TABLE OF CONTENT

INTRODUCTION

FEATURES/REQUIREMENTS OF THE PROJECT

Following are the requirements of this project:

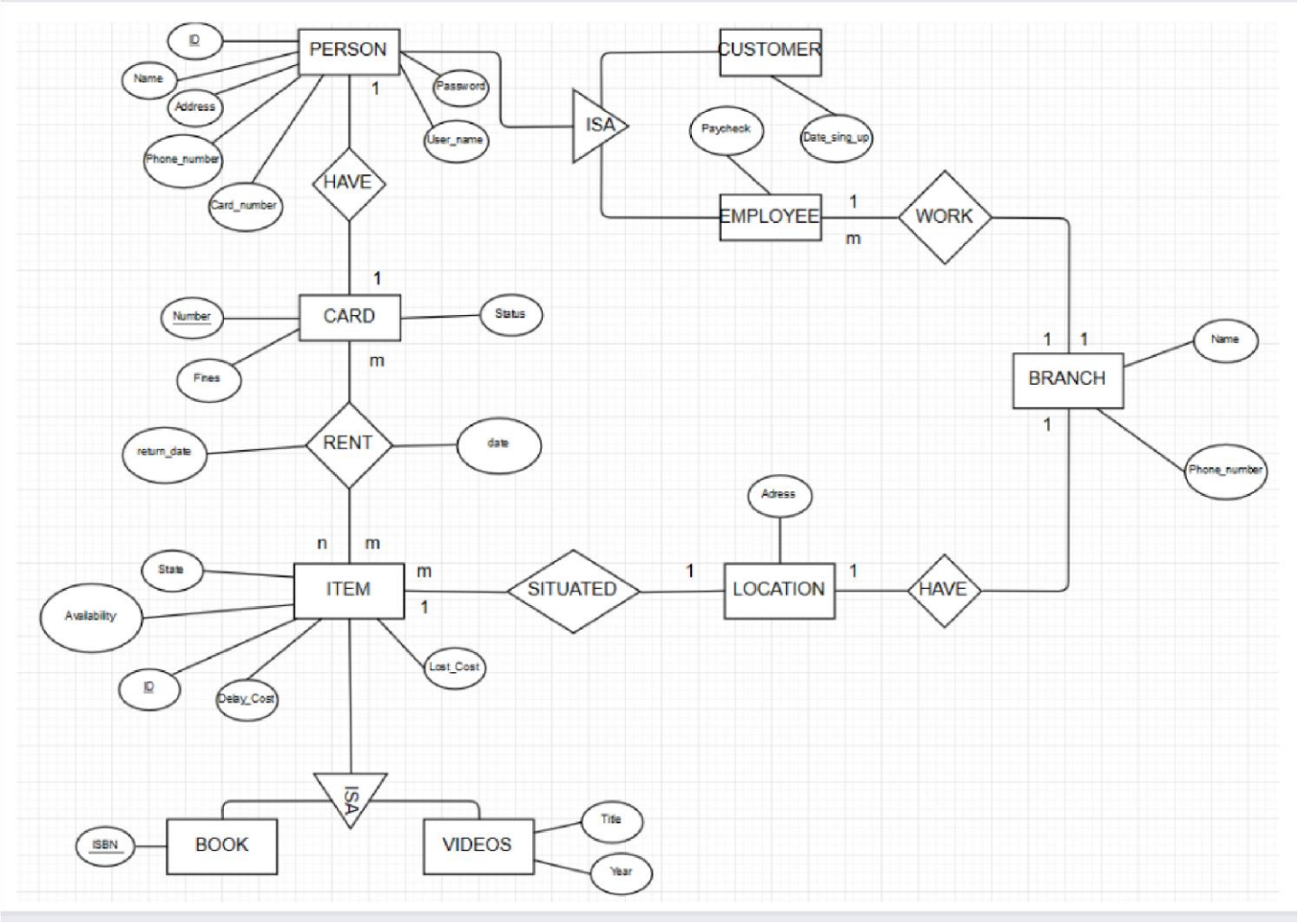
- A database which maintains employee information.
- A separate database for maintaining information related to the book issued and customer.
- Information regarding all the books and videos available in the library.
- Login credentials for customers as well as the employee.

Based on the above requirements, we have:

- Formed the most relevant tables and functions that mimic the library management system and are sufficient to provide the core and fundamental functionalities and facilities of the same.
- Added the proper integrity constraints to the tables, also used several checks in the functions, procedure which ensures the proper functioning of our database.
- Introduced different roles which have different privileges over our database.

ER DIAGRAM

Using the requirements of the library management system, the following are entity sets and relational sets:



RELATIONAL SCHEMA

- CARD (*Number*, Fines, Status)
- CUSTOMER (*ID*, Name, Address, Phone_number, Card_number [References CARD(Number)], Password, User_name, Date_sign_up)
- EMPLOYEE (*ID*, Name, Address, Phone_number, Card_number [References CARD(Number)], Password, User_name, Paycheck, Branch_name [References BRANCH(Name)])
- BRANCH (*Name*, Address [References LOCATION(Address)], Phone_number)
- LOCATION (Address)
- RENT (*Card_ID [References CARD(Number)], Item_ID [References BOOK or VIDEO(ID)]*, Date, Return_date)
- BOOK (*ISBN, ID*, State, Availability, Deby_cost, Lost_cost, Address [References LOCATION(Address)])
- VIDEO (*Title, Year, ID*, State, Availability, Deby_cost, Lost_cost, Address [References LOCATION(Address)])

STRUCTURE,INTEGRITY AND GENERAL CONSTRAINTS OF THE DATABASE

Here we are displaying the tables and their constraints using PL SQL description of tables to display the results better.

