**Group-13:** **Integrated Banking Information System (IBIS)**

Durga Prasanna Basireddy (Team Coordinator)

Pavan Kumar Reddy Bhumireddy

Roshan Dappu

Harshitha Devabhaktuni

Prakashraj Javvaji

**INFO 5707 Data Modelling**

Dr. Tozammel Hossain

**Contents**

Project Description 3

Objectives 3

Scope 4

Project Requirements 4

Database Requirements 4

User Requirements 5

Business Rules 6

Entity Relationship Diagram 7

Data Dictionary 8

Queries and Operations

Section 1: Entity Generation and Data Entry 10-25

Section 2: Data Retrieval and Analytical Reports 26-32

Appendix 1 33

Integrated Banking Information System (IBIS)

## Project Description

The Integrated Banking Information System (IBIS) was developed to improve the way banking works more effectively and productively. The design is a complete, safe, and extensible system. This serves as an extremely helpful instrument for banking. This can be used to manage money, make transactions, and repay loans, among other functions. The business known as IBIS, ensures sure that all deals are fair. This additionally makes clear that arrangements can be modified to comply with the regulations of various nations. It is designed for handling both easy and hard cash duties.

Protection and the customer journey are essential to IBIS. The organization performs this by offering customers a set of personal banking features and improved protection of data. Bankers can easily talk with customers, identify hazards, and send in reports on compliance with this application. IBIS wants to help banks identify fraudulent activities, manage hazards, and maintain a step ahead of the competition in the rapid-fire financial sector by improving and streamlining processes.

## Objectives:

* To develop one application for managing all banking assignments, which include loans, accounts for consumers, payment transfers, and employee records.
* With continuous monitoring, you can make sure that addresses are handled securely and efficiently.
* Analysis of data may be utilized to look at economic patterns, measure potential hazards and provide better service to customers.
* Better awareness of requirements by using automatic check records and findings.
* To provide a uniform customer experience throughout all banking methods, which includes the Internet, mobile devices, and in-branch operations.
* Establish machine learning algorithms to identify and block fraudulent transactions and to review a customer's reputation.
* Support management of several banking accounts, offering customers and bank managers an integrated overview of resources.
* Connect to foreign banking systems to make it simple to perform business as well as exchange funds across boundaries.
* Prepare an atmosphere for the digital transformation of banking institutions so that fresh financial products and services can be launched.

## Scope:

* The Integrated Banking Information System will manage all the banking responsibilities and become the primary centre for both the front office (which deals with customers) and the back office (which operates the business).
* This will be utilized by every department in the bank and fulfil the requirements in various areas, like funds, loans, regulation, and employees.
* The design will have the capacity to manage plenty of transactions including complex financial products, as well as various currencies and regulations.
* To maintain customer trust and protect confidential financial information, modern safety measures will be important.
* Bank Sphere will continue to provide additional features for individuals in retail banking, business banking, economic banking, and asset management.
* This will be able to establish connections to exterior banking systems like gateways for payments, stock exchanges, and credit report agencies.
* The platform will include a great deal of information storage facility features that will allow users to review historical information, develop forecasts, and prepare effectively.
* The design will also make it simpler for people who have disabilities to access as well as make sure it satisfies worldwide accessibility instructions.
* As the bank's customer base and the number of transactions expand, the system will be designed to be flexible, which means it can manage them.
* The bank's foreign clients and their various needs will profit from soft cross-border trade and following financial rules using connection with global financial and banking systems.

## Project Requirements

**Operating System:** Windows

**Database:** MySQL

**Applications:** MYSQL Workbench, Lucidchart, Microsoft Word

## Database Requirements

Tables:

1. Customer Table

2. Employees Table

3. Accounts Table

4. User sessions Table

5. Transactions Table

6. Loan Table

7. Payment Table

8. Report Details Table

## User requirements:

* Each customer will be given a unique ID number and will be able to safely access information about their accounts, transactions, and loans.
* The employees who work in banks can manage deals and documentation, maintain records of customer information, and prepare financial statements.
* Customers will be able to submit requests for loans and check on the advancement of their requests through the system.
* Loan supervisors will be ready to perform checks for creditworthiness, make interest rates, and maintain a watchful eye on the payment of loans.
* Automatic notifications for purchases which appear to be correct to stop fraudulent activities while maintaining the funds safe.
* The approach will be utilized by accountability employees to generate the findings that government groups need and to make sure the bank follows financial rules.
* Customers will be able to do their banking on the system, which includes beginning fresh accounts and keeping track of their money saved.
* Managers of relationships will be able to give personalized banking tips based on customer data analytics thanks to Bank Sphere.
* The approach allows rates of interest and fees charged by banks to change immediately based on the way the economy performs along with how the bank runs.
* The technique will be utilized by internal examiners for continual monitoring and evaluating danger in real-time.
* The system's architecture will deal with various languages so it may operate in a variety of places and with an extensive number of customers.
* Make out encrypted messages and recognize methods to ensure customers and bank employees can speak with one another straight.
* Ensure that the platform includes an interface to feed customers to raise difficulties and recommend approaches to improve it.
* Provide a comprehensive instructional resource within the system for new customers and employees.
* The system will have an effective help and services component that responds rapidly to client inquiries and worries, ensuring customer satisfaction.

## Business Rules:

1. For customers to have access to their banking information, each customer needs a unique ID.
2. The primary objective of employees is to manage communications with customers, maintain information, and prepare reports on finances.
3. Customers need to be ready to make applications for loans and report on the status of their queries via the system.
4. Loan agents review people's finances, create rates of interest, as well as make sure they pay to return the loans on time.
5. For the sake of safety reasons, transactions of more than a certain amount should generate alerts to stop fraud.
6. Compliance employees utilize the technique to prepare reports for government agencies and make sure that financial regulations are obeyed.
7. The system requires us to be ready to create and manage new customer accounts as well as keep up to date on existing ones.
8. customers' personal banking advice and relationship supervisors are required to utilize customer information analytics.
9. The system must be able to modify interest rates and bank charges depending on how the market is performing and how banks are operating.
10. The procedure must be utilized by internal officials to maintain an eye on activities and determine problems constantly.
11. The system is required to be ready to function in many various locations and languages.
12. Every interaction between customers and the system requires to be recorded to ensure that they can be traced and verified.
13. The system must be linked to foreign banking systems as well as provide a secure means for people to conduct operations with one another globally.
14. The most up-to-date safety precautions must be utilized to keep all confidential customer and bank data safe.
15. The structure of the system must be ready to expand as the total number of users and interactions occurs.
16. Global requirements for accessibility indicate that user platforms need to be able to be utilized by people with challenges.
17. For modern financial products and services to eventually become available, the system must make it simple for banking institutions to become digital.
18. To identify and prevent fake operations, methods of machine learning must be utilized.
19. The program needs to be available to keep a great deal of data to ensure knowledge is useful for forecasting and evaluating recent events.
20. For a uniform customer experience, each of the banking activities needs to be performed by one user platform.

