LIBRARY MANAGEMENT SYSTEM

UCS310 DataBase Management System Project Report

Submitted by:

(102117150) KASHISH

(102117158) DRISHTI

(102117161) CHELSI

(102117165) SHATAKSHI

BE Second Year, CSE

Group No: 2CS6

Submitted to:

Dr. Geeta Kasana

Computer Science and Engineering Department

TIET, Patiala

May 2023

TABLE OF CONTENTS

S.No.	Topic	Page no
1	Introduction	3
2	Requirement Analysis	4
3	ER diagram	5
4	ER to Tables	6
5	Normalised tables	7
6	Normalisation	8
7	Functional Dependencies	9
8	SQL/PL-SQL Queries	16
9	Screenshots of outputs	34
10	Conclusion	60

INTRODUCTION

The Library Management System is a software application that facilitates the management of books and videos in a library with multiple branches. Each branch can be located in different geographical locations and can hold a varying number of books and videos. Users of the system can be classified as either employees or customers. The library can have multiple employees working in each branch, and each customer is issued a card that allows them to borrow a specified number of items.

The software application has a Book Issue form that records the issuance and return dates of borrowed books. The due date for returning the book is also recorded in the system. This feature ensures that customers do not exceed the borrowing period and incur fines for late returns. The application enables customers to rent any number of items within the borrowing limit specified by the library.

The system is designed to cater to the needs of library administrators, employees, and customers. The administrators can use the application to manage the library's inventory, monitor the borrowing patterns, and generate reports on the usage of library resources. Employees can use the system to issue books, record returns, and check the availability of books. Customers can use the application to search for books, reserve them, and view their borrowing history.

In summary, the Library Management System is a comprehensive software application that simplifies the management of books and videos in a library with multiple branches. The system ensures efficient tracking of inventory, issuance and return of books, and borrowing patterns. The application provides an excellent user experience for library administrators, employees, and customers alike.

REQUIREMENT ANALYSIS

Requirement analysis is a crucial step in the development of the Library Management System, which is designed to manage the receipt and issuance of books in the library. The following requirements need to be analyzed for the successful development of the software application:

Functional Requirements:

a) User Management: The system should have different user types, including librarian, customer, and administrator. The system should provide access control based on user roles and privileges.

b) Inventory Management: The system should manage the library's inventory by keeping track of books and videos in each branch. The system should also enable the librarian to add, modify, and delete books and videos from the inventory.

c) Borrowing Management: The system should enable customers to borrow and return books and videos. The system should keep track of the borrowing history, due dates, and fines for late returns.

Non-Functional Requirements:

a) Security: The system should have robust security features to prevent unauthorized access to the database and sensitive information.

b) Performance: The system should be fast and responsive, with minimum latency.

c) Reliability: The system should be reliable and available 24/7, with minimum downtime.

Data Requirements:

a) Book and Video Records: The system should have a database to store information about books and videos, including the title, author, publisher, publication year, ISBN, and genre.

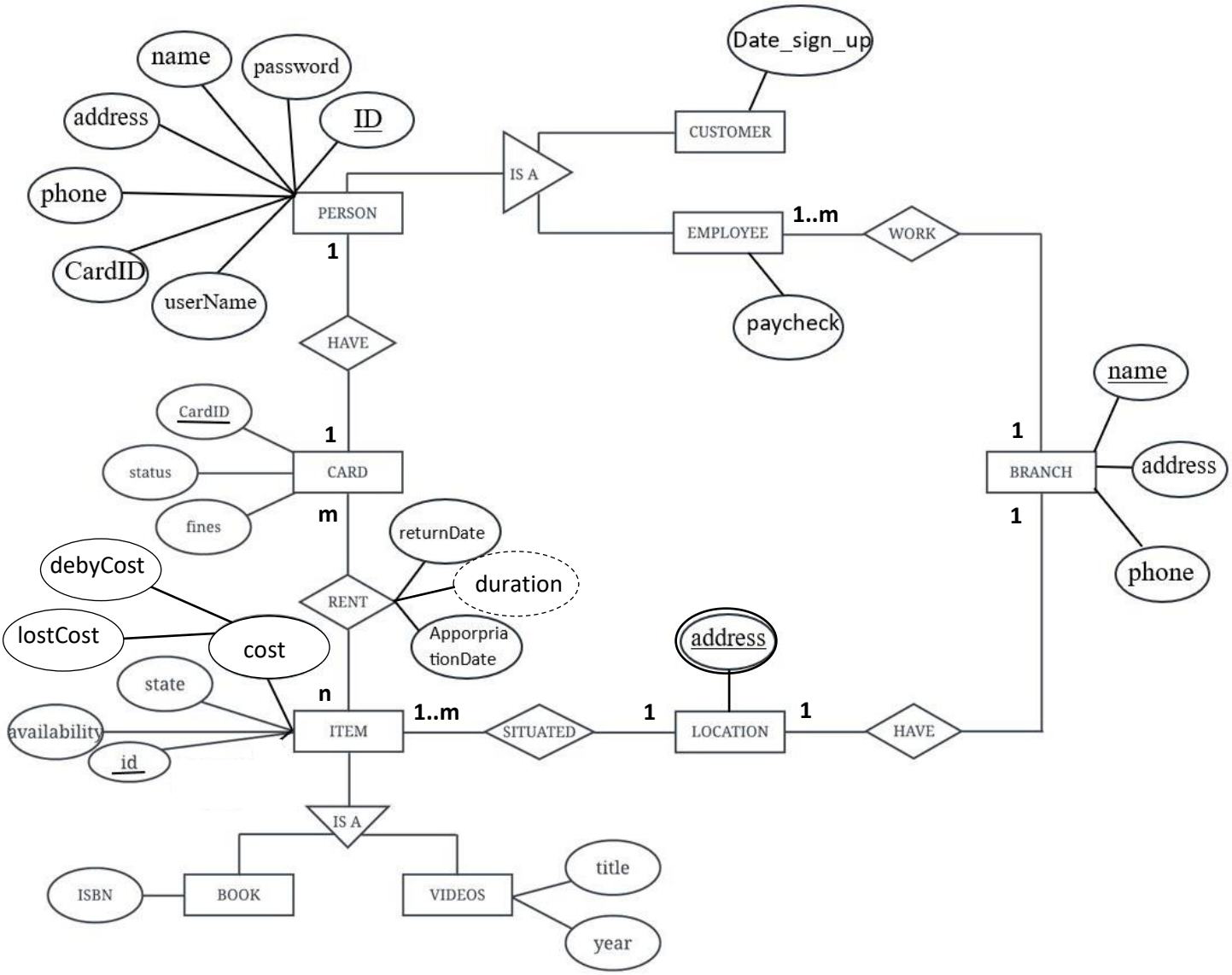
b) Customer Records: The system should have a database to store information about customers, including their name, address, contact details, and borrowing history.

c) Transaction Records: The system should have a database to record transactions such as borrowing, returning, and reserving books and videos.

d) Branch Records: The system should have a database to store information about the library branches, including their location, address, and contact details.

In conclusion, the Library Management System should meet the above functional, non-functional, and data requirements to manage the library's inventory, borrowing and reservation, and reporting effectively. The software application should also be secure, reliable, fast, user-friendly, and scalable to meet the needs of the library administrators, employees, and customers.

ER-DIAGRAM


name
password
address
ID
phone
CardID
userName
PERSON
1
HAVE
Date_sign_up
CUSTOMER
IS A
EMPLOYEE
1..m
WORK
paycheck
CardID
status
fines
CARD
m
1
returnDate
duration
RENT
Apporpria
tionDate
debyCost
lostCost
cost
state
availability
id
n
ITEM
1..m
SITUATED
1
address
LOCATION
1
HAVE
1
1
name
address
phone
BRANCH
IS A
ISBN
BOOK
VIDEOS
title
year

ER to Table (Denormalised)

● **CARD** (CardID, fines, status)

● **PERSON** (ID, name, address, phone, CardID (references CARD(CardID)), userName, password)

● **CUSTOMER** (customerID, name, address, phone, cardNumber (References CARD(CardID)), password, userName, dateSignUp)

● **HAVE** (ID (References Person(ID)), CardID (References CARD (CardID)))

● **ITEM** (itemID, state, availability, debyCost, lostCost)

● **EMPLOYEE** (employeeID, name, address, phone, cardNumber (References CARD(CardID)), password, userName, paycheck, branchName (References BRANCH(name)))

● **BRANCH** (name, address (References LOCATION(address)]), phone)

● **WORK** (employeeID (References Employee(ID)) , name (References Branch(name)))

● **LOCATION** (address)

● **SITUATED** (itemID (References BOOK(bookID) or VIDEO(videoID)), address (References Location(address)))

● **RENT** (CardID (References CARD(CardID)), itemID (References BOOK(book_ID)) apporpriationDate, returnDate, duration int)

● **BOOK** (ISBN, bookID, state, availability, debyCost, lostCost, address (References LOCATION(address)))

● **VIDEO** (title, year, videoID, state, availability, debyCost, lostCost, address (References LOCATION (address)))

NORMALISED TABLES

● **CARD** (CardID, fines, status)

● **CUSTOMER** (customerID, name, address, phone, cardNumber (References CARD(CardID)), password, userName, dateSignUp)

● **EMPLOYEE** (employeeID, name, address, phone, cardNumber (References CARD(CardID)), password, userName, paycheck, branchName (References BRANCH(name)))

● **BRANCH** (name, address (References LOCATION(address)]), phone)

● **LOCATION** (address)

● **RENT** (CardID (References CARD(CardID)), itemID (References BOOK(book_ID)) apporpriationDate, returnDate)

● **BOOK** (ISBN, bookID, state, availability, debyCost, lostCost, address (References LOCATION (address)))

● **VIDEO** (title, year, videoID, state, availability, debyCost, lostCost, address (References LOCATION (address)))

FUNCTIONAL DEPENDENCIES:

Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

NORMALISATION

Normalisation is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate keyof table.

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super keyshould be removed.

In other words, 3NF can be explained like this: A table is in 3NF if it is in 2NF and foreach functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

CARD

```
SQL> CREATE TABLE Card(
  2  cardID NUMBER,
  3  status VARCHAR2(1) CHECK ((status = 'A') OR (status = 'B')),
  4  fines NUMBER,
  5* CONSTRAINT Card_PK PRIMARY KEY (cardID));

Table CARD created.
```

```
SQL> desc Card;

     Name       Null?           Type
--------- ----------- --------------
CARDID    NOT NULL    NUMBER
STATUS                VARCHAR2(1)
FINES                 NUMBER
```

Functional Dependency: CardID -> fines,status

Candidate Keys: CardID

Non-prime attributes: fines, status

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

Boyce-Codd NF: Since every field of the table is dependent only on primary key, it is in BCNF.

4NF: The table is in BCNF and has no multi-valued dependency, so it is in 4th Normal form.

5NF: There is no Lossless Decomposition, so it is in 5th Normal form.

CUSTOMER

```
SQL> CREATE TABLE Customer(
  2  customerID NUMBER,
  3  name VARCHAR2(40),
  4  customerAddress VARCHAR2(50),
  5  phone NUMBER(10),
  6  password VARCHAR2(20),
  7  userName VARCHAR2(30),
  8  dateSignUp DATE,
  9  cardNumber NUMBER REFERENCES Card(cardID),
 10* CONSTRAINT Customer_PK PRIMARY KEY (customerID));

Table CUSTOMER created.
```

```
SQL> desc Customer;

             Name       Null?            Type
------------------ ----------- ---------------
CUSTOMERID          NOT NULL    NUMBER
NAME                            VARCHAR2(40)
CUSTOMERADDRESS                 VARCHAR2(50)
PHONE                           NUMBER(10)
PASSWORD                        VARCHAR2(20)
USERNAME                        VARCHAR2(30)
DATESIGNUP                      DATE
CARDNUMBER                      NUMBER
```

Functional Dependency: customerID -> name, address, phone, password, userName, dateSignUp

Candidate Keys: customerID

Non-prime attributes: name, address, phone, password, userName, dateSignUp, cardID

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

Boyce-Codd NF: Since every field of the table is dependent only on primary key, it is in BCNF.

4NF: The table is in BCNF and has no multi-valued dependency, so it is in 4th Normal form.

5NF: There is no Lossless Decomposition, so it is in 5th Normal form.

LOCATION

```
SQL> CREATE TABLE Location(
  2  address VARCHAR2(50),
  3* CONSTRAINT Location_PK PRIMARY KEY (address));

Table LOCATION created.
```

```
SQL> desc Location;

      Name       Null?            Type
---------- ----------- ---------------
ADDRESS    NOT NULL    VARCHAR2(50)
```

Candidate Keys: address

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

Boyce-Codd NF: Since every field of the table is dependent only on primary key, it is in BCNF.

