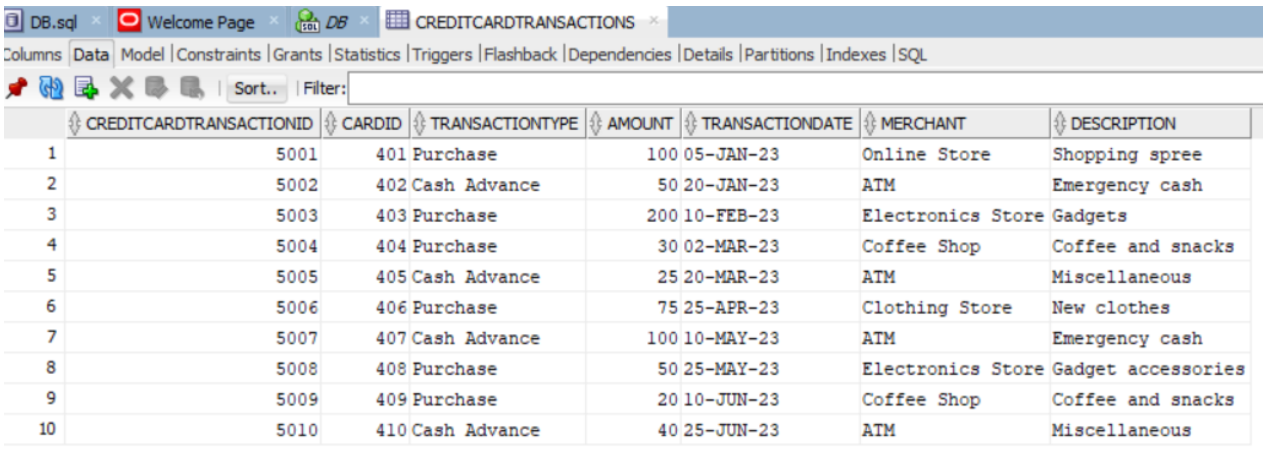
	LOANID	CUSTOMERID	LOANTYPE	AMOUNT	INTERESTRATE	TERM	APPROVALDATE	STATUS
1	301	1	Home Loan	200000	4.5	30	01-FEB-23	Approved
2	302	2	Auto Loan	30000	3.5	5	10-FEB-23	Pending
3	303	3	Education Loan	5000	2.5	10	01-MAR-23	Approved
4	304	4	Personal Loan	10000	6	12	15-MAR-23	Pending
5	305	5	Home Improvement Loan	15000	5	24	01-APR-23	Approved
6	306	6	Personal Loan	8000	5.5	12	10-APR-23	Approved
7	307	7	Auto Loan	20000	4	24	01-MAY-23	Pending
8	308	8	Education Loan	5000	3	10	15-MAY-23	Approved
9	309	9	Home Loan	300000	6	30	01-JUN-23	Pending
10	310	10	Personal Loan	10000	4.5	12	15-JUN-23	Approved

CreditCards Table:



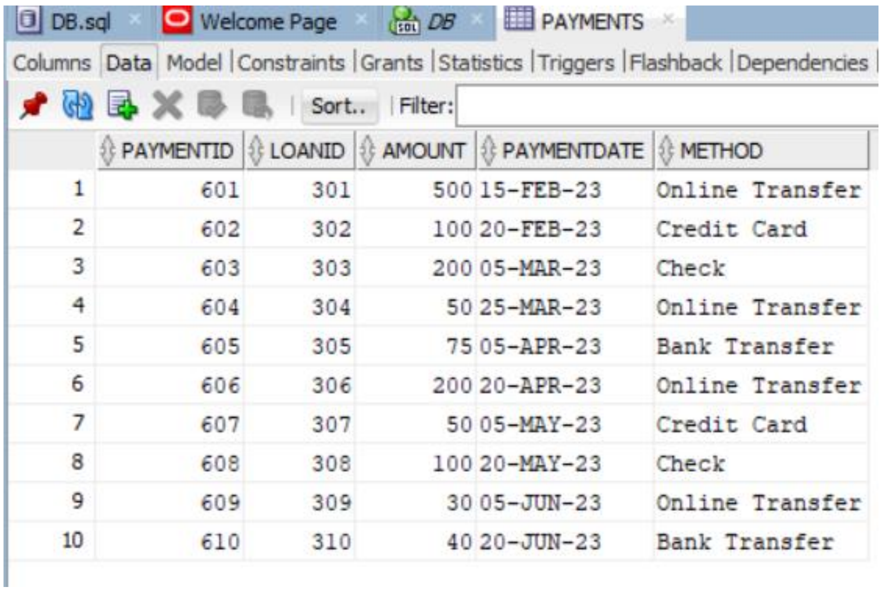
	CARDID	CUSTOMERID	CARDTYPE	CREDITLIMIT	EXPIRYDATE	OUTSTANDINGBALANCE
1	401	1	Visa	5000	01-DEC-24	0
2	402	2	MasterCard	8000	01-OCT-23	100
3	403	3	Visa	10000	01-FEB-24	500
4	404	4	Discover	2000	01-SEP-23	0
5	405	5	Visa	3000	01-NOV-23	50
6	406	6	MasterCard	6000	01-JUN-24	0
7	407	7	Visa	12000	01-DEC-23	50
8	408	8	Discover	8000	01-JUN-24	0
9	409	9	Visa	2500	01-DEC-23	30
10	410	10	MasterCard	5000	01-DEC-23	0