## Entity Relationship Diagram:

A diagram of a company

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity | Relationship | Connectivity | Related Entity | Entity |
| CUSTOMER | Has | 1:M | PAYMENT | CUSTOMER |
| CUSTOMER | Has | 1:M | LOAN | CUSTOMER |
| CUSTOMER | Has | 1:M | ACCOUNTS | CUSTOMER |
| CUSTOMER | Initiates | 1:M | TRANSACTIONS | CUSTOMER |
| EMPLOYEES | Manages | 1:M | LOAN | EMPLOYEES |
| EMPLOYEES | Conducts | 1:M | TRANSACTIONS | EMPLOYEES |
| EMPLOYEES | Logs Into | 1:M | USER\_SESSIONS | EMPLOYEES |
| ACCOUNTS | Recorded in | 1:M | REPORT\_DETAILS | ACCOUNTS |
| LOAN | Repaid by | 1:M | PAYMENT | LOAN |
| TRANSACTIONS | Recorded in | 1:M | REPORT\_DETAILS | TRANSACTIONS |
| USER\_SESSIONS | Tracked in | 1:M | REPORT\_DETAILS | USER\_SESSIONS |

**Data Dictionary**

During the procedure of constructing the Integrated Banking Information System (IBIS), the data dictionary serves as a crucial component. It serves as a complete catalogue, providing information on the features and relationships of the data included within the system. Clear communication between participants can be made easier by the data dictionary, as well as guarantees that data management procedures are followed to and that consistency is preserved. This is achieved by regulating the meaning of data objects.

**Key Concepts of the IBIS Data Dictionary:**

* Ensures that all members of the financial system have a consistent interpretation of the data.
* The data rules and formats are documented to ensure clarity and uniform functioning.
* This framework offers support for collaborative endeavours by offering a consistent data structure.
* Establishes governance for the use of data inside the context of the IBIS tool.
* provides a fast reference to facilitate the making of well-informed decisions and the planning of strategic actions.

**Data Dictionary:**





**Queries and Operations**

**Section 1: Entity Generation and Data Entry: Create Database, Create Tables, and Insert Values to the Tables**

**Overview:**

The Integrated Banking Information System (IBIS) is an advanced system that will transform the way banking run their companies, communicate with consumers, and process back-end responsibilities. With an emphasis on efficiency and precision, IBIS mixes core banking duties like maintaining accounts, establishing loans, and handling operations into one, simple, and smooth system. The solid database represents the place where all banking measures interact, making sure that all records are stored confidentially and securely.   
  
IBIS makes customers more interested in banking by offering these individuals unique experiences and makes work easier for staff members by streamlining typical duties. This not only helps service better, but it additionally provides financial institutions with the resources required for dealing with the complicated legal environment and changes in the market's wants. IBIS helps businesses create smart choices and promotes a proactive approach to creativity by employing real-time data analysis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Database | Table Name | Table Explanation | Records | Insert Order |
| CustomerManagement | CUSTOMER | Stores customer details like name, address, contact, etc. | 30 | 1 |
| EmployeeAndTransaction | EMPLOYEES | Holds information on employees including contact details. | 30 | 2 |
| CustomerManagement | ACCOUNTS | Stores account details linked to customers. | 30 | 3 |
| CustomerManagement | USER\_SESSIONS | Tracks user login and logout sessions. | 30 | 4 |
| EmployeeAndTransaction | TRANSACTIONS | Records details of financial transactions. | 30 | 5 |
| LoanManagement | LOAN | Details of loans taken out by customers, with terms. | 30 | 6 |
| CustomerManagement | PAYMENT | Records payments made towards loans. | 30 | 7 |
| EmployeeAndTransaction | REPORT\_DETAILS | Stores reports on accounts, transactions, and sessions. | 30 | 8 |

**Query Steps and Explanations:**

|  |  |  |
| --- | --- | --- |
| **Steps** | **Query** | **Example** |
| First, the database is created. | CREATE DATABASE DatabaseName; | CREATE DATABASE CustomerManagement; |
| Second, the table is created. | USE DatabaseName; CREATE TABLE TableName (Column1 datatype, Column2 datatype, ...); | USE CustomerManagement; CREATE TABLE CUSTOMER ( Cust\_ID INT PRIMARY KEY,  First\_Name VARCHAR(50), ...); |
| Third, the data is inserted into the table. | INSERT INTO DatabaseName.TableName (Column1, Column2, ...) VALUES (Value1, Value2, ...); | INSERT INTO CUSTOMER (Cust\_ID, First\_Name, Last\_Name,...) VALUES (1, 'Aarav', 'Kumar', ...); |
| Last, retrieving the inserted values to the table. | SELECT \* FROM DatabaseName.TableName; | SELECT \* FROM CustomerManagement.CUSTOMER; |

**Creation of Databases:**

CREATE DATABASE CustomerManagement;

CREATE DATABASE EmployeeAndTransaction;

CREATE DATABASE LoanManagement;

1. CustomerManagement Database:
   * **CUSTOMER**
   * **ACCOUNTS**
   * **USER\_SESSIONS**
2. LoanManagement Database:
   * **LOAN**
   * **PAYMENT**
3. EmployeeAndTransaction Database:
   * **EMPLOYEES**
   * **TRANSACTIONS**
   * **REPORT\_DETAILS**

**Tables:**

1. **CUSTOMER:** Contains the personal and login details of customers.

* First, we ensure that we are using the CustomerManagement database with "USE CustomerManagement".
* Second, we create the CUSTOMER table using "CREATE TABLE CUSTOMER".
* Third, we populate CUSTOMER with customer data via "INSERT INTO CUSTOMER".
* Finally, we display all customer records using "SELECT \* from CustomerManagement.CUSTOMER".