4NF: The table is in BCNF and has no multi-valued dependency, so it is in 4th Normal form.

5NF: There is no Lossless Decomposition, so it is in 5th Normal form.

BRANCH

```
SQL> CREATE TABLE Branch(
  2  name VARCHAR2(40),
  3  address VARCHAR2(50) REFERENCES Location(address),
  4  phone NUMBER(10),
  5* CONSTRAINT Branch_PK PRIMARY KEY (name));

Table BRANCH created.
```

```
SQL> desc Branch;

      Name       Null?            Type
---------- ---------- ---------------
NAME       NOT NULL    VARCHAR2(40)
ADDRESS                VARCHAR2(50)
PHONE                  NUMBER(10)
```

Functional Dependency: name -> address, phone

Candidate Keys: name

Non-prime attributes: address, phone

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

Boyce-Codd NF: Since every field of the table is dependent only on primary key, it is in BCNF.

4NF: The table is in BCNF and has no multi-valued dependency, so it is in 4th Normal form.

5NF: There is no Lossless Decomposition, so it is in 5th Normal form.

EMPLOYEE

```
SQL> CREATE TABLE Employee(
  2  employeeID NUMBER,
  3  ename VARCHAR2(40),
  4  employeeAddress VARCHAR2(50),
  5  phone NUMBER(10),
  6  password VARCHAR2(20),
  7  userName VARCHAR2(30),
  8  paycheck NUMBER (8, 2),
  9  branchName VARCHAR2(40) REFERENCES Branch(name) ,
 10  cardNumber NUMBER REFERENCES Card(cardID),
 11* CONSTRAINT Employee_PK PRIMARY KEY (employeeID));

Table EMPLOYEE created.
```

```
SQL> desc Employee;

              Name       Null?            Type
------------------ ----------- ---------------
EMPLOYEEID         NOT NULL    NUMBER
ENAME                          VARCHAR2(40)
EMPLOYEEADDRESS                VARCHAR2(50)
PHONE                          NUMBER(10)
PASSWORD                       VARCHAR2(20)
USERNAME                       VARCHAR2(30)
PAYCHECK                       NUMBER(8,2)
BRANCHNAME                     VARCHAR2(40)
CARDNUMBER                     NUMBER
```

Functional Dependency: employeeID -> name, address, phone,password, userName, paycheck

Candidate Keys: employeeID

Non-prime attributes: name, address, phone, cardNumber, password, userName, paycheck, branchName

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

Boyce-Codd NF: Since every field of the table is dependent only on primary key, it is in BCNF.

4NF: The table is in BCNF and has no multi-valued dependency, so it is in 4th Normal form.

5NF: There is no Lossless Decomposition, so it is in 5th Normal form.

BOOK

```
SQL> CREATE TABLE Book(
  2  ISBN VARCHAR2(4),
  3  bookID VARCHAR2(6),
  4  state VARCHAR2(10),
  5  availability VARCHAR2(1) CHECK ((availability = 'A') OR (availability = 'O')),
  6  debyCost NUMBER(10,2),
  7  lostCost NUMBER(10,2),
  8  address VARCHAR2(50) REFERENCES Location(address),
  9* CONSTRAINT Book_PK PRIMARY KEY (bookID));

Table BOOK created.
```

```
SQL> desc Book;

           Name       Null?            Type
--------------- ----------- ---------------
ISBN                        VARCHAR2(4)
BOOKID          NOT NULL    VARCHAR2(6)
STATE                       VARCHAR2(10)
AVAILABILITY                VARCHAR2(1)
DEBYCOST                    NUMBER(10,2)
LOSTCOST                    NUMBER(10,2)
ADDRESS                     VARCHAR2(50)
```

Functional Dependency: bookID -> ISBN,state, availability, debyCost, lostCost

Candidate Keys: bookID

Non-prime attributes: ISBN,state, availability, debyCost, lostCost, address

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

Boyce-Codd NF: Since every field of the table is dependent only on primary key, it is in BCNF.

4NF: The table is in BCNF and has no multi-valued dependency, so it is in 4th Normal form.

5NF: There is no Lossless Decomposition, so it is in 5th Normal form.

RENT

```
SQL> CREATE TABLE Rent(
  2  cardID NUMBER REFERENCES Card(cardID),
  3  itemID VARCHAR2(6) ,
  4  apporpriationDate DATE,
  5  returnDate DATE,
  6* CONSTRAINT Rent_PK PRIMARY KEY (cardID,itemID), constraint rent_fk foreign key (itemID) reference
s Book(bookID));

Table RENT created.
```

```
SQL> desc Rent;

                Name       Null?           Type
____________________ ___________ ______________
CARDID               NOT NULL    NUMBER
ITEMID               NOT NULL    VARCHAR2(6)
APPORPRIATIONDATE                DATE
RETURNDATE                       DATE
```

Functional Dependency: CardID,itemID -> apporpriationDate , returnDate

Candidate Keys: cardID, itemID

Non-prime attributes: apporpriationDate , returnDate

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

Boyce-Codd NF: Since every field of the table is dependent only on primary key, it is in BCNF.

4NF: The table is in BCNF and has no multi-valued dependency, so it is in 4th Normal form.

5NF: There is no Lossless Decomposition, so it is in 5th Normal form.

SQL/PL-SQL COMMANDS

CREATION

```
CREATE TABLE Card(
cardID NUMBER,
status VARCHAR2(1) CHECK ((status = 'A') OR (status = 'B')),
fines NUMBER,
CONSTRAINT Card_PK PRIMARY KEY (cardID));
```

```
CREATE TABLE Customer(
customerID NUMBER,
name VARCHAR2(40),
customerAddress VARCHAR2(50),
phone NUMBER(10),
password VARCHAR2(20),
userName VARCHAR2(30),
dateSignUp DATE,
cardNumber NUMBER REFERENCES Card(cardID),
CONSTRAINT Customer_PK PRIMARY KEY (customerID));
```

```
CREATE TABLE Employee(
employeeID NUMBER,
ename VARCHAR2(40),
employeeAddress VARCHAR2(50),
phone NUMBER(10),
password VARCHAR2(20),
userName VARCHAR2(30),
paycheck NUMBER (8, 2),
branchName VARCHAR2(40) REFERENCES Branch(name) ,
cardNumber NUMBER REFERENCES Card(cardID),
CONSTRAINT Employee_PK PRIMARY KEY (employeeID));
```

```
CREATE TABLE Branch(
name VARCHAR2(40),
address VARCHAR2(50) REFERENCES Location(address),
phone NUMBER(10),
CONSTRAINT Branch_PK PRIMARY KEY (name));
```

```
CREATE TABLE Location(
address VARCHAR2(50),
CONSTRAINT Location_PK PRIMARY KEY (address));
```

```
CREATE TABLE Book(
ISBN VARCHAR2(4),
bookID VARCHAR2(6),
state VARCHAR2(10),
availability VARCHAR2(1) CHECK ((availability = 'A') OR (availability = 'O')),
debyCost NUMBER(10,2),
lostCost NUMBER(10,2),
address VARCHAR2(50) REFERENCES Location(address),
CONSTRAINT Book_PK PRIMARY KEY (bookID));


CREATE TABLE Rent(
cardID NUMBER REFERENCES Card(cardID),
itemID VARCHAR2(6) ,
apporpriationDate DATE,
returnDate DATE,
CONSTRAINT Rent_PK PRIMARY KEY (cardID,itemID), constraint rent_fk foreign key
(itemID) references Book(bookID));
```

DISPLAYING RECORDS

```
SELECT * FROM Card;
SELECT * FROM Customer;
SELECT * FROM Employee;
SELECT * FROM Branch;
SELECT * FROM Location;
SELECT * FROM Book;
SELECT * FROM Rent;
```

DELETION OF TABLE

```
DROP TABLE Card;
DROP TABLE Customer;
DROP TABLE Employee;
DROP TABLE Branch;
DROP TABLE Location;
DROP TABLE Book;
DROP TABLE Rent;
```

INSERTION

```
INSERT INTO Card VALUES (101,'A',0);
INSERT INTO Card VALUES (102,'A',0);
INSERT INTO Card VALUES (103,'A',0);
INSERT INTO Card VALUES (104,'A',0);
INSERT INTO Card VALUES (105,'A',0);
INSERT INTO Card VALUES (106,'A',0);
INSERT INTO Card VALUES (107,'B',50);
INSERT INTO Card VALUES (108,'B',10);
INSERT INTO Card VALUES (109,'B',25.5);
INSERT INTO Card VALUES (110,'B',15.25);
INSERT INTO Card VALUES (111,'A',0);
INSERT INTO Card VALUES (112,'A',0);
INSERT INTO Card VALUES (150,'A',0);
INSERT INTO Card VALUES (158,'A',0);
INSERT INTO Card VALUES (161,'A',0);
INSERT INTO Card VALUES (165,'A',0);
```

```
INSERT INTO Customer VALUES (1, 'Abhi', 'Leela Bhawan', 6236236230, 'abhi123', 'ab1',
'12-05-2022', 101);
INSERT INTO Customer VALUES (2, 'Priya', 'Adarsh Nagar', 6596596509, 'priya123', 'pr1',
'10-01-2023', 102);
INSERT INTO Customer VALUES (3, 'Garima', 'South City', 6546546054, 'garima123',
'ga1', '21-06-2019', 103);
INSERT INTO Customer VALUES (4, 'Tarika', 'Sarojini Nagar.', 6586586058, 'tarika123',
'ta1', '05-12-2022', 104);
INSERT INTO Customer VALUES (5, 'Piyush', 'Sarabha Nagar', 6526052652, 'piyush123',
'pi1', '09-08-2016', 105);
INSERT INTO Customer VALUES (6, 'Janvi', 'Kitchlu Nagar', 6501651651, 'janvi123', 'ja1',
'30-04-2021', 106);
INSERT INTO Customer VALUES (7, 'Riya', 'Dugri Estate I', 6576576507, 'riya123', 'ri1',
'28-02-2020', 107);
INSERT INTO Customer VALUES (8, 'Monica', 'Ram Vihar', 6390639639, 'monica123',
'mo1', '15-01-2021', 108);
INSERT INTO Customer VALUES (9, 'Pihu', 'Pitampura', 6786786708, 'pihu123', 'pi1', '25-
03-2023', 109);
INSERT INTO Customer VALUES (10, 'Avnee', 'Dholakpur', 6087687687, 'avnee123', 'av1',
'01-09-2019', 110);
```

```
INSERT INTO Employee VALUES (211, 'Kashish', 'Dugri', 9876578900, 'kashish123', 'ka1',
12000, 'AI', 150);
INSERT INTO Employee VALUES (212, ' Drishti ', 'Ram Ganga Vihar', 6886886808,
'drishti123', 'dr1', 11500, 'Cyber Security', 158);
INSERT INTO Employee VALUES (213, 'Chelsi', 'Pakhowal Road', 6286208628,
```

'chelsi123', 'ch1', 97500, 'Robotics', 161);
INSERT INTO Employee VALUES (214, 'Shatakshi', 'Krishna Nagar', 6543021987, 'shatakshi123', 'sh14', 22000, 'Cyber Forensics', 165);

INSERT INTO Branch VALUES ('AI', 'AI_LT', 9875674300);
INSERT INTO Branch VALUES ('Robotics', 'ROBO_LT', 8761213190);
INSERT INTO Branch VALUES ('Cyber Security', 'CS_LT', 8788987760);
INSERT INTO Branch VALUES ('Cyber Forensics', 'CF_LT', 9213241220);

INSERT INTO Location VALUES ('AI_LT');
INSERT INTO Location VALUES ('ROBO_LT');
INSERT INTO Location VALUES ('CS_LT');
INSERT INTO Location VALUES ('CF_LT');

INSERT INTO Book VALUES ('A123', 'B1A123', 'GOOD', 'A', 5, 20, 'AI_LT');
INSERT INTO Book VALUES ('A123', 'B2A123', 'NEW', 'O', 6, 30, 'AI_LT');
INSERT INTO Book VALUES ('R234', 'B1R234', 'NEW', 'A', 2, 15, 'ROBO_LT');
INSERT INTO Book VALUES ('C321', 'B1C321', 'BAD', 'A', 1, 10, 'CS_LT');
INSERT INTO Book VALUES ('R123', 'B1R123', 'GOOD', 'A', 3, 15, 'ROBO_LT');
INSERT INTO Book VALUES ('F123', 'B1F123', 'GOOD', 'O', 4, 20, 'CF_LT');
INSERT INTO Book VALUES ('F321', 'B1F321', 'NEW', 'O', 4, 20, 'CF_LT');
INSERT INTO Book VALUES ('R321', 'B1R321', 'USED', 'A', 2, 12, 'ROBO_LT');

