Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2024), B.Sc. in CSE (Day)

Database of Bank Management System

Course Title: Database System Lab
Course Code: CSE 210
Section: 221 D1

Students Details

Name	ID
Tanvir Ahmed	221002461

Submission Date: 06/07/2024
Course Teacher's Name: Dr. Faiz Al Faisal

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The Bank Management System is a comprehensive software solution crafted to revolutionize the operational efficiency of financial institutions. This system integrates a set of modules to automate and optimize key banking processes, ranging from customer management to transaction processing. By leveraging technology, the system aims to enhance accuracy, speed, and security in handling diverse financial transactions.

1.2 Motivation

•**Relevance to Industry Trends:** The banking sector is undergoing a digital transformation, and a Bank Management System project aligns with current industry trends.

•**Practical Application of Knowledge:** Provides a practical application of programming, database management, and system design concepts learned in coursework.

•**Career Readiness:** Developing expertise in building financial systems enhances my career readiness for roles in software development, fintech, or the broader IT industry.

•**Problem-Solving and Innovation:** Think critically and innovatively about how to address existing issues in traditional banking systems..

•**Interdisciplinary Learning:** The project inherently involves understanding both technical and financial aspects, fostering interdisciplinary learning.

•**Social Impact:** Enhancing banking processes contribute o the broader goal of financial inclusion and accessibility..

•**Project Management Skills**: Provides an opportunity to enhance project management skills.

•**Portfolio Development:** Adds a significant and tangible item to your portfolio..

•**Learning from Industry Challenges:** Helps me to understand the practical constraints and demands of real-world applications.

Educational Growth: Academic requirements and contributes to your personal and educational growth. [**?**].

1.3 Problem Definition

1.3.1 Problem Statement

The existing banking systems often face challenges related to manual processes, data redundancies, and a lack of centralized control. These issues can lead to errors, security vulnerabilities, and prolonged service delivery times. The need for a more streamlined and integrated approach to bank management is evident. Inefficient customer onboarding, time-consuming transaction processing, and difficulty in managing employee workflows are some of the key problems that the Bank Management System project seeks to address. The project aims to provide a solution that not only automates routine tasks but also enhances the overall management and security of banking operations. By identifying and defining these problems, the project sets the stage for the development of a comprehensive system that will revolutionize how banks operate, ensuring a smoother and more efficient experience for both customers and employees.

1.3.2 Complex Engineering Problem

Table 1.1: Summary of the attributes touched by the projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	Computer Science and Software Development DBMS (e.g., MySQL, MongoDB) to build a robust system.
P2: Range of conflicting requirements	—-
P3: Depth of analysis required	Data Analytics and Decision-Making
P4: Familiarity of issues	Finance and Banking Operations
P5: Extent of applicable codes	Industry Standards and Regulations
P6: Extent of stakeholder involvement and conflicting requirements	Collaboration and Communication
P7: Interdependence	Integrated System Components

1.4 Design Goals/Objectives

• Efficiency Enhancement • User-Friendly Interfaces
• Data Security and Compliance
• Scalability and Adaptability
• Comprehensive Transaction Tracking
• Automation of Routine Tasks
• Integration of Emerging Technologies
• Enhanced Customer Service
• Facilitation of Reporting and Analytic

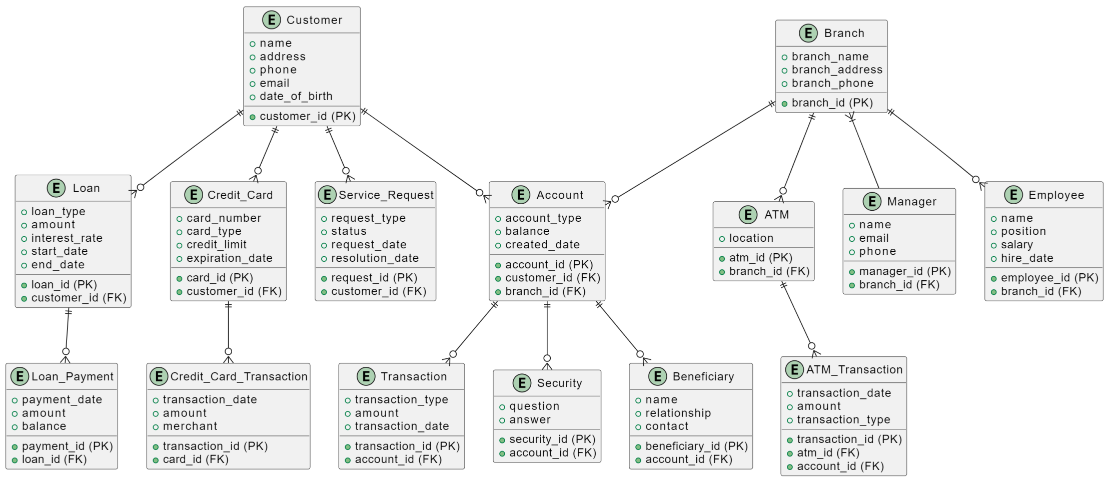


Figure 1.1: Schema Diagram

1.5 Application

Modernizing Banking Operations.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Since the a good overview of Bank Management System are covered in 1st chapter, this one deals with complete development process that includes its designing as well. It details the strategic process used to convert high-level abstract requirements into a working software solution. Some key highlights include how they set up their system, database design, module build and some of the crucial integrations that have helped them to optimize banking operations. The goals were to render the system quite robust, scaleable and secure within functionality of a high transactional volume industrial grade application architecture for which several technical decisions methodologies tools demonstrated in this chapter.