- CARD:

  This table contains all the information related to the library card including card-ID, Fine on the book issued and status of the book issued(returned or not). In this table, **Card-ID is the primary key**.

```
SQL> desc card;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CARDID                                    NOT NULL NUMBER
 STATUS                                             VARCHAR2(1)
 FINES                                              NUMBER
```

- CUSTOMER:

  This table contains all the information related to the customers including CUSTOMERID, name, customer address, phone, password, username, date of sign up and card number. Here, **CUSTOMERID is the primary key and CARDNUMBER is the foreign key (from table card(CARDID))**.

```
SQL> desc customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMERID                                NOT NULL NUMBER
 NAME                                               VARCHAR2(40)
 CUSTOMERADDRESS                                    VARCHAR2(50)
 PHONE                                              NUMBER(9)
 PASSWORD                                           VARCHAR2(20)
 USERNAME                                           VARCHAR2(10)
 DATESIGNUP                                         DATE
 CARDNUMBER                                         NUMBER
```

- EMPLOYEE:

This table contains all the information related to the library employee including name, employee address, employeeID, phone, password, username, paycheck, branchname, card number. Here, **employeeID is the primary key branchname is foreign key (from branch(name)) and cardnumber is foreign key(from card(cardID)).**

```
SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEEID                                NOT NULL NUMBER
 NAME                                               VARCHAR2(40)
 EMPLOYEEADDRESS                                    VARCHAR2(50)
 PHONE                                              NUMBER(9)
 PASSWORD                                           VARCHAR2(20)
 USERNAME                                           VARCHAR2(10)
 PAYCHECK                                           NUMBER(8,2)
 BRANCHNAME                                         VARCHAR2(40)
 CARDNUMBER                                         NUMBER
```

- LOCATION:

  This table contains addresses of the different branches of the institute. Here, **address is the primary key.**

```
SQL> desc location;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ADDRESS                                   NOT NULL VARCHAR2(50)
```

- BOOK:

  This table contains information regarding all the books available in the library including ISBN, BookID, state, availability, address etc. Here, **ISBN and BookID are primary keys while address is the foreign key.**

```
SQL> desc book;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ISBN                                      NOT NULL VARCHAR2(4)
 BOOKID                                    NOT NULL VARCHAR2(6)
 STATE                                              VARCHAR2(10)
 AVALABILITY                                        VARCHAR2(1)
 DEBYCOST                                           NUMBER(10,2)
 LOSTCOST                                           NUMBER(10,2)
 ADDRESS                                            VARCHAR2(50)
```

- BRANCH:

  This table contains all the information which are branch specific including-name of the branch, address and phone number. Here, **name is the primary key and address is the foreign key(from the table location(address))**.

```
SQL> desc branch;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                      NOT NULL VARCHAR2(40)
 ADDRESS                                            VARCHAR2(50)
 PHONE                                              NUMBER(9)
```

- VIDEO:

  This table contains information regarding different video products available in the library including-Title of the video, VideoID, year made, availability etc. Here, **<u>title, year and videoID are primary key while address is the foreign key(from table location(address)).</u>**

```
SQL> desc video;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TITLE                                     NOT NULL VARCHAR2(50)
 YEAR                                      NOT NULL NUMBER(38)
 VIDEOID                                   NOT NULL VARCHAR2(6)
 STATE                                              VARCHAR2(10)
 AVALABILITY                                        VARCHAR2(1)
 DEBYCOST                                           NUMBER(10,2)
 LOSTCOST                                           NUMBER(10,2)
 ADDRESS                                            VARCHAR2(50)
```

- RENT:

  Contains columns such as CardID, ItemID, return date etc. Here, **<u>cardID and itemID are primary keys while CardID(from table card(cardID)),itemID (from table book(bookID)) and itemID(from table video(itemID)) are foreign keys.</u>**

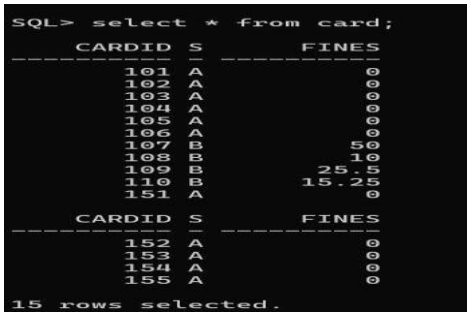
```
SQL> desc rent;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CARDID                                    NOT NULL NUMBER
 VIDEOID                                            VARCHAR2(6)
 APPORPRIATIONDATE                         NOT NULL DATE
 RETURNDATE                                NOT NULL DATE
 BOOKID                                             VARCHAR2(6)
```

<u>STORED VALUES IN DATABASE</u>

- BOOK:

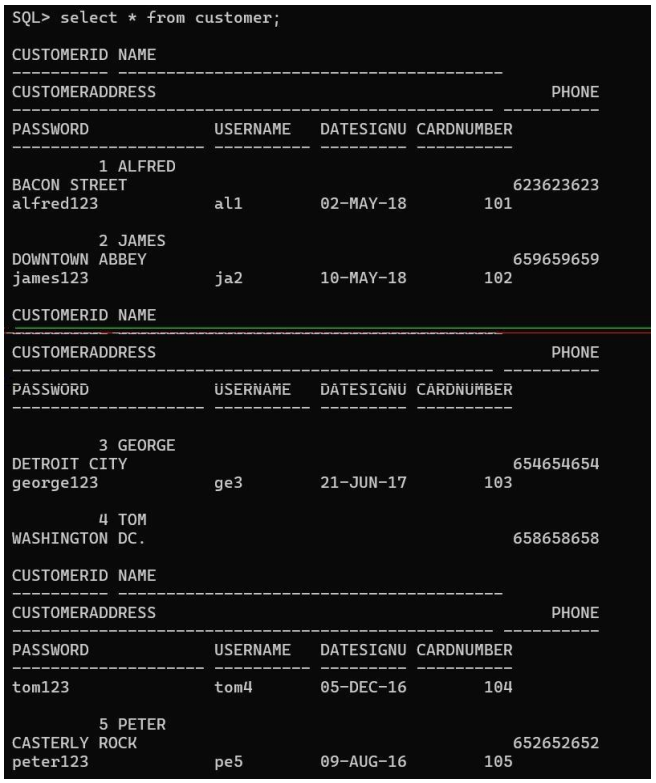
```
SQL> select * from book;

ISBN BOOKID STATE      A   DEBYCOST   LOSTCOST
---- ------ ---------- - ---------- ----------
ADDRESS
--------------------------------------------------
A123 B1A123 GOOD       A          5         20
ARCHEOLOGY ROAD

A123 B2A123 NEW        O          6         30
ARCHEOLOGY ROAD

B234 B1B234 NEW        A          2         15
CHEMISTRY ROAD


ISBN BOOKID STATE      A   DEBYCOST   LOSTCOST
---- ------ ---------- - ---------- ----------
ADDRESS
--------------------------------------------------
C321 B1C321 BAD        A          1         10
PHYSICS ROAD

H123 B1H123 GOOD       A          3         15
CHEMISTRY ROAD

Z123 B1Z123 GOOD       O          4         20
COMPUTING ROAD


ISBN BOOKID STATE      A   DEBYCOST   LOSTCOST
---- ------ ---------- - ---------- ----------
ADDRESS
--------------------------------------------------
L321 B1L321 NEW        O          4         20
COMPUTING ROAD

P321 B1P321 USED       A          2         12
CHEMISTRY ROAD


8 rows selected.
```

- CARD:

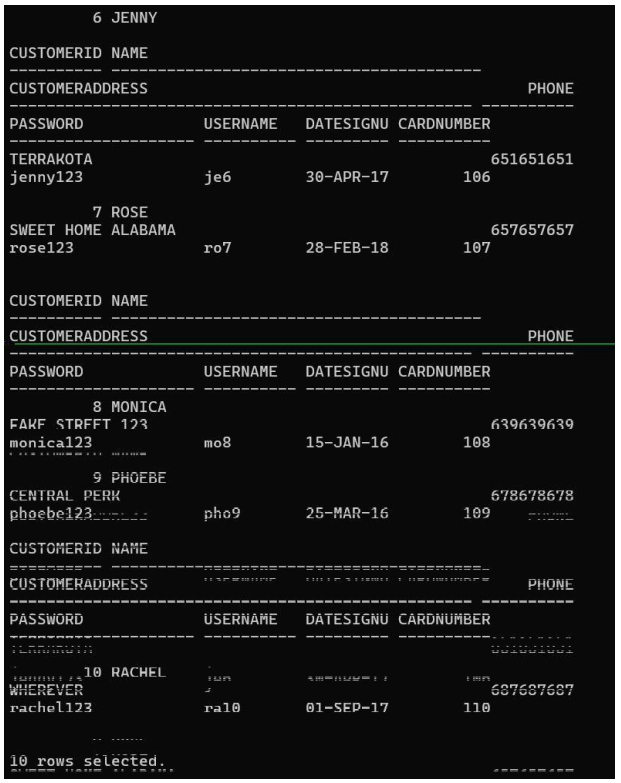


```
SQL> select * from card;

    CARDID S        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
       151 A            0

    CARDID S        FINES
---------- - ----------
       152 A            0
       153 A            0
       154 A            0
       155 A            0

15 rows selected.
```