**-- Creation of Customer Table**

CREATE TABLE CUSTOMER (

Cust\_ID INT PRIMARY KEY,

First\_Name VARCHAR(50),

Last\_Name VARCHAR(50),

Address VARCHAR(250),

Contact\_Details VARCHAR(50),

User\_Name VARCHAR(50),

Password VARCHAR(50)

);

**-- Insert Customers**

INSERT INTO CUSTOMER (Cust\_ID, First\_Name, Last\_Name, Address, Contact\_Details, User\_Name, Password) VALUES

(1, 'Aarav', 'Kumar', '1234 Elm St, Bangalore', '9900112233', 'aaravk', 'pass123'),

(2, 'Vivaan', 'Singh', '2345 Maple St, Delhi', '9900223344', 'vivaans', 'pass234'),

(3, 'Aditya', 'Sharma', '3456 Oak St, Chennai', '9900334455', 'adityas', 'pass345'),

(4, 'Vihaan', 'Dutta', '4567 Pine St, Kolkata', '9900445566', 'vihaand', 'pass456'),

(5, 'Arjun', 'Roy', '5678 Cedar St, Mumbai', '9900556677', 'arjunr', 'pass567'),

(6, 'Ananya', 'Bose', '6789 Fir St, Jaipur', '9900667788', 'ananyab', 'pass678'),

(7, 'Ishaan', 'Mishra', '7890 Spruce St, Lucknow', '9900778899', 'ishaanm', 'pass789'),

(8, 'Pihu', 'Agarwal', '8901 Cedar St, Patna', '9900889900', 'pihua', 'pass890'),

(9, 'Riya', 'Khan', '9012 Pine St, Bhopal', '9900990011', 'riyak', 'pass901'),

(10, 'Aadhya', 'Patel', '0123 Maple St, Chandigarh', '9900101122', 'aadhya', 'pass012'),

(11, 'Vihaan', 'Chopra', '1234 Oak St, Gurgaon', '9900112233', 'vihaanc', 'pass123'),

(12, 'Manav', 'Jain', '2345 Birch St, Shimla', '9900223344', 'manavj', 'pass234'),

(13, 'Navya', 'Gupta', '3456 Redwood St, Srinagar', '9900334455', 'navyag', 'pass345'),

(14, 'Aarush', 'Sharma', '4567 Elm St, Ranchi', '9900445566', 'aarushs', 'pass456'),

(15, 'Advik', 'Sen', '5678 Maple St, Cochin', '9900556677', 'adviks', 'pass567'),

(16, 'Tanvi', 'Kumar', '6789 Oak St, Pune', '9900667788', 'tanvik', 'pass678'),

(17, 'Reyansh', 'Singh', '7890 Birch St, Indore', '9900778899', 'reyanshs', 'pass789'),

(18, 'Zara', 'Dutta', '8901 Cedar St, Nagpur', '9900889900', 'zarad', 'pass890'),

(19, 'Sara', 'Roy', '9012 Pine St, Bhubaneswar', '9900990011', 'sarar', 'pass901'),

(20, 'Yash', 'Bose', '0123 Fir St, Ahmedabad', '9900101122', 'yashb', 'pass012'),

(21, 'Siya', 'Mishra', '1234 Spruce St, Surat', '9900112233', 'siyam', 'pass123'),

(22, 'Arnav', 'Agarwal', '2345 Cedar St, Visakhapatnam', '9900223344', 'arnava', 'pass234'),

(23, 'Diya', 'Khan', '3456 Pine St, Mysore', '9900334455', 'diyak', 'pass345'),

(24, 'Nikhil', 'Patel', '4567 Maple St, Jodhpur', '9900445566', 'nikhilp', 'pass456'),

(25, 'Sanya', 'Chopra', '5678 Oak St, Vadodara', '9900556677', 'sanyac', 'pass567'),

(26, 'Kabir', 'Jain', '6789 Birch St, Amritsar', '9900667788', 'kabirj', 'pass678'),

(27, 'Anvi', 'Gupta', '7890 Redwood St, Kanpur', '9900778899', 'anvig', 'pass789'),

(28, 'Ishaan', 'Sharma', '8901 Elm St, Thane', '9900889900', 'ishaans', 'pass890'),

(29, 'Mira', 'Sen', '9012 Maple St, Jamshedpur', '9900990011', 'miras', 'pass901'),

(30, 'Rohan', 'Kumar', '0123 Oak St, Ghaziabad', '9900101122', 'rohank', 'pass012');

**Results:**

1. **CUSTOMER:** Contains personal and login details of the bank's customers.

- Query: **SELECT \* FROM CustomerManagement.CUSTOMER;**

A screenshot of a computer

Description automatically generated

1. **EMPLOYEES**: Stores information about the employees of the organization.

* First, we ensure that we are using the EmployeeAndTransaction database with "USE EmployeeAndTransaction".
* Second, we create the EMPLOYEES table using "CREATE TABLE EMPLOYEES".
* Third, we populate EMPLOYEES with employee data via "INSERT INTO EMPLOYEES".
* Finally, we display all employee records using "SELECT \* from EmployeeAndTransaction.EMPLOYEES".

USE EmployeeAndTransaction;

**-- Creation of Employees Table**

CREATE TABLE EMPLOYEES (

Emp\_ID INT PRIMARY KEY,

First\_Name VARCHAR(50),

Last\_Name VARCHAR(50),

Address VARCHAR(250),

Contact\_Details VARCHAR(50),

User\_Name VARCHAR(50),

Password VARCHAR(50)

);

**-- Insert Employees**

INSERT INTO EMPLOYEES (Emp\_ID, First\_Name, Last\_Name, Address, Contact\_Details, User\_Name, Password) VALUES

(201, 'Soham', 'Gupta', '6789 Spruce St, Pune', '9900667788', 'sohamg', 'pass678'),

(202, 'Ayaan', 'Khan', '7890 Birch St, Hyderabad', '9900778899', 'ayaank', 'pass789'),

(203, 'Ansh', 'Mehta', '8901 Redwood St, Ahmedabad', '9900889900', 'anshm', 'pass890'),

(204, 'Dhruv', 'Patel', '9012 Douglas St, Surat', '9900990011', 'dhruvp', 'pass901'),

(205, 'Ritvik', 'Bose', '0123 Elm St, Jaipur', '9900101122', 'ritvikb', 'pass012'),

(206, 'Priya', 'Nair', '0134 Oak St, Lucknow', '9900112233', 'priyan', 'pass013'),