2.2 Project Details

The Bank Management System is a centralized database solution intended to optimize banking operations. Built on Oracle RDBMS, the system has a strong schema that ensures data integrity and scalability. Customer Management, Account Management, Transaction Processing, Loan Management, and Security Management are carefully designed with normalized tables and strong foreign key constraints to efficiently handle complex banking transactions. The architecture prioritizes security and performance, meeting the needs of modern banking environments with a stable and scalable database foundation.

2.3 Implementation

```
Creating Tables:
```

```
CREATE TABLE Customer (
    customer_id NUMBER PRIMARY KEY,
    name VARCHAR2(100),
    address VARCHAR2(200),
    phone VARCHAR2(20),
    email VARCHAR2(100),
    date_of_birth DATE
);

CREATE TABLE Account (
    account_id NUMBER PRIMARY KEY,
    customer_id NUMBER,
    branch_id NUMBER,
    account_type VARCHAR2(50),
    balance NUMBER,
    created_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
);

CREATE TABLE Branch (
    branch_id NUMBER PRIMARY KEY,
    branch_name VARCHAR2(100),
    branch_address VARCHAR2(200),
    branch_phone VARCHAR2(20)
);

CREATE TABLE Employee (
    employee_id NUMBER PRIMARY KEY,
    branch_id NUMBER,
    name VARCHAR2(100),
    position VARCHAR2(50),
    salary NUMBER,
    hire_date DATE,
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
);

CREATE TABLE Transaction (
    transaction_id NUMBER PRIMARY KEY,
    account_id NUMBER,
    transaction_type VARCHAR2(50),
    amount NUMBER,
    transaction_date DATE,
    FOREIGN KEY (account_id) REFERENCES Account(account_id)
);

CREATE TABLE Loan (
    loan_id NUMBER PRIMARY KEY,
    customer_id NUMBER,
    loan_type VARCHAR2(50),
    amount NUMBER,
    interest_rate NUMBER,
    start_date DATE,
    end_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Loan_Payment (
    payment_id NUMBER PRIMARY KEY,
    loan_id NUMBER,
    payment_date DATE,
    amount NUMBER,
    balance NUMBER,
    FOREIGN KEY (loan_id) REFERENCES Loan(loan_id)
);

CREATE TABLE Credit_Card (
    card_id NUMBER PRIMARY KEY,
    customer_id NUMBER,
    card_number VARCHAR2(20),
    card_type VARCHAR2(50),
    credit_limit NUMBER,
    expiration_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Credit_Card_Transaction (
    transaction_id NUMBER PRIMARY KEY,
    card_id NUMBER,
    transaction_date DATE,
    amount NUMBER,
    merchant VARCHAR2(100),
    FOREIGN KEY (card_id) REFERENCES Credit_Card(card_id)
);

CREATE TABLE ATM (
    atm_id NUMBER PRIMARY KEY,
    location VARCHAR2(200),
    branch_id NUMBER,
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
);

CREATE TABLE ATM_Transaction (
    transaction_id NUMBER PRIMARY KEY,
    atm_id NUMBER,
    account_id NUMBER,
    transaction_date DATE,
    amount NUMBER,
    transaction_type VARCHAR2(50),
    FOREIGN KEY (atm_id) REFERENCES ATM(atm_id),
    FOREIGN KEY (account_id) REFERENCES Account(account_id)
);

CREATE TABLE Manager (
    manager_id NUMBER PRIMARY KEY,
    branch_id NUMBER,
    name VARCHAR2(100),
    email VARCHAR2(100),
    phone VARCHAR2(20),
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
);

CREATE TABLE Service_Request (
    request_id NUMBER PRIMARY KEY,
    customer_id NUMBER,
    request_type VARCHAR2(50),
    status VARCHAR2(20),
    request_date DATE,
    resolution_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Security (
    security_id NUMBER PRIMARY KEY,