- CUSTOMER:



```
SQL> select * from customer;

CUSTOMERID NAME
---------- ----------------------------------------
CUSTOMERADDRESS                                           PHONE
--------------------------------------------------- ----------
PASSWORD             USERNAME   DATESIGNU CARDNUMBER
-------------------- ---------- --------- ----------
         1 ALFRED
BACON STREET                                         623623623
alfred123            al1        02-MAY-18        101

         2 JAMES
DOWNTOWN ABBEY                                       659659659
james123             ja2        10-MAY-18        102

CUSTOMERID NAME
---------- ----------------------------------------
CUSTOMERADDRESS                                           PHONE
--------------------------------------------------- ----------
PASSWORD             USERNAME   DATESIGNU CARDNUMBER
-------------------- ---------- --------- ----------

         3 GEORGE
DETROIT CITY                                         654654654
george123            ge3        21-JUN-17        103

         4 TOM
WASHINGTON DC.                                       658658658

CUSTOMERID NAME
---------- ----------------------------------------
CUSTOMERADDRESS                                           PHONE
--------------------------------------------------- ----------
PASSWORD             USERNAME   DATESIGNU CARDNUMBER
-------------------- ---------- --------- ----------
tom123               tom4       05-DEC-16        104

         5 PETER
CASTERLY ROCK                                        652652652
peter123             pe5        09-AUG-16        105
```



```
         6 JENNY

CUSTOMERID NAME
---------- ----------------------------------------
CUSTOMERADDRESS                                           PHONE
--------------------------------------------------- ----------
PASSWORD             USERNAME   DATESIGNU CARDNUMBER
-------------------- ---------- --------- ----------
TERRAKOTA                                            651651651
jenny123             je6        30-APR-17        106

         7 ROSE
SWEET HOME ALABAMA                                   657657657
rose123              ro7        28-FEB-18        107


CUSTOMERID NAME
---------- ----------------------------------------
CUSTOMERADDRESS                                           PHONE
--------------------------------------------------- ----------
PASSWORD             USERNAME   DATESIGNU CARDNUMBER
-------------------- ---------- --------- ----------
         8 MONICA
FAKE STREET 123                                      639639639
monica123            mo8        15-JAN-16        108

         9 PHOEBE
CENTRAL PERK                                         678678678
phoebe123            pho9       25-MAR-16        109

CUSTOMERID NAME
---------- ----------------------------------------
CUSTOMERADDRESS                                           PHONE
--------------------------------------------------- ----------
PASSWORD             USERNAME   DATESIGNU CARDNUMBER
-------------------- ---------- --------- ----------
        10 RACHEL
WHEREVER                                             687687687
rachel123            ra10       01-SEP-17        110


10 rows selected.
```

- RENT:

```
SQL> select * from rent;

    CARDID VIDEOI APPORPRIA RETURNDAT BOOKID
---------- ------ --------- --------- ------
       101        10-MAY-18 20-MAY-18 B2A123
       102        10-MAY-18 25-MAY-18 B1Z123
       104 V1JA15 01-MAY-18 21-MAY-18
       105 V1DI00 02-MAY-18 25-MAY-18
       154        04-MAY-18 26-MAY-18 B1L321
       155 V1CH16 29-APR-18 29-MAY-18

6 rows selected.
```

- VIDEO:

```
SQL> select * from video;

TITLE                                                YEAR VIDEOI STATE
-------------------------------------------------- ---------- ------ ----------
A   DEBYCOST   LOSTCOST ADDRESS
- ---------- ---------- --------------------------------------------------------
CHEMISTRY FOR DUMMIES                                2016 V1CH16 NEW
O         10         50 CHEMISTRY ROAD

CHEMISTRY FOR DUMMIES                                2016 V2CH16 BAD
A          5         20 CHEMISTRY ROAD

COMPUTING MANAGER                                    2014 V1CO14 GOOD
A          4         20 COMPUTING ROAD


TITLE                                                YEAR VIDEOI STATE
-------------------------------------------------- ---------- ------ ----------
A   DEBYCOST   LOSTCOST ADDRESS
- ---------- ---------- --------------------------------------------------------
JAVA LANGUAGE                                        2015 V1JA15 USED
O          4         20 COMPUTING ROAD

DINOSAURS                                            2000 V1DI00 GOOD
O          5         25 ARCHEOLOGY ROAD

T-REX, DEADLY KING                                   1992 V1TR92 USED
A         10         50 ARCHEOLOGY ROAD


TITLE                                                YEAR VIDEOI STATE
-------------------------------------------------- ---------- ------ ----------
A   DEBYCOST   LOSTCOST ADDRESS
- ---------- ---------- --------------------------------------------------------
ANCESTORS OF THE HUMANITY                            1998 V1AN98 BAD
A          3         15 ARCHEOLOGY ROAD

PHYSICS, MOST BORING SH*T                            2018 V1PH18 NEW
A          1          5 PHYSICS ROAD


8 rows selected.
```

- LOCATION:

```
SQL> select * from location;

ADDRESS
--------------------------------
ARCHEOLOGY ROAD
CHEMISTRY ROAD
COMPUTING ROAD
PHYSICS ROAD
```

- EMPLOYEE:

```
SQL> select * from employee;

EMPLOYEEID NAME
---------- ----------------------------------------
EMPLOYEEADDRESS                                          PHONE
-------------------------------------------------- ----------
PASSWORD             USERNAME     PAYCHECK
-------------------- ---------- ----------
BRANCHNAME                            CARDNUMBER
------------------------------------- ----------
       211 ROSS
HIS HOUSE                                           671671671
ross123              ro11             1200
ARCHEOLOGY                                   155


EMPLOYEEID NAME
---------- ----------------------------------------
EMPLOYEEADDRESS                                          PHONE
-------------------------------------------------- ----------
PASSWORD             USERNAME     PAYCHECK
-------------------- ---------- ----------
BRANCHNAME                            CARDNUMBER
------------------------------------- ----------
       212 CHANDLER
OUR HEARTHS                                         688688688
chandler123          chand12        1150.5
ARCHEOLOGY                                   110


EMPLOYEEID NAME
---------- ----------------------------------------
EMPLOYEEADDRESS                                          PHONE
-------------------------------------------------- ----------
PASSWORD             USERNAME     PAYCHECK
-------------------- ---------- ----------
BRANCHNAME                            CARDNUMBER
------------------------------------- ----------
       213 JOEY
LITTLE ITAYLY                                       628628628
joey123              jo13           975.75
ARCHEOLOGY                                   151
```

- BRANCH:

```
SQL> select * from branch;

NAME
------------------------------------
ADDRESS                                          PHONE
------------------------------------------------ ----------
ARCHEOLOGY
ARCHEOLOGY ROAD                                  645645645

CHEMISTRY
CHEMISTRY ROAD                                   622622622

COMPUTING
COMPUTING ROAD                                   644644644


NAME
------------------------------------
ADDRESS                                          PHONE
------------------------------------------------ ----------
PHYSICS
PHYSICS ROAD                                     666666666
```

FUNCTIONALITIES OF THE DATABASE

List of all functions, procedures and cursors and how they help in preserving the consistency of the database.