(207, 'Nikhil', 'Verma', '0145 Pine St, Chandigarh', '9900223344', 'nikhilv', 'pass014'),

(208, 'Tanvi', 'Chopra', '0156 Maple St, Gurgaon', '9900334455', 'tanvic', 'pass015'),

(209, 'Siddharth', 'Jain', '0167 Cedar St, Bhopal', '9900445566', 'siddj', 'pass016'),

(210, 'Manisha', 'Kaur', '0178 Redwood St, Patna', '9900556677', 'manishak', 'pass017'),

(211, 'Vikram', 'Shetty', '0189 Birch St, Indore', '9900667788', 'vikrams', 'pass018'),

(212, 'Lata', 'Raj', '0200 Spruce St, Kochi', '9900778899', 'lata', 'pass020'),

(213, 'Rohan', 'Gupta', '0211 Cedar St, Mysore', '9900889900', 'rohang', 'pass021'),

(214, 'Simran', 'Mehta', '0222 Oak St, Amritsar', '9900990011', 'simranm', 'pass022'),

(215, 'Ishan', 'Patel', '0233 Pine St, Coimbatore', '9900101122', 'ishanp', 'pass023'),

(216, 'Monika', 'Deshmukh', '0244 Maple St, Visakhapatnam', '9900112233', 'monikad', 'pass024'),

(217, 'Karan', 'Kumar', '0255 Birch St, Bhubaneswar', '9900223344', 'karank', 'pass025'),

(218, 'Sneha', 'Sharma', '0266 Cedar St, Guwahati', '9900334455', 'snehas', 'pass026'),

(219, 'Arpit', 'Singh', '0277 Spruce St, Thiruvananthapuram', '9900445566', 'arpits', 'pass027'),

(220, 'Heena', 'Raj', '0288 Oak St, Nagpur', '9900556677', 'heenar', 'pass028'),

(221, 'Ravi', 'Bose', '0299 Pine St, Jaipur', '9900667788', 'ravib', 'pass029'),

(222, 'Yash', 'Nair', '0301 Maple St, Gandhinagar', '9900778899', 'yashn', 'pass030'),

(223, 'Riya', 'Chopra', '0312 Birch St, Shillong', '9900889900', 'riyac', 'pass031'),

(224, 'Nikhil', 'Jain', '0323 Cedar St, Panaji', '9900990011', 'nikhilj', 'pass032'),

(225, 'Anjali', 'Kaur', '0334 Redwood St, Ranchi', '9900101122', 'anjalik', 'pass033'),

(226, 'Sagar', 'Shetty', '0345 Birch St, Shimla', '9900112233', 'sagars', 'pass034'),

(227, 'Mira', 'Raj', '0356 Spruce St, Srinagar', '9900223344', 'mirar', 'pass035'),

(228, 'Amit', 'Gupta', '0367 Cedar St, Imphal', '9900334455', 'amitg', 'pass036'),

(229, 'Tina', 'Mehta', '0378 Oak St, Agartala', '9900445566', 'tinam', 'pass037'),

(230, 'Rajesh', 'Patel', '0389 Pine St, Aizawl', '9900556677', 'rajeshp', 'pass038');

**Results:**

**EMPLOYEES**: Stores information on the bank's employees.

* + Query: **SELECT \* FROM EmployeeAndTransaction.EMPLOYEES;**

**A screenshot of a computer

Description automatically generated**

1. **ACCOUNTS:** Manages the customer's account information.

* First, we ensure that we are using the CustomerManagement database with "USE CustomerManagement".
* Second, we create the ACCOUNTS table using "CREATE TABLE ACCOUNTS".
* Third, we populate ACCOUNTS with account data via "INSERT INTO ACCOUNTS".
* Finally, we display all account records using "SELECT \* from CustomerManagement.ACCOUNTS".

**-- CustomerManagement Database --**

USE CustomerManagement;

**-- Creation of Accounts Table**

CREATE TABLE ACCOUNTS (

Account\_ID INT PRIMARY KEY,

Cust\_ID INT,

Account\_Name VARCHAR(50),

Login\_Time TIMESTAMP,

FOREIGN KEY (Cust\_ID) REFERENCES CUSTOMER(Cust\_ID)

);

**-- Insert Accounts**

INSERT INTO ACCOUNTS (Account\_ID, Cust\_ID, Account\_Name, Login\_Time) VALUES

(101, 1, 'account1', '2023-04-01 08:00:00'),

(102, 2, 'account2', '2023-04-01 09:00:00'),

(103, 3, 'account3', '2023-04-01 10:00:00'),

(104, 4, 'account4', '2023-04-01 11:00:00'),

(105, 5, 'account5', '2023-04-01 12:00:00'),

(106, 6, 'account6', '2023-04-01 13:00:00'),

(107, 7, 'account7', '2023-04-01 14:00:00'),

(108, 8, 'account8', '2023-04-01 15:00:00'),

(109, 9, 'account9', '2023-04-01 16:00:00'),

(110, 10, 'account10', '2023-04-01 17:00:00'),

(111, 11, 'account11', '2023-04-01 18:00:00'),

(112, 12, 'account12', '2023-04-01 19:00:00'),

(113, 13, 'account13', '2023-04-01 20:00:00'),

(114, 14, 'account14', '2023-04-01 21:00:00'),

(115, 15, 'account15', '2023-04-01 22:00:00'),

(116, 16, 'account16', '2023-04-02 08:00:00'),

(117, 17, 'account17', '2023-04-02 09:00:00'),

(118, 18, 'account18', '2023-04-02 10:00:00'),

(119, 19, 'account19', '2023-04-02 11:00:00'),

(120, 20, 'account20', '2023-04-02 12:00:00'),

(121, 21, 'account21', '2023-04-02 13:00:00'),

(122, 22, 'account22', '2023-04-02 14:00:00'),

(123, 23, 'account23', '2023-04-02 15:00:00'),

(124, 24, 'account24', '2023-04-02 16:00:00'),

(125, 25, 'account25', '2023-04-02 17:00:00'),

(126, 26, 'account26', '2023-04-02 18:00:00'),

(127, 27, 'account27', '2023-04-02 19:00:00'),

(128, 28, 'account28', '2023-04-02 20:00:00'),

(129, 29, 'account29', '2023-04-02 21:00:00'),

(130, 30, 'account30', '2023-04-02 22:00:00');

**Results:**

**ACCOUNTS:** Manages the customers' account information.

- Query: **SELECT \* FROM CustomerManagement.ACCOUNTS;**

A screenshot of a computer

Description automatically generated

1. **USER\_SESSIONS:** Tracks the session activity of users in the system.
   * First, we ensure that we are using the CustomerManagement database with "USE CustomerManagement".
   * Second, we create the USER\_SESSIONS table using "CREATE TABLE USER\_SESSIONS".
   * Third, we populate USER\_SESSIONS with session data via "INSERT INTO USER\_SESSIONS".
   * Finally, we display all user session records using "SELECT \* from CustomerManagement.USER\_SESSIONS".