CreditCardTransactions Table:



	CREDITCARDTRANSACTIONID	CARDID	TRANSACTIONTYPE	AMOUNT	TRANSACTIONDATE	MERCHANT	DESCRIPTION
1	5001	401	Purchase	100	05-JAN-23	Online Store	Shopping spree
2	5002	402	Cash Advance	50	20-JAN-23	ATM	Emergency cash
3	5003	403	Purchase	200	10-FEB-23	Electronics Store	Gadgets
4	5004	404	Purchase	30	02-MAR-23	Coffee Shop	Coffee and snacks
5	5005	405	Cash Advance	25	20-MAR-23	ATM	Miscellaneous
6	5006	406	Purchase	75	25-APR-23	Clothing Store	New clothes
7	5007	407	Cash Advance	100	10-MAY-23	ATM	Emergency cash
8	5008	408	Purchase	50	25-MAY-23	Electronics Store	Gadget accessories
9	5009	409	Purchase	20	10-JUN-23	Coffee Shop	Coffee and snacks
10	5010	410	Cash Advance	40	25-JUN-23	ATM	Miscellaneous

Payments Table:



	PAYMENTID	LOANID	AMOUNT	PAYMENTDATE	METHOD
1	601	301	500	15-FEB-23	Online Transfer
2	602	302	100	20-FEB-23	Credit Card
3	603	303	200	05-MAR-23	Check
4	604	304	50	25-MAR-23	Online Transfer
5	605	305	75	05-APR-23	Bank Transfer
6	606	306	200	20-APR-23	Online Transfer
7	607	307	50	05-MAY-23	Credit Card
8	608	308	100	20-MAY-23	Check
9	609	309	30	05-JUN-23	Online Transfer
10	610	310	40	20-JUN-23	Bank Transfer

Table-Level Constraints:

1. Customers Table:
   - Unique constraint on **CustomerID** to ensure each customer has a distinct identifier.
   - Ensure uniqueness of the combination of **Email** and **Phone** to avoid duplicate customer entries.

2. Accounts Table:
   - Primary key constraint on **AccountID** to ensure a unique identifier for each account.
   - Foreign key constraint on **CustomerID** referencing **Customers(CustomerID)** to maintain referential integrity.

3. Transactions Table:
   - Primary key constraint on **TransactionID** for a unique identifier for each transaction.
   - Foreign key constraint on **AccountID** referencing **Accounts(AccountID)** to link transactions to specific accounts.

4. Branches Table:
   - Primary key constraint on **BranchID** for a unique identifier for each branch.
   - Foreign key constraint on **ManagerID** referencing **Employees(EmployeeID)** for the branch manager.

5. Employees Table:
   - Primary key constraint on **EmployeeID** for a unique identifier for each employee.

6. Loans Table:
   - Primary key constraint on **LoanID** for a unique identifier for each loan.
   - Foreign key constraint on **CustomerID** referencing **Customers(CustomerID)** to link loans to specific customers.

7. CreditCards Table:
   - Primary key constraint on **CardID** for a unique identifier for each credit card.
   - Foreign key constraint on **CustomerID** referencing **Customers(CustomerID)** to associate credit cards with customers.

8. CreditCardTransactions Table:
   - Primary key constraint on **CreditCardTransactionID** for a unique identifier for each credit card transaction.
   - Foreign key constraint on **CardID** referencing **CreditCards(CardID)** to link transactions to specific credit cards.

9. Payments Table:
   - Primary key constraint on **PaymentID** for a unique identifier for each payment.
   - Foreign key constraint on **LoanID** referencing **Loans(LoanID)** to associate payments with loans.

Project-Level Constraints:

1. Data Consistency:
   - Ensure consistency of customer information across tables by updating customer details in a centralized manner to prevent discrepancies.

2. Transaction Integrity:
   - Guarantee the consistency of transactions by enforcing foreign key constraints to link transactions to valid accounts or credit cards.

3. Security Constraints:
   - Implement access controls and authentication mechanisms to ensure that only authorized users can interact with the database.

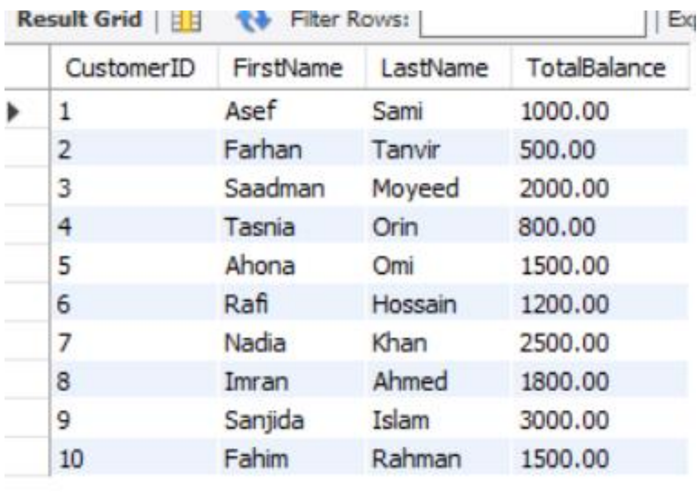
4. Business Rules:
   - Enforce specific business rules such as minimum balance requirements, maximum credit limits, and loan approval criteria.