- LOGIN CREDENTIALS:
  The login credential function in the library management system ensures secure access to the system by requiring users to authenticate with unique usernames and passwords. It provides granular access control, allowing administrators to assign specific privileges based on user roles, thus safeguarding sensitive library data from unauthorized access. Both the employee and customers have their own login credentials for accessing the said database.
  example: CORRECT CUSTOMER USERNAME: all
  CORRECT CUSTOMER PASSWORD: alfred123

```
  1  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;
SQL> /
Enter value for username: al1
old   5:  user := '&Username';
new   5:  user := 'al1';
Enter value for password: alfred123
old   6:  pass := '&Password';
new   6:  pass := 'alfred123';
User al1 loging succesfull

PL/SQL procedure successfully completed.

SQL> edit
Wrote file afiedt.buf

  1  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;
SQL> /
Enter value for username: gvf
old   5:  user := '&Username';
new   5:  user := 'gvf';
Enter value for password: bhj
old   6:  pass := '&Password';
new   6:  pass := 'bhj';
Incorrect username or password

PL/SQL procedure successfully completed.
```

- CHECK ITEM STATUS:

This function allows the customer to check availability of different books and video resources present in the library.

```
  1  DECLARE
  2    auxItemID VARCHAR2(10);
  3  BEGIN
  4    auxItemID := '&Item_ID';
  5    viewItem_library(auxItemID);
  6* END;
SQL> /
Enter value for item_id: B1A123
old   4:   auxItemID := '&Item_ID';
new   4:   auxItemID := 'B1A123';
BOOK B1A123 INFO
----------------------------------------
ISBN: A123
STATE: GOOD
AVALABILITY: A
DEBY COST: 5
LOST COST: 20
ADDRESS: ARCHEOLOGY ROAD
----------------------------------------

PL/SQL procedure successfully completed.
```

- PAY FINES:
  This function allows the library to maintain a record of the pending fines and alerts the customers of the pending payments.

```
SQL> edit
Wrote file afiedt.buf

  1  DECLARE
  2    auxCard card.cardid%TYPE;
  3    money NUMBER;
  4  BEGIN
  5    auxCard := '&Card_ID';
  6    money := '&Money_To_Pay';
  7    payFines_library(auxCard,money);
  8* END;
SQL> /
Enter value for card_id: 105
old   5:   auxCard := '&Card_ID';
new   5:   auxCard := '105';
Enter value for money_to_pay: 100
old   6:   money := '&Money_To_Pay';
new   6:   money := '100';
YOU PAY ALL YOUR FINES AND YOU HAVE 100 MONEY BACK

PL/SQL procedure successfully completed.
```

- RENT ITEM:
  This function allows the user to borrow books or video resources using the database. It reduces the manual interference in the library management system and makes the renting process much more efficient and seamless.

```
SQL> DECLARE
  2    auxCard NUMBER;
  3    auxItemID VARCHAR2(10);
  4    itemType VARCHAR2(20);
  5    auxDate DATE;
  6  BEGIN
  7    auxCard := &Card_ID;
  8    itemType := '&Item_Type_book_or_video';
  9    auxItemID := '&ID_Item';
  10    auxDate := '&Return_date';
  11    rentItem_library(auxCard,auxItemID,itemType,auxDate);
  12  END;

Enter value for Card_ID: 110
old   7:   auxCard := &Card_ID;
new   7:   auxCard := 110;
Enter value for Item_Type_book_or_video: B
old   8:   itemType := '&Item_Type_book_or_video';
new   8:   itemType := 'B';
Enter value for ID_Item: B2C321
old   9:   auxItemID := '&ID_Item';
new   9:   auxItemID := 'B2C321';
Enter value for Return_date: 10-MAY-2024
old   10:   auxDate := '&Return_date';
new   10:   auxDate := '10-MAY-2024';

PL/SQL procedure successfully completed.
Commit complete.
```

CURSORS USED IN DATABASE

Cursors in databases are virtual pointers used to traverse through the result set of a query. They enable sequential access to query results, allowing retrieval of one row at a time, which is particularly useful in iterative processing or when dealing with large datasets. Cursors facilitate efficient data manipulation and analysis within database applications, offering flexibility in handling query results programmatically.