USE CustomerManagement;

**-- Creation of User\_Sessions Table**

CREATE TABLE USER\_SESSIONS (

Session\_ID INT PRIMARY KEY,

Emp\_ID INT,

Session\_Start TIMESTAMP,

Session\_End TIMESTAMP,

FOREIGN KEY (Emp\_ID) REFERENCES EmployeeAndTransaction.EMPLOYEES(Emp\_ID) -- Note cross-database reference

);

**-- Insert User Sessions**

INSERT INTO USER\_SESSIONS (Session\_ID, Emp\_ID, Session\_Start, Session\_End) VALUES

(501, 201, '2023-04-01 08:30:00', '2023-04-01 12:30:00'),

(502, 202, '2023-04-01 09:30:00', '2023-04-01 13:30:00'),

(503, 203, '2023-04-01 10:30:00', '2023-04-01 14:30:00'),

(504, 204, '2023-04-01 11:30:00', '2023-04-01 15:30:00'),

(505, 205, '2023-04-01 12:30:00', '2023-04-01 16:30:00'),

(506, 206, '2023-04-01 08:45:00', '2023-04-01 12:45:00'),

(507, 207, '2023-04-01 09:45:00', '2023-04-01 13:45:00'),

(508, 208, '2023-04-01 10:45:00', '2023-04-01 14:45:00'),

(509, 209, '2023-04-01 11:45:00', '2023-04-01 15:45:00'),

(510, 210, '2023-04-01 12:45:00', '2023-04-01 16:45:00'),

(511, 211, '2023-04-02 08:30:00', '2023-04-02 12:30:00'),

(512, 212, '2023-04-02 09:30:00', '2023-04-02 13:30:00'),

(513, 213, '2023-04-02 10:30:00', '2023-04-02 14:30:00'),

(514, 214, '2023-04-02 11:30:00', '2023-04-02 15:30:00'),

(515, 215, '2023-04-02 12:30:00', '2023-04-02 16:30:00'),

(516, 216, '2023-04-02 08:45:00', '2023-04-02 12:45:00'),

(517, 217, '2023-04-02 09:45:00', '2023-04-02 13:45:00'),

(518, 218, '2023-04-02 10:45:00', '2023-04-02 14:45:00'),

(519, 219, '2023-04-02 11:45:00', '2023-04-02 15:45:00'),

(520, 220, '2023-04-02 12:45:00', '2023-04-02 16:45:00'),

(521, 221, '2023-04-03 08:30:00', '2023-04-03 12:30:00'),

(522, 222, '2023-04-03 09:30:00', '2023-04-03 13:30:00'),

(523, 223, '2023-04-03 10:30:00', '2023-04-03 14:30:00'),

(524, 224, '2023-04-03 11:30:00', '2023-04-03 15:30:00'),

(525, 225, '2023-04-03 12:30:00', '2023-04-03 16:30:00'),

(526, 226, '2023-04-03 08:45:00', '2023-04-03 12:45:00'),

(527, 227, '2023-04-03 09:45:00', '2023-04-03 13:45:00'),

(528, 228, '2023-04-03 10:45:00', '2023-04-03 14:45:00'),

(529, 229, '2023-04-03 11:45:00', '2023-04-03 15:45:00'),

(530, 230, '2023-04-03 12:45:00', '2023-04-03 16:45:00');

**Results:**

**USER\_SESSIONS:** Tracks the login and logout sessions of users.

Query: **SELECT \* FROM CustomerManagement.USER\_SESSIONS;**

A screenshot of a computer

Description automatically generated

1. **TRANSACTIONS**: Records the transactions handled by employees for customers.
   * First, we ensure that we are using the EmployeeAndTransaction database with "USE EmployeeAndTransaction".
   * Second, we create the TRANSACTIONS table using "CREATE TABLE TRANSACTIONS".
   * Third, we populate TRANSACTIONS with transaction data via "INSERT INTO TRANSACTIONS".
   * Finally, we display all transaction records using "SELECT \* from EmployeeAndTransaction.TRANSACTIONS".

**-- Creation of Transactions Table**

CREATE TABLE TRANSACTIONS (

Transaction\_ID INT PRIMARY KEY,

Emp\_ID INT,

Cust\_ID INT,

Trans\_Name VARCHAR(100),

FOREIGN KEY (Emp\_ID) REFERENCES EMPLOYEES(Emp\_ID),

FOREIGN KEY (Cust\_ID) REFERENCES CustomerManagement.CUSTOMER(Cust\_ID)

);

**-- Insert Transactions**

INSERT INTO TRANSACTIONS (Transaction\_ID, Emp\_ID, Cust\_ID, Trans\_Name) VALUES

(301, 201, 1, 'Transaction1'),

(302, 202, 2, 'Transaction2'),

(303, 203, 3, 'Transaction3'),

(304, 204, 4, 'Transaction4'),

(305, 205, 5, 'Transaction5'),

(306, 201, 6, 'Transaction6'),

(307, 202, 7, 'Transaction7'),

(308, 203, 8, 'Transaction8'),

(309, 204, 9, 'Transaction9'),

(310, 205, 10, 'Transaction10'),

(311, 201, 11, 'Transaction11'),

(312, 202, 12, 'Transaction12'),

(313, 203, 13, 'Transaction13'),

(314, 204, 14, 'Transaction14'),

(315, 205, 15, 'Transaction15'),

(316, 201, 16, 'Transaction16'),

(317, 202, 17, 'Transaction17'),

(318, 203, 18, 'Transaction18'),

(319, 204, 19, 'Transaction19'),

(320, 205, 20, 'Transaction20'),

(321, 201, 21, 'Transaction21'),

(322, 202, 22, 'Transaction22'),

(323, 203, 23, 'Transaction23'),

(324, 204, 24, 'Transaction24'),

(325, 205, 25, 'Transaction25'),

(326, 201, 26, 'Transaction26'),

(327, 202, 27, 'Transaction27'),

(328, 203, 28, 'Transaction28'),

(329, 204, 29, 'Transaction29'),

(330, 205, 30, 'Transaction30');

**Results:**

**TRANSACTIONS:** Records all financial transactions conducted.

- Query: **SELECT \* FROM EmployeeAndTransaction.TRANSACTIONS;**

A screenshot of a computer

Description automatically generated

1. **LOAN:** Contains details of loans taken out by customers.
   * First, we ensure that we are using the LoanManagement database with "USE LoanManagement".
   * Second, we create the LOAN table using "CREATE TABLE LOAN".
   * Third, we populate LOAN with loan data via "INSERT INTO LOAN".
   * Finally, we display all loan records using "SELECT \* from LoanManagement.LOAN".

