CIS 3400
Professor Edsn Kensington
Group Project Banking (#4)
Pujan Ketheeswarapaskaran, Daniel Furmanov, Zohar Balter, Daniel Tlaxcaltecatl, Faiza Nawar

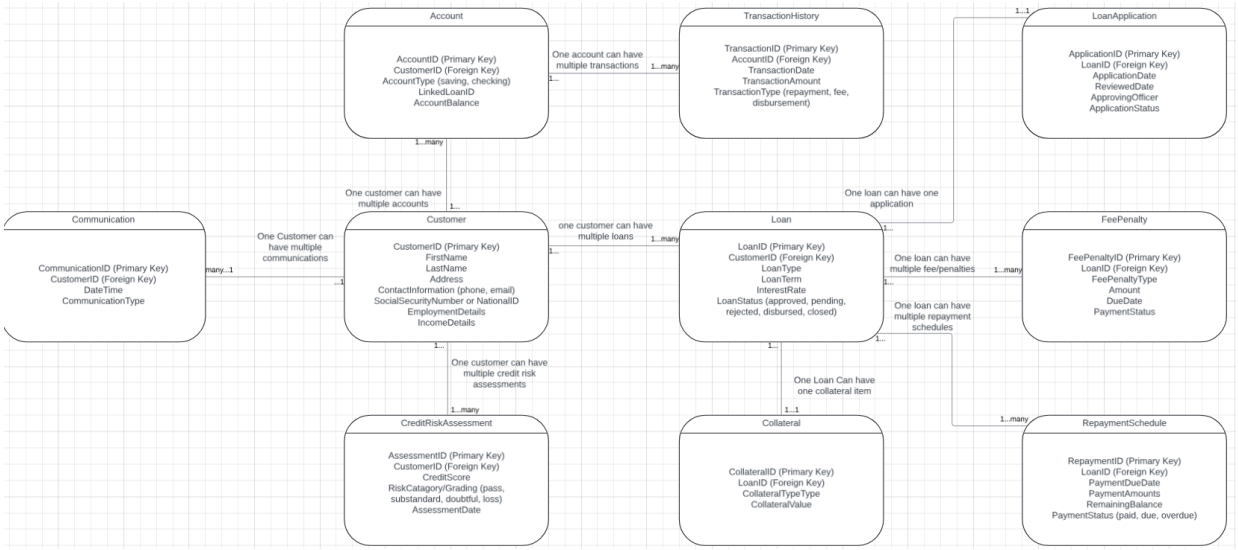
I. Business Scenario

Our bank has been in operation for the past two years, during which we have relied on Microsoft Excel and paper logs to manage customer information, accounts, and loans. However, these methods have proven increasingly inadequate as our operations grow, prompting us to consider implementing a modern database system to enhance efficiency, accuracy, and security. Currently, customer accounts and loans are managed through a paper-based system and fragmented data-entry practices, leading to significant operational challenges. These outdated methods result in inconsistencies, discrepancies, and gaps in the recorded data, which can severely impact decision-making. The inefficiencies in retrieving and analyzing critical information cause delays in responding to customer inquiries and making business decisions.

Data redundancy and security are also critical concerns with our current system, where duplicate information is often stored across multiple spreadsheets and paper records. This not only wastes storage space but also increases the risk of inconsistencies and errors. Additionally, without proper security measures, sensitive customer data is vulnerable to unauthorized access and potential breaches. Implementing a modern database would eliminate redundancy by centralizing data storage and ensuring each piece of information is stored only once. It would also enhance security through encryption, access controls, and audit trails, protecting our data from unauthorized access and ensuring compliance with regulatory standards.

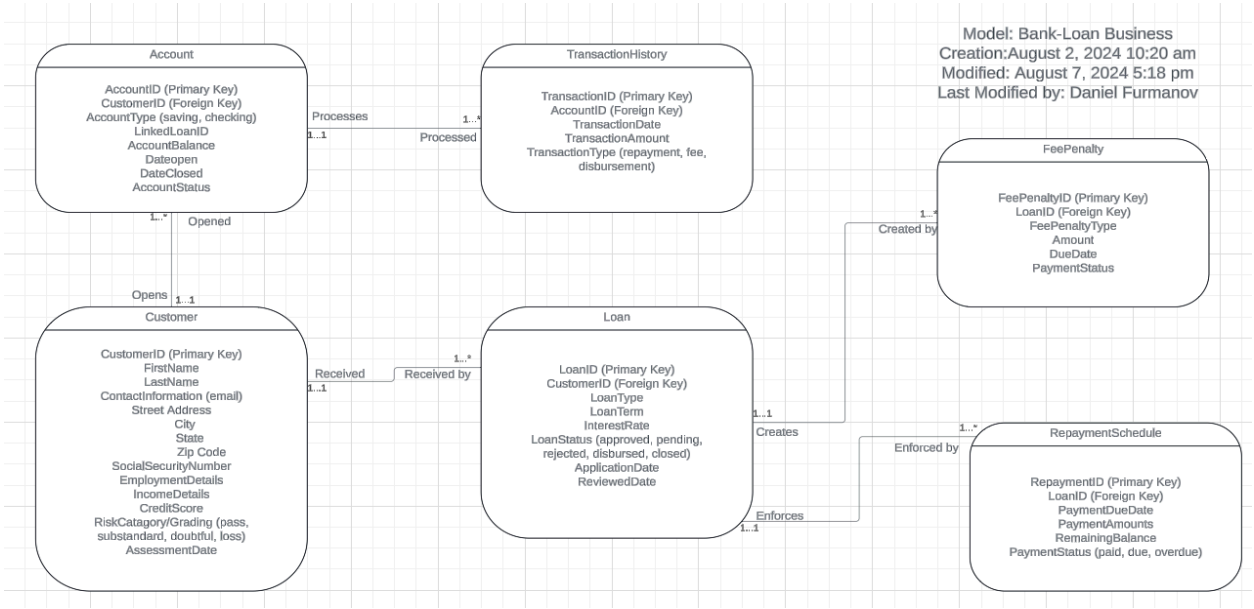
Additionally, we need to keep meticulous track of repayments, fees, and transaction histories to ensure accurate financial management. Our current manual system makes it difficult to maintain up-to-date information, leading to potential errors, missed payments, and customer dissatisfaction. A modern database would automate the recording of repayments, calculation of fees, and updating of transaction histories, providing real-time visibility into financial activities. This would enable us to send timely payment reminders, apply late fees consistently, and generate detailed reports for better financial planning.

II. ER Model using UML Notation



This is our first draft of the ER diagram. There were a few issues with it. The first issue that was solved was that we made it smaller to keep it more simple. The second and most important issue was that it was not in the standard UML notation. The next image is our final draft of the ER model that solved both problems.

Final ER Model



Relationship Sentences:

One **Customer** must open one or more **Accounts.**
One **Account** must only be opened by one and only one **Customer**.
One **Customer** can receive one or more **Loans**.
One **Loan** is received by one and only one **Customer.**
One **Loan** can create one or more **Fee Penalties**.
One **Fee Penalty** is created by one and only one **Loan.**
One **Loan** must enforce one or more **Repayment Schedule**.
One **Repayment Schedule** must be enforced by one and only one **Loan.**
One **Account** can process one or more **Transaction Histories.**
One **Transaction History** can only be processed by one and only one **Account.**

III. Conversion to a Relational Database

Account (Account Id (key), Customer Id (foreign key), Account type, Linked loan, Account balance, Date open, Date closed, Account status)
Customer (Customer Id (key), First name, Last name, Contact information, Street Address, State, City, Zip Code, Social security, Employment details, Income details, Credit score, Risk category or grading, Assessment date)
Transaction History (Transaction Id (key), Account Id (foreign key), Transaction date, Transaction amount, Transaction type)
Loan (Loan Id (key), Customer Id (foreign key), Loan type, Loan term, Interest type, Loan status, Application date, Reviewed date)

Fee Penalty (Fee penalty Id (key), Loan Id (foreign key), Fee penalty type, Amount, Due date, Payment status)
Repayment Schedule (Repayment schedule Id (key), Loan Id (foreign key), Payment due date, Payment amounts, Remaining balance, Payment status)

IV. Normalization

FeePenalty | RepaymentSchedule

FeePenaltyI	LoanID	FeePenaltyT	Amount	DueDate	FeePayment
1	1	early	-00	2024-02-27	paid
2	2	early	-00	2024-05-19	paid
3	3	overdue	69.00	2024-01-28	paid
4	4	early	-00	2024-07-18	paid
5	5	early	-00	2024-03-27	paid
6	6	late	-00	2024-05-30	due
7	7	overdue	19.00	2024-03-16	paid
8	8	early	-00	2024-05-11	paid
9	9	early	-00	2024-04-17	paid
10	10	early	-00	2024-05-28	paid

We ran into an error when importing our data into Access. Our 0s have turned into a -00, in the FeePenalty and RepaymentSchedule tables. In order to fix this issue we had to go back to the excel file. After some editing and taking some time to play with the data. The problem that was found was that the numbers in the column are not in the same number format. So, we adjusted the column and made every number in the column the same number format and it fixed our issue.