Here, cursors are used to fetch data from the 'book' and 'video' tables based on the media type provided as input. Cursors enable the sequential retrieval of rows from the query results, facilitating the processing and output of book and video information separately. Cursors 'cBooks' and 'cVideos' are defined to fetch data from the 'book' and 'video' tables, respectively, based on the input media type ('books' or 'videos'). These cursors are then opened, fetched row by row, and their contents are printed using DBMS_OUTPUT.PUT_LINE. Cursors simplify the retrieval and processing of data from database tables, enabling efficient handling of query results within the PL/SQL procedure.

```
SQL> DECLARE
  2    typeItem VARCHAR2(10);
  3  BEGIN
  4    typeItem := '&Select_between_books_or_videos';
  5    allMedia_library(typeItem);
  6  END;

Enter value for Select_between_books_or_videos: books
old   4:   typeItem := '&Select_between_books_or_videos';
new   4:   typeItem := 'books';

ISBN     ID             STATE   AVALABILITY     DEBY_COST       LOST_COST       LOCATION
-------------------------------------------------------------------------------------------
A123     B1A123         GOOD    A               5               20              ARCHEOLOGY ROAD
A123     B2A123         NEW     O               6               30              ARCHEOLOGY ROAD
B234     B1B234         NEW     A               2               15              CHEMISTRY ROAD
C321     B1C321         BAD     A               1               10              PHYSICS ROAD
H123     B1H123         GOOD    A               3               15              CHEMISTRY ROAD
Z123     B1Z123         GOOD    O               4               20              COMPUTING ROAD
L321     B1L321         NEW     O               4               20              COMPUTING ROAD
P321     B1P321         USED    A               2               12              CHEMISTRY ROAD

PL/SQL procedure successfully completed.
Commit complete.
```

SOURCE CODE

```
--LIBRARY PROJECT--

--CREATE TABLES--
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(9),
  password VARCHAR2(20),
  userName VARCHAR2(10),
  dateSignUp DATE,
  cardNumber NUMBER,
  CONSTRAINT Customer_PK PRIMARY KEY (customerID));

CREATE TABLE Employee(
  employeeID NUMBER,
  name VARCHAR2(40),
  employeeAddress
  VARCHAR2(50), phone
  NUMBER(9), password
  VARCHAR2(20), userName
  VARCHAR2(10), paycheck
  NUMBER (8, 2),
```

```
  branchName
  VARCHAR2(40),
  cardNumber NUMBER,
  CONSTRAINT Employee_PK PRIMARY KEY (employeeID));

CREATE TABLE Branch(
  name VARCHAR2(40),
  address VARCHAR2(50),
  phone NUMBER(9),
  CONSTRAINT Branch_PK PRIMARY KEY (name));

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

CREATE TABLE Rent(
  cardID NUMBER, itemID
  VARCHAR2(6),
  apporpriationDate DATE,
  returnDate DATE,
  CONSTRAINT Rent_PK PRIMARY KEY (cardID,apporpriationDate,returnDate));

CREATE TABLE Book(
  ISBN VARCHAR2(4),
  ItemID VARCHAR2(6),
  state VARCHAR2(10),
  avalability VARCHAR2(1) CHECK ((avalability = 'A') OR (avalability =
  'O')), debyCost NUMBER(10,2), lostCost NUMBER(10,2), address
  VARCHAR2(50),
  CONSTRAINT Book_PK PRIMARY KEY (ISBN,itemID));

CREATE TABLE Video(
  title VARCHAR2(50),
  year INT, itemIID
  VARCHAR2(6), state
  VARCHAR2(10),
  avalability VARCHAR2(1) CHECK ((avalability = 'A') OR (avalability =
  'O')), debyCost NUMBER(10,2), lostCost NUMBER(10,2), address
  VARCHAR(50),
  CONSTRAINT Video_PK PRIMARY KEY (title,year,itemID));


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

```
SELECT * FROM Video;
SELECT * FROM Rent;

--FOREIGN KEYS-ALTER
TABLE Customer
ADD CONSTRAINT Customer_FK
FOREIGN KEY (cardNumber)
REFERENCES Card(cardID);

ALTER TABLE Employee
ADD CONSTRAINT Employee_FK_Card
FOREIGN KEY (cardNumber)
REFERENCES Card(cardID);

ALTER TABLE Employee
ADD CONSTRAINT Employee_FK_Branch
FOREIGN KEY (branchName)
REFERENCES Branch(name);

ALTER TABLE Branch
ADD CONSTRAINT Branch_FK
FOREIGN KEY (address)
REFERENCES Location(address);

ALTER TABLE Book
ADD CONSTRAINT Book_FK
FOREIGN KEY (address)
REFERENCES Location(address);

ALTER TABLE Video
ADD CONSTRAINT Video_FK
FOREIGN KEY (address)
REFERENCES Location(address);

ALTER TABLE Rent
ADD CONSTRAINT Rent_FK_Card
FOREIGN KEY (cardID)
REFERENCES Card(cardID);

ALTER TABLE Rent
ADD CONSTRAINT Rent_FK_Book
FOREIGN KEY (itemID)
REFERENCES Book(itemID);

ALTER TABLE Rent
ADD CONSTRAINT Rent_FK_Video
FOREIGN KEY (itemID)
REFERENCES Video(itemID);
```

--**INSERTS--**

```
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 (151,'A',0);
INSERT INTO Card VALUES (152,'A',0);
INSERT INTO Card VALUES (153,'A',0);
INSERT INTO Card VALUES (154,'A',0);
INSERT INTO Card VALUES (155,'A',0);