USE LoanManagement;

**-- Creation of Loan Table**

CREATE TABLE LOAN (

LOAN\_ID INT PRIMARY KEY,

Cust\_ID INT,

Emp\_ID INT,

Loan\_amount DECIMAL(19,4),

Interest\_rate DECIMAL(5,2),

Loan\_Start\_Date DATE,

Loan\_End\_Date DATE,

Loan\_Status VARCHAR(50),

FOREIGN KEY (Cust\_ID) REFERENCES CustomerManagement.CUSTOMER(Cust\_ID),

FOREIGN KEY (Emp\_ID) REFERENCES EmployeeAndTransaction.EMPLOYEES(Emp\_ID)

);

**-- Insert Loans**

INSERT INTO LOAN (LOAN\_ID, Cust\_ID, Emp\_ID, Loan\_amount, Interest\_rate, Loan\_Start\_Date, Loan\_End\_Date, Loan\_Status) VALUES

(401, 1, 201, 50000.00, 5.5, '2023-04-01', '2028-04-01', 'Active'),

(402, 2, 202, 100000.00, 6.5, '2023-04-02', '2028-04-02', 'Active'),

(403, 3, 203, 150000.00, 7.5, '2023-04-03', '2028-04-03', 'Active'),

(404, 4, 204, 200000.00, 8.5, '2023-04-04', '2028-04-04', 'Active'),

(405, 5, 205, 250000.00, 9.5, '2023-04-05', '2028-04-05', 'Active'),

(406, 6, 206, 300000.00, 10.0, '2023-04-06', '2028-04-06', 'Active'),

(407, 7, 207, 350000.00, 10.5, '2023-04-07', '2028-04-07', 'Active'),

(408, 8, 208, 400000.00, 11.0, '2023-04-08', '2028-04-08', 'Active'),

(409, 9, 209, 450000.00, 11.5, '2023-04-09', '2028-04-09', 'Active'),

(410, 10, 210, 500000.00, 12.0, '2023-04-10', '2028-04-10', 'Active'),

(411, 11, 211, 550000.00, 12.5, '2023-04-11', '2028-04-11', 'Active'),

(412, 12, 212, 600000.00, 13.0, '2023-04-12', '2028-04-12', 'Active'),

(413, 13, 213, 650000.00, 13.5, '2023-04-13', '2028-04-13', 'Active'),

(414, 14, 214, 700000.00, 14.0, '2023-04-14', '2028-04-14', 'Active'),

(415, 15, 215, 750000.00, 14.5, '2023-04-15', '2028-04-15', 'Active'),

(416, 16, 216, 800000.00, 15.0, '2023-04-16', '2028-04-16', 'Active'),

(417, 17, 217, 850000.00, 15.5, '2023-04-17', '2028-04-17', 'Active'),

(418, 18, 218, 900000.00, 16.0, '2023-04-18', '2028-04-18', 'Active'),

(419, 19, 219, 950000.00, 16.5, '2023-04-19', '2028-04-19', 'Active'),

(420, 20, 220, 1000000.00, 17.0, '2023-04-20', '2028-04-20', 'Active'),

(421, 21, 221, 1050000.00, 17.5, '2023-04-21', '2028-04-21', 'Active'),

(422, 22, 222, 1100000.00, 18.0, '2023-04-22', '2028-04-22', 'Active'),

(423, 23, 223, 1150000.00, 18.5, '2023-04-23', '2028-04-23', 'Active'),

(424, 24, 224, 1200000.00, 19.0, '2023-04-24', '2028-04-24', 'Active'),

(425, 25, 225, 1250000.00, 19.5, '2023-04-25', '2028-04-25', 'Active'),

(426, 26, 226, 1300000.00, 20.0, '2023-04-26', '2028-04-26', 'Active'),

(427, 27, 227, 1350000.00, 20.5, '2023-04-27', '2028-04-27', 'Active'),

(428, 28, 228, 1400000.00, 21.0, '2023-04-28', '2028-04-28', 'Active'),

(429, 29, 229, 1450000.00, 21.5, '2023-04-29', '2028-04-29', 'Active'),

(430, 30, 230, 1500000.00, 22.0, '2023-04-30', '2028-04-30', 'Active');

**Results:**

**LOAN**: Details loans taken out by customers, including terms and conditions.

-Query: **SELECT \* FROM LoanManagement.LOAN;**

A screenshot of a computer

Description automatically generated

1. **PAYMENT**: Manages payment records towards loans by customers.
   * First, we ensure that we are using the CustomerManagement database with "USE CustomerManagement".
   * Second, we create the PAYMENT table using "CREATE TABLE PAYMENT".
   * Third, we populate PAYMENT with payment data via "INSERT INTO PAYMENT".
   * Finally, we display all payment records using "SELECT \* from CustomerManagement.PAYMENT".