INSERT INTO Rent VALUES (101, 'B2A123', '10-06-2022', '15-06-2022');
INSERT INTO Rent VALUES (102, ' B1R234', '14-01-2023', '18-01-2023');
INSERT INTO Rent VALUES (104, ' B1F123', '30-06-2020', '05-07-2020');
INSERT INTO Rent VALUES (105, ' B1R321', '06-04-2023', '18-04-2023');
INSERT INTO Rent VALUES (150, ' B1C321', '04-02-2023', '20-02-2023');
INSERT INTO Rent VALUES (158, ' B1A123', '29-04-2022', '17-05-2022');

SQL Query to display details of a rented book

```
SQL> SELECT * FROM BOOK B, RENT R WHERE B.BOOKID=R.ITEMID AND BOOKID='B2A123';

   ISBN    BOOKID    STATE    AVAILABILITY    DEBYCOST    LOSTCOST    ADDRESS    CARDID    ITEMID    APPORPRIATIONDATE    RETURNDATE
------- --------- -------- --------------- ----------- ----------- ---------- --------- --------- -------------------- -------------
A123    B2A123    NEW      O                         6          30 AI_LT           101 B2A123    10-06-22             15-06-22
```

SQL Query to display name, id, address of a particular employee

```
SQL> SELECT ENAME,EMPLOYEEID,ADDRESS FROM EMPLOYEE E, BRANCH B WHERE E.BRANCHNAME=B.NAME AND E.EMPLOYEEID = 211;

     ENAME    EMPLOYEEID    ADDRESS
---------- ------------- ----------
Kashish              211 AI_LT
```

SQL Query to display details of a particular book

```
SQL> SELECT * FROM BOOK WHERE BOOKID='B2A123';

   ISBN    BOOKID    STATE    AVAILABILITY    DEBYCOST    LOSTCOST    ADDRESS
------- --------- -------- --------------- ----------- ----------- ----------
A123    B2A123    NEW      O                         6          30 AI_LT
```

SQL Query to display selected details for a particular book

```
SQL> SELECT BOOKID,CUSTOMERID,NAME,CARDID,ITEMID,RETURNDATE FROM BOOK B, CUSTOMER C, RENT R WHERE B.BOOKID=R.ITEMID AND R.CARDID=C.CARDNUMBER AND BOOKID='B2A123';

   BOOKID    CUSTOMERID    NAME    CARDID    ITEMID    RETURNDATE
--------- ------------- ------- --------- --------- -------------
B2A123                1 Abhi          101 B2A123    15-06-22
```

PROCEDURES / TRIGGERS/ CURSORS

1. Procedure to login the customer .

```
CREATE OR REPLACE PROCEDURE loginCustomer_library(user IN VARCHAR2, pass
IN VARCHAR2)
IS
passAux customer.password%TYPE;
incorrect_password EXCEPTION;
BEGIN
SELECT password INTO passAux
FROM customer
WHERE username LIKE user;
IF passAux LIKE pass THEN
DBMS_OUTPUT.PUT_LINE('User ' || user || ' loging succesfull');
ELSE
RAISE incorrect_password;
END IF;
EXCEPTION
WHEN no_data_found OR incorrect_password THEN
DBMS_OUTPUT.PUT_LINE('Incorrect username or password');
END;
```

```
DECLARE
user customer.userName%TYPE;
pass customer.password%TYPE;
BEGIN
user := '&username';
pass := '&password';
loginCustomer_library(user,pass);
END;
/
```

2. Procedure to login the employee.

```
CREATE OR REPLACE PROCEDURE loginEmployee_library(user IN VARCHAR2, pass
IN VARCHAR2)
IS
passAux employee.password%TYPE;
incorrect_password EXCEPTION;
BEGIN
SELECT password INTO passAux
FROM employee
```

```
WHERE username LIKE user;
IF passAux LIKE pass THEN
DBMS_OUTPUT.PUT_LINE('User ' || user || ' login succesfull');
ELSE
RAISE incorrect_password;
END IF;
EXCEPTION
WHEN no_data_found OR incorrect_password THEN
DBMS_OUTPUT.PUT_LINE('Incorrect username or password');
END;
SET SERVEROUTPUT ON;
DECLARE
user employee.userName%TYPE;
pass employee.password%TYPE;
BEGIN
user := '&Username';
pass := '&Password';
loginEmployee_library(user,pass);
END;
/
```

3. Procedure to view item details.

```
CREATE OR REPLACE PROCEDURE viewItem_library(auxItemID IN VARCHAR2)
IS
auxISBN VARCHAR2(4);
auxTitle VARCHAR2(50);
auxYear NUMBER;
auxState VARCHAR2(10);
auxDebyCost NUMBER(10,2);
auxLostCost NUMBER(10,2);
auxAddress VARCHAR2(50);
auxAbala VARCHAR2(1);
auxVideo NUMBER;
auxBook NUMBER;
BEGIN
SELECT COUNT(*) INTO auxBook
FROM book
WHERE bookid LIKE auxItemID;
SELECT COUNT(*) INTO auxVideo
FROM video
WHERE videoid LIKE auxItemID;
IF auxBook > 0 THEN
SELECT isbn, state, availability, debycost, lostcost, address
INTO auxISBN, auxState, auxAbala, auxDebyCost, auxLostCost, auxAddress
FROM book
WHERE bookid LIKE auxItemID;
DBMS_OUTPUT.PUT_LINE('BOOK ' || auxItemID || ' INFO');
DBMS_OUTPUT.PUT_LINE('----------------------------------------');
DBMS_OUTPUT.PUT_LINE('ISBN: ' || auxISBN);
```

```
DBMS_OUTPUT.PUT_LINE('STATE: ' || auxState);
DBMS_OUTPUT.PUT_LINE('AVAILABILITY: ' || auxAbala);
DBMS_OUTPUT.PUT_LINE('DEBY COST: ' || auxDebyCost);
DBMS_OUTPUT.PUT_LINE('LOST COST: ' || auxLostCost);
DBMS_OUTPUT.PUT_LINE('ADDRESS: ' || auxAddress);
DBMS_OUTPUT.PUT_LINE('----------------------------------------');
ELSIF auxVideo > 0 THEN
SELECT title, year, state, availability, debycost, lostcost, address
INTO auxTitle, auxYear, auxState, auxAbala, auxDebyCost, auxLostCost, auxAddress
FROM video
WHERE videoid LIKE auxItemID;
DBMS_OUTPUT.PUT_LINE('VIDEO ' || auxItemID || ' INFO');
DBMS_OUTPUT.PUT_LINE('----------------------------------------');
DBMS_OUTPUT.PUT_LINE('TITLE: ' || auxTitle);
DBMS_OUTPUT.PUT_LINE('YEAR: ' || auxYear);
DBMS_OUTPUT.PUT_LINE('STATE: ' || auxState);
DBMS_OUTPUT.PUT_LINE('AVAILABILITY: ' || auxAbala);
DBMS_OUTPUT.PUT_LINE('DEBY COST: ' || auxDebyCost);
DBMS_OUTPUT.PUT_LINE('LOST COST: ' || auxLostCost);
DBMS_OUTPUT.PUT_LINE('ADDRESS: ' || auxAddress);
DBMS_OUTPUT.PUT_LINE('----------------------------------------');
END IF;
END;


SET SERVEROUTPUT ON;
DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := '&Item_ID';
viewItem_library(auxItemID);
END;
```

4. Procedure to view customer's rented books and fines .

```
CREATE OR REPLACE PROCEDURE customerAccount_library(custoID IN
customer.customerid%TYPE)
IS
auxCard NUMBER;
auxFines NUMBER;
auxItem VARCHAR(6);
rented number := 0;
BEGIN
SELECT cardnumber INTO auxCard
FROM customer
WHERE customerid LIKE custoID;
SELECT COUNT(*) INTO rented
FROM rent
WHERE rent.cardid LIKE auxcard;
```

```
DBMS_OUTPUT.PUT_LINE('The user card is ' || auxCard);
IF (rented > 0) THEN
SELECT rent.itemid INTO auxItem
FROM rent,card
WHERE card.cardid = rent.cardid
AND card.cardid LIKE auxCard;
DBMS_OUTPUT.PUT_LINE('The user has ' || auxItem || ' rented');
ELSE
DBMS_OUTPUT.PUT_LINE('This user has no rents');
END IF;
SELECT fines INTO auxFines
FROM card
WHERE cardid LIKE auxcard;
DBMS_OUTPUT.PUT_LINE('The user fines are ' || auxFines);
EXCEPTION WHEN no_data_found THEN
DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
END;


SET SERVEROUTPUT ON;
DECLARE
custoID customer.customerID%TYPE;
BEGIN
custoID := &Customer_ID;
customerAccount_library(custoID);
END;
```

5. Procedure to view employee's rented books and fines .

```
CREATE OR REPLACE PROCEDURE employeeAccount_library(emploID IN
employee.employeeid%TYPE)
IS
auxCard NUMBER;
auxFines NUMBER;
auxItem VARCHAR(6);
rented number := 0;
BEGIN
SELECT cardnumber INTO auxCard
FROM employee
WHERE employeeid LIKE emploID;
SELECT COUNT(*) INTO rented
FROM rent
WHERE rent.cardid LIKE auxcard;
DBMS_OUTPUT.PUT_LINE('The user card is ' || auxCard);
IF (rented > 0) THEN
SELECT rent.itemid INTO auxItem
FROM rent,card
WHERE card.cardid = rent.cardid
AND card.cardid LIKE auxCard;
```

```
DBMS_OUTPUT.PUT_LINE('The user has ' || auxItem || ' rented');
ELSE
DBMS_OUTPUT.PUT_LINE('This user has no rents');
END IF;
SELECT fines INTO auxFines
FROM card
WHERE cardid LIKE auxcard;
DBMS_OUTPUT.PUT_LINE('The user fines are ' || auxFines);
EXCEPTION WHEN no_data_found THEN
DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
END;


SET SERVEROUTPUT ON;
DECLARE
emploID employee.employeeID%TYPE;
BEGIN
emploID := &Employee_ID;
employeeAccount_library(emploID);
END;
```

6. Procedure to rent an item from library.

```
CREATE OR REPLACE PROCEDURE rentItem_library(auxCard IN NUMBER, auxItemID
IN VARCHAR2, itemType IN VARCHAR2, auxDate
IN DATE)
IS
statusAux VARCHAR2(1);
itemStatus VARCHAR2(1);
BEGIN
SELECT status INTO statusAux
FROM card
WHERE cardid LIKE auxCard;
IF statusAux LIKE 'A' THEN
IF itemType LIKE 'book' THEN
SELECT availability INTO itemStatus
FROM book
WHERE bookid LIKE auxItemID;
IF itemStatus LIKE 'A' THEN
UPDATE book
SET availability = 'O'
WHERE bookid LIKE auxItemID;
INSERT INTO rent
VALUES (auxCard,auxItemID,sysdate,auxDate);
DBMS_OUTPUT.PUT_LINE('Item ' || auxItemID || ' rented');
ELSE
DBMS_OUTPUT.PUT_LINE('The item is already rented');
END IF;
ELSIF itemType LIKE 'video' THEN
SELECT availability INTO itemStatus
```

```
FROM video
WHERE videoid LIKE auxItemID;
IF itemStatus LIKE 'A' THEN
UPDATE video
SET availability = 'O'
WHERE videoid LIKE auxItemID;
INSERT INTO rent
VALUES (auxCard,auxItemID,sysdate,auxDate);
DBMS_OUTPUT.PUT_LINE('Item ' || auxItemID || ' rented');
ELSE
DBMS_OUTPUT.PUT_LINE('The item is already rented');
END IF;
ELSE
DBMS_OUTPUT.PUT_LINE('The user is blocked');
END IF;
end if;
END;


SET SERVEROUTPUT ON;
DECLARE
auxCard NUMBER;
auxItemID VARCHAR2(10);
itemType VARCHAR2(20);
auxDate DATE;
BEGIN
auxCard := &Card_ID;
itemType := '&Enter_book';
auxItemID := '&ID_Item';
auxDate := '&Return_date';
rentItem_library(auxCard,auxItemID,itemType,auxDate);
END;
```