5. Referential Integrity:
   - Maintain referential integrity by ensuring that foreign keys always reference valid primary keys in the referenced tables.

SQLs to Implement the Project with Example Outputs:

Query 1: Retrieve Customer Information with Total Balance

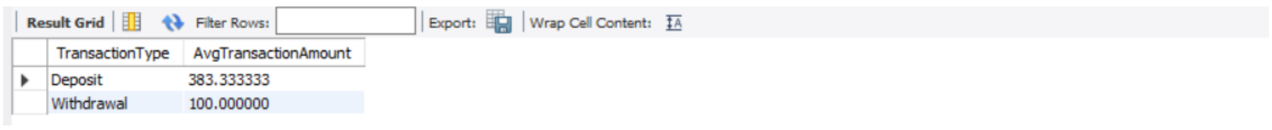
```
SELECT Customers.CustomerID, FirstName, LastName, SUM(Balance) AS
TotalBalance
FROM Customers
JOIN Accounts ON Customers.CustomerID = Accounts.CustomerID
GROUP BY Customers.CustomerID, FirstName, LastName;
```



CustomerID	FirstName	LastName	TotalBalance
1	Asef	Sami	1000.00
2	Farhan	Tanvir	500.00
3	Saadman	Moyeed	2000.00
4	Tasnia	Orin	800.00
5	Ahona	Omi	1500.00
6	Rafi	Hossain	1200.00
7	Nadia	Khan	2500.00
8	Imran	Ahmed	1800.00
9	Sanjida	Islam	3000.00
10	Fahim	Rahman	1500.00

Query 2: Find Average Transaction Amount

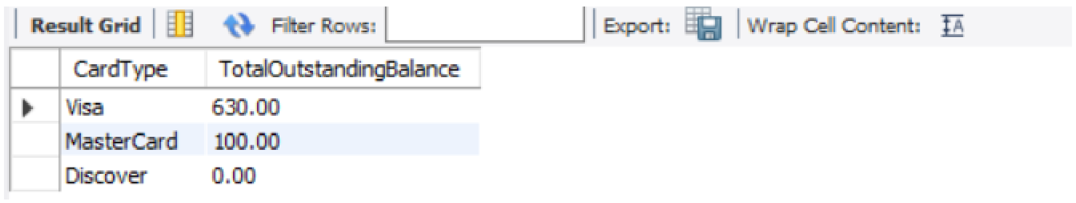
```
SELECT TransactionType, AVG(Amount) AS AvgTransactionAmount
FROM Transactions
GROUP BY TransactionType;
```



TransactionType	AvgTransactionAmount
Deposit	383.333333
Withdrawal	100.000000

Query 3: Total Outstanding Balance per Credit Card Type

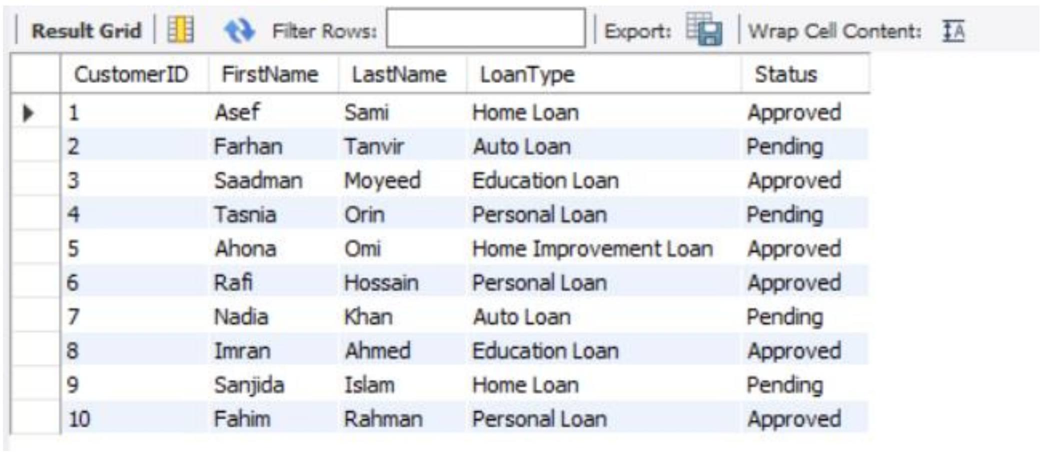
```
SELECT CreditCards.CardType, SUM(OutstandingBalance) AS
TotalOutstandingBalance
FROM CreditCards
GROUP BY CreditCards.CardType;
```