USE LoanManagement;

**-- Creation of Payment Table**

CREATE TABLE PAYMENT (

Pay\_ID INT PRIMARY KEY,

Cust\_ID INT,

LOAN\_ID INT,

Pay\_amount DECIMAL(19,4),

Pay\_Method VARCHAR(50),

Pay\_Date DATE,

FOREIGN KEY (Cust\_ID) REFERENCES CustomerManagement.CUSTOMER(Cust\_ID), -- Note cross-database reference

FOREIGN KEY (LOAN\_ID) REFERENCES LoanManagement.LOAN(LOAN\_ID)

);

**-- Insert Payments**

INSERT INTO PAYMENT (Pay\_ID, Cust\_ID, LOAN\_ID, Pay\_amount, Pay\_Method, Pay\_Date) VALUES

(501, 1, 401, 1000.00, 'Online', '2023-05-01'),

(502, 2, 402, 2000.00, 'Cheque', '2023-05-02'),

(503, 3, 403, 3000.00, 'Cash', '2023-05-03'),

(504, 4, 404, 4000.00, 'Online', '2023-05-04'),

(505, 5, 405, 5000.00, 'Cheque', '2023-05-05'),

(506, 6, 406, 6000.00, 'Cash', '2023-05-06'),

(507, 7, 407, 7000.00, 'Online', '2023-05-07'),

(508, 8, 408, 8000.00, 'Cheque', '2023-05-08'),

(509, 9, 409, 9000.00, 'Cash', '2023-05-09'),

(510, 10, 410, 10000.00, 'Online', '2023-05-10'),

(511, 11, 411, 11000.00, 'Cheque', '2023-05-11'),

(512, 12, 412, 12000.00, 'Cash', '2023-05-12'),

(513, 13, 413, 13000.00, 'Online', '2023-05-13'),

(514, 14, 414, 14000.00, 'Cheque', '2023-05-14'),

(515, 15, 415, 15000.00, 'Cash', '2023-05-15'),

(516, 16, 416, 16000.00, 'Online', '2023-05-16'),

(517, 17, 417, 17000.00, 'Cheque', '2023-05-17'),

(518, 18, 418, 18000.00, 'Cash', '2023-05-18'),

(519, 19, 419, 19000.00, 'Online', '2023-05-19'),

(520, 20, 420, 20000.00, 'Cheque', '2023-05-20'),

(521, 21, 421, 21000.00, 'Cash', '2023-05-21'),

(522, 22, 422, 22000.00, 'Online', '2023-05-22'),

(523, 23, 423, 23000.00, 'Cheque', '2023-05-23'),

(524, 24, 424, 24000.00, 'Cash', '2023-05-24'),

(525, 25, 425, 25000.00, 'Online', '2023-05-25'),

(526, 26, 426, 26000.00, 'Cheque', '2023-05-26'),

(527, 27, 427, 27000.00, 'Cash', '2023-05-27'),

(528, 28, 428, 28000.00, 'Online', '2023-05-28'),

(529, 29, 429, 29000.00, 'Cheque', '2023-05-29'),

(530, 30, 430, 30000.00, 'Cash', '2023-05-30');

**Results:**

**PAYMENT:** Manages records of payments made towards loans by customers.

Query**: SELECT \* FROM LoanManagement.PAYMENT;**

**A screenshot of a computer

Description automatically generated**

1. **REPORT\_DETAILS:** Stores detailed reports related to transactions, accounts, and sessions.
   * First, we ensure that we are using the EmployeeAndTransaction database with "USE EmployeeAndTransaction".
   * Second, we create the REPORT\_DETAILS table using "CREATE TABLE REPORT\_DETAILS".
   * Third, we populate REPORT\_DETAILS with report data via "INSERT INTO REPORT\_DETAILS".
   * Finally, we display all report detail records using "SELECT \* from EmployeeAndTransaction.REPORT\_DETAILS".

USE EmployeeAndTransaction;

**-- Creation of REPORT\_DETAILS Table**

CREATE TABLE REPORT\_DETAILS (

Detail\_ID INT PRIMARY KEY,

Account\_ID INT,

Transaction\_ID INT,

Session\_ID INT,

Detail\_Type VARCHAR(50),

Detail\_Date DATE,

FOREIGN KEY (Account\_ID) REFERENCES CustomerManagement.ACCOUNTS(Account\_ID), -- Note cross-database reference

FOREIGN KEY (Transaction\_ID) REFERENCES TRANSACTIONS(Transaction\_ID),

FOREIGN KEY (Session\_ID) REFERENCES CustomerManagement.USER\_SESSIONS(Session\_ID) -- Note cross-database reference

);

**-- Insert Report Details**

INSERT INTO REPORT\_DETAILS (Detail\_ID, Account\_ID, Transaction\_ID, Session\_ID, Detail\_Type, Detail\_Date) VALUES

(601, 101, 301, 501, 'Payment', '2023-05-01'),

(602, 102, 302, 502, 'Withdrawal', '2023-05-02'),

(603, 103, 303, 503, 'Deposit', '2023-05-03'),

(604, 104, 304, 504, 'Payment', '2023-05-04'),

(605, 105, 305, 505, 'Withdrawal', '2023-05-05'),

(606, 106, 306, 506, 'Query', '2023-05-06'),

(607, 107, 307, 507, 'Payment', '2023-05-07'),

(608, 108, 308, 508, 'Withdrawal', '2023-05-08'),

(609, 109, 309, 509, 'Deposit', '2023-05-09'),

(610, 110, 310, 510, 'Payment', '2023-05-10'),

(611, 111, 311, 511, 'Withdrawal', '2023-05-11'),

(612, 112, 312, 512, 'Query', '2023-05-12'),

(613, 113, 313, 513, 'Payment', '2023-05-13'),

(614, 114, 314, 514, 'Withdrawal', '2023-05-14'),

(615, 115, 315, 515, 'Deposit', '2023-05-15'),

(616, 116, 316, 516, 'Payment', '2023-05-16'),

(617, 117, 317, 517, 'Withdrawal', '2023-05-17'),

(618, 118, 318, 518, 'Deposit', '2023-05-18'),

(619, 119, 319, 519, 'Payment', '2023-05-19'),

(620, 120, 320, 520, 'Withdrawal', '2023-05-20'),

(621, 121, 321, 521, 'Query', '2023-05-21'),

(622, 122, 322, 522, 'Payment', '2023-05-22'),

(623, 123, 323, 523, 'Withdrawal', '2023-05-23'),

(624, 124, 324, 524, 'Deposit', '2023-05-24'),

(625, 125, 325, 525, 'Payment', '2023-05-25'),

(626, 126, 326, 526, 'Withdrawal', '2023-05-26'),

(627, 127, 327, 527, 'Query', '2023-05-27'),

(628, 128, 328, 528, 'Payment', '2023-05-28'),

(629, 129, 329, 529, 'Withdrawal', '2023-05-29'),

(630, 130, 330, 530, 'Deposit', '2023-05-30');

**Results:**

**REPORT\_DETAILS:** Holds generated reports concerning accounts, transactions, and sessions.

Query: **SELECT \* FROM EmployeeAndTransaction.REPORT\_DETAILS;**

A screenshot of a computer

Description automatically generated

**Section 2: Data Retrieval and Analytical Reports**

1. **Report on Customer Loan Details**

Which customers have loans, and what are the details of those loans?