INSERT INTO Branch VALUES ('ARCHEOLOGY', 'ARCHEOLOGY ROAD', 645645645);
INSERT INTO Branch VALUES ('CHEMISTRY', 'CHEMISTRY ROAD', 622622622);
INSERT INTO Branch VALUES ('COMPUTING', 'COMPUTING ROAD', 644644644);
INSERT INTO Branch VALUES ('PHYSICS', 'PHYSICS ROAD', 666666666);

INSERT INTO Customer VALUES (1, 'ALFRED', 'BACON STREET', 623623623, 'alfred123', 'al1', '12-05-2018', 101);
INSERT INTO Customer VALUES (2, 'JAMES', 'DOWNTOWN ABBEY', 659659659, 'james123', 'ja2', '10-05-2018',
102);
INSERT INTO Customer VALUES (3, 'GEORGE', 'DETROIT CITY', 654654654, 'george123', 'ge3', '21-06-2017',
103);
INSERT INTO Customer VALUES (4, 'TOM', 'WASHINGTON DC.', 658658658, 'tom123', 'tom4', '05-12-2016',
104);
INSERT INTO Customer VALUES (5, 'PETER', 'CASTERLY ROCK', 652652652, 'peter123', 'pe5', '09-08-2016', 105);
INSERT INTO Customer VALUES (6, 'JENNY', 'TERRAKOTA', 651651651, 'jenny123', 'je6', '30-04-2017', 106);
INSERT INTO Customer VALUES (7, 'ROSE', 'SWEET HOME ALABAMA', 657657657, 'rose123', 'ro7', '28-02-
2018', 107);
INSERT INTO Customer VALUES (8, 'MONICA', 'FAKE STREET 123', 639639639, 'monica123', 'mo8', '15-01-2016',
108);
INSERT INTO Customer VALUES (9, 'PHOEBE', 'CENTRAL PERK', 678678678, 'phoebe123', 'pho9', '25-03-2016',
109);
INSERT INTO Customer VALUES (10, 'RACHEL', 'WHEREVER', 687687687, 'rachel123', 'ra10', '01-09-2017', 110);

INSERT INTO Employee VALUES (211, 'ROSS', 'HIS HOUSE', 671671671, 'ross123', 'ro11', 1200, 'ARCHEOLOGY',
551);
INSERT INTO Employee VALUES (212, 'CHANDLER', 'OUR HEARTHS', 688688688, 'chandler123', 'chand12',
1150.50, 'ARCHEOLOGY', 552);
INSERT INTO Employee VALUES (213, 'JOEY', 'LITTLE ITAYLY', 628628628, 'joey123', 'jo13', 975.75,
'ARCHEOLOGY', 553);
INSERT INTO Employee VALUES (214, 'VICTOR', 'SANTA FE', 654321987, 'victor123', 'vic14', 2200,
'COMPUTING', 554);
INSERT INTO Employee VALUES (215, 'JAIRO', 'ARMILLA', 698754321, 'jairo123', 'ja15', 2200.50, 'CHEMISTRY',
555);
```

```
INSERT INTO Location VALUES ('ARCHEOLOGY ROAD');
INSERT INTO Location VALUES ('CHEMISTRY ROAD');
INSERT INTO Location VALUES ('COMPUTING ROAD');
INSERT INTO Location VALUES ('PHYSICS ROAD');
INSERT INTO Book VALUES ('A123', 'B1A123', 'GOOD', 'A', 5, 20, 'ARCHEOLOGY ROAD');
INSERT INTO Book VALUES ('A123', 'B2A123', 'NEW', 'O', 6, 30, 'ARCHEOLOGY ROAD');
INSERT INTO Book VALUES ('B234', 'B1B234', 'NEW', 'A', 2, 15, 'CHEMISTRY ROAD');
INSERT INTO Book VALUES ('C321', 'B1C321', 'BAD', 'A', 1, 10, 'PHYSICS ROAD');
INSERT INTO Book VALUES ('H123', 'B1H123', 'GOOD', 'A', 3, 15, 'CHEMISTRY ROAD');
INSERT INTO Book VALUES ('Z123', 'B1Z123', 'GOOD', 'O', 4, 20, 'COMPUTING ROAD');
INSERT INTO Book VALUES ('L321', 'B1L321', 'NEW', 'O', 4, 20, 'COMPUTING ROAD');
INSERT INTO Book VALUES ('P321', 'B1P321', 'USED', 'A', 2, 12, 'CHEMISTRY ROAD');

INSERT INTO Video VALUES ('CHEMISTRY FOR DUMMIES', 2016, 'V1CH16', 'NEW', 'O', 10, 50, 'CHEMISTRY
ROAD');
INSERT INTO Video VALUES ('CHEMISTRY FOR DUMMIES', 2016, 'V2CH16', 'BAD', 'A', 5, 20, 'CHEMISTRY
ROAD');
INSERT INTO Video VALUES ('COMPUTING MANAGER', 2014, 'V1CO14', 'GOOD', 'A', 4, 20, 'COMPUTING
ROAD');
INSERT INTO Video VALUES ('JAVA LANGUAGE', 2015, 'V1JA15', 'USED', 'O', 4, 20, 'COMPUTING ROAD');
INSERT INTO Video VALUES ('DINOSAURS', 2000, 'V1DI00', 'GOOD', 'O', 5, 25, 'ARCHEOLOGY ROAD');
INSERT INTO Video VALUES ('T-REX, DEADLY KING', 1992, 'V1TR92', 'USED', 'A', 10, 50, 'ARCHEOLOGY ROAD');
INSERT INTO Video VALUES ('ANCESTORS OF THE HUMANITY', 1998, 'V1AN98', 'BAD', 'A', 3, 15, 'ARCHEOLOGY
ROAD');
INSERT INTO Video VALUES ('PHYSICS, MOST BORING SH*T', 2018, 'V1PH18', 'NEW', 'A', 1, 5, 'PHYSICS ROAD');