7. Procedure to pay fines and collect the change.

```
CREATE OR REPLACE PROCEDURE payFines_library(auxCard IN card.cardid%TYPE,
money IN NUMBER)
IS
finesAmount NUMBER;
total NUMBER;
BEGIN
SELECT fines INTO finesAmount
FROM card
WHERE cardid LIKE auxCard;
IF finesAmount < money THEN
total := money - finesAmount;
DBMS_OUTPUT.PUT_LINE('ALL FINES HAVE BEEN PAID. PLEASE COLLECT
CHANGE OF ' || total || ' RS.');
UPDATE card
SET status = 'A', fines = 0
```

```
WHERE cardid = auxCard;
ELSIF finesAmount = money THEN
total := money - finesAmount;
DBMS_OUTPUT.PUT_LINE('ALL FINES PAID.');
UPDATE card
SET status = 'A', fines = 0
WHERE cardid = auxCard;
ELSE
total := finesAmount - money;
DBMS_OUTPUT.PUT_LINE('YOU WILL NEED TO PAY ' || total || ' RS. MORE TO
UNLOCK YOUR CARD.');
UPDATE card
SET fines = total
WHERE cardid = auxCard;
END IF;
END;


SET SERVEROUTPUT ON;
DECLARE
auxCard card.cardID%TYPE;
money NUMBER;
BEGIN
auxCard := &Card_ID;
money := &Money_To_Pay;
payFines_library(auxCard,money);
END;
```

8. Procedure to update customer information.

```
CREATE OR REPLACE PROCEDURE updateInfoCusto_library(auxCustomer IN
customer.customerID%TYPE, pNumber NUMBER, address
VARCHAR2, newPass VARCHAR2)
IS
BEGIN
UPDATE customer
SET phone = pNumber, customerAddress = address, password = newPass
WHERE customerID = auxCustomer;
END;

SET SERVEROUTPUT ON;
DECLARE
auxCustomer customer.customerID%TYPE;
pNumber NUMBER;
address VARCHAR2(30);
newPass VARCHAR2(30);
BEGIN
auxCustomer := &Customer_ID;
pNumber := &Write_new_old_ForNoChange_contact;
address := '&Write_new_old_ForNoChange_address';
```

```
newPass := '&Write_new_old_ForNoChange_password';
updateInfoCusto_library(auxCustomer,pNumber,address,newPass);
END;

select * from customer;
```

9. Procedure to update employee information.

```
CREATE OR REPLACE PROCEDURE updateInfoEmp_library(auxEmployee IN
employee.employeeID%TYPE, pNumber NUMBER, address
VARCHAR2, newPass VARCHAR2, newPayCheck NUMBER,
newBranch VARCHAR2)
IS
BEGIN
UPDATE employee
SET phone = pNumber, employeeAddress = address, password = newPass, paycheck =
newPayCheck, branchName = newBranch
WHERE employeeID = auxEmployee;
END;


SET SERVEROUTPUT ON;
DECLARE
auxEmployee employee.employeeID%TYPE;
pNumber NUMBER;
address VARCHAR2(30);
newPass VARCHAR2(30);
newPayCheck NUMBER;
newBranch VARCHAR2(30);
BEGIN
auxEmployee := &emp_ID;
pNumber := &Write_new_old_ForNoChange_contact;
address := '&Write_new_old_ForNoChange_address';
newPass := '&Write_new_old_ForNoChange_password';
newPayCheck := &Write_new_old_ForNoChange_paycheck;
newBranch := '&Write_new_old_ForNoChange_branch';
updateInfoEmp_library(auxEmployee,pNumber,address,newPass,newPayCheck,newBranch);
END;

select * from employee;
```

10. Procedure to add a customer in database.

```
CREATE OR REPLACE PROCEDURE addCustomer_library(auxCustomerId IN
NUMBER, auxName IN VARCHAR2, auxCustomerAddress IN
VARCHAR2, auxPhone IN NUMBER,
auxPass IN VARCHAR2, auxUserName IN VARCHAR2, auxCardNumber IN NUMBER)
```

```
IS
BEGIN
INSERT INTO customer
VALUES
(auxCustomerId,auxName,auxCustomerAddress,auxPhone,auxPass,auxUserName,sysdate,au
xCardNumber);
END;


SET SERVEROUTPUT ON;
DECLARE
auxCustomerId NUMBER;
auxName VARCHAR2(20);
auxCustomerAddress VARCHAR2(20);
auxPhone NUMBER;
auxPass VARCHAR2(20);
auxUserName VARCHAR2(20);
auxCardNumber NUMBER;
BEGIN
auxCustomerId := &Customer_ID;
auxName := '&Name';
auxCustomerAddress := '&Address';
auxPhone := &Phone;
auxPass := '&Password';
auxUserName := '&User_Name';
auxCardNumber := &Card_Numeber;
addCustomer_library(auxCustomerId,auxName,auxCustomerAddress,auxPhone,auxPass,aux
UserName,auxCardNumber);
END;

select * from customer;
```

11. Trigger to add a customer card.

```
CREATE OR REPLACE TRIGGER addCardCusto_library
AFTER INSERT
ON customer
FOR EACH ROW
DECLARE
BEGIN
INSERT INTO card
VALUES (:new.cardnumber,'A',0);
DBMS_OUTPUT.PUT_LINE('Card created');
END;
```

12. Trigger to add an employee card.

```
CREATE OR REPLACE TRIGGER addCardEmp_library
```

```
AFTER INSERT
ON employee
FOR EACH ROW
DECLARE
BEGIN
INSERT INTO card
VALUES (:new.cardnumber,'A',0);
DBMS_OUTPUT.PUT_LINE('Card created');
END;


INSERT INTO customer
VALUES (23,'Yash' ,'Lajpat Nagar',6045892456,'yash123','ya1',sysdate,119);
```

13. Cursor to display item information.

```
CREATE OR REPLACE PROCEDURE allMedia_library(mediaType VARCHAR2)
IS
CURSOR cBooks
IS
SELECT *
FROM book;
CURSOR cVideos
IS
SELECT *
FROM video;
xBooks cBooks%ROWTYPE;
xVideos cVideos%ROWTYPE;
BEGIN
IF mediaType LIKE 'books' THEN
OPEN cBooks;
DBMS_OUTPUT.PUT_LINE('ISBN ID STATE AVAILABILITY DEBY_COST
LOST_COST LOCATION');
DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------
---');
LOOP
FETCH cBooks
INTO xBooks;
EXIT WHEN cBooks%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(xBooks.isbn || ' ' || xBooks.bookid || ' ' || xBooks.state || ' ' ||
xBooks.availability || ' ' || xBooks.debycost || ' ' ||
xBooks.lostcost || ' ' || xBooks.address);
END LOOP;
ELSIF mediaType LIKE 'videos' THEN
OPEN cVideos;
DBMS_OUTPUT.PUT_LINE('TITLE YEAR ID STATE AVAILABILITY DEBY_COST
LOST_COST LOCATION');
DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------
-------------');
LOOP
FETCH cVideos
```

```
INTO xVideos;
EXIT WHEN cVideos%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(xVideos.title || ' ' || xVideos.year || ' ' || xVideos.videoid || ' ' ||
xVideos.state || ' ' || xVideos.availability || ' ' || xVideos.debycost || ' ' ||
xVideos.lostcost || ' ' || xVideos.address);
END LOOP;
CLOSE cBooks;
CLOSE cVideos;
ELSE
DBMS_OUTPUT.PUT_LINE('TYPE INCORRECT, please enter book');
END IF;
END;


SET SERVEROUTPUT ON;
DECLARE
typeItem VARCHAR2(10);
BEGIN
typeItem := '&enter_book';
allMedia_library(typeItem);
END;
```

14. Procedure for returning an item.

```
CREATE OR REPLACE PROCEDURE handleReturns_library(auxItemID IN VARCHAR2)
IS
auxRented NUMBER;
auxBook NUMBER;
auxVideo NUMBER;
BEGIN
SELECT COUNT(*) INTO auxRented
FROM rent
WHERE itemid LIKE auxItemID;
SELECT COUNT(*) INTO auxBook
FROM book
WHERE bookid LIKE auxItemID;
SELECT COUNT(*) INTO auxVideo
FROM video
WHERE videoid LIKE auxItemID;
IF auxRented > 0 THEN
DELETE FROM rent
WHERE itemid = auxItemID;
IF auxBook > 0 THEN
UPDATE book
SET availability = 'A'
WHERE bookid LIKE auxItemID;
DBMS_OUTPUT.PUT_LINE('The book ' || auxItemID || ' is now available.');
ELSIF auxVideo > 0 THEN
```

```
UPDATE video
SET availability = 'A'
WHERE videoid LIKE auxItemID;
DBMS_OUTPUT.PUT_LINE('The video ' || auxItemID || ' is now available.');
END IF;
ELSE
DBMS_OUTPUT.PUT_LINE('This item is not rented at the moment');
END IF;
EXCEPTION WHEN no_data_found THEN
DBMS_OUTPUT.PUT_LINE('Item ID incorrect');
END;

SET SERVEROUTPUT ON;
DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := '&ItemID_to_return';
handleReturns_library(auxItemID);
END;

SELECT * FROM rent;
SELECT * FROM book;
```

15. Procedure to add a book to library.

```
CREATE OR REPLACE PROCEDURE addBook_library(auxISBN IN VARCHAR2,
auxBookID IN VARCHAR2, auxState IN VARCHAR2,
auxDebyCost IN NUMBER,
auxLostCost IN NUMBER, auxAddress IN VARCHAR2)
IS
BEGIN
INSERT INTO book
VALUES(auxISBN,auxBookID,auxState,'A',auxDebyCost,auxLostCost,auxAddress);
DBMS_OUTPUT.PUT_LINE('Book inserted correctly');
END;


SET SERVEROUTPUT ON;
DECLARE
auxISBN VARCHAR2(4);
auxItemID VARCHAR2(6);
auxState VARCHAR2(10);
auxDebyCost NUMBER(10,2);
auxLostCost NUMBER(10,2);
auxAddress VARCHAR2(50);
BEGIN
auxISBN := '&ISBN';
auxItemID := '&ItemID' ;
auxState := '&State';
auxDebyCost := &Deby_Cost;
```

```
auxLostCost := &Lost_Cost;
auxAddress := '&Location';
addBook_library(auxISBN, auxItemID, auxState, auxDebyCost, auxLostCost, auxAddress);
END;
SELECT * FROM Book;
```

16. Procedure to remove an item from the library.

```
CREATE OR REPLACE PROCEDURE removeItem_library(auxItemID IN VARCHAR2)
IS
auxBook NUMBER;
auxVideo NUMBER;
BEGIN
SELECT COUNT(*) INTO auxBook
FROM book
WHERE bookid LIKE auxItemID;
SELECT COUNT(*) INTO auxVideo
FROM video
WHERE videoid LIKE auxItemID;
IF auxBook > 0 THEN
DELETE FROM book
WHERE bookid LIKE auxItemID;
DBMS_OUTPUT.PUT_LINE('Book removed correctly');
ELSIF auxVideo > 0 THEN
DELETE FROM video
WHERE videoid LIKE auxItemID;
DBMS_OUTPUT.PUT_LINE('Video removed correctly');
END IF;
END;


SET SERVEROUTPUT ON;
DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := '&ItemSEID_to_remove';
removeItem_library(auxItemID);
END;

SELECT * FROM BOOK;
```

17. Procedure to view customer details.

```
CREATE OR REPLACE PROCEDURE viewCustomer_library(auxCustomerID IN
NUMBER)
IS
custoName VARCHAR2(40);
custoAdd VARCHAR2(50);
custoPhone NUMBER(10);
userNaM VARCHAR2(10);
custoDate DATE;
custoCard NUMBER;
BEGIN
SELECT name,customeraddress,phone,username,datesignup,cardnumber
INTO custoName, custoAdd, custoPhone, userNaM, custoDate, custoCard
FROM customer
WHERE customerID LIKE auxCustomerID;
DBMS_OUTPUT.PUT_LINE('CUSTOMER ' || auxCustomerID || ' INFO');
DBMS_OUTPUT.PUT_LINE('----------------------------------------');
DBMS_OUTPUT.PUT_LINE('NAME: ' || custoName);
DBMS_OUTPUT.PUT_LINE('ADDRESS: ' || custoAdd);
DBMS_OUTPUT.PUT_LINE('PHONE: ' || custoPhone);
DBMS_OUTPUT.PUT_LINE('USER NAME: ' || userNaM);
DBMS_OUTPUT.PUT_LINE('DATE OF SIGN UP: ' || custoDate);
DBMS_OUTPUT.PUT_LINE('CARD NUMBER: ' || custoCard);
DBMS_OUTPUT.PUT_LINE('----------------------------------------');
END;



SET SERVEROUTPUT ON;
DECLARE
auxCustomerID NUMBER;
BEGIN
auxCustomerID :=&CustomerID;
viewCustomer_library(auxCustomerID);
END;
```