CardType	TotalOutstandingBalance
Visa	630.00
MasterCard	100.00
Discover	0.00

Query 4: List Customers and Their Loan Status

```
SELECT Customers.CustomerID, FirstName, LastName, LoanType, Status
FROM Customers
LEFT JOIN Loans ON Customers.CustomerID = Loans.CustomerID;
```

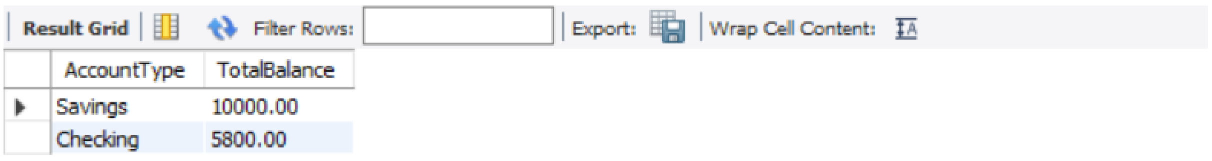


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerID	FirstName	LastName	LoanType	Status
1	Asef	Sami	Home Loan	Approved
2	Farhan	Tanvir	Auto Loan	Pending
3	Saadman	Moyeed	Education Loan	Approved
4	Tasnia	Orin	Personal Loan	Pending
5	Ahona	Omi	Home Improvement Loan	Approved
6	Rafi	Hossain	Personal Loan	Approved
7	Nadia	Khan	Auto Loan	Pending
8	Imran	Ahmed	Education Loan	Approved
9	Sanjida	Islam	Home Loan	Pending
10	Fahim	Rahman	Personal Loan	Approved

Query 5: Total Balance per Account Type

```
SELECT AccountType, SUM(Balance) AS TotalBalance
FROM Accounts
GROUP BY AccountType;
```

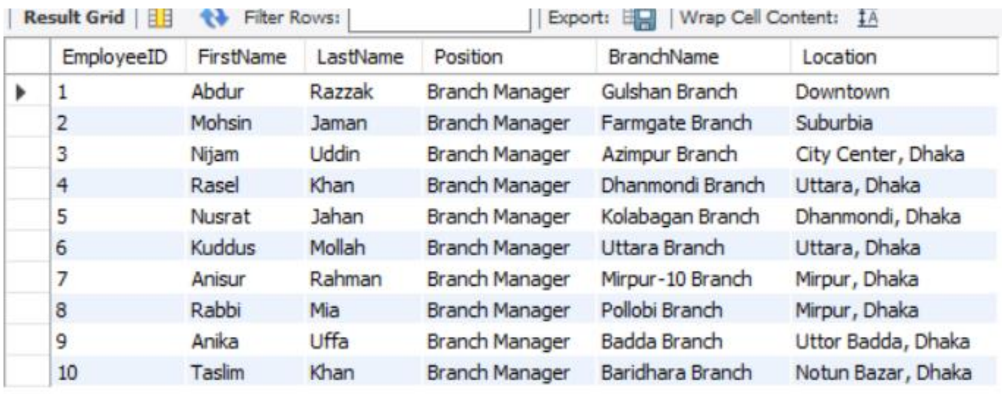


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AccountType	TotalBalance
Savings	10000.00
Checking	5800.00

Query 6: List Employees with Their Branch Information

```
SELECT Employees.EmployeeID, FirstName, LastName, Position, BranchName,
Location
FROM Employees
JOIN Branches ON Employees.BranchID = Branches.BranchID;
```