Customer

CustomerID	FirstName	LastName	Address	ContactInfor	Soc
1	Kimberly	Nguyen	8737 Charles Ce	scotttaylor@gr	868
2	Maurice	Mccarthy	64514 Benjamii	glee@smith.or	875
3	Daniel	Stone	67973 David Sq	cannonkaren@	353
4	Scott	Powell	12227 Matthew	sean79@gmail	719
5	Anna	Cisneros	USNS PadillaFP	luis75@housto	436
6	Tyrone	Thomas	PSC 8523, Box 3	mariamendez@	172
7	Wesley	Bush	7207 Jose Wayl	reevesangela@	425

Here, Our Customer table was in an unnormalized normal form. In order to get it into 1st normal form. We had to make the Address column into its atomic value. In the screenshot above, the address is all in one column, it needs to be separated into Street Address, City, State, and Zip code. After fixing this issue, we now have all tables in normal form as shown below.

1. Account Relation

Relation:

- Account (AccountID (PK), CustomerID (FK), AccountType, LinkedLoanID, AccountBalance, DateOpened, DateClosed, AccountStatus)

Sample Data:

AccountID	CustomerID	AccountType	LinkedLoanID	AccountBalance	DateOpened	DateClosed	AccountStatus
1	1	saving	1	66,667.00	1993-12-21	N/A	Active
2	2	saving	2	49,207.00	1982-07-16	N/A	Active
3	3	saving	3	27,503.00	1984-07-01	N/A	Active
4	4	checking	4	41,327.00	2006-08-29	N/A	Active
5	5	saving	5	41,524.00	2014-11-17	N/A	Active

Key:

- AccountID

Functional Dependencies (FDs):

- FD1: AccountID → CustomerID(fk), AccountType, LinkedLoanID, AccountBalance, DateOpened, DateClosed, AccountStatus

Normalization:

- 1NF: Meets the definition of a relation (No repeating groups, atomic attributes)
- 2NF: No partial key dependencies (AccountID is the only key, all other attributes fully dependent)
- 3NF: No transitive dependencies (All attributes directly dependent on the primary key)
- BCNF: All determinants are candidate keys

2. Customer Relation

Relation:

- Customer (CustomerID (key), FirstName, LastName, ContactInformation, StreetAddress, City, State, ZipCode, SocialSecurityNumber, EmploymentDetails, IncomeDetails, CreditScore, RiskCategoryGrading, AssessmentDate)

Sample Data:

CustomerID	FirstName	LastName	ContactInformation	Street Address	City	State	Zip Code	SocialSecurityNumber	EmploymentDetails	IncomeDetails	CreditScore	RiskCategoryGrading	AssessmentDate
1	Kimberly	Nguyen	scotttaylor@gmail.com	8737 Charles Centers Apt. 890	East Isaiah	NV	87803	868-12-5031	Therapist, sports	$136708	750	pass	2024-02-07
2	Maurice	Mccarthy	glee@smith.org	64514 Benjamin Port Suite 666	South Jeffrey	WV	75316	875-04-8670	Accountant, chartered certified	$145862	708	pass	2024-02-13
3	Daniel	Stone	cannonkaren@miller.net	67973 David Squares Apt. 066	Carrollville	CT	00724	353-83-0805	Gaffer	$139642	580	pass	2024-06-28
4	Scott	Powell	sean79@gmail.com	12227 Matthew Creek Suite 267	Robertstad	MI	02466	719-64-0365	Horticultural therapist	$124421	657	pass	2024-05-10
5	Anna	Cisneros	luis75@houston.com	7207 Jose Way	North Mason	NY	68873	436-74-5309	Comptroller	$47803	740	pass	2024-06-09

Key:

- CustomerID

Functional Dependencies (FDs):

- FD1: CustomerID → FirstName, LastName, ContactInformation, StreetAddress, City, State, ZipCode, SocialSecurityNumber, EmploymentDetails, IncomeDetails, CreditScore, RiskCategoryGrading, AssessmentDate

Normalization:

- 1NF: Meets the definition of a relation (No repeating groups, atomic attributes)
- 2NF: No partial key dependencies (CustomerID is the only key, all other attributes fully dependent)
- 3NF: No transitive dependencies (All attributes directly dependent on the primary key)
- BCNF: All determinants are candidate keys

3. FeePenalty Relation

Relation:

- FeePenaltyID → (FeePenaltyID (key), LoanID (fk), FeePenaltyType, Amount, DueDate, FeePaymentStatus)

Sample Data:

FeePenaltyID	LoanID	FeePenaltyType	Amount	DueDate	FeePaymentStatus
1	1	early	0	2024-02-27	paid
2	2	early	0	2024-05-19	paid
3	3	overdue	69	2024-01-28	paid
4	4	early	0	2024-07-18	paid
5	5	early	0	2024-03-27	paid

Key:

- FeePenaltyID

Functional Dependencies (FDs):

- FD1: FeePenaltyID → LoanID(fk), FeePenaltyType, Amount, DueDate, FeePaymentStatus

Normalization:

- 1NF: Meets the definition of a relation (No repeating groups, atomic attributes)
- 2NF: No partial key dependencies (CustomerID is the only key, all other attributes fully dependent)
- 3NF: No transitive dependencies (All attributes directly dependent on the primary key)
- BCNF: All determinants are candidate keys

4. Loan Relation

Relation:

- Loan (LoanID (key), CustomerID (fk), LoanType, LoanTerm, InterestRate, LoanStatus, ApplicationDate, ReviewDate)

Sample Data:

LoanID	CustomerID	LoanType	LoanTerm	InterestRate	LoanStatus	ApplicationDate	ReviewDate
1	1	personal	12	0.0749	Approved	2024-07-28	2024-09-17
2	2	car	24	0.1478	Approved	2024-07-11	2024-12-02
3	3	personal	12	0.0874	Approved	2024-06-13	2024-12-06
4	4	personal	12	0.0874	Approved	2024-07-02	2024-11-04
5	5	car	24	0.0564	Approved	2024-06-19	2024-08-12

Key:

- LoanID

Functional Dependencies (FDs):

- FD1: LoanID → CustomerID(fk), LoanType, LoanTerm, InterestRate, LoanStatus, ApplicationDate, ReviewDate

Normalization:

- 1NF: Meets the definition of a relation (No repeating groups, atomic attributes)
- 2NF: No partial key dependencies (CustomerID is the only key, all other attributes fully dependent)
- 3NF: No transitive dependencies (All attributes directly dependent on the primary key)
- BCNF: All determinants are candidate keys

5. RepaymentSchedule Relation

Relation:

- RepaymentSchedule (RepaymentID (key), LoanID (fk), PaymentDueDate, PaymentAmounts, RemainingBalance, LoanPaymentStatus)

Sample Data:

RepaymentID	LoanID	PaymentDueDate	PaymentAmounts	RemainingBalance	LoanPaymentStatus
1	1	2024-07-26	131.00	0.00	paid
2	2	2024-06-28	601.00	11.00	paid
3	3	2024-06-02	853.00	8374.00	paid
4	4	2024-05-24	467.00	1812.00	paid
5	5	2024-03-04	923.00	2575.00	paid

Key:

- RepaymentScheduleID

Functional Dependencies (FDs):

- FD1: RepaymentID → LoanID(fk), PaymentDueDate, PaymentAmounts, RemainingBalance, LoanPaymentStatus

Normalization:

- 1NF: Meets the definition of a relation (No repeating groups, atomic attributes)
- 2NF: No partial key dependencies (CustomerID is the only key, all other attributes fully dependent)
- 3NF: No transitive dependencies (All attributes directly dependent on the primary key)
- BCNF: All determinants are candidate keys

6. TransactionHistory Table

Relation:

- TransactionHistory (TransactionID (key), AccountID (fk), TransactionDate, TransactionAmount, TransactionType)

Sample Data:

TransactionID	AccountID	TransactionDate	TransactionAmount	TransactionType
1	1	2024-02-09	6,116.00	fee
2	2	2024-05-04	188.00	repayment
3	3	2024-02-19	5,772.00	disbursement
4	4	2024-04-11	9,961.00	fee
5	5	2024-01-05	1,796.00	fee