SCREENSHOTS OF OUTPUTS

Displaying records of tables.

```
SQL> SELECT * FROM Card;

   CARDID    STATUS    FINES
--------- --------- --------
      101 A                0
      102 A                0
      103 A                0
      104 A                0
      105 A                0
      106 A                0
      107 B               50
      108 B               10
      109 B             25.5
      110 B            15.25
      111 A                0
      112 A                0
      150 A                0
      158 A                0

   CARDID    STATUS    FINES
--------- --------- --------
      161 A                0
      165 A                0

16 rows selected.
```

```
SQL> SELECT * FROM Customer;

   CUSTOMERID      NAME    CUSTOMERADDRESS         PHONE      PASSWORD    USERNAME    DATESIGNUP    CARDNUMBER
------------- --------- ------------------ ------------- ------------ ----------- ------------- -------------
            1 Abhi      Leela Bhawan          6236236230 abhi123      ab1         12-05-22                101
            2 Priya     Adarsh Nagar          6596596509 priya123     pr1         10-01-23                102
            3 Garima    South City            6546546054 garima123    ga1         21-06-19                103
            4 Tarika    Sarojini Nagar.       6586586058 tarika123    ta1         05-12-22                104
            5 Piyush    Sarabha Nagar         6526052652 piyush123    pi1         09-08-16                105
            6 Janvi     Kitchlu Nagar         6501651651 janvi123     ja1         30-04-21                106
            7 Riya      Dugri Estate I        6576576507 riya123      ri1         28-02-20                107
            8 Monica    Ram Vihar             6390639639 monica123    mo1         15-01-21                108
            9 Pihu      Pitampura             6786786708 pihu123      pi1         25-03-23                109
           10 Avnee     Dholakpur             6087687687 avnee123     av1         01-09-19                110

10 rows selected.
```

```
SQL> SELECT * FROM Employee;

   EMPLOYEEID       ENAME    EMPLOYEEADDRESS         PHONE        PASSWORD    USERNAME    PAYCHECK         BRANCHNAME    CARDNUMBER
------------- ------------ ------------------ ------------ --------------- ----------- ----------- ------------------ -------------
          211 Kashish      Dugri               9876578900 kashish123      ka1              12000 AI                             150
          212  Drishti     Ram Ganga Vihar     6886886808 drishti123      dr1              11500 Cyber Security                 158
          213 Chelsi       Pakhowal Road       6286208628 chelsi123       ch1              97500 Robotics                       161
          214 Shatakshi    Krishna Nagar       6543021987 shatakshi123    sh14             22000 Cyber Forensics                165
```

```
SQL> SELECT * FROM Branch;

               NAME    ADDRESS         PHONE
------------------ ---------- -------------
AI                 AI_LT         9875674300
Robotics           ROBO_LT       8761213190
Cyber Security     CS_LT         8788987760
Cyber Forensics    CF_LT         9213241220
```

```
SQL> SELECT * FROM Location;

   ADDRESS
----------
AI_LT
CF_LT
CS_LT
ROBO_LT
```

```
SQL> SELECT * FROM Book;

   ISBN    BOOKID    STATE    AVAILABILITY    DEBYCOST    LOSTCOST    ADDRESS
------- --------- -------- -------------- ---------- ---------- ----------
A123    B1A123    GOOD     A                       5          20 AI_LT
A123    B2A123    NEW      O                       6          30 AI_LT
R234    B1R234    NEW      A                       2          15 ROBO_LT
C321    B1C321    BAD      A                       1          10 CS_LT
R123    B1R123    GOOD     A                       3          15 ROBO_LT
F123    B1F123    GOOD     O                       4          20 CF_LT
F321    B1F321    NEW      O                       4          20 CF_LT
R321    B1R321    USED     A                       2          12 ROBO_LT

8 rows selected.
```

```
SQL> SELECT * FROM Rent;

   CARDID    ITEMID    APPORPRIATIONDATE    RETURNDATE
--------- --------- -------------------- -------------
      101 B2A123    10-06-22             15-06-22
      102 B1R234    14-01-23             18-01-23
      104 B1F123    30-06-20             05-07-20
      105 B1R321    06-04-23             18-04-23
      150 B1C321    04-02-23             20-02-23
      158 B1A123    29-04-22             17-05-22

6 rows selected.
```

Procedures / Triggers / Cursors

1. Procedure to login the customer .

```
SQL> CREATE OR REPLACE PROCEDURE loginCustomer_library(user IN VARCHAR2, pass IN VARCHAR2)
  2  IS
  3  passAux customer.password%TYPE;
  4  incorrect_password EXCEPTION;
  5  BEGIN
  6  SELECT password INTO passAux
  7  FROM customer
  8  WHERE username LIKE user;
  9  IF passAux LIKE pass THEN
 10  DBMS_OUTPUT.PUT_LINE('User ' || user || ' loging succesfull');
 11  ELSE
 12  RAISE incorrect_password;
 13  END IF;
 14  EXCEPTION
 15  WHEN no_data_found OR incorrect_password THEN
 16  DBMS_OUTPUT.PUT_LINE('Incorrect username or password');
 17  END;
 18* /

Procedure LOGINCUSTOMER_LIBRARY compiled

SQL> DECLARE
  2  user customer.userName%TYPE;
  3  pass customer.password%TYPE;
  4  BEGIN
  5  user := '&username';
  6  pass := '&password';
  7  loginCustomer_library(user,pass);
  8  END;
  9* /
Enter value for username: ab1
Enter value for password: abhi123
old:DECLARE
user customer.userName%TYPE;
pass customer.password%TYPE;
BEGIN
user := '&username';
pass := '&password';
loginCustomer_library(user,pass);
END;

new:DECLARE
user customer.userName%TYPE;
pass customer.password%TYPE;
BEGIN
user := 'ab1';
pass := 'abhi123';
loginCustomer_library(user,pass);
END;

PL/SQL procedure successfully completed.
```

2. Procedure to login the employee.

```
SQL> CREATE OR REPLACE PROCEDURE loginEmployee_library(user IN VARCHAR2, pass IN VARCHAR2)
  2  IS
  3  passAux employee.password%TYPE;
  4  incorrect_password EXCEPTION;
  5  BEGIN
  6  SELECT password INTO passAux
  7  FROM employee
  8  WHERE username LIKE user;
  9  IF passAux LIKE pass THEN
 10  DBMS_OUTPUT.PUT_LINE('User ' || user || ' login succesfull');
 11  ELSE
 12  RAISE incorrect_password;
 13  END IF;
 14  EXCEPTION
 15  WHEN no_data_found OR incorrect_password THEN
 16  DBMS_OUTPUT.PUT_LINE('Incorrect username or password');
 17  END;
 18* /

Procedure LOGINEMPLOYEE_LIBRARY compiled

SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  user employee.userName%TYPE;
  3  pass employee.password%TYPE;
  4  BEGIN
  5  user := '&Username';
  6  pass := '&Password';
  7  loginEmployee_library(user,pass);
  8  END;
  9* /
Enter value for Username: ka1
Enter value for Password: kashish123
old:DECLARE
user employee.userName%TYPE;
pass employee.password%TYPE;
BEGIN
user := '&Username';
pass := '&Password';
loginEmployee_library(user,pass);
END;

new:DECLARE
user employee.userName%TYPE;
pass employee.password%TYPE;
BEGIN
user := 'ka1';
pass := 'kashish123';
loginEmployee_library(user,pass);
END;
User ka1 login succesfull
```

```
SQL> /
Enter value for Username: owl
Enter value for Password: owl123
old:DECLARE
user employee.userName%TYPE;
pass employee.password%TYPE;
BEGIN
user := '&Username';
pass := '&Password';
loginEmployee_library(user,pass);
END;

new:DECLARE
user employee.userName%TYPE;
pass employee.password%TYPE;
BEGIN
user := 'owl';
pass := 'owl123';
loginEmployee_library(user,pass);
END;
Incorrect username or password


PL/SQL procedure successfully completed.
```

3. Procedure to view item details.

```
SQL> CREATE OR REPLACE PROCEDURE viewItem_library(auxItemID IN VARCHAR2)
  2  IS
  3  auxISBN VARCHAR2(4);
  4  auxTitle VARCHAR2(50);
  5  auxYear NUMBER;
  6  auxState VARCHAR2(10);
  7  auxDebyCost NUMBER(10,2);
  8  auxLostCost NUMBER(10,2);
  9  auxAddress VARCHAR2(50);
 10  auxAbala VARCHAR2(1);
 11  auxVideo NUMBER;
 12  auxBook NUMBER;
 13  BEGIN
 14  SELECT COUNT(*) INTO auxBook
 15  FROM book
 16  WHERE bookid LIKE auxItemID;
 17  SELECT COUNT(*) INTO auxVideo
 18  FROM video
 19  WHERE videoid LIKE auxItemID;
 20  IF auxBook > 0 THEN
 21  SELECT isbn, state, availability, debycost, lostcost, address
 22  INTO auxISBN, auxState, auxAbala, auxDebyCost, auxLostCost, auxAddress
 23  FROM book
 24  WHERE bookid LIKE auxItemID;
 25  DBMS_OUTPUT.PUT_LINE('BOOK ' || auxItemID || ' INFO');
 26  DBMS_OUTPUT.PUT_LINE('----------------------------------------');
 27  DBMS_OUTPUT.PUT_LINE('ISBN: ' || auxISBN);
 28  DBMS_OUTPUT.PUT_LINE('STATE: ' || auxState);
 29  DBMS_OUTPUT.PUT_LINE('AVAILABILITY: ' || auxAbala);
 30  DBMS_OUTPUT.PUT_LINE('DEBY COST: ' || auxDebyCost);
 31  DBMS_OUTPUT.PUT_LINE('LOST COST: ' || auxLostCost);
 32  DBMS_OUTPUT.PUT_LINE('ADDRESS: ' || auxAddress);
 33  DBMS_OUTPUT.PUT_LINE('----------------------------------------');
 34  ELSIF auxVideo > 0 THEN
 35  SELECT title, year, state, availability, debycost, lostcost, address
 36  INTO auxTitle, auxYear, auxState, auxAbala, auxDebyCost, auxLostCost, auxAddress
 37  FROM video
 38  WHERE videoid LIKE auxItemID;
 39  DBMS_OUTPUT.PUT_LINE('VIDEO ' || auxItemID || ' INFO');
 40  DBMS_OUTPUT.PUT_LINE('----------------------------------------');
 41  DBMS_OUTPUT.PUT_LINE('TITLE: ' || auxTitle);
 42  DBMS_OUTPUT.PUT_LINE('YEAR: ' || auxYear);
 43  DBMS_OUTPUT.PUT_LINE('STATE: ' || auxState);
 44  DBMS_OUTPUT.PUT_LINE('AVAILABILITY: ' || auxAbala);
 45  DBMS_OUTPUT.PUT_LINE('DEBY COST: ' || auxDebyCost);
 46  DBMS_OUTPUT.PUT_LINE('LOST COST: ' || auxLostCost);
 47  DBMS_OUTPUT.PUT_LINE('ADDRESS: ' || auxAddress);
 48  DBMS_OUTPUT.PUT_LINE('----------------------------------------');
 49  END IF;
 50  END;
 51* /

Procedure VIEWITEM_LIBRARY compiled
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  auxItemID VARCHAR2(10);
  3  BEGIN
  4  auxItemID := '&Item_ID';
  5  viewItem_library(auxItemID);
  6  END;
  7* /
Enter value for Item_ID: B1A123
old:DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := '&Item_ID';
viewItem_library(auxItemID);
END;

new:DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := 'B1A123';
viewItem_library(auxItemID);
END;
BOOK B1A123 INFO
----------------------------------------
ISBN: A123
STATE: GOOD
AVAILABILITY: A
DEBY COST: 5
LOST COST: 20
ADDRESS: AI_LT
----------------------------------------


PL/SQL procedure successfully completed.
```