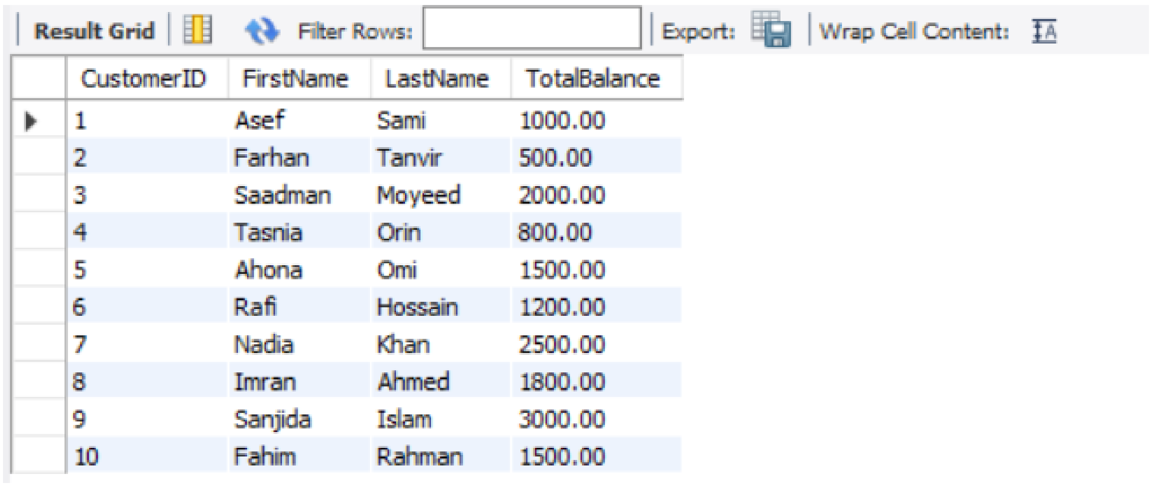


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EmployeeID	FirstName	LastName	Position	BranchName	Location
1	Abdur	Razzak	Branch Manager	Gulshan Branch	Downtown
2	Mohsin	Jaman	Branch Manager	Farmgate Branch	Suburbia
3	Nijam	Uddin	Branch Manager	Azimpur Branch	City Center, Dhaka
4	Rasel	Khan	Branch Manager	Dhanmondi Branch	Uttara, Dhaka
5	Nusrat	Jahan	Branch Manager	Kolabagan Branch	Dhanmondi, Dhaka
6	Kuddus	Mollah	Branch Manager	Uttara Branch	Uttara, Dhaka
7	Anisur	Rahman	Branch Manager	Mirpur-10 Branch	Mirpur, Dhaka
8	Rabbi	Mia	Branch Manager	Pollobi Branch	Mirpur, Dhaka
9	Anika	Uffa	Branch Manager	Badda Branch	Uttor Badda, Dhaka
10	Taslim	Khan	Branch Manager	Baridhara Branch	Notun Bazar, Dhaka

Query 7: Retrieve Customer Information with Total Balance

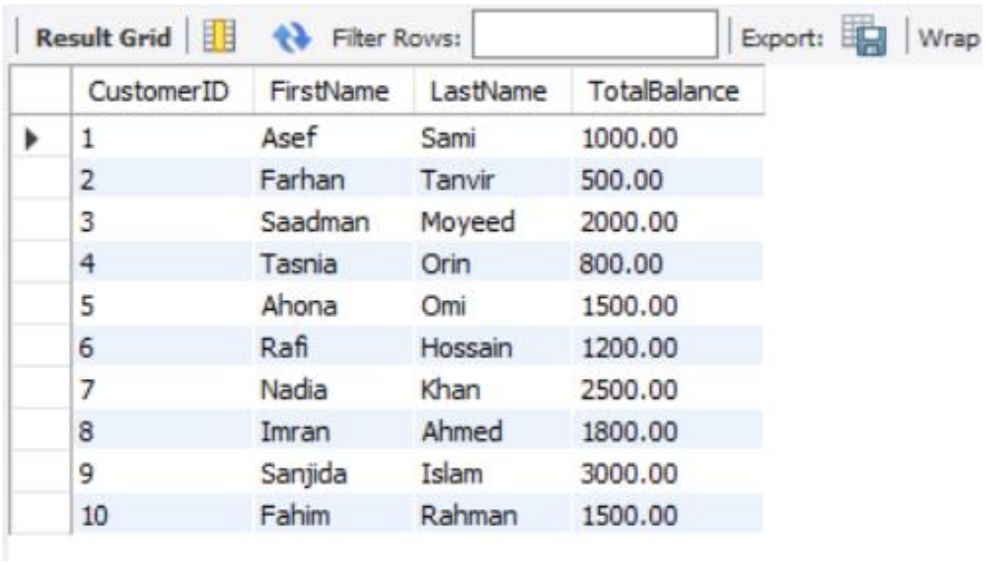
```
SELECT Customers.CustomerID, FirstName, LastName, SUM(Balance) AS
TotalBalance
FROM Customers
JOIN Accounts ON Customers.CustomerID = Accounts.CustomerID
GROUP BY Customers.CustomerID, FirstName, LastName;
```



CustomerID	FirstName	LastName	TotalBalance
1	Asef	Sami	1000.00
2	Farhan	Tanvir	500.00
3	Saadman	Moyeed	2000.00
4	Tasnia	Orin	800.00
5	Ahona	Omi	1500.00
6	Rafi	Hossain	1200.00
7	Nadia	Khan	2500.00
8	Imran	Ahmed	1800.00
9	Sanjida	Islam	3000.00
10	Fahim	Rahman	1500.00

Query 8: Find Average Transaction Amount

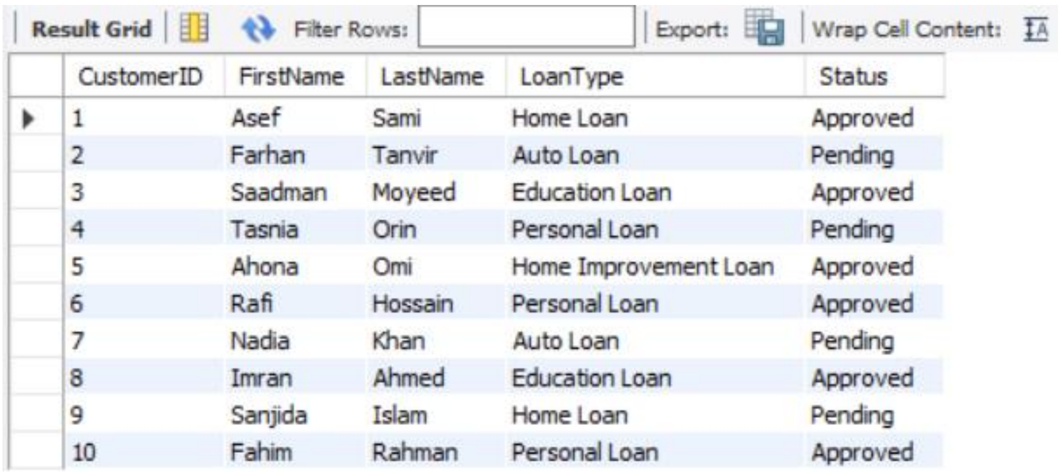
```
SELECT TransactionType, AVG(Amount) AS AvgTransactionAmount
FROM Transactions
GROUP BY TransactionType;
```