Key:

- TransactionID

Functional Dependencies (FDs):

- FD1: TransactionID → AccountID(fk), TransactionDate, TransactionAmount, TransactionType

Normalization:

- 1NF: Meets the definition of a relation (No repeating groups, atomic attributes)
- 2NF: No partial key dependencies (CustomerID is the only key, all other attributes fully dependent)
- 3NF: No transitive dependencies (All attributes directly dependent on the primary key)
- BCNF: All determinants are candidate keys

Final Set of Relations (Including TransactionHistory)

- Account (AccountID (key), CustomerID (fk), AccountType, LinkedLoanID, AccountBalance, DateOpened, DateClosed, AccountStatus)
- Customer (CustomerID (key), FirstName, LastName, ContactInformation, StreetAddress, City, State, ZipCode, SocialSecurityNumber, EmploymentDetails, IncomeDetails, CreditScore, RiskCategoryGrading, AssessmentDate)

- FeePenalty (FeePenaltyID (key), LoanID (fk), FeePenaltyType, Amount, DueDate, FeePaymentStatus)
- Loan (LoanID (key), CustomerID (fk), LoanType, LoanTerm, InterestRate, LoanStatus, ApplicationDate, ReviewDate)
- RepaymentSchedule (RepaymentID (key), LoanID (fk), PaymentDueDate, PaymentAmounts, RemainingBalance, LoanPaymentStatus)
- TransactionHistory (TransactionID (key), AccountID (fk), TransactionDate, TransactionAmount, TransactionType)

V. Creating the Database Schema with Structured Query Language

The following SQL code will create the data tables and assign a primary key to them.

```
CREATE TABLE Account
(
     AccountID    VARCHAR(255) NOT NULL,
     CustomerID   VARCHAR(255),
     AccountType  VARCHAR(255),
     LinkedLoanID VARCHAR(255),
     AccountBalance VARCHAR(255),
     DateOpened VARCHAR(255)
     DateClosed VARCHAR(255),
     AccountStatus VARCHAR(255)
     CONSTRAINT pk_Account
          PRIMARY KEY (AccountID)
);


CREATE TABLE Customer
(
     CustomerID   VARCHAR(255) NOT NULL,
     FirstName    VARCHAR(255),
     LastName     VARCHAR(255),
     ContactInformation  VARCHAR(255),
     Street Address       VARCHAR(255),
     City      VARCHAR(255),
```

```
      State       VARCHAR(255),
      Zip Code VARCHAR(255),
      SocialSecurityNumber VARCHAR(255),
      EmploymentDetails VARCHAR(255),
      CreditScore VARCHAR(255),
      RiskCategoryGrading VARCHAR(255),
      AssessmentDate VARCHAR(255),
      CONSTRAINT pk_customer
           PRIMARY KEY (CustomerID)
);

CREATE TABLE FeePenalty
(
      FeePenaltyID  VARCHAR(255) NOT NULL,
      LoanID     VARCHAR(255),
      FeePenaltyType VARCHAR(255),
      Amount VARCHAR(255),
      DueDate VARCHAR(255),
      FeePaymentStatus VARCHAR(255),
      CONSTRAINT pk_FeePenalty
           PRIMARY KEY (FeePenaltyID)
);

CREATE TABLE Loan
(
      LoanID         VARCHAR(255) NOT NULL,
      CustomerID      VARCHAR(255),
      LoanType   VARCHAR(255),
      LoanTerm      VARCHAR(255),
      InterestRate  VARCHAR(255),
      LoanStatus VARCHAR(255),
      ApplicationDate VARCHAR(255),
      ReviewDate VARCHAR(255),
      CONSTRAINT pk_Loan
         PRIMARY KEY (LoanID)
);

CREATE TABLE RepaymentSchedule
(
    RepaymentID     VARCHAR(255) NOT NULL,
    LoanID      VARCHAR(255),
    PaymentDueDate     VARCHAR(255),
    PaymentAmounts     VARCHAR(255),
    RemainingBalance    VARCHAR(255),
    LoanPaymentStatus   VARCHAR(255),
```

```
     CONSTRAINT pk_RepaymentSchedule
       PRIMARY KEY (RepaymentID)
);

CREATE TABLE TransactionHistory
(
    TransactionID    VARCHAR(255) NOT NULL,
    AccountID        VARCHAR(255),
    TransactionDate   VARCHAR(255),
    TransactionAmount  VARCHAR(255),
    TransactionType    VARCHAR(255),
    CONSTRAINT TransactionHistory
        PRIMARY KEY (TransactionID)
);
```

The following is the SQL code to insert the data into the table.
*Note this is only some of the code, since we have 100 entries per table. Due to time constraints we were not able to make the code for the 100 of each table.

```
INSERT INTO RepaymentSchedule VALUES
('1','1','2024-07-26','131.00','0.00','paid');

INSERT INTO RepaymentSchedule VALUES
('2','2','2024-06-28','601.00','11.00','paid');

INSERT INTO RepaymentSchedule VALUES
('3','3','2024-06-02','853.00','8374.00','paid');

INSERT INTO RepaymentSchedule VALUES
('4','4','2024-05-24','467.00','1812.00','paid');

INSERT INTO RepaymentSchedule VALUES
('5','5','2024-03-04','923.00','2575.00','paid');

INSERT INTO RepaymentSchedule VALUES
('6','6','2024-05-28','702.00','0.00','due');

INSERT INTO RepaymentSchedule VALUES
('7','7','2024-03-19','267.00','0.00','paid');

INSERT INTO RepaymentSchedule VALUES
('8','8','2024-02-23','412.00','0.00','paid');

INSERT INTO RepaymentSchedule VALUES
('9','9','2024-02-03','124.00','0.00','paid');
```