4. Procedure to view customer's rented books and fines .

```
SQL> CREATE OR REPLACE PROCEDURE customerAccount_library(custoID IN customer.customerid%TYPE)
  2  IS
  3  auxCard NUMBER;
  4  auxFines NUMBER;
  5  auxItem VARCHAR(6);
  6  rented number := 0;
  7  BEGIN
  8  SELECT cardnumber INTO auxCard
  9  FROM customer
 10  WHERE customerid LIKE custoID;
 11  SELECT COUNT(*) INTO rented
 12  FROM rent
 13  WHERE rent.cardid LIKE auxcard;
 14  DBMS_OUTPUT.PUT_LINE('The user card is ' || auxCard);
 15  IF (rented > 0) THEN
 16  SELECT rent.itemid INTO auxItem
 17  FROM rent,card
 18  WHERE card.cardid = rent.cardid
 19  AND card.cardid LIKE auxCard;
 20  DBMS_OUTPUT.PUT_LINE('The user has ' || auxItem || ' rented');
 21  ELSE
 22  DBMS_OUTPUT.PUT_LINE('This user has no rents');
 23  END IF;
 24  SELECT fines INTO auxFines
 25  FROM card
 26  WHERE cardid LIKE auxcard;
 27  DBMS_OUTPUT.PUT_LINE('The user fines are ' || auxFines);
 28  EXCEPTION WHEN no_data_found THEN
 29  DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
 30  END;
 31* /

Procedure CUSTOMERACCOUNT_LIBRARY compiled
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  custoID customer.customerID%TYPE;
  3  BEGIN
  4  custoID := &Customer_ID;
  5  customerAccount_library(custoID);
  6  END;
  7
  8* /
Enter value for Customer_ID: 1
old:DECLARE
custoID customer.customerID%TYPE;
BEGIN
custoID := &Customer_ID;
customerAccount_library(custoID);
END;

new:DECLARE
custoID customer.customerID%TYPE;
BEGIN
custoID := 1;
customerAccount_library(custoID);
END;
The user card is 101
The user has B2A123 rented
The user fines are 0


PL/SQL procedure successfully completed.
```

5. Procedure to view employee's rented books and fines .

```
SQL> CREATE OR REPLACE PROCEDURE employeeAccount_library(emploID IN employee.employeeid%TYPE)
  2  IS
  3  auxCard NUMBER;
  4  auxFines NUMBER;
  5  auxItem VARCHAR(6);
  6  rented number := 0;
  7  BEGIN
  8  SELECT cardnumber INTO auxCard
  9  FROM employee
 10  WHERE employeeid LIKE emploID;
 11  SELECT COUNT(*) INTO rented
 12  FROM rent
 13  WHERE rent.cardid LIKE auxcard;
 14  DBMS_OUTPUT.PUT_LINE('The user card is ' || auxCard);
 15  IF (rented > 0) THEN
 16  SELECT rent.itemid INTO auxItem
 17  FROM rent,card
 18  WHERE card.cardid = rent.cardid
 19  AND card.cardid LIKE auxCard;
 20  DBMS_OUTPUT.PUT_LINE('The user has ' || auxItem || ' rented');
 21  ELSE
 22  DBMS_OUTPUT.PUT_LINE('This user has no rents');
 23  END IF;
 24  SELECT fines INTO auxFines
 25  FROM card
 26  WHERE cardid LIKE auxcard;
 27  DBMS_OUTPUT.PUT_LINE('The user fines are ' || auxFines);
 28  EXCEPTION WHEN no_data_found THEN
 29  DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
 30  END;
 31
 32* /

Procedure EMPLOYEEACCOUNT_LIBRARY compiled
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  emploID employee.employeeID%TYPE;
  3  BEGIN
  4  emploID := &Employee_ID;
  5  employeeAccount_library(emploID);
  6  END;
  7
  8* /
Enter value for Employee_ID: 211
old:DECLARE
emploID employee.employeeID%TYPE;
BEGIN
emploID := &Employee_ID;
employeeAccount_library(emploID);
END;

new:DECLARE
emploID employee.employeeID%TYPE;
BEGIN
emploID := 211;
employeeAccount_library(emploID);
END;
The user card is 150
The user has B1C321 rented
The user fines are 0


PL/SQL procedure successfully completed.
```

6. Procedure to rent an item from library.

```
SQL> CREATE OR REPLACE PROCEDURE rentItem_library(auxCard IN NUMBER, auxItemID IN VARCHAR2, itemType I
N VARCHAR2, auxDate
  2  IN DATE)
  3  IS
  4  statusAux VARCHAR2(1);
  5  itemStatus VARCHAR2(1);
  6  BEGIN
  7  SELECT status INTO statusAux
  8  FROM card
  9  WHERE cardid LIKE auxCard;
 10  IF statusAux LIKE 'A' THEN
 11  IF itemType LIKE 'book' THEN
 12  SELECT availability INTO itemStatus
 13  FROM book
 14  WHERE bookid LIKE auxItemID;
 15  IF itemStatus LIKE 'A' THEN
 16  UPDATE book
 17  SET availability = 'O'
 18  WHERE bookid LIKE auxItemID;
 19  INSERT INTO rent
 20  VALUES (auxCard,auxItemID,sysdate,auxDate);
 21  DBMS_OUTPUT.PUT_LINE('Item ' || auxItemID || ' rented');
 22  ELSE
 23  DBMS_OUTPUT.PUT_LINE('The item is already rented');
 24  END IF;
 25  ELSIF itemType LIKE 'video' THEN
 26  SELECT availability INTO itemStatus
 27  FROM video
 28  WHERE videoid LIKE auxItemID;
 29  IF itemStatus LIKE 'A' THEN
 30  UPDATE video
 31  SET availability = 'O'
 32  WHERE videoid LIKE auxItemID;
 33  INSERT INTO rent
 34  VALUES (auxCard,auxItemID,sysdate,auxDate);
 35  DBMS_OUTPUT.PUT_LINE('Item ' || auxItemID || ' rented');
 36  ELSE
 37  DBMS_OUTPUT.PUT_LINE('The item is already rented');
 38  END IF;
 39  ELSE
 40  DBMS_OUTPUT.PUT_LINE('The user is blocked');
 41  END IF;
 42  END IF;
 43  END;
 44* /

Procedure RENTITEM_LIBRARY compiled
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  auxCard NUMBER;
  3  auxItemID VARCHAR2(10);
  4  itemType VARCHAR2(20);
  5  auxDate DATE;
  6  BEGIN
  7  auxCard := &Card_ID;
  8  itemType := '&Enter_book';
  9  auxItemID := '&ID_Item';
 10  auxDate := '&Return_date';
 11  rentItem_library(auxCard,auxItemID,itemType,auxDate);
 12  END;
 13* /
Enter value for Card_ID: 101
Enter value for Enter_book: book
Enter value for ID_Item: B1A123
Enter value for Return_date: 17-05-2022
old:DECLARE
auxCard NUMBER;
auxItemID VARCHAR2(10);
itemType VARCHAR2(20);
auxDate DATE;
BEGIN
auxCard := &Card_ID;
itemType := '&Enter_book';
auxItemID := '&ID_Item';
auxDate := '&Return_date';
rentItem_library(auxCard,auxItemID,itemType,auxDate);
END;

new:DECLARE
auxCard NUMBER;
auxItemID VARCHAR2(10);
itemType VARCHAR2(20);
auxDate DATE;
BEGIN
auxCard := 101;
itemType := 'book';
auxItemID := 'B1A123';
auxDate := '17-05-2022';
rentItem_library(auxCard,auxItemID,itemType,auxDate);
END;
Item B1A123 rented


PL/SQL procedure successfully completed.
```

7. Procedure to pay fines and collect the change.

```
SQL> CREATE OR REPLACE PROCEDURE payFines_library(auxCard IN card.cardid%TYPE, money IN NUMBER)
  2  IS
  3  finesAmount NUMBER;
  4  total NUMBER;
  5  BEGIN
  6  SELECT fines INTO finesAmount
  7  FROM card
  8  WHERE cardid LIKE auxCard;
  9  IF finesAmount < money THEN
 10  total := money - finesAmount;
 11  DBMS_OUTPUT.PUT_LINE('ALL FINES HAVE BEEN PAID. PLEASE COLLECT  CHANGE OF ' || total || ' RS.');
 12  UPDATE card
 13  SET status = 'A', fines = 0
 14  WHERE cardid = auxCard;
 15  ELSIF finesAmount = money THEN
 16  total := money - finesAmount;
 17  DBMS_OUTPUT.PUT_LINE('ALL FINES PAID.');
 18  UPDATE card
 19  SET status = 'A', fines = 0
 20  WHERE cardid = auxCard;
 21  ELSE
 22  total := finesAmount - money;
 23  DBMS_OUTPUT.PUT_LINE('YOU WILL NEED TO PAY ' || total || ' RS. MORE TO UNLOCK YOUR CARD.');
 24  UPDATE card
 25  SET fines = total
 26  WHERE cardid = auxCard;
 27  END IF;
 28  END;
 29* /

Procedure PAYFINES_LIBRARY compiled
```

```
Enter value for Card_ID: 102
Enter value for Money_To_Pay: 200
old:DECLARE
auxCard card.cardID%TYPE;
money NUMBER;
BEGIN
auxCard := &Card_ID;
money := &Money_To_Pay;
payFines_library(auxCard,money);
END;

new:DECLARE
auxCard card.cardID%TYPE;
money NUMBER;
BEGIN
auxCard := 102;
money := 200;
payFines_library(auxCard,money);
END;
ALL FINES HAVE BEEN PAID. PLEASE COLLECT  CHANGE OF 200 RS.


PL/SQL procedure successfully completed.

SQL> /
Enter value for Card_ID: 107
Enter value for Money_To_Pay: 100
old:DECLARE
auxCard card.cardID%TYPE;
money NUMBER;
BEGIN
auxCard := &Card_ID;
money := &Money_To_Pay;
payFines_library(auxCard,money);
END;

new:DECLARE
auxCard card.cardID%TYPE;
money NUMBER;
BEGIN
auxCard := 107;
money := 100;
payFines_library(auxCard,money);
END;
ALL FINES HAVE BEEN PAID. PLEASE COLLECT  CHANGE OF 50 RS.


PL/SQL procedure successfully completed.
```

8. Procedure to update customer information.

```
SQL> CREATE OR REPLACE PROCEDURE updateInfoCusto_library(auxCustomer IN customer.customerID%TYPE, pNum
ber NUMBER, address
  2  VARCHAR2, newPass VARCHAR2)
  3  IS
  4  BEGIN
  5  UPDATE customer
  6  SET phone = pNumber, customerAddress = address, password = newPass
  7  WHERE customerID = auxCustomer;
  8  END;
  9* /

Procedure UPDATEINFOCUSTO_LIBRARY compiled
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  auxCustomer customer.customerID%TYPE;
  3  pNumber NUMBER;
  4  address VARCHAR2(30);
  5  newPass VARCHAR2(30);
  6  BEGIN
  7  auxCustomer := &Customer_ID;
  8  pNumber := &Write_new_old_ForNoChange_contact;
  9  address := '&Write_new_old_ForNoChange_address';
 10  newPass := '&Write_new_old_ForNoChange_password';
 11  updateInfoCusto_library(auxCustomer,pNumber,address,newPass);
 12  END;
 13* /
Enter value for Customer_ID: 1
Enter value for Write_new_old_ForNoChange_contact: 8526354230
Enter value for Write_new_old_ForNoChange_address: bacon street
Enter value for Write_new_old_ForNoChange_password: abhi123
old:DECLARE
auxCustomer customer.customerID%TYPE;
pNumber NUMBER;
address VARCHAR2(30);
newPass VARCHAR2(30);
BEGIN
auxCustomer := &Customer_ID;
pNumber := &Write_new_old_ForNoChange_contact;
address := '&Write_new_old_ForNoChange_address';
newPass := '&Write_new_old_ForNoChange_password';
updateInfoCusto_library(auxCustomer,pNumber,address,newPass);
END;

new:DECLARE
auxCustomer customer.customerID%TYPE;
pNumber NUMBER;
address VARCHAR2(30);
newPass VARCHAR2(30);
BEGIN
auxCustomer := 1;
pNumber := 8526354230;
address := 'bacon street';
newPass := 'abhi123';
updateInfoCusto_library(auxCustomer,pNumber,address,newPass);
END;

PL/SQL procedure successfully completed.
```