    account_id NUMBER,
    question VARCHAR2(200),
    answer VARCHAR2(200),
    FOREIGN KEY (account_id) REFERENCES Account(account_id)
);

CREATE TABLE Beneficiary (
    beneficiary_id NUMBER PRIMARY KEY,
    account_id NUMBER,
    name VARCHAR2(100),
    relationship VARCHAR2(50),
    contact VARCHAR2(20),
    FOREIGN KEY (account_id) REFERENCES Account(account_id)
);

Two see the tables:

SELECT table_name
FROM user_tables
WHERE table_name IN (
    'CUSTOMER',
    'ACCOUNT',
    'BRANCH',
    'EMPLOYEE',
    'TRANSACTION',
    'LOAN',
    'LOAN_PAYMENT',
    'CREDIT_CARD',
    'CREDIT_CARD_TRANSACTION',
    'ATM',
    'ATM_TRANSACTION',
    'MANAGER',
    'SERVICE_REQUEST',
    'SECURITY',
    'BENEFICIARY'
);

Triggers:

CREATE OR REPLACE TRIGGER update_account_balance_after_transaction
AFTER INSERT ON Transaction
FOR EACH ROW
DECLARE
    from_account_balance NUMBER;
BEGIN
    IF :NEW.transaction_type = 'Deposit' THEN
        UPDATE Account
        SET balance = balance + :NEW.amount
        WHERE account_id = :NEW.account_id;

    ELSIF :NEW.transaction_type = 'Withdrawal' THEN
        UPDATE Account
        SET balance = balance - :NEW.amount
        WHERE account_id = :NEW.account_id;

    ELSIF :NEW.transaction_type = 'Transfer' THEN
        -- Get the current balance of the from_account
        SELECT balance INTO from_account_balance
        FROM Account
        WHERE account_id = :NEW.account_id;

        -- Update balance for from_account (sender)
        UPDATE Account
        SET balance = from_account_balance - :NEW.amount
        WHERE account_id = :NEW.account_id;

    ELSIF :NEW.transaction_type = 'Receive' THEN
        -- Update balance for to_account (recipient)
        UPDATE Account
        SET balance = balance + :NEW.amount
        WHERE account_id = :NEW.account_id;  -- Assuming :NEW.account_i
    END IF;
END;
/



CREATE OR REPLACE TRIGGER update_loan_balance_after_payment
AFTER INSERT ON Loan_Payment
FOR EACH ROW
BEGIN
    UPDATE Loan
    SET amount = amount - :NEW.amount
    WHERE loan_id = :NEW.loan_id;
END;
/

CREATE OR REPLACE TRIGGER prevent_overdrawn_account
BEFORE INSERT ON Transaction
FOR EACH ROW
DECLARE
    v_balance NUMBER;
BEGIN
    SELECT balance INTO v_balance FROM Account WHERE account_id = :NEW.
    IF :NEW.transaction_type = 'withdrawal' AND v_balance < :NEW.amoun
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds for withdra
    END IF;
END;
/

CREATE OR REPLACE TRIGGER set_default_status_service_request
BEFORE INSERT ON Service_Request
FOR EACH ROW
BEGIN
    IF :NEW.status IS NULL THEN
        :NEW.status := 'Pending';
    END IF;
END;
/

views:

CREATE VIEW CustomerDetails AS
SELECT customer_id, name, address, phone, email, date_of_birth
FROM Customer;
```

```
CREATE VIEW AccountSummary AS
SELECT a.account_id, a.customer_id, c.name AS customer_name, a.branch_i
FROM Account a
JOIN Customer c ON a.customer_id = c.customer_id;

CREATE VIEW TransactionDetails AS
SELECT t.transaction_id, t.account_id, a.customer_id, c.name AS custome
FROM Transaction t
JOIN Account a ON t.account_id = a.account_id
JOIN Customer c ON a.customer_id = c.customer_id;

CREATE VIEW LoanDetails AS
SELECT l.loan_id, l.customer_id, c.name AS customer_name, l.loan_type,
FROM Loan l
JOIN Customer c ON l.customer_id = c.customer_id;

CREATE VIEW ServiceRequestStatus AS
SELECT sr.request_id, sr.customer_id, c.name AS customer_name, sr.reque
FROM Service_Request sr
JOIN Customer c ON sr.customer_id = c.customer_id;
```