* + ***Query Used:***

SELECT c.First\_Name, c.Last\_Name, l.Loan\_amount, l.Loan\_Status FROM CustomerManagement.CUSTOMER c JOIN LoanManagement.LOAN l ON c.Cust\_ID = l.Cust\_ID;

* + ***SQL Command Explanation:***
    1. **SELECT**: Choose the customer's first and last name and loan details.
    2. **FROM**: Indicates the CUSTOMER table from the CustomerManagement database as the starting point.
    3. **JOIN**: Connects the CUSTOMER table to the LOAN table on the matching customer ID.
    4. **ON**: Specifies the condition for the join, which is the matching customer IDs between the two tables.

A screenshot of a computer

Description automatically generated

1. **Employee Loan Management Report**

Which employees manage the largest loans?

* + ***Query Used:*** SELECT e.First\_Name, e.Last\_Name, MAX(l.Loan\_amount) as Largest\_Loan\_Managed FROM EmployeeAndTransaction.EMPLOYEES e JOIN LoanManagement.LOAN l ON e.Emp\_ID = l.Emp\_ID GROUP BY e.Emp\_ID;
  + ***SQL Command Explanation:***
    1. **SELECT**: Choose the employee's first and last name and the maximum loan amount they manage.
    2. **FROM**: Indicates the EMPLOYEES table from the EmployeeAndTransaction database as the starting point.
    3. **JOIN**: Connects the EMPLOYEES table to the LOAN table on the matching employee ID.
    4. **GROUP BY**: Aggregates the results by the employee ID to calculate the maximum loan amount per employee.

**A screenshot of a computer

Description automatically generated**

1. **Customer Payment Summary Report**

How much has each customer paid in total?

* + ***Query Used:*** SELECT c.First\_Name, c.Last\_Name, SUM(p.Pay\_amount) as Total\_Paid FROM CustomerManagement.CUSTOMER c JOIN CustomerManagement.PAYMENT p ON c.Cust\_ID = p.Cust\_ID GROUP BY c.Cust\_ID;
  + ***SQL Command Explanation:***
    1. **SELECT**: Choose the customer's first and last name and the sum of their payment amounts.
    2. **FROM**: Indicates the CUSTOMER table from the CustomerManagement database as the starting point.
    3. **JOIN**: Connects the CUSTOMER table to the PAYMENT table on the matching customer ID.
    4. **GROUP BY**: Aggregates the results by the customer ID to calculate the total paid amount per customer.

A screenshot of a computer

Description automatically generated

1. **Account Transactions Frequency Report**

Which accounts have the highest number of transactions?

* + ***Query Used:*** SELECT a.Account\_ID, COUNT(t.Transaction\_ID) as TransactionCount FROM CustomerManagement.ACCOUNTS a JOIN EmployeeAndTransaction.TRANSACTIONS t ON a.Cust\_ID = t.Cust\_ID GROUP BY a.Account\_ID ORDER BY TransactionCount DESC;
  + ***SQL Command Explanation:***
    1. **SELECT**: Choose the account ID and count the number of transactions associated with each account.
    2. **FROM**: Indicates the ACCOUNTS table from the CustomerManagement database as the starting point.
    3. **JOIN**: Connects the ACCOUNTS table to the TRANSACTIONS table where the customer ID matches.
    4. **GROUP BY**: Aggregates the results by the account ID to count transactions for each account.
    5. **ORDER BY**: Orders the results by the count of transactions in descending order.

A screenshot of a computer

Description automatically generated

1. **Employee Attendance Report**

How frequently do employees log into the system?

* + ***Query Used:*** SELECT e.First\_Name, e.Last\_Name, COUNT(s.Session\_ID) as LoginFrequency FROM EmployeeAndTransaction.EMPLOYEES e JOIN CustomerManagement.USER\_SESSIONS s ON e.Emp\_ID = s.Emp\_ID GROUP BY e.Emp\_ID;
  + ***SQL Command Explanation:***
    1. **SELECT**: Choose the employee's first and last name and count their login sessions.
    2. **FROM**: Indicates the EMPLOYEES table from the EmployeeAndTransaction database as the starting point.
    3. **JOIN**: Connects the EMPLOYEES table to the USER\_SESSIONS table on the matching employee ID.
    4. **GROUP BY**: Aggregates the results by employee ID to count their login sessions.

A screenshot of a computer

Description automatically generated

1. **Total Loan Distribution by Employees Report**

How much in total loan amounts have each of the employees distributed?

* + ***Query Used:*** SELECT e.First\_Name, e.Last\_Name, SUM(l.Loan\_amount) as TotalLoansDistributed FROM EmployeeAndTransaction.EMPLOYEES e JOIN LoanManagement.LOAN l ON e.Emp\_ID = l.Emp\_ID GROUP BY e.Emp\_ID;
  + ***SQL Command Explanation:***
    1. **SELECT**: Choose the employee's first and last name and sum the loan amounts they have distributed.
    2. **FROM**: Indicates the EMPLOYEES table from the EmployeeAndTransaction database as the starting point.
    3. **JOIN**: Connects the EMPLOYEES table to the LOAN table on the matching employee ID.
    4. **GROUP BY**: Aggregates the results by employee ID to sum the loan amounts for each.

A screenshot of a computer

Description automatically generated

1. **Customer Loan Absence Report**

Which customers do not have any loans?

* + ***Query Used:*** SELECT c.First\_Name, c.Last\_Name FROM CustomerManagement.CUSTOMER c LEFT JOIN LoanManagement.LOAN l ON c.Cust\_ID = l.Cust\_ID WHERE l.Cust\_ID IS NULL;
  + ***SQL Command Explanation:***
    1. **SELECT**: Choose the first and last names of customers.
    2. **FROM**: Indicates the CUSTOMER table from the CustomerManagement database as the starting point.
    3. **LEFT JOIN**: Connects the CUSTOMER table to the LOAN table but retains all customers even if they don't have loans.
    4. **WHERE**: Filters the results to show only those customers who do not have a corresponding loan ID, implying they have no loans.

A screenshot of a computer program

Description automatically generated