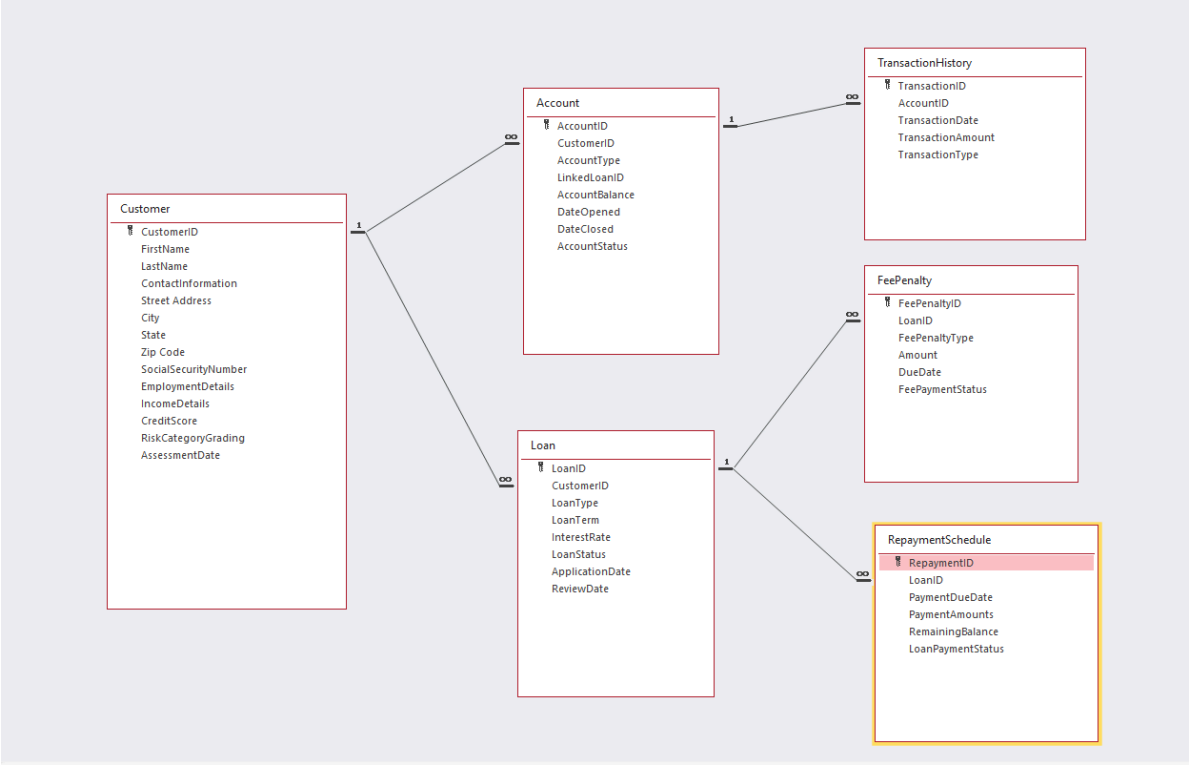
```
INSERT INTO RepaymentSchedule VALUES
('10','10','2024-05-05','624.00','0.00','paid');

INSERT INTO RepaymentSchedule VALUES
('11','11','2024-02-21','436.00','9629.00','due');

INSERT INTO RepaymentSchedule VALUES
('12','12','2024-06-10','107.00','5049.00','paid');

INSERT INTO RepaymentSchedule VALUES
('13','13','2024-06-19','841.00','7542.00','due');

INSERT INTO RepaymentSchedule VALUES
('14','14','2024-06-24','180.00','6737.00','paid');

INSERT INTO RepaymentSchedule VALUES
('15','15','2024-02-11','86.00','0.00','paid');

INSERT INTO RepaymentSchedule VALUES
('16','16','2024-04-09','826.00','9405.00','due');

INSERT INTO RepaymentSchedule VALUES
('17','17','2024-07-15','608.00','4681.00','due');

INSERT INTO RepaymentSchedule VALUES
('18','18','2024-05-16','604.00','4740.00','due');

INSERT INTO RepaymentSchedule VALUES
('19','19','2024-06-20','465.00','2481.00','paid');

INSERT INTO RepaymentSchedule VALUES
('20','20','2024-05-30','108.00','6256.00','paid');

INSERT INTO RepaymentSchedule VALUES
('21','21','2024-06-10','894.00','3298.00','paid');

INSERT INTO RepaymentSchedule VALUES
('22','22','2024-06-18','553.00','1316.00','paid');

INSERT INTO RepaymentSchedule VALUES
('23','23','2024-01-15','608.00','8713.00','due');

INSERT INTO RepaymentSchedule VALUES
('24','24','2024-03-04','54.00','5564.00','paid');

INSERT INTO RepaymentSchedule VALUES
('25','25','2024-07-06','478.00','9607.00','due');
```

```
INSERT INTO RepaymentSchedule VALUES
('26','26','2024-06-19','403.00','5495.00','paid');

INSERT INTO RepaymentSchedule VALUES
('27','27','2024-05-02','8.00','7051.00','due');

INSERT INTO RepaymentSchedule VALUES
('28','28','2024-01-23','644.00','0.00','due');

INSERT INTO RepaymentSchedule VALUES
('29','29','2024-03-16','782.00','833.00','paid');

INSERT INTO RepaymentSchedule VALUES
('30','30','2024-07-03','630.00','2948.00','paid');

INSERT INTO FeePenalty VALUES
('1','1','early','0','2024-02-27','paid');

INSERT INTO FeePenalty VALUES
('2','2','early','0','2024-05-19','paid');

INSERT INTO FeePenalty VALUES
('3','3','overdue','69.00','2024-01-28','paid');

INSERT INTO FeePenalty VALUES
('4','4','early','0','2024-07-18','paid');

INSERT INTO FeePenalty VALUES
('5','5','early','0','2024-03-27','paid');

INSERT INTO FeePenalty VALUES
('6','6','late','0','2024-05-30','due');

INSERT INTO FeePenalty VALUES
('7','7','overdue','19.00','2024-03-16','paid');

INSERT INTO FeePenalty VALUES
('8','8','early','0','2024-05-11','paid');

INSERT INTO FeePenalty VALUES
('9','9','early','0','2024-04-17','paid');

INSERT INTO FeePenalty VALUES
('10','10','early','0','2024-05-28','paid');

INSERT INTO FeePenalty VALUES
('11','11','late','0','2024-06-07','due');
```

```
INSERT INTO FeePenalty VALUES
('12','12','overdue','8.00','2024-05-26','paid');

INSERT INTO FeePenalty VALUES
('13','13','late','0','2024-05-02','due');

INSERT INTO FeePenalty VALUES
('14','14','early','0','2024-02-12','paid');

INSERT INTO FeePenalty VALUES
('15','15','early','0','2024-02-27','paid');

INSERT INTO FeePenalty VALUES
('16','16','late','0','2024-08-03','due');

INSERT INTO FeePenalty VALUES
('17','17','late','0','2024-06-01','due');

INSERT INTO FeePenalty VALUES
('18','18','late','0','2024-02-08','due');

INSERT INTO FeePenalty VALUES
('19','19','overdue','57.00','2024-01-17','paid');

INSERT INTO FeePenalty VALUES
('20','20','overdue','83.00','2024-05-15','paid');

INSERT INTO FeePenalty VALUES
('21','21','early','0','2024-04-24','paid');

INSERT INTO FeePenalty VALUES
('22','22','early','0','2024-02-16','paid');

INSERT INTO FeePenalty VALUES
('23','23','overdue','4.00','2024-02-15','due');

INSERT INTO FeePenalty VALUES
('24','24','early','0','2024-06-08','paid');

INSERT INTO FeePenalty VALUES
('25','25','late','0','2024-05-28','due');

INSERT INTO FeePenalty VALUES
('26','26','early','0','2024-01-14','paid');

INSERT INTO FeePenalty VALUES
('27','27','overdue','56.00','2024-02-20','due');
```

```
INSERT INTO FeePenalty VALUES
('28','28','overdue','47.00','2024-05-13','due');

INSERT INTO FeePenalty VALUES
('29','29','overdue','33.00','2024-06-15','paid');

INSERT INTO FeePenalty VALUES
('30','30','early','0','2024-01-12','paid');
```

Relationship View



Making Queries with SQL

```
UPDATE RepaymentSchedule
SET LoanPaymentStatus = 'completed'
WHERE RemainingBalance = 0;
```

RepaymentI	LoanID	PaymentDue	PaymentAm	RemainingB	LoanPaymer	Click to Add
S	1	2024-07-26	131.00	0.00	completed	
2	2	2024-06-28	601.00	11.00	paid	
3	3	2024-06-02	853.00	8374.00	paid	
4	4	2024-05-24	467.00	1812.00	paid	
5	5	2024-03-04	923.00	2575.00	paid	
6	6	2024-05-28	702.00	0.00	completed	
7	7	2024-03-19	267.00	0.00	completed	
8	8	2024-02-23	412.00	0.00	completed	
9	9	2024-02-03	124.00	0.00	completed	
10	10	2024-05-05	624.00	0.00	completed	

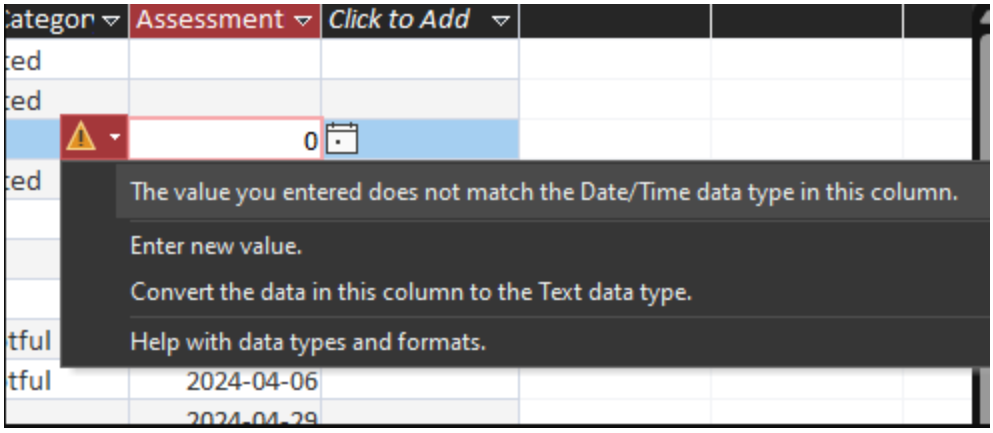
This Query marks all loan payments as “completed” in the RepaymentSchedule table for entries where the RemainingBalance is 0 indicating that the loan has been fully paid off. This ensures that only fully settled loans are marked as completed in the database