2.4 Algorithms

- Connect with the Oracle Database
- Creating tables Customer, Account, Brach, Employee, Transaction, Loan, Loan Payment, Credit Card, Credit Card Transaction, ATM, ATM Transaction, Manager, Service Request, Security, Beneficiary.
- Creating necessary triggers, and views.
- Inserting a huge number of data to each table.
- Make use of the database works perfectly for every table.

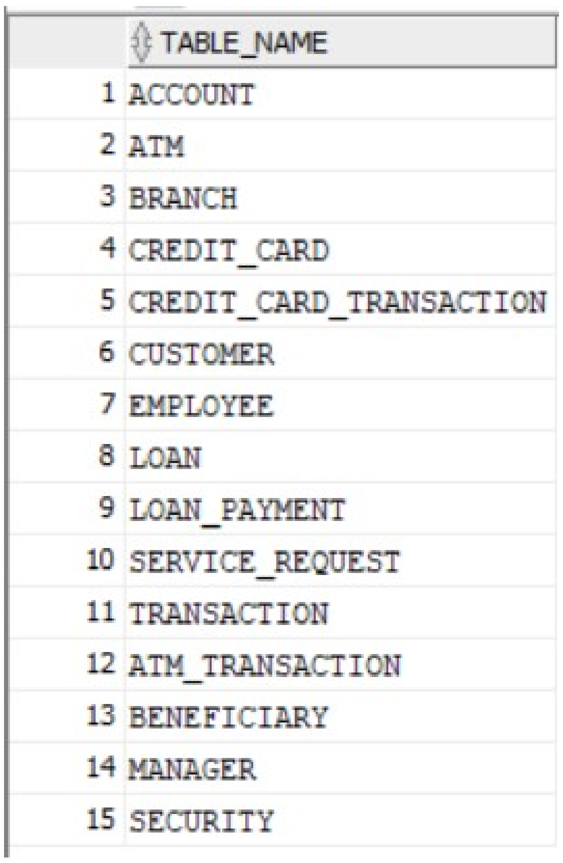
Chapter 3

Performance Evaluation

3.1 Simulation Environment and Procedure

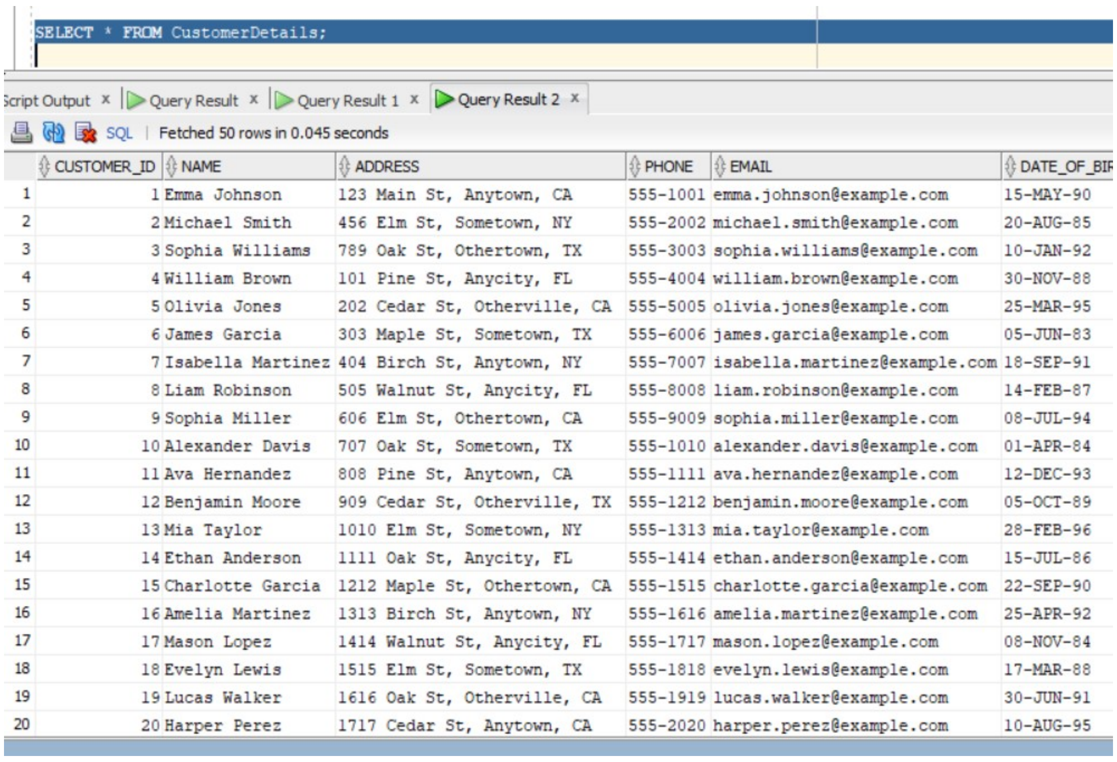
The Bank Management System's performance evaluation focuses on determining its operational efficiency and scalability in a controlled simulation environment. The simulation uses Oracle as the backend database and consists of executing a series of predefined transactions and operations that are representative of real-world banking activities. The procedure entails creating a test database with simulated customer information, accounts, transactions, loans, and service requests. Several scenarios are run to evaluate system response times, transaction processing speeds, and resource utilization under various load conditions. Performance metrics such as transaction throughput, response times, and database query performance are measured and analyzed to determine the system's robustness and scalability.

3.2 Results Analysis/Testing



	TABLE_NAME
1	ACCOUNT
2	ATM
3	BRANCH
4	CREDIT_CARD
5	CREDIT_CARD_TRANSACTION
6	CUSTOMER
7	EMPLOYEE
8	LOAN
9	LOAN_PAYMENT
10	SERVICE_REQUEST
11	TRANSACTION
12	ATM_TRANSACTION
13	BENEFICIARY
14	MANAGER
15	SECURITY