CustomerID	FirstName	LastName	TotalBalance
1	Asef	Sami	1000.00
2	Farhan	Tanvir	500.00
3	Saadman	Moyeed	2000.00
4	Tasnia	Orin	800.00
5	Ahona	Omi	1500.00
6	Rafi	Hossain	1200.00
7	Nadia	Khan	2500.00
8	Imran	Ahmed	1800.00
9	Sanjida	Islam	3000.00
10	Fahim	Rahman	1500.00

Query 9: List Customers and Their Loan Status

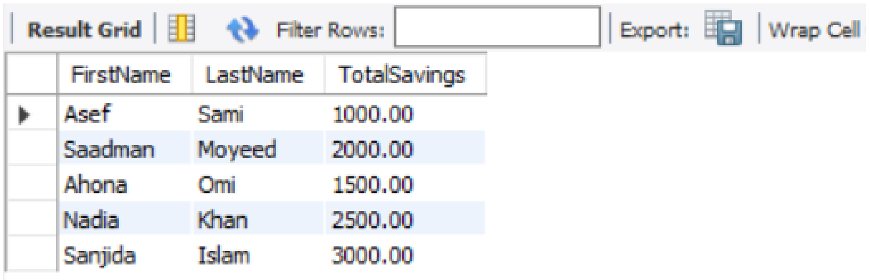
```
SELECT Customers.CustomerID, FirstName, LastName, LoanType, Status
FROM Customers
LEFT JOIN Loans ON Customers.CustomerID = Loans.CustomerID;
```



CustomerID	FirstName	LastName	LoanType	Status
1	Asef	Sami	Home Loan	Approved
2	Farhan	Tanvir	Auto Loan	Pending
3	Saadman	Moyeed	Education Loan	Approved
4	Tasnia	Orin	Personal Loan	Pending
5	Ahona	Omi	Home Improvement Loan	Approved
6	Rafi	Hossain	Personal Loan	Approved
7	Nadia	Khan	Auto Loan	Pending
8	Imran	Ahmed	Education Loan	Approved
9	Sanjida	Islam	Home Loan	Pending
10	Fahim	Rahman	Personal Loan	Approved

Query 10: Uncover Customer of account Savings

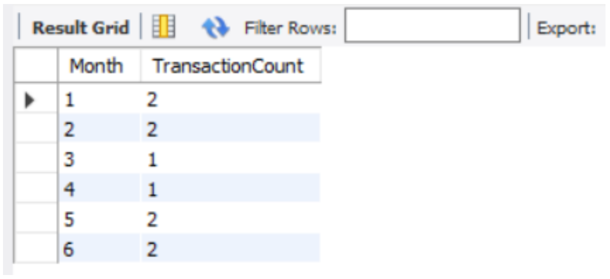
```
SELECT FirstName, LastName, SUM(Balance) AS TotalSavings
FROM Customers
JOIN Accounts ON Customers.CustomerID = Accounts.CustomerID
WHERE AccountType = 'Savings'
GROUP BY Customers.CustomerID, FirstName, LastName;
```



FirstName	LastName	TotalSavings
Asef	Sami	1000.00
Saadman	Moyeed	2000.00
Ahona	Omi	1500.00
Nadia	Khan	2500.00
Sanjida	Islam	3000.00

Query 11: Count Monthly Transactions

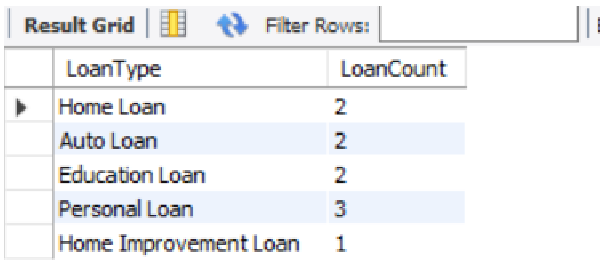
```
SELECT MONTH(TransactionDate) AS Month, COUNT(*) AS TransactionCount
FROM Transactions
GROUP BY Month;
```



Month	TransactionCount
1	2
2	2
3	1
4	1
5	2
6	2

Query 12: Loan Status count

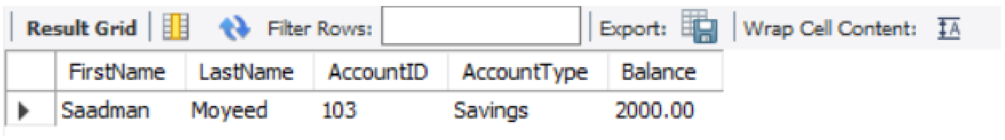
```
SELECT LoanType, COUNT(*) AS LoanCount
FROM Loans
GROUP BY LoanType;
```