9. Procedure to update employee information.

```
SQL> CREATE OR REPLACE PROCEDURE updateInfoEmp_library(auxEmployee IN employee.employeeID%TYPE, pNumber NUMBER, address
  2  VARCHAR2, newPass VARCHAR2, newPayCheck NUMBER,
  3  newBranch VARCHAR2)
  4  IS
  5  BEGIN
  6  UPDATE employee
  7  SET phone = pNumber, employeeAddress = address, password = newPass, paycheck = newPayCheck, branchName = newBranch
  8  WHERE employeeID = auxEmployee;
  9  END;
 10* /

Procedure UPDATEINFOEMP_LIBRARY compiled
```

```
Enter value for emp_ID: 213
Enter value for Write_new_old_ForNoChange_contact: 8569745820
Enter value for Write_new_old_ForNoChange_address: ludhiana
Enter value for Write_new_old_ForNoChange_password: chelsi123
Enter value for Write_new_old_ForNoChange_paycheck: 96000
Enter value for Write_new_old_ForNoChange_branch: Robotics
old:DECLARE
auxEmployee employee.employeeID%TYPE;
pNumber NUMBER;
address VARCHAR2(30);
newPass VARCHAR2(30);
newPayCheck NUMBER;
newBranch VARCHAR2(30);
BEGIN
auxEmployee := &emp_ID;
pNumber := &Write_new_old_ForNoChange_contact;
address := '&Write_new_old_ForNoChange_address';
newPass := '&Write_new_old_ForNoChange_password';
newPayCheck := &Write_new_old_ForNoChange_paycheck;
newBranch := '&Write_new_old_ForNoChange_branch';
updateInfoEmp_library(auxEmployee,pNumber,address,newPass,newPayCheck,newBranch);
END;

new:DECLARE
auxEmployee employee.employeeID%TYPE;
pNumber NUMBER;
address VARCHAR2(30);
newPass VARCHAR2(30);
newPayCheck NUMBER;
newBranch VARCHAR2(30);
BEGIN
auxEmployee := 213;
pNumber := 8569745820;
address := 'ludhiana';
newPass := 'chelsi123';
newPayCheck := 96000;
newBranch := 'Robotics';
updateInfoEmp_library(auxEmployee,pNumber,address,newPass,newPayCheck,newBranch);
END;

PL/SQL procedure successfully completed.

SQL> select * from employee;

   EMPLOYEEID       ENAME    EMPLOYEEADDRESS         PHONE        PASSWORD    USERNAME    PAYCHECK         BRANCHNAME    CARDNUMBER
------------ ------------ ------------------ ------------ ---------- ---------- ---------- ------------------ ------------
          211 Kashish      Dugri              9876578900 kashish123 ka1              12000 AI                              150
          212  Drishti     Ram Ganga Vihar    6886886808 drishti123 dr1              11500 Cyber Security                  158
          213 Chelsi       ludhiana           8569745820 chelsi123  ch1              96000 Robotics                        161
          214 Shatakshi    Krishna Nagar      6543021987 shatakshi123 sh14           22000 Cyber Forensics                 165
```

```
SQL> select * from customer;

   CUSTOMERID      NAME    CUSTOMERADDRESS          PHONE     PASSWORD    USERNAME    DATESIGNUP    CARDNUMBER
------------ --------- ------------------ ------------ ---------- ---------- ------------ ------------
           1 Abhi      bacon street       8526354230 abhi123    ab1        12-05-22              101
           2 Priya     Adarsh Nagar       6596596509 priya123   pr1        10-01-23              102
           3 Garima    South City         6546546054 garima123  ga1        21-06-19              103
           4 Tarika    Sarojini Nagar.    6586586058 tarika123  ta1        05-12-22              104
           5 Piyush    Sarabha Nagar      6526052652 piyush123  pi1        09-08-16              105
           6 Janvi     Kitchlu Nagar      6501651651 janvi123   ja1        30-04-21              106
           7 Riya      Dugri Estate I     6576576507 riya123    ri1        28-02-20              107
           8 Monica    Ram Vihar          6390639639 monica123  mo1        15-01-21              108
           9 Pihu      Pitampura          6786786708 pihu123    pi1        25-03-23              109
          10 Avnee     Dholakpur          6087687687 avnee123   av1        01-09-19              110
          20 alex      srilanka           9876543456 alex123    al1        02-05-23              111

11 rows selected.
```

10. Trigger to add a customer card.

11. Trigger to add an employee card.

```
SQL> CREATE OR REPLACE TRIGGER addCardCusto_library
  2  AFTER INSERT
  3  ON customer
  4  FOR EACH ROW
  5  DECLARE
  6  BEGIN
  7  INSERT INTO card
  8  VALUES (:new.cardnumber,'A',0);
  9  DBMS_OUTPUT.PUT_LINE('Card created');
 10  END;
 11* /

Trigger ADDCARDCUSTO_LIBRARY compiled

SQL> CREATE OR REPLACE TRIGGER addCardEmp_library
  2  AFTER INSERT
  3  ON employee
  4  FOR EACH ROW
  5  DECLARE
  6  BEGIN
  7  INSERT INTO card
  8  VALUES (:new.cardnumber,'A',0);
  9  DBMS_OUTPUT.PUT_LINE('Card created');
 10  END;
 11* /

Trigger ADDCARDEMP_LIBRARY compiled
```

```
SQL> INSERT INTO customer
  2* VALUES (23,'Yash' ,'Lajpat Nagar',6045892456,'yash123','ya1',sysdate,119);
Card created


SQL> select * from customer;

   CUSTOMERID      NAME    CUSTOMERADDRESS         PHONE     PASSWORD    USERNAME    DATESIGNUP    CARDNUMBER
------------- --------- ------------------ ------------- ------------ ----------- ------------- -------------
            1 Abhi      bacon street          8526354230 abhi123      ab1         12-05-22                101
            2 Priya     Adarsh Nagar          6596596509 priya123     pr1         10-01-23                102
            3 Garima    South City            6546546054 garima123    ga1         21-06-19                103
            4 Tarika    Sarojini Nagar.       6586586058 tarika123    ta1         05-12-22                104
            5 Piyush    Sarabha Nagar         6526052652 piyush123    pi1         09-08-16                105
            6 Janvi     Kitchlu Nagar         6501651651 janvi123     ja1         30-04-21                106
            7 Riya      Dugri Estate I        6576576507 riya123      ri1         28-02-20                107
            8 Monica    Ram Vihar             6390639639 monica123    mo1         15-01-21                108
            9 Pihu      Pitampura             6786786708 pihu123      pi1         25-03-23                109
           10 Avnee     Dholakpur             6087687687 avnee123     av1         01-09-19                110
           20 alex      srilanka              9876543456 alex123      al1         02-05-23                111
           23 Yash      Lajpat Nagar          6045892456 yash123      ya1         02-05-23                119

12 rows selected.
```

```
SQL> select * from card;

    CARDID    STATUS     FINES
---------- ---------- --------
       101 A                 0
       102 A                 0
       103 A                 0
       104 A                 0
       105 A                 0
       106 A                 0
       107 A                 0
       108 B                10
       109 B              25.5
       110 B             15.25
       111 A                 0
       112 A                 0
       150 A                 0
       158 A                 0

    CARDID    STATUS     FINES
---------- ---------- --------
       161 A                 0
       165 A                 0
       119 A                 0

17 rows selected.
```

12. Cursor to display item information.

```
SQL> CREATE OR REPLACE PROCEDURE allMedia_library(mediaType VARCHAR2)
  2  IS
  3  CURSOR cBooks
  4  IS
  5  SELECT *
  6  FROM book;
  7  CURSOR cVideos
  8  IS
  9  SELECT *
 10  FROM video;
 11  xBooks cBooks%ROWTYPE;
 12  xVideos cVideos%ROWTYPE;
 13  BEGIN
 14  IF mediaType LIKE 'books' THEN
 15  OPEN cBooks;
 16  DBMS_OUTPUT.PUT_LINE('ISBN ID STATE AVAILABILITY DEBY_COST LOST_COST LOCATION');
 17  DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------------
--');
 18  LOOP
 19  FETCH cBooks
 20  INTO xBooks;
 21  EXIT WHEN cBooks%NOTFOUND;
 22  DBMS_OUTPUT.PUT_LINE(xBooks.isbn || ' ' || xBooks.bookid || ' ' || xBooks.state || ' ' ||
 23  xBooks.availability || ' ' || xBooks.debycost || ' ' ||
 24  xBooks.lostcost || ' ' || xBooks.address);
 25  END LOOP;
 26  ELSIF mediaType LIKE 'videos' THEN
 27  OPEN cVideos;
 28  DBMS_OUTPUT.PUT_LINE('TITLE YEAR ID STATE AVAILABILITY DEBY_COST LOST_COST LOCATION');
 29  DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------------
------------');
 30  LOOP
 31  FETCH cVideos
 32  INTO xVideos;
 33  EXIT WHEN cVideos%NOTFOUND;
 34  DBMS_OUTPUT.PUT_LINE(xVideos.title || ' ' || xVideos.year || ' ' || xVideos.videoid || ' ' ||
 35  xVideos.state || ' ' || xVideos.availability || ' ' || xVideos.debycost || ' ' ||
 36  xVideos.lostcost || ' ' || xVideos.address);
 37  END LOOP;
 38  CLOSE cBooks;
 39  CLOSE cVideos;
 40  ELSE
 41  DBMS_OUTPUT.PUT_LINE('TYPE INCORRECT, please enter book');
 42  END IF;
 43  END;
 44* /

Procedure ALLMEDIA_LIBRARY compiled
```

```
SQL> DECLARE
  2  typeItem VARCHAR2(10);
  3  BEGIN
  4  typeItem := '&enter_book';
  5  allMedia_library(typeItem);
  6  END;
  7* /
Enter value for enter_book: books
old:DECLARE
typeItem VARCHAR2(10);
BEGIN
typeItem := '&enter_book';
allMedia_library(typeItem);
END;

new:DECLARE
typeItem VARCHAR2(10);
BEGIN
typeItem := 'books';
allMedia_library(typeItem);
END;
ISBN ID STATE AVAILABILITY DEBY_COST LOST_COST LOCATION
--------------------------------------------------------------------------------
A123 B1A123 GOOD O 5 20 AI_LT
A123 B2A123 NEW O 6 30 AI_LT
R234 B1R234 NEW A 2 15 ROBO_LT
C321 B1C321 BAD A 1 10 CS_LT
R123 B1R123 GOOD A 3 15 ROBO_LT
F123 B1F123 GOOD O 4 20 CF_LT
F321 B1F321 NEW O 4 20 CF_LT
R321 B1R321 USED A 2 12 ROBO_LT


PL/SQL procedure successfully completed.
```

13 . Procedure for returning an item.

```
SQL> CREATE OR REPLACE PROCEDURE handleReturns_library(auxItemID IN VARCHAR2)
  2  IS
  3  auxRented NUMBER;
  4  auxBook NUMBER;
  5  auxVideo NUMBER;
  6  BEGIN
  7  SELECT COUNT(*) INTO auxRented
  8  FROM rent
  9  WHERE itemid LIKE auxItemID;
 10  SELECT COUNT(*) INTO auxBook
 11  FROM book
 12  WHERE bookid LIKE auxItemID;
 13  SELECT COUNT(*) INTO auxVideo
 14  FROM video
 15  WHERE videoid LIKE auxItemID;
 16  IF auxRented > 0 THEN
 17  DELETE FROM rent
 18  WHERE itemid = auxItemID;
 19  IF auxBook > 0 THEN
 20  UPDATE book
 21  SET availability = 'A'
 22  WHERE bookid LIKE auxItemID;
 23  DBMS_OUTPUT.PUT_LINE('The book ' || auxItemID || ' is now available.');
 24  ELSIF auxVideo > 0 THEN
 25  UPDATE video
 26  SET availability = 'A'
 27  WHERE videoid LIKE auxItemID;
 28  DBMS_OUTPUT.PUT_LINE('The video ' || auxItemID || ' is now available.');
 29  END IF;
 30  ELSE
 31  DBMS_OUTPUT.PUT_LINE('This item is not rented at the moment');
 32  END IF;
 33  EXCEPTION WHEN no_data_found THEN
 34  DBMS_OUTPUT.PUT_LINE('Item ID incorrect');
 35  END;
 36* /

Procedure HANDLERETURNS_LIBRARY compiled
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  auxItemID VARCHAR2(10);
  3  BEGIN
  4  auxItemID := '&ItemID_to_return';
  5  handleReturns_library(auxItemID);
  6  END;
  7* /
Enter value for ItemID_to_return: B1A123
old:DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := '&ItemID_to_return';
handleReturns_library(auxItemID);
END;

new:DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := 'B1A123';
handleReturns_library(auxItemID);
END;
The book B1A123 is now available.


PL/SQL procedure successfully completed.
```

```
SQL> SELECT * FROM rent;

   CARDID    ITEMID    APPORPRIATIONDATE    RETURNDATE
--------- --------- ------------------- -------------
      101 B2A123    10-06-22            15-06-22
      102 B1R234    14-01-23            18-01-23
      104 B1F123    30-06-20            05-07-20
      105 B1R321    06-04-23            18-04-23
      150 B1C321    04-02-23            20-02-23

SQL> SELECT * FROM book;

   ISBN    BOOKID    STATE    AVAILABILITY    DEBYCOST    LOSTCOST    ADDRESS
------- --------- -------- --------------- ---------- ---------- ----------
A123    B1A123    GOOD     A                        5         20 AI_LT
A123    B2A123    NEW      O                        6         30 AI_LT
R234    B1R234    NEW      A                        2         15 ROBO_LT
C321    B1C321    BAD      A                        1         10 CS_LT
R123    B1R123    GOOD     A                        3         15 ROBO_LT
F123    B1F123    GOOD     O                        4         20 CF_LT
F321    B1F321    NEW      O                        4         20 CF_LT
R321    B1R321    USED     A                        2         12 ROBO_LT

8 rows selected.
```

```
SQL> DECLARE
  2  auxCard NUMBER;
  3  auxItemID VARCHAR2(10);
  4  itemType VARCHAR2(20);
  5  auxDate DATE;
  6  BEGIN
  7  auxCard := &Card_ID;
  8  itemType := '&Enter_book';
  9  auxItemID := '&ID_Item';
 10  auxDate := '&Return_date';
 11  rentItem_library(auxCard,auxItemID,itemType,auxDate);
 12  END;
 13* /
Enter value for Card_ID: 104
Enter value for Enter_book: BOOK
Enter value for ID_Item: B1F123
Enter value for Return_date: 05-07-2020
old:DECLARE
auxCard NUMBER;
auxItemID VARCHAR2(10);
itemType VARCHAR2(20);
auxDate DATE;
BEGIN
auxCard := &Card_ID;
itemType := '&Enter_book';
auxItemID := '&ID_Item';
auxDate := '&Return_date';
rentItem_library(auxCard,auxItemID,itemType,auxDate);
END;

new:DECLARE
auxCard NUMBER;
auxItemID VARCHAR2(10);
itemType VARCHAR2(20);
auxDate DATE;
BEGIN
auxCard := 104;
itemType := 'BOOK';
auxItemID := 'B1F123';
auxDate := '05-07-2020';
rentItem_library(auxCard,auxItemID,itemType,auxDate);
END;
The user is blocked


PL/SQL procedure successfully completed.
```