Figure 3.1: Tables



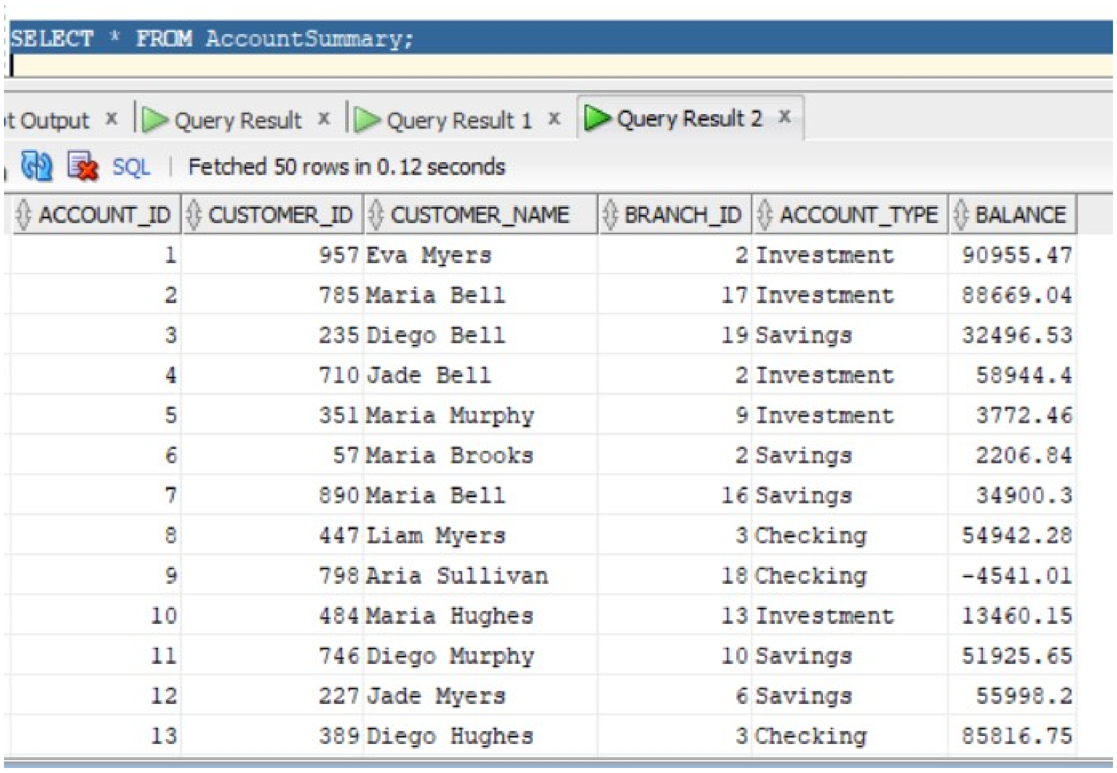
```
SELECT * FROM CustomerDetails;
```

Script Output × | Query Result × | Query Result 1 × | Query Result 2 ×

SQL | Fetched 50 rows in 0.045 seconds

	CUSTOMER_ID	NAME	ADDRESS	PHONE	EMAIL	DATE_OF_BIR
1	1	Emma Johnson	123 Main St, Anytown, CA	555-1001	emma.johnson@example.com	15-MAY-90
2	2	Michael Smith	456 Elm St, Sometown, NY	555-2002	michael.smith@example.com	20-AUG-85
3	3	Sophia Williams	789 Oak St, Othertown, TX	555-3003	sophia.williams@example.com	10-JAN-92
4	4	William Brown	101 Pine St, Anycity, FL	555-4004	william.brown@example.com	30-NOV-88
5	5	Olivia Jones	202 Cedar St, Otherville, CA	555-5005	olivia.jones@example.com	25-MAR-95
6	6	James Garcia	303 Maple St, Sometown, TX	555-6006	james.garcia@example.com	05-JUN-83
7	7	Isabella Martinez	404 Birch St, Anytown, NY	555-7007	isabella.martinez@example.com	18-SEP-91
8	8	Liam Robinson	505 Walnut St, Anycity, FL	555-8008	liam.robinson@example.com	14-FEB-87
9	9	Sophia Miller	606 Elm St, Othertown, CA	555-9009	sophia.miller@example.com	08-JUL-94
10	10	Alexander Davis	707 Oak St, Sometown, TX	555-1010	alexander.davis@example.com	01-APR-84
11	11	Ava Hernandez	808 Pine St, Anytown, CA	555-1111	ava.hernandez@example.com	12-DEC-93
12	12	Benjamin Moore	909 Cedar St, Otherville, TX	555-1212	benjamin.moore@example.com	05-OCT-89
13	13	Mia Taylor	1010 Elm St, Sometown, NY	555-1313	mia.taylor@example.com	28-FEB-96
14	14	Ethan Anderson	1111 Oak St, Anycity, FL	555-1414	ethan.anderson@example.com	15-JUL-86
15	15	Charlotte Garcia	1212 Maple St, Othertown, CA	555-1515	charlotte.garcia@example.com	22-SEP-90
16	16	Amelia Martinez	1313 Birch St, Anytown, NY	555-1616	amelia.martinez@example.com	25-APR-92
17	17	Mason Lopez	1414 Walnut St, Anycity, FL	555-1717	mason.lopez@example.com	08-NOV-84
18	18	Evelyn Lewis	1515 Elm St, Sometown, TX	555-1818	evelyn.lewis@example.com	17-MAR-88
19	19	Lucas Walker	1616 Oak St, Otherville, CA	555-1919	lucas.walker@example.com	30-JUN-91
20	20	Harper Perez	1717 Cedar St, Anytown, CA	555-2020	harper.perez@example.com	10-AUG-95