INSERT INTO Rent VALUES (101, 'B2A123', '10-05-2018', '20-05-2018');
INSERT INTO Rent VALUES (102, 'B1Z123', '10-05-2018', '25-05-2018');
INSERT INTO Rent VALUES (104, 'V1JA15', '01-05-2018', '21-05-2018');
INSERT INTO Rent VALUES (105, 'V1DI00', '02-05-2018', '25-05-2018');
INSERT INTO Rent VALUES (154, 'B1L321', '04-05-2018', '26-05-2018');
INSERT INTO Rent VALUES (155, 'V1CH16', '29-04-2018', '29-05-2018');
```

--FUNCTIONS---1--

```
--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;

SET SERVEROUTPUT ON;
DECLARE user
  customer.username%TYPE; pass
  customer.password%TYPE;
BEGIN user :=
  &Username; pass :=
  &Password;
  login_library(user,pass);
END;


--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 || ' 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;

SET SERVEROUTPUT ON;
DECLARE user
  employee.username%TYPE; pass
  employee.password%TYPE;
BEGIN user := &Username; pass :=
  &Password;
  login_employee_library(user,pass);
END;
```

--2--

```
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 itemId LIKE auxItemID;

  SELECT COUNT(*) INTO auxVideo
  FROM video
  WHERE itemId LIKE auxItemID;

  IF auxBook > 0 THEN
    SELECT isbn, state, avalability, debycost, lostcost, address
    INTO auxISBN, auxState, auxAbala, auxDebyCost, auxLostCost, auxAddress
    FROM book
    WHERE itemId 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('AVALABILITY: ' || 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, avalability, debycost, lostcost, address
    INTO auxTitle, auxYear, auxState, auxAbala, auxDebyCost, auxLostCost, auxAddress
    FROM video
    WHERE itemId 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('AVALABILITY: ' || 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;
```

--3--

```
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 avalability INTO itemStatus
      FROM book
      WHERE itemId LIKE auxItemID;

      IF itemStatus LIKE 'A' THEN
        UPDATE book
        SET avalability = 'O'
        WHERE itemId 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 avalability INTO itemStatus
      FROM video
      WHERE itemId LIKE auxItemID;
```

```
    IF itemStatus LIKE 'A' THEN
      UPDATE video
      SET avalability = 'O'
      WHERE itemId 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;

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

SELECT * FROM customer;
SELECT * FROM rent;
SELECT * FROM card;
```

--4--

```
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('YOU PAY ALL YOUR FINES AND YOU HAVE ' || total || ' MONEY BACK');
```

```
    UPDATE card
    SET status = 'A', fines = 0
    WHERE cardid = auxCard;

  ELSIF finesAmount = money THEN
    total := money - finesAmount;
    DBMS_OUTPUT.PUT_LINE('YOU PAY ALL YOUR FINES');

    UPDATE card
    SET status = 'A', fines = 0
    WHERE cardid = auxCard;

  ELSE
    total := finesAmount - money;
    DBMS_OUTPUT.PUT_LINE('YOU WILL NEED TO PAY ' || total || ' MORE DOLLARS 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(custoID);
END;


--cursor--
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    AVALABILITY    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.itemId || '' || xBooks.state || '   ' ||
xBooks.avalability || ' ' || xBooks.debycost || '        ' || xBooks.lostcost || '          ' ||
xBooks.address);
   END LOOP;
  ELSIF mediaType LIKE 'videos' THEN
   OPEN cVideos;
DBMS_OUTPUT.PUT_LINE('TITLE        YEAR    ID    STATE    AVALABILITY    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.itemId || '
' || xVideos.state || ' ' || xVideos.avalability || '     ' || xVideos.debycost || '       ' ||
xVideos.lostcost || '    ' || xVideos.address);
   END LOOP;
  ELSE
   DBMS_OUTPUT.PUT_LINE('TYPE INCORRECT, you must choose between books or videos');
  END IF;
END;

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

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 avalability INTO itemStatus
    FROM book
    WHERE itemid LIKE auxItemID;

    IF itemStatus LIKE 'A' THEN
     UPDATE book
     SET avalability = 'O'
     WHERE itemid LIKE auxItemID;

     INSERT INTO rent
     VALUES (auxCard,null,sysdate,auxDate,auxItemID);
     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 avalability INTO itemStatus
FROM video
    WHERE itemid LIKE auxItemID;

    IF itemStatus LIKE 'A' THEN
     UPDATE video
     SET avalability = 'O'
     WHERE itemid LIKE auxItemID;

     INSERT INTO rent
     VALUES (auxCard,auxItemID,sysdate,auxDate,null);
     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;

DECLARE auxCard
  NUMBER; auxItemID
  VARCHAR2(10); itemType
  VARCHAR2(20); auxDate
  DATE;
BEGIN auxCard :=
  &Card_ID;
  itemType := '&Item_Type_book_or_video'; auxItemID :=
'&ID_Item'; auxDate := '&Return_date';
```

```
rentItem_library(auxCard,auxItemID,itemType,auxDate);
END;
```