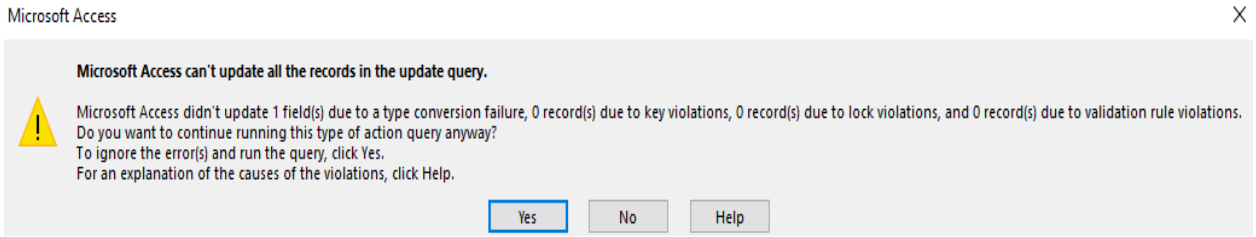
```
UPDATE Customer
SET
    FirstName = 'deleted',
    LastName = 'deleted',
    ContactInformation = 'deleted',
    [Street Address] = 'deleted',
    City = 'deleted',
    State = 'deleted',
    [Zip Code] = 'deleted',
    EmploymentDetails = 'deleted',
    IncomeDetails = 0,
    CreditScore = 0,
    RiskCategoryGrading = 'deleted',
    AssessmentDate = #1900-01-01#
WHERE CustomerID = [EnterCustomerID];
```

Customer

CustomerID	FirstName	LastName	ContactInfor	Street Addre	City	State	Zip Code	SocialSecuri	Employmen	IncomeDeta	CreditScore	RiskCategor	Assessment
1	deleted	deleted	deleted	deleted	deleted	deleted	deleted	868-12-5031	deleted	0	0	deleted	1900-01-01
2	deleted	deleted	deleted	deleted	deleted	deleted	deleted	875-04-8670	deleted	0	0	deleted	1900-01-01
3	Daniel	Stone	cannonkaren@	67973 David Sq	Carrollville	CT	00724	353-83-0805	Gaffer	$139642	580	pass	2024-06-28
4	deleted	deleted	deleted	deleted	deleted	deleted	deleted	719-64-0365	deleted	0	0	deleted	1900-01-01
5	Anna	Cisneros	luis75@housto	7207 Jose Way	North Mason	NY	68873	436-74-5309	Comptroller	$47803	740	pass	2024-06-09
6	deleted	deleted	deleted	deleted	deleted	deleted	deleted	172-53-1726	deleted	0	0	deleted	1900-01-01

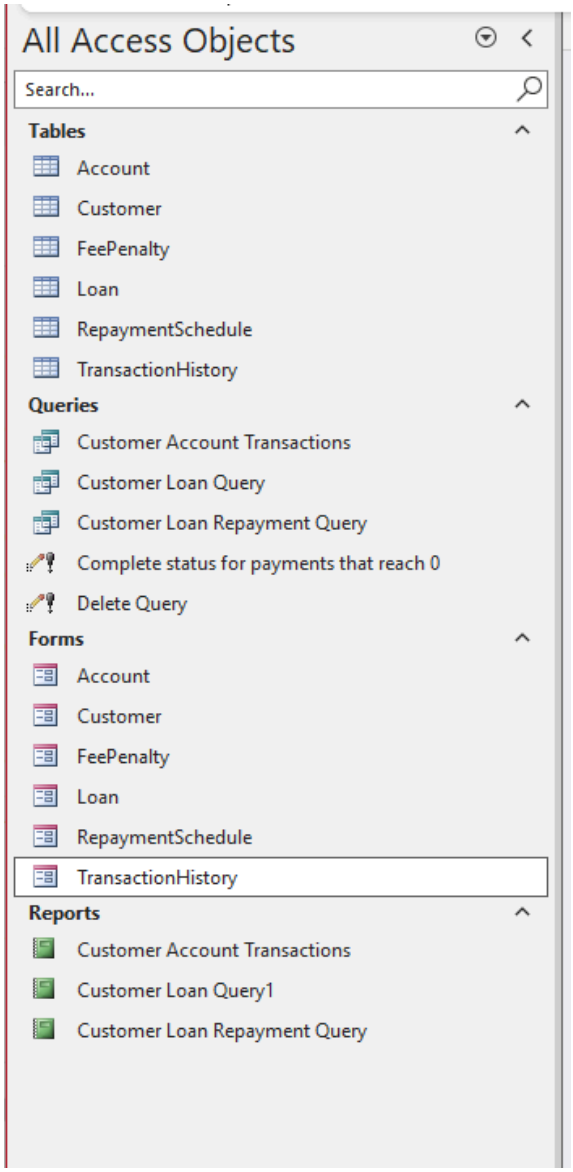
This query updates specific fields in the Customer table for customers identified by CustomerID. It replaces personal information such as the first name, last name, contact details, address, and risk category with the “deleted”. While creating the query we encounter an issue with certain columns requiring numerical values or date format so we created 0 and 1900-01-01 as the default deleted option





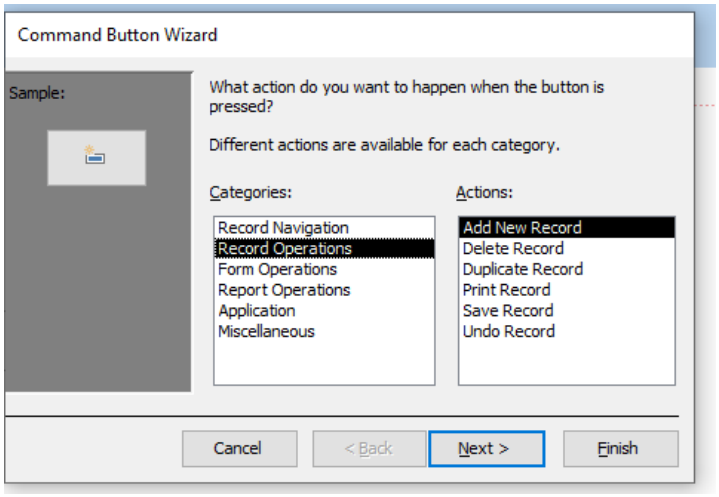
VI. Database Application

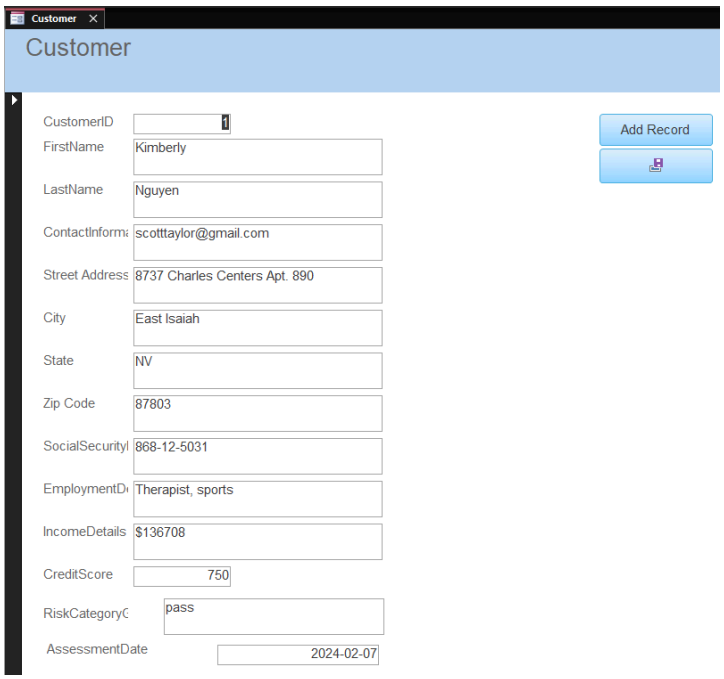
Navigation Form



All the tables, queries, forms and reports can be seen on the left- hand side of the database. Above is the final navigation form. Next we move on to making all the forms.