Figure 3.2: Customer Details



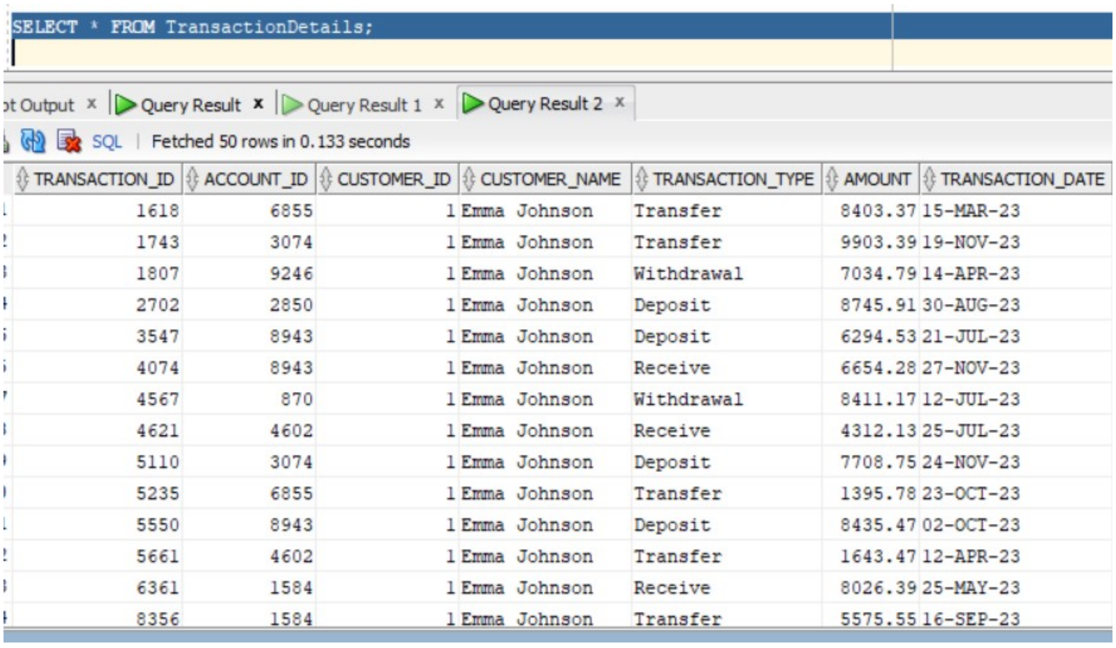
```
SELECT * FROM AccountSummary;
```

t Output × | Query Result × | Query Result 1 × | Query Result 2 ×

SQL | Fetched 50 rows in 0.12 seconds

ACCOUNT_ID	CUSTOMER_ID	CUSTOMER_NAME	BRANCH_ID	ACCOUNT_TYPE	BALANCE
1	957	Eva Myers	2	Investment	90955.47
2	785	Maria Bell	17	Investment	88669.04
3	235	Diego Bell	19	Savings	32496.53
4	710	Jade Bell	2	Investment	58944.4
5	351	Maria Murphy	9	Investment	3772.46
6	57	Maria Brooks	2	Savings	2206.84
7	890	Maria Bell	16	Savings	34900.3
8	447	Liam Myers	3	Checking	54942.28
9	798	Aria Sullivan	18	Checking	-4541.01
10	484	Maria Hughes	13	Investment	13460.15
11	746	Diego Murphy	10	Savings	51925.65
12	227	Jade Myers	6	Savings	55998.2
13	389	Diego Hughes	3	Checking	85816.75

Figure 3.3: Account Summary

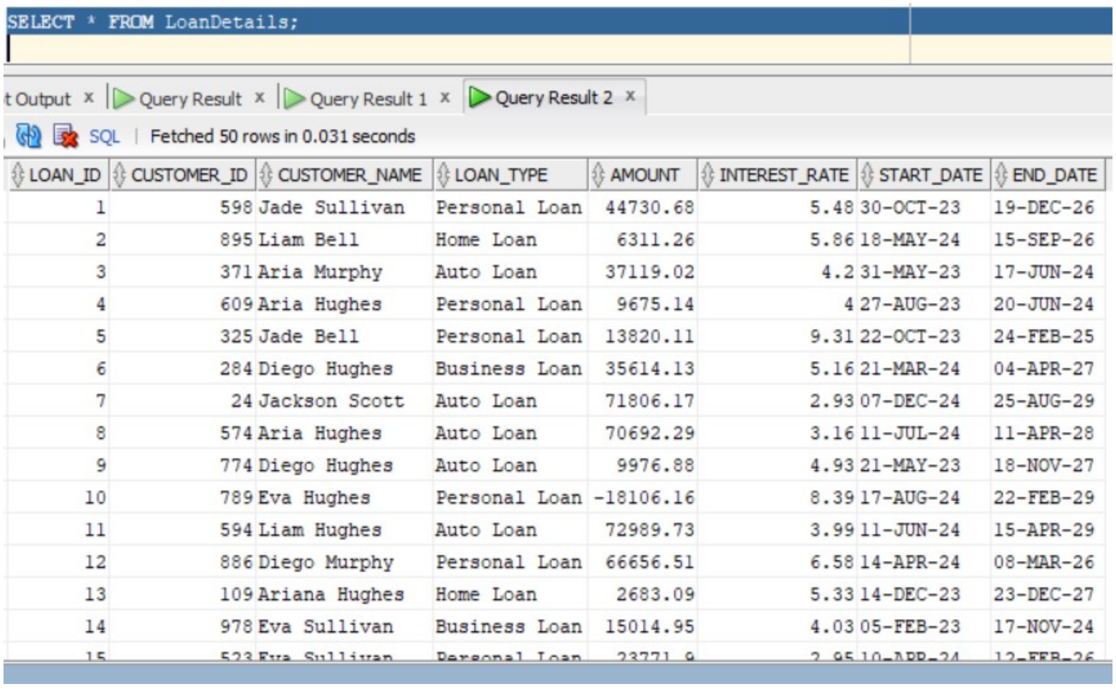


```
SELECT * FROM TransactionDetails;
```