14. Procedure to add a book to library.

```
SQL> CREATE OR REPLACE PROCEDURE addBook_library(auxISBN IN VARCHAR2, auxBookID IN VARCHAR2, auxState
IN VARCHAR2,
  2  auxDebyCost IN NUMBER,
  3  auxLostCost IN NUMBER, auxAddress IN VARCHAR2)
  4  IS
  5  BEGIN
  6  INSERT INTO book
  7  VALUES(auxISBN,auxBookID,auxState,'A',auxDebyCost,auxLostCost,auxAddress);
  8  DBMS_OUTPUT.PUT_LINE('Book inserted correctly');
  9  END;
 10* /

Procedure ADDBOOK_LIBRARY compiled
```

```
Enter value for ISBN: P12
Enter value for ItemID: B1P12
Enter value for State: New
Enter value for Deby_Cost: 20
Enter value for Lost_Cost: 100
Enter value for Location: ROBO_LT
old:DECLARE
auxISBN VARCHAR2(4);
auxItemID VARCHAR2(6);
auxState VARCHAR2(10);
auxDebyCost NUMBER(10,2);
auxLostCost NUMBER(10,2);
auxAddress VARCHAR2(50);
BEGIN
auxISBN := '&ISBN';
auxItemID := '&ItemID' ;
auxState := '&State';
auxDebyCost := &Deby_Cost;
auxLostCost := &Lost_Cost;
auxAddress := '&Location';
addBook_library(auxISBN, auxItemID, auxState, auxDebyCost, auxLostCost, auxAddress);
END;

new:DECLARE
auxISBN VARCHAR2(4);
auxItemID VARCHAR2(6);
auxState VARCHAR2(10);
auxDebyCost NUMBER(10,2);
auxLostCost NUMBER(10,2);
auxAddress VARCHAR2(50);
BEGIN
auxISBN := 'P12';
auxItemID := 'B1P12' ;
auxState := 'New';
auxDebyCost := 20;
auxLostCost := 100;
auxAddress := 'ROBO_LT';
addBook_library(auxISBN, auxItemID, auxState, auxDebyCost, auxLostCost, auxAddress);
END;
Book inserted correctly


PL/SQL procedure successfully completed.
```

```
SQL> SELECT * FROM Book;

   ISBN    BOOKID    STATE    AVAILABILITY    DEBYCOST    LOSTCOST    ADDRESS
_______ _________ ________ _____________ __________ __________ __________
A123    B1A123    GOOD     A                        5          20 AI_LT
A123    B2A123    NEW      O                        6          30 AI_LT
R234    B1R234    NEW      A                        2          15 ROBO_LT
C321    B1C321    BAD      O                        1          10 CS_LT
R123    B1R123    GOOD     O                        3          15 ROBO_LT
F123    B1F123    GOOD     O                        4          20 CF_LT
F321    B1F321    NEW      O                        4          20 CF_LT
R321    B1R321    USED     A                        2          12 ROBO_LT
P12     B1P12     New      A                       20         100 ROBO_LT

9 rows selected.
```

15. Procedure to remove an item from the library.

```
SQL> CREATE OR REPLACE PROCEDURE removeItem_library(auxItemID IN VARCHAR2)
  2  IS
  3  auxBook NUMBER;
  4  auxVideo NUMBER;
  5  BEGIN
  6  SELECT COUNT(*) INTO auxBook
  7  FROM book
  8  WHERE bookid LIKE auxItemID;
  9  SELECT COUNT(*) INTO auxVideo
 10  FROM video
 11  WHERE videoid LIKE auxItemID;
 12  IF auxBook > 0 THEN
 13  DELETE FROM book
 14  WHERE bookid LIKE auxItemID;
 15  DBMS_OUTPUT.PUT_LINE('Book removed correctly');
 16  ELSIF auxVideo > 0 THEN
 17  DELETE FROM video
 18  WHERE videoid LIKE auxItemID;
 19  DBMS_OUTPUT.PUT_LINE('Video removed correctly');
 20  END IF;
 21  END;
 22* /

Procedure REMOVEITEM_LIBRARY compiled

SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  auxItemID VARCHAR2(10);
  3  BEGIN
  4  auxItemID := '&ItemID_to_remove';
  5  removeItem_library(auxItemID);
  6  END;
  7* /
Enter value for ItemID_to_remove: B1P12
old:DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := '&ItemID_to_remove';
removeItem_library(auxItemID);
END;

new:DECLARE
auxItemID VARCHAR2(10);
BEGIN
auxItemID := 'B1P12';
removeItem_library(auxItemID);
END;
Book removed correctly
```

```
SQL> SELECT * FROM Book;

   ISBN    BOOKID    STATE    AVAILABILITY    DEBYCOST    LOSTCOST    ADDRESS
------- --------- -------- --------------- ---------- ----------- ----------
A123    B1A123    GOOD     A                        5          20 AI_LT
A123    B2A123    NEW      O                        6          30 AI_LT
R234    B1R234    NEW      A                        2          15 ROBO_LT
C321    B1C321    BAD      O                        1          10 CS_LT
R123    B1R123    GOOD     O                        3          15 ROBO_LT
F123    B1F123    GOOD     O                        4          20 CF_LT
F321    B1F321    NEW      O                        4          20 CF_LT
R321    B1R321    USED     A                        2          12 ROBO_LT

8 rows selected.
```

16. Procedure to view customer details

```
SQL> CREATE OR REPLACE PROCEDURE viewCustomer_library(auxCustomerID IN NUMBER)
  2  IS
  3  custoName VARCHAR2(40);
  4  custoAdd VARCHAR2(50);
  5  custoPhone NUMBER(9);
  6  userNaM VARCHAR2(10);
  7  custoDate DATE;
  8  custoCard NUMBER;
  9  BEGIN
 10  SELECT name,customeraddress,phone,username,datesignup,cardnumber
 11  INTO custoName, custoAdd, custoPhone, userNaM, custoDate, custoCard
 12  FROM customer
 13  WHERE customerid LIKE auxCustomerID;
 14  DBMS_OUTPUT.PUT_LINE('CUSTOMER ' || auxCustomerID || ' INFO');
 15  DBMS_OUTPUT.PUT_LINE('----------------------------------------');
 16  DBMS_OUTPUT.PUT_LINE('NAME: ' || custoName);
 17  DBMS_OUTPUT.PUT_LINE('ADDRESS: ' || custoAdd);
 18  DBMS_OUTPUT.PUT_LINE('PHONE: ' || custoPhone);
 19  DBMS_OUTPUT.PUT_LINE('USER NAME: ' || userNaM);
 20  DBMS_OUTPUT.PUT_LINE('DATE OF SIGN UP: ' || custoDate);
 21  DBMS_OUTPUT.PUT_LINE('CARD NUMBER: ' || custoCard);
 22  DBMS_OUTPUT.PUT_LINE('----------------------------------------');
 23  END;
 24* /

Procedure VIEWCUSTOMER_LIBRARY compiled
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  auxCustomerID NUMBER;
  3  BEGIN
  4  auxCustomerID :=&CustomerID;
  5  viewCustomer_library(auxCustomerID);
  6  END;
  7* /
Enter value for CustomerID: 1
old:DECLARE
auxCustomerID NUMBER;
BEGIN
auxCustomerID :=&CustomerID;
viewCustomer_library(auxCustomerID);
END;

new:DECLARE
auxCustomerID NUMBER;
BEGIN
auxCustomerID :=1;
viewCustomer_library(auxCustomerID);
END;
CUSTOMER 1 INFO
----------------------------------------
NAME: Abhi
ADDRESS: bacon street
PHONE: 8526354230
USER NAME: ab1
DATE OF SIGN UP: 12-05-22
CARD NUMBER: 101
----------------------------------------


PL/SQL procedure successfully completed.
```

17. Procedure to add new customer.

```
Enter value for Customer_ID: 20
Enter value for Name: alex
Enter value for Address: srilanka
Enter value for Phone: 9876543456
Enter value for Password: alex123
Enter value for User_Name: al1
Enter value for Card_Numeber: 111
old:DECLARE
auxCustomerId NUMBER;
auxName VARCHAR2(20);
auxCustomerAddress VARCHAR2(20);
auxPhone NUMBER;
auxPass VARCHAR2(20);
auxUserName VARCHAR2(20);
auxCardNumber NUMBER;
BEGIN
auxCustomerId := &Customer_ID;
auxName := '&Name';
auxCustomerAddress := '&Address';
auxPhone := &Phone;
auxPass := '&Password';
auxUserName := '&User_Name';
auxCardNumber := &Card_Numeber;
addCustomer_library(auxCustomerId,auxName,auxCustomerAddress,auxPhone,auxPass,auxUserName,auxCardNumbe
r);
END;

new:DECLARE
auxCustomerId NUMBER;
auxName VARCHAR2(20);
auxCustomerAddress VARCHAR2(20);
auxPhone NUMBER;
auxPass VARCHAR2(20);
auxUserName VARCHAR2(20);
auxCardNumber NUMBER;
BEGIN
auxCustomerId := 20;
auxName := 'alex';
auxCustomerAddress := 'srilanka';
auxPhone := 9876543456;
auxPass := 'alex123';
auxUserName := 'al1';
auxCardNumber := 111;
addCustomer_library(auxCustomerId,auxName,auxCustomerAddress,auxPhone,auxPass,auxUserName,auxCardNumbe
r);
END;

PL/SQL procedure successfully completed.
```

```
SQL> select* from customer;

   CUSTOMERID      NAME    CUSTOMERADDRESS         PHONE     PASSWORD    USERNAME    DATESIGNUP    CARDNUMBER
------------ --------- ------------------ ------------ ------------ ----------- ------------ ------------
           1 Abhi      bacon street         8526354230 abhi123      ab1         12-05-22                101
           2 Priya     Adarsh Nagar         6596596509 priya123     pr1         10-01-23                102
           3 Garima    South City           6546546054 garima123    ga1         21-06-19                103
           4 Tarika    Sarojini Nagar.      6586586058 tarika123    ta1         05-12-22                104
           5 Piyush    Sarabha Nagar        6526052652 piyush123    pi1         09-08-16                105
           6 Janvi     Kitchlu Nagar        6501651651 janvi123     ja1         30-04-21                106
           7 Riya      Dugri Estate I       6576576507 riya123      ri1         28-02-20                107
           8 Monica    Ram Vihar            6390639639 monica123    mo1         15-01-21                108
           9 Pihu      Pitampura            6786786708 pihu123      pi1         25-03-23                109
          10 Avnee     Dholakpur            6087687687 avnee123     av1         01-09-19                110

10 rows selected.
```

CONCLUSION

In conclusion, the Library Management System provides a comprehensive solution for managing the renting of books in a library. With its ability to handle multiple branches and their inventory of books, the system offers a scalable solution for libraries of varying sizes. The system also accommodates both employees and customers with unique privileges and access levels. The inclusion of card system and rent tracking provides a streamlined and efficient experience for all involved parties. Overall, this system offers a valuable tool for libraries to better manage their operations and provide an enhanced experience for their customers as well as employees.