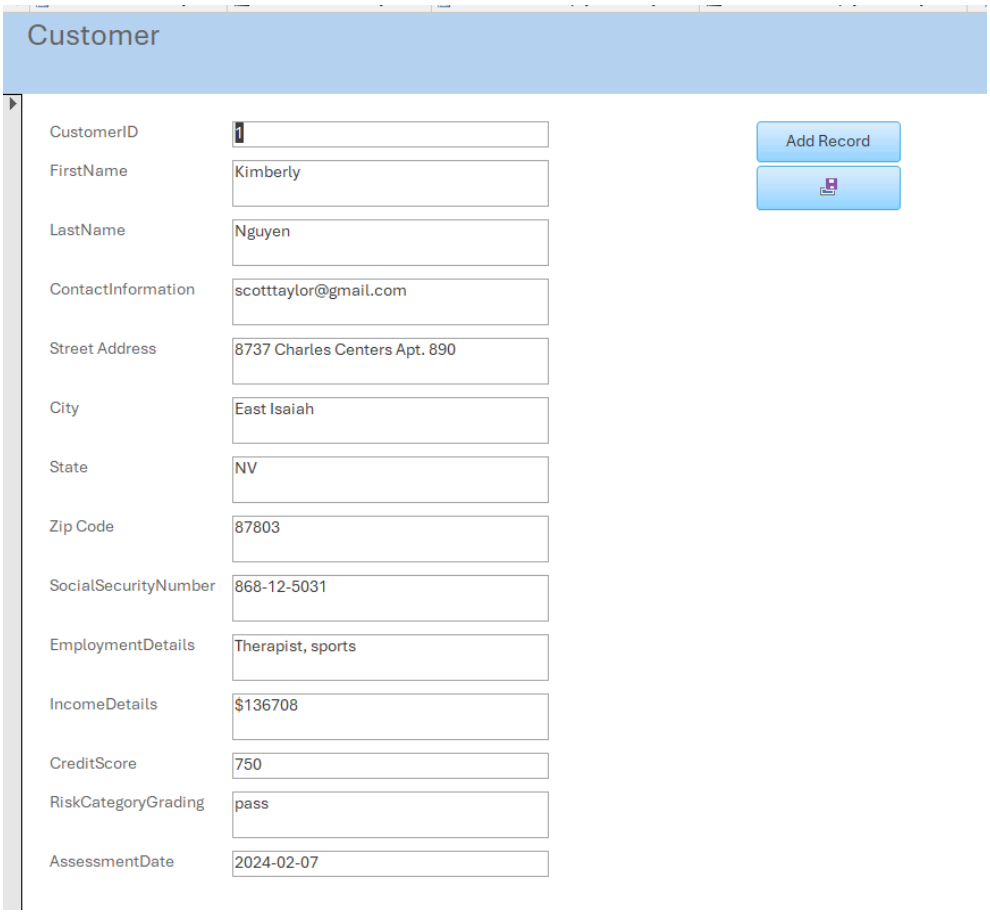
We proceeded to create a form using the Form Wizard. We selected the relevant fields from tables and then utilized the command button wizard to add functionality to our form. Next, we chose to add a button for record operation, specifically the “Add New Record” action. This allows users to easily add new repayment records through the form interface.





We adjusted the size of the text boxes and labels to ensure that they fit within the designated space without overlapping. This involved resizing and repositioning the controls to create a clear and organized layout

Customer Data Entry Form

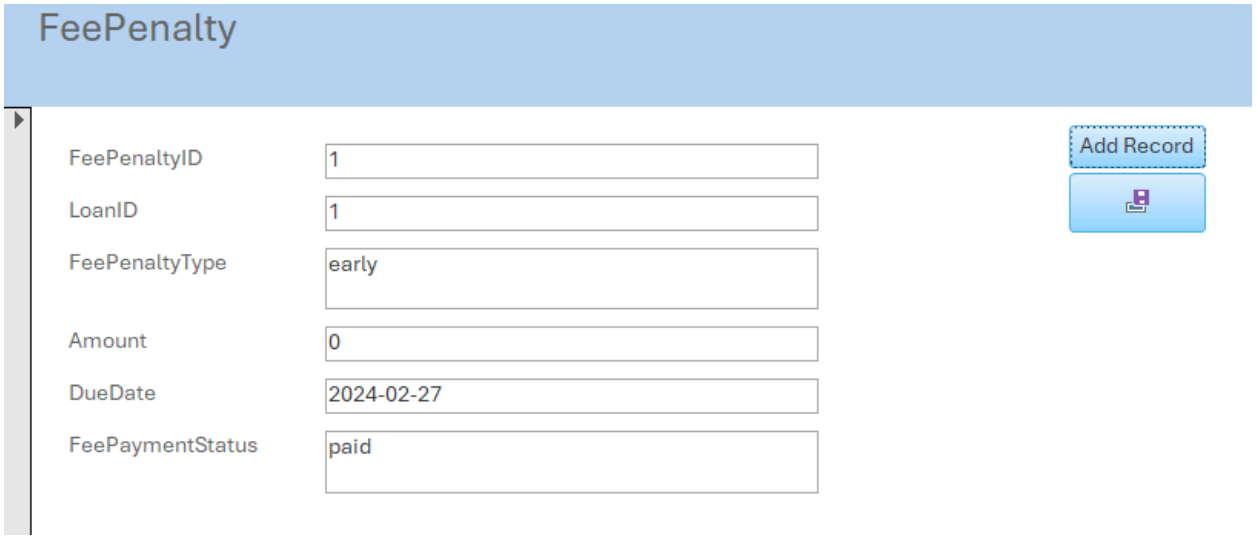


The Customer Data Entry form is used to look up existing customers. We can also add new customers, as well edit existing customer information and save the information.

Account Data Entry Form

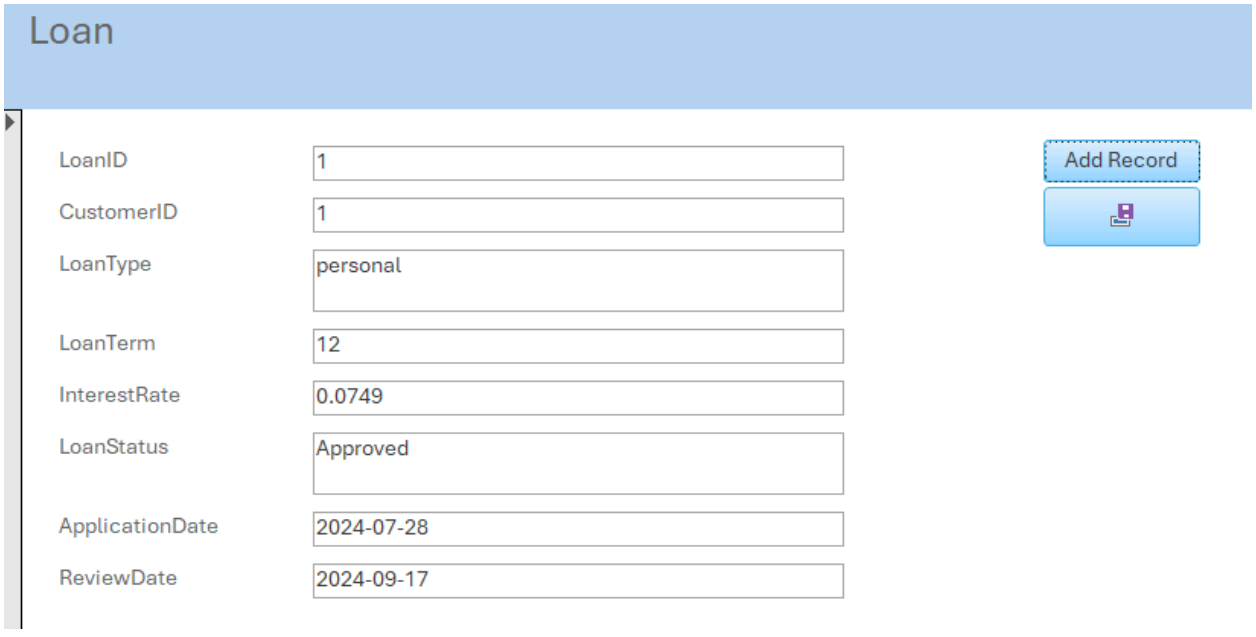
The Account Data Entry form can be used to update, edit and add accounts into the database.

FeePenalty Data Entry Form



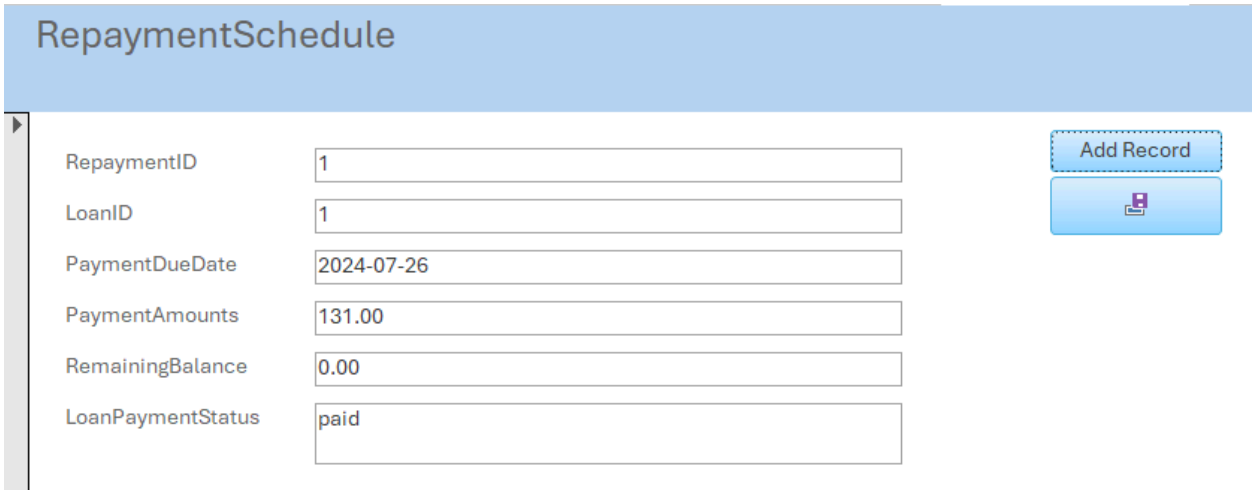
The Fee Penalty Data Entry form is used to query, update, or add a new fee penalty into the database.