Fetched 50 rows in 0.133 seconds

TRANSACTION_ID	ACCOUNT_ID	CUSTOMER_ID	CUSTOMER_NAME	TRANSACTION_TYPE	AMOUNT	TRANSACTION_DATE
1618	6855	1	Emma Johnson	Transfer	8403.37	15-MAR-23
1743	3074	1	Emma Johnson	Transfer	9903.39	19-NOV-23
1807	9246	1	Emma Johnson	Withdrawal	7034.79	14-APR-23
2702	2850	1	Emma Johnson	Deposit	8745.91	30-AUG-23
3547	8943	1	Emma Johnson	Deposit	6294.53	21-JUL-23
4074	8943	1	Emma Johnson	Receive	6654.28	27-NOV-23
4567	870	1	Emma Johnson	Withdrawal	8411.17	12-JUL-23
4621	4602	1	Emma Johnson	Receive	4312.13	25-JUL-23
5110	3074	1	Emma Johnson	Deposit	7708.75	24-NOV-23
5235	6855	1	Emma Johnson	Transfer	1395.78	23-OCT-23
5550	8943	1	Emma Johnson	Deposit	8435.47	02-OCT-23
5661	4602	1	Emma Johnson	Transfer	1643.47	12-APR-23
6361	1584	1	Emma Johnson	Receive	8026.39	25-MAY-23
8356	1584	1	Emma Johnson	Transfer	5575.55	16-SEP-23

Figure 3.4: Transaction Details

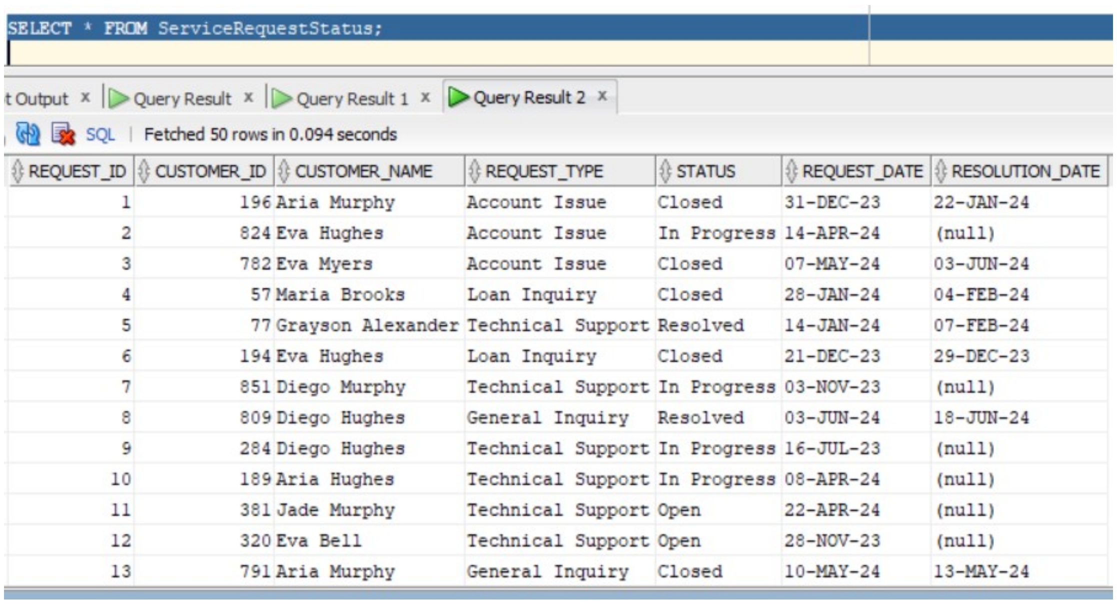


```sql
SELECT * FROM LoanDetails;
```

Fetched 50 rows in 0.031 seconds

LOAN_ID	CUSTOMER_ID	CUSTOMER_NAME	LOAN_TYPE	AMOUNT	INTEREST_RATE	START_DATE	END_DATE
1	598	Jade Sullivan	Personal Loan	44730.68	5.48	30-OCT-23	19-DEC-26
2	895	Liam Bell	Home Loan	6311.26	5.86	18-MAY-24	15-SEP-26
3	371	Aria Murphy	Auto Loan	37119.02	4.2	31-MAY-23	17-JUN-24
4	609	Aria Hughes	Personal Loan	9675.14	4	27-AUG-23	20-JUN-24
5	325	Jade Bell	Personal Loan	13820.11	9.31	22-OCT-23	24-FEB-25
6	284	Diego Hughes	Business Loan	35614.13	5.16	21-MAR-24	04-APR-27
7	24	Jackson Scott	Auto Loan	71806.17	2.93	07-DEC-24	25-AUG-29
8	574	Aria Hughes	Auto Loan	70692.29	3.16	11-JUL-24	11-APR-28
9	774	Diego Hughes	Auto Loan	9976.88	4.93	21-MAY-23	18-NOV-27
10	789	Eva Hughes	Personal Loan	-18106.16	8.39	17-AUG-24	22-FEB-29
11	594	Liam Hughes	Auto Loan	72989.73	3.99	11-JUN-24	15-APR-29
12	886	Diego Murphy	Personal Loan	66656.51	6.58	14-APR-24	08-MAR-26
13	109	Ariana Hughes	Home Loan	2683.09	5.33	14-DEC-23	23-DEC-27
14	978	Eva Sullivan	Business Loan	15014.95	4.03	05-FEB-23	17-NOV-24
15	523	Eva Sullivan	Personal Loan	23771.9	2.95	10-APR-24	12-FEB-26