LoanType	LoanCount
Home Loan	2
Auto Loan	2
Education Loan	2
Personal Loan	3
Home Improvement Loan	1

Query 13: List Accounts for a Specific Customer

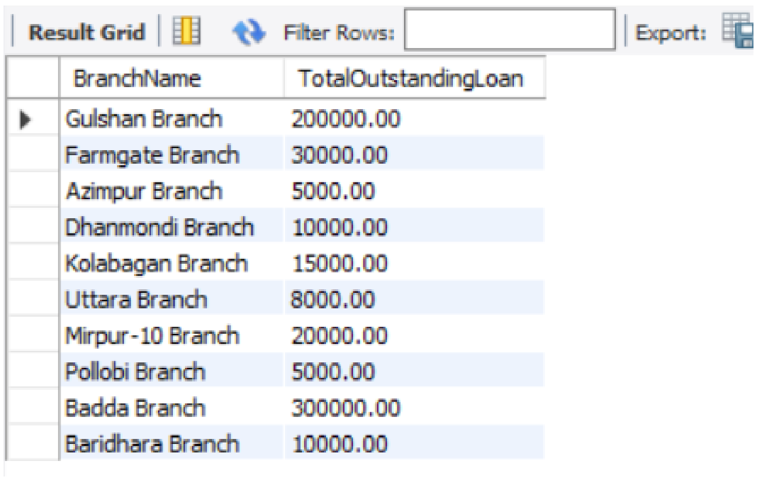
```
SELECT Customers.FirstName, Customers.LastName, Accounts.AccountID,
Accounts.AccountType, Accounts.Balance
FROM Customers
JOIN Accounts ON Customers.CustomerID = Accounts.CustomerID
WHERE Customers.CustomerID = 3;
```



FirstName	LastName	AccountID	AccountType	Balance
Saadman	Moyeed	103	Savings	2000.00

Query 14: Branches with Total Outstanding Loan Amount

```
SELECT Branches.BranchName, SUM(Loans.Amount) AS TotalOutstandingLoan
FROM Branches
LEFT JOIN Employees ON Branches.ManagerID = Employees.EmployeeID
LEFT JOIN Loans ON Employees.EmployeeID = Loans.CustomerID
GROUP BY Branches.BranchName;
```



BranchName	TotalOutstandingLoan
Gulshan Branch	200000.00
Farmgate Branch	30000.00
Azimpur Branch	5000.00
Dhanmondi Branch	10000.00
Kolabagan Branch	15000.00
Uttara Branch	8000.00
Mirpur-10 Branch	20000.00
Pollobi Branch	5000.00
Badda Branch	300000.00
Baridhara Branch	10000.00

Finding the Normal Form:

Customers Table:
Functional Dependencies:
- CustomerID → FirstName, LastName, Address, Email, Phone

Normalization Analysis:
- The table is in 1st Normal Form (1NF) with no transitive dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (CustomerID).
- The absence of transitive dependencies and full functional dependencies ensures the removal of redundancy.
- In 3rd Normal Form (3NF), each attribute is directly dependent on the primary key, minimizing redundancy and enhancing data integrity.

Accounts Table:
Functional Dependencies:
- AccountID → CustomerID, AccountType, Balance, OpenDate, Status
- CustomerID → FirstName, LastName, Address, Email, Phone

Normalization Analysis:
- The table is in 2nd Normal Form (2NF) with no partial dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (AccountID) or the super key (CustomerID).
- The absence of partial and transitive dependencies ensures the removal of redundancy.
- In 3rd Normal Form (3NF), the table further reduces redundancy, supporting efficient data management.

Transactions Table:
Functional Dependencies:
- TransactionID → AccountID, TransactionType, Amount, TransactionDate, Description

Normalization Analysis:
- The table is in 2nd Normal Form (2NF) with no partial dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (TransactionID).
- The absence of partial and transitive dependencies ensures the removal of redundancy.

- In 3rd Normal Form (3NF), the table maintains a high level of normalization, enhancing data integrity.

Branches Table:
Functional Dependencies:
- BranchID → BranchName, Location, ManagerID
- ManagerID → FirstName, LastName, Position

Normalization Analysis:
- The table is in 2nd Normal Form (2NF) with no partial dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (BranchID) or the super key (ManagerID).
- The absence of partial and transitive dependencies ensures the removal of redundancy.
- In 3rd Normal Form (3NF), the table further reduces redundancy, supporting efficient data management.

Employees Table:
Functional Dependencies:
- EmployeeID → FirstName, LastName, Position, BranchID

Normalization Analysis:
- The table is in 2nd Normal Form (2NF) with no partial dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (EmployeeID).
- The absence of partial and transitive dependencies ensures the removal of redundancy.
- In 3rd Normal Form (3NF), the table maintains a high level of normalization, enhancing data integrity.