Loan Data Entry Form



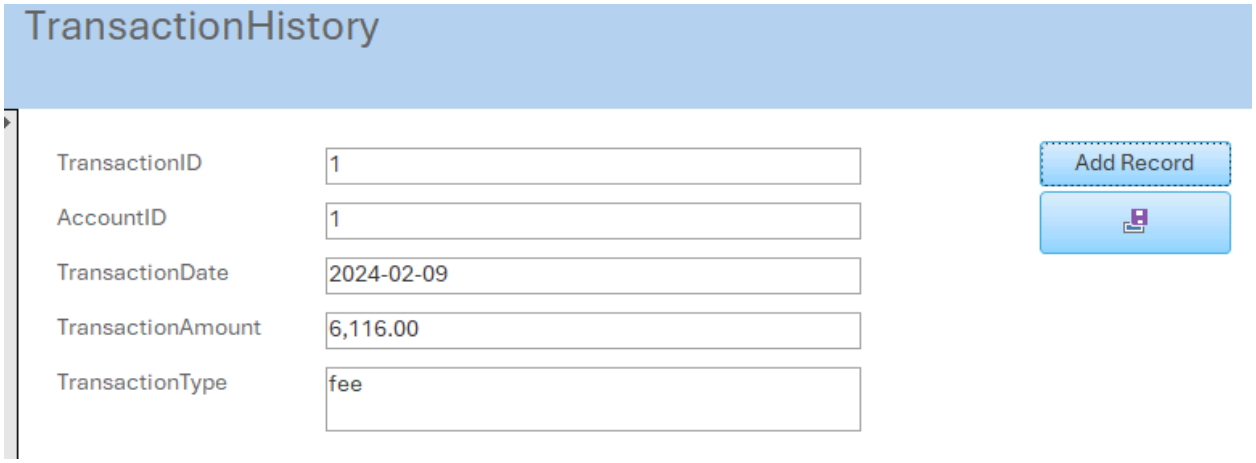
The Loan Data Entry form is used to query, update, or add a new loan into the database.

Repayment Schedule Data Entry Form



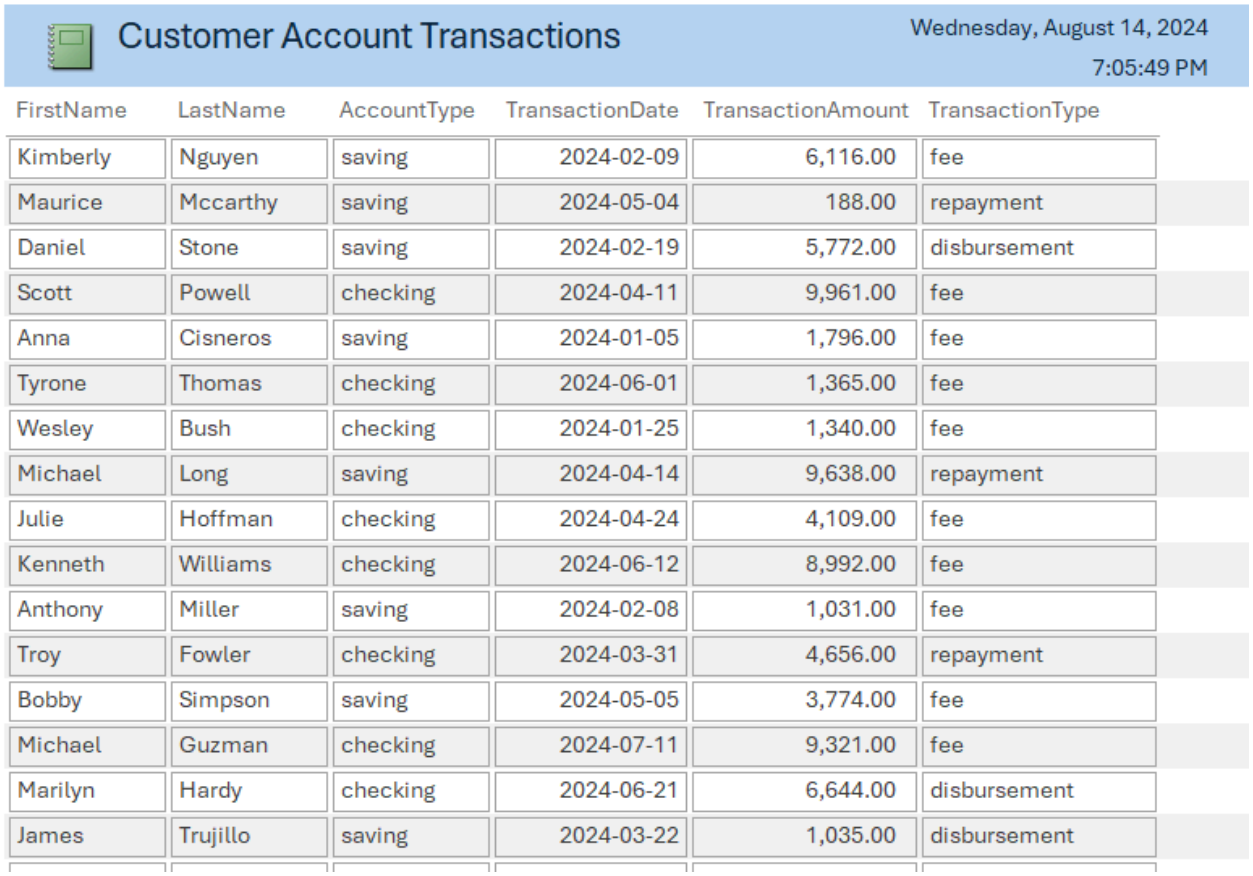
The Repayment Data Entry form is used to query, update, or add a new loan into the repayment schedule table.

Transaction History Data Entry Form



The Transaction History Data Entry form is used to query, update, or add a new loan into the transaction history table.

Customer Account Transactions Report



Customer Account Transactions

Wednesday, August 14, 2024
7:05:49 PM

FirstName	LastName	AccountType	TransactionDate	TransactionAmount	TransactionType
Kimberly	Nguyen	saving	2024-02-09	6,116.00	fee
Maurice	Mccarthy	saving	2024-05-04	188.00	repayment
Daniel	Stone	saving	2024-02-19	5,772.00	disbursement
Scott	Powell	checking	2024-04-11	9,961.00	fee
Anna	Cisneros	saving	2024-01-05	1,796.00	fee
Tyrone	Thomas	checking	2024-06-01	1,365.00	fee
Wesley	Bush	checking	2024-01-25	1,340.00	fee
Michael	Long	saving	2024-04-14	9,638.00	repayment
Julie	Hoffman	checking	2024-04-24	4,109.00	fee
Kenneth	Williams	checking	2024-06-12	8,992.00	fee
Anthony	Miller	saving	2024-02-08	1,031.00	fee
Troy	Fowler	checking	2024-03-31	4,656.00	repayment
Bobby	Simpson	saving	2024-05-05	3,774.00	fee
Michael	Guzman	checking	2024-07-11	9,321.00	fee
Marilyn	Hardy	checking	2024-06-21	6,644.00	disbursement
James	Trujillo	saving	2024-03-22	1,035.00	disbursement

This report shows all the customers (First and Last name), their account type, the date and amount of the transaction. It also shows what kind of transaction. This report allows us to see all the information together all in one print page instead of separately.