Figure 3.5: Loan Details



```
SELECT * FROM ServiceRequestStatus;
```

t Output × | Query Result × | Query Result 1 × | Query Result 2 ×

SQL | Fetched 50 rows in 0.094 seconds

REQUEST_ID	CUSTOMER_ID	CUSTOMER_NAME	REQUEST_TYPE	STATUS	REQUEST_DATE	RESOLUTION_DATE
1	196	Aria Murphy	Account Issue	Closed	31-DEC-23	22-JAN-24
2	824	Eva Hughes	Account Issue	In Progress	14-APR-24	(null)
3	782	Eva Myers	Account Issue	Closed	07-MAY-24	03-JUN-24
4	57	Maria Brooks	Loan Inquiry	Closed	28-JAN-24	04-FEB-24
5	77	Grayson Alexander	Technical Support	Resolved	14-JAN-24	07-FEB-24
6	194	Eva Hughes	Loan Inquiry	Closed	21-DEC-23	29-DEC-23
7	851	Diego Murphy	Technical Support	In Progress	03-NOV-23	(null)
8	809	Diego Hughes	General Inquiry	Resolved	03-JUN-24	18-JUN-24
9	284	Diego Hughes	Technical Support	In Progress	16-JUL-23	(null)
10	189	Aria Hughes	Technical Support	In Progress	08-APR-24	(null)
11	381	Jade Murphy	Technical Support	Open	22-APR-24	(null)
12	320	Eva Bell	Technical Support	Open	28-NOV-23	(null)
13	791	Aria Murphy	General Inquiry	Closed	10-MAY-24	13-MAY-24

Figure 3.6: Service Request Details

3.3 Results Overall Discussion

Here I am showing the results of some views. I am not showing details here, Because details are huge. There are in some tables 10000 rows available. Showing everything will be expensive.

3.3.1 Complex Engineering Problem Discussion

- The development of a Bank Management System involves tackling a range of intricate engineering challenges that span various domains. These challenges are pivotal in shaping the architecture, functionality, and overall effectiveness of the system.
- P1: Depth of Knowledge Required in Database Management Systems (DBMS): Implementing a robust bank management system necessitates a profound understanding of DBMS principles. Choosing an appropriate database, such as Oracle, involves considerations like scalability, data integrity, and transaction management. Optimizing database queries, ensuring ACID (Atomicity, Consistency, Isolation, Durability) properties, and designing efficient data models are crucial to meet performance and reliability requirements.
- P2: Balancing Conflicting Requirements: The project encounters conflicting requirements that demand careful trade-offs. For instance, ensuring stringent security measures while maintaining high system availability poses a challenge. Balancing regulatory compliance with the need for operational flexibility and user convenience requires meticulous design and strategic decision-making.
- P3: Depth of Analysis Required for Data Analytics and Decision-Making: Incorporating data analytics capabilities into the system involves complex analysis of transaction patterns, customer behavior, and risk assessment. Implementing algorithms for fraud detection, credit scoring, and financial forecasting requires sophisticated data processing techniques and statistical models. Integrating these analyses seamlessly into operational workflows enhances decision-making accuracy and operational efficiency.
- P4: Familiarity with Financial and Banking Operations: A deep understanding of financial operations and banking regulations is essential. Designing features like loan management, interest calculation, and compliance reporting demands adherence to industry standards such as Basel III. Integrating these functionalities while ensuring legal compliance and operational efficiency is critical for the system's success.
- P5: Adherence to Applicable Codes, Standards, and Regulations: Meeting industry standards and regulatory requirements, such as GDPR (General Data Protection Regulation) and PCI DSS (Payment Card Industry Data Security Standard), adds complexity. Implementing robust data encryption, access controls, and audit trails to safeguard sensitive information while ensurin operational continuity is imperative.

Chapter 4

Conclusion

4.1 Discussion

The development of the Bank Management System demonstrates the importance of integrating advanced technological solutions with financial institutions' operational needs. Throughout the project, critical engineering challenges such as database management, regulatory compliance, and stakeholder alignment were addressed methodically. The system ensures data security and regulatory compliance by utilizing robust database systems like Oracle and adhering to industry standards such as GDPR and PCI DSS. The project's success in balancing conflicting requirements, improving operational efficiency, and integrating complex functionalities demonstrates its potential to transform banking operations. Moving forward, continuous evaluation and adaptation will be critical for meeting evolving technological trends and regulatory landscapes, ensuring the financial services industry's long-term relevance and effectiveness.

4.2 Limitations

While the Bank Management System represents a significant step forward in automating and optimizing banking processes, it is important to note a few limitations. For starters, rapid increases in transaction volume or customer base may put the system's scalability under strain, necessitating ongoing infrastructure upgrades. Second, reliance on external factors like network stability and third-party service providers may have an impact on system reliability. Furthermore, the initial implementation may necessitate extensive training for bank employees in order to fully leverage its capabilities, potentially affecting immediate operational efficiency. Addressing these limitations is critical to ensuring the system's effectiveness and seamless integration into existing banking frameworks.

4.3 Scope of Future Work

The Bank Management System provides a solid foundation for future enhancements and expansions. Future work could include integrating advanced analytics to provide realtime insights into customer behavior and financial trends, thereby improving decision-making processes. Implementing artificial intelligence and machine learning algorithms could help to automate routine tasks, optimize resource allocation, and detect fraudulent activities more accurately. Furthermore, investigating blockchain technology for improved transaction security and the development of mobile banking applications may increase service accessibility and customer engagement. Continuous refinement and adaptation to changing regulatory requirements and technological advancements will keep the system at the cutting edge of modern banking operations. .

Chapter 5

References

- GeeksForGeeks website
- Rick van der Lans, SQL for MySQL Developers
- Kevin Loney, Oracle Database