Loans Table:
Functional Dependencies:
- LoanID → CustomerID, LoanType, Amount, InterestRate, Term, ApprovalDate, Status

Normalization Analysis:
- The table is in 2nd Normal Form (2NF) with no partial dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (LoanID).
- The absence of partial and transitive dependencies ensures the removal of redundancy.

- In 3rd Normal Form (3NF), the table maintains a high level of normalization, supporting efficient data management.

CreditCards Table:
Functional Dependencies:
- CardID → CustomerID, CardType, CreditLimit, ExpiryDate, OutstandingBalance

Normalization Analysis:
- The table is in 2nd Normal Form (2NF) with no partial dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (CardID).
- The absence of partial and transitive dependencies ensures the removal of redundancy.
- In 3rd Normal Form (3NF), the table maintains a high level of normalization, supporting efficient data management.

CreditCardTransactions Table:
Functional Dependencies:
- CreditCardTransactionID → CardID, TransactionType, Amount, TransactionDate, Merchant, Description

Normalization Analysis:
- The table is in 2nd Normal Form (2NF) with no partial dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (CreditCardTransactionID).
- The absence of partial and transitive dependencies ensures the removal of redundancy.
- In 3rd Normal Form (3NF), the table maintains a high level of normalization, enhancing data integrity.

Payments Table:
Functional Dependencies:
- PaymentID → LoanID, Amount, PaymentDate, Method

Normalization Analysis:
- The table is in 2nd Normal Form (2NF) with no partial dependencies.
- All non-prime attributes are fully functionally dependent on the primary key (PaymentID).
- The absence of partial and transitive dependencies ensures the removal of redundancy.

- In 3rd Normal Form (3NF), the table maintains a high level of normalization, supporting efficient data management.

The provided database exhibits a high level of normalization (3NF) across its tables, ensuring data integrity, minimizing redundancy, and supporting efficient data management practices. This normalization level is essential for creating a robust and scalable relational database system.

Future Works:

1. Enhanced Security Measures:
Implement advanced security features such as encryption for sensitive data, secure access controls, and regular security audits to ensure the confidentiality and integrity of customer information.

2. Improved Reporting and Analytics:
Integrate a robust reporting system for detailed financial analysis, customer behavior insights, and performance tracking. This can aid in making informed business decisions and enhancing customer services.

3. Mobile Application Development:
Consider developing a mobile application for customers to easily access their account information, perform transactions, and receive real-time alerts. This enhances customer convenience and engagement.

4. Machine Learning for Fraud Detection:
Explore the implementation of machine learning algorithms to detect unusual patterns or anomalies in transactions, helping in the early identification of potential fraudulent activities.

5. Customer Relationship Management (CRM) Integration:
Integrate a CRM system to streamline customer interactions, manage communication effectively, and provide personalized services based on customer preferences and behaviors.

Conclusions:

The development and implementation of the Banking Management System mark a significant milestone in achieving a comprehensive and efficient platform for managing financial operations. Throughout the project, careful consideration was given to database design, data relationships, and query functionalities to ensure a robust and scalable solution.

Key Findings:

1. **Relational Integrity:** The database design successfully establishes and maintains relationships between entities, fostering data consistency and accuracy. Tables, such as Customers, Accounts, Transactions, Loans, and Credit Cards, are interconnected, reflecting a holistic representation of the banking domain.
2. **Query Flexibility:** Utilizing aggregate functions and outer joins in queries enhances the system's analytical capabilities. This allows for insightful reporting, facilitating data-driven decision-making processes for financial analysis and customer relationship management.

3. **Data Constraints:** The application of table-level constraints, such as primary and foreign keys, ensures adherence to business rules and data integrity. This guarantees that the database remains a reliable source of truth for banking-related information.

4. **Scalability:** The current system is designed with scalability in mind, allowing for the seamless integration of additional features and functionalities. This adaptability positions the Banking Management System to accommodate future business requirements and technological advancements.

The next phase of development should focus on implementing advanced security protocols, including encryption, multi-factor authentication, and continuous monitoring. This will fortify the system against potential security threats and safeguard sensitive customer data. Enhancing the reporting and analytics capabilities will empower financial institutions to derive deeper insights from their data. Incorporating advanced analytical tools and machine learning algorithms can facilitate trend analysis, risk assessment, and personalized customer experiences. The development of a user-friendly mobile application will provide customers with seamless access to their accounts, real-time transaction updates, and personalized notifications. This will elevate user experience and engagement levels. Integrating a CRM system will streamline customer interactions, allowing financial institutions to tailor services based on individual preferences. This customer-centric approach fosters loyalty and satisfaction.

In conclusion, the Banking Management System stands as a testament to effective database design and implementation. It not only addresses the core functionalities of banking but also sets the stage for a dynamic and responsive financial platform. The success of this project lays the groundwork for continued innovation, ensuring that the system remains at the forefront of technology and industry best practices.
As the financial landscape evolves, the Banking Management System is poised to adapt and grow, meeting the ever-changing needs of both financial institutions and their customers. This project exemplifies a commitment to excellence in managing financial data, fostering secure transactions, and providing a foundation for future advancements in the dynamic realm of banking technology.