The report was based on the following query:

```
SELECT Customer.FirstName, Customer.LastName,
Account.AccountType, TransactionHistory.TransactionDate,
TransactionHistory.TransactionAmount,
TransactionHistory.TransactionType
FROM (Customer INNER JOIN Account ON Customer.[CustomerID] =
Account.[CustomerID]) INNER JOIN TransactionHistory ON
Account.[AccountID] = TransactionHistory.[AccountID];
```

Customer Loan Information Report

Customer Loan Information

Wednesday, August 14, 2024
8:12:01 PM

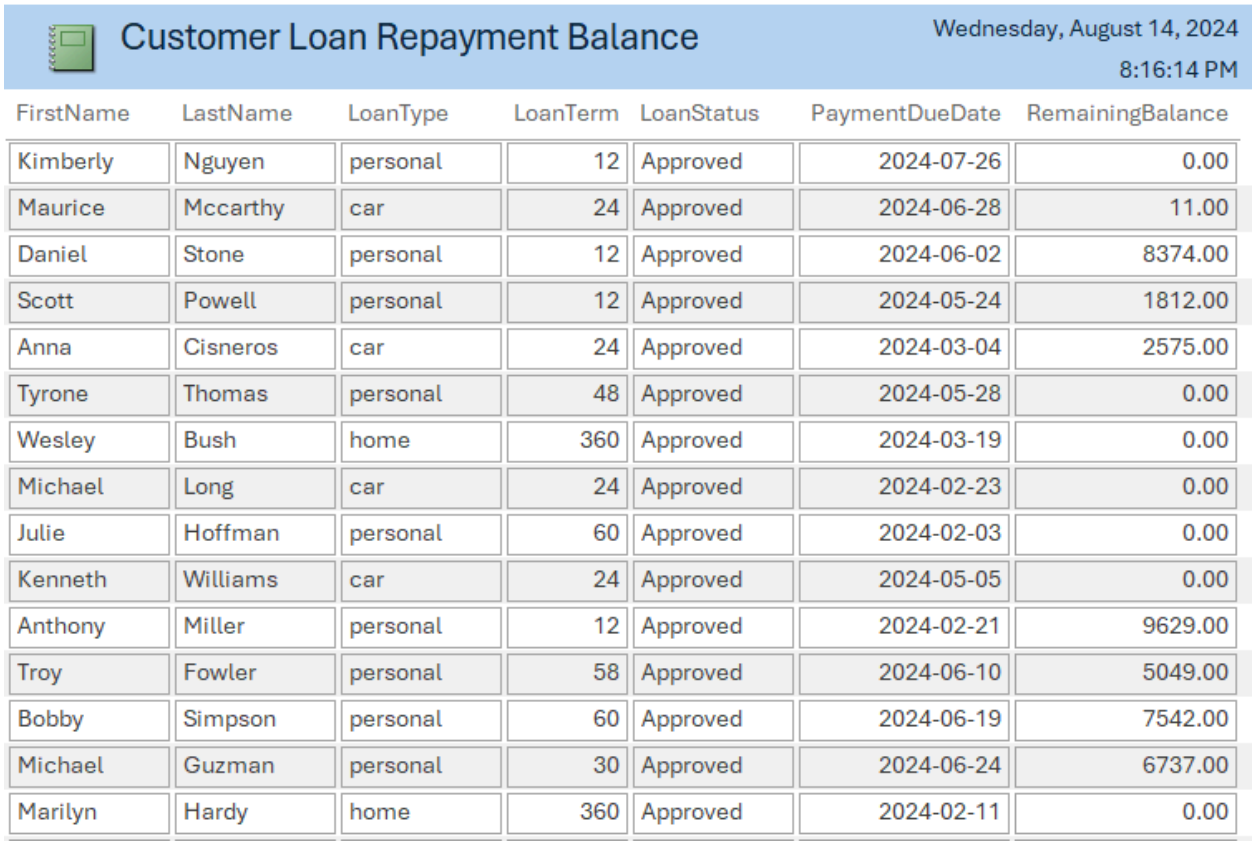
FirstName	LastName	ContactInformation	LoanType	LoanTerm	InterestRate	LoanStatus
Kimberly	Nguyen	scotttaylor@gmail.com	personal	12	0.0749	Approved
Maurice	Mccarthy	glee@smith.org	car	24	0.1478	Approved
Daniel	Stone	cannonkaren@miller.net	personal	12	0.0874	Approved
Scott	Powell	sean79@gmail.com	personal	12	0.0874	Approved
Anna	Cisneros	luis75@houston.com	car	24	0.0564	Approved
Tyrone	Thomas	mariamendez@jones-trujillo.info	personal	48	0.0749	Approved
Wesley	Bush	reevesangela@cameron.com	home	360	0.0638	Approved
Michael	Long	pnelson@martinez.com	car	24	0.0875	Approved
Julie	Hoffman	gary18@ward.org	personal	60	0.0749	Approved
Kenneth	Williams	pkelly@king.com	car	24	0.0564	Approved
Anthony	Miller	larry22@hernandez.info	personal	12	0.0689	Approved
Troy	Fowler	jesseparker@gmail.com	personal	58	0.0689	Approved
Bobby	Simpson	imaldonado@gmail.com	personal	60	0.0749	Approved
Michael	Guzman	kstout@hotmail.com	personal	30	0.0689	Approved
Marilyn	Hardy	martinezcharles@hotmail.com	home	360	0.0619	Approved

This report shows all the customers (First and Last name), their contact information. It also shows the loan type, loan term, interest rate, and the status of the loan for each customer. This report allows us to see all the information together all in one print page instead of separately, for easy access.

The report was based on the following query:

```
SELECT Customer.FirstName, Customer.LastName,
Customer.ContactInformation, Loan.LoanType, Loan.LoanTerm,
Loan.InterestRate, Loan.LoanStatus
FROM Customer INNER JOIN Loan ON Customer.[CustomerID] =
Loan.[CustomerID];
```

Customer Loan Repayment Balance Report



Customer Loan Repayment Balance

Wednesday, August 14, 2024
8:16:14 PM

FirstName	LastName	LoanType	LoanTerm	LoanStatus	PaymentDueDate	RemainingBalance
Kimberly	Nguyen	personal	12	Approved	2024-07-26	0.00
Maurice	Mccarthy	car	24	Approved	2024-06-28	11.00
Daniel	Stone	personal	12	Approved	2024-06-02	8374.00
Scott	Powell	personal	12	Approved	2024-05-24	1812.00
Anna	Cisneros	car	24	Approved	2024-03-04	2575.00
Tyrone	Thomas	personal	48	Approved	2024-05-28	0.00
Wesley	Bush	home	360	Approved	2024-03-19	0.00
Michael	Long	car	24	Approved	2024-02-23	0.00
Julie	Hoffman	personal	60	Approved	2024-02-03	0.00
Kenneth	Williams	car	24	Approved	2024-05-05	0.00
Anthony	Miller	personal	12	Approved	2024-02-21	9629.00
Troy	Fowler	personal	58	Approved	2024-06-10	5049.00
Bobby	Simpson	personal	60	Approved	2024-06-19	7542.00
Michael	Guzman	personal	30	Approved	2024-06-24	6737.00
Marilyn	Hardy	home	360	Approved	2024-02-11	0.00

This report shows all the customers (First and Last name) who have an active loan. It provides the loan type, loan term and the status of the loan. It also shows when their payment is due and how much balance is remaining in the loan. This report allows us to see all the information together all in one print page instead of separately, for easy access.

The report was based on the following query:

```
SELECT Customer.FirstName, Customer.LastName, Loan.LoanType,
Loan.LoanTerm, Loan.LoanStatus,
RepaymentSchedule.PaymentDueDate,
RepaymentSchedule.RemainingBalance
FROM (Customer INNER JOIN Loan ON Customer.[CustomerID] =
Loan.[CustomerID]) INNER JOIN RepaymentSchedule ON Loan.[LoanID]
= RepaymentSchedule.[LoanID];
```

VII. Conclusions

The development of our banking database application has followed a systematic and structured approach, aligning with the typical system development life cycle. Beginning with the **Business Scenario**, we identified the need to replace outdated and inefficient manual processes with a modern database system to enhance accuracy, security, and operational efficiency. This was followed by creating an **ER Model Using UML Notations**, which provided a clear conceptual representation of the key entities and relationships within our system.

We then moved on to the **Conversion to Relational Model**, where we translated the conceptual model into a structured set of relations, ensuring that each entity was properly normalized to reduce redundancy and improve data integrity. During the **Normalization** phase, we encountered and resolved specific issues, such as data formatting errors, which were crucial in ensuring that our database adhered to best practices and standards. **The Creating the Database Schema With SQL** step involved implementing the logical model into a physical database structure using SQL, where we defined tables, relationships, and constraints to enforce data integrity.

Finally, in the **Database Application** phase, we developed the user interface, including forms and reports, to facilitate easy interaction with the database. While the project has successfully addressed the primary objectives, there are still areas that could benefit from further refinement. For instance, additional normalization might be needed to handle complex transactions more efficiently, and more detailed documentation of the application's features and functionality could improve usability. This project provides a strong foundation for future enhancements, ensuring that the system remains scalable and adaptable.