“Rendevouz: A Library Management System”

UCS301: Database Management Systems

Submitted by

102003370	Mukul Singhal
102003372	Jahnvi Gangwar
102003375	Aryan Baluja
102053038	Arnav Barman

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Jan - June 2022

TABLE OF CONTENTS

“Rendevouz: A Library Management System”

Introduction

A library management system is **software that is designed to manage all the functions of a library**. It helps librarian to maintain the database of new books and the books that are borrowed by members along with their due dates. This system completely automates all your library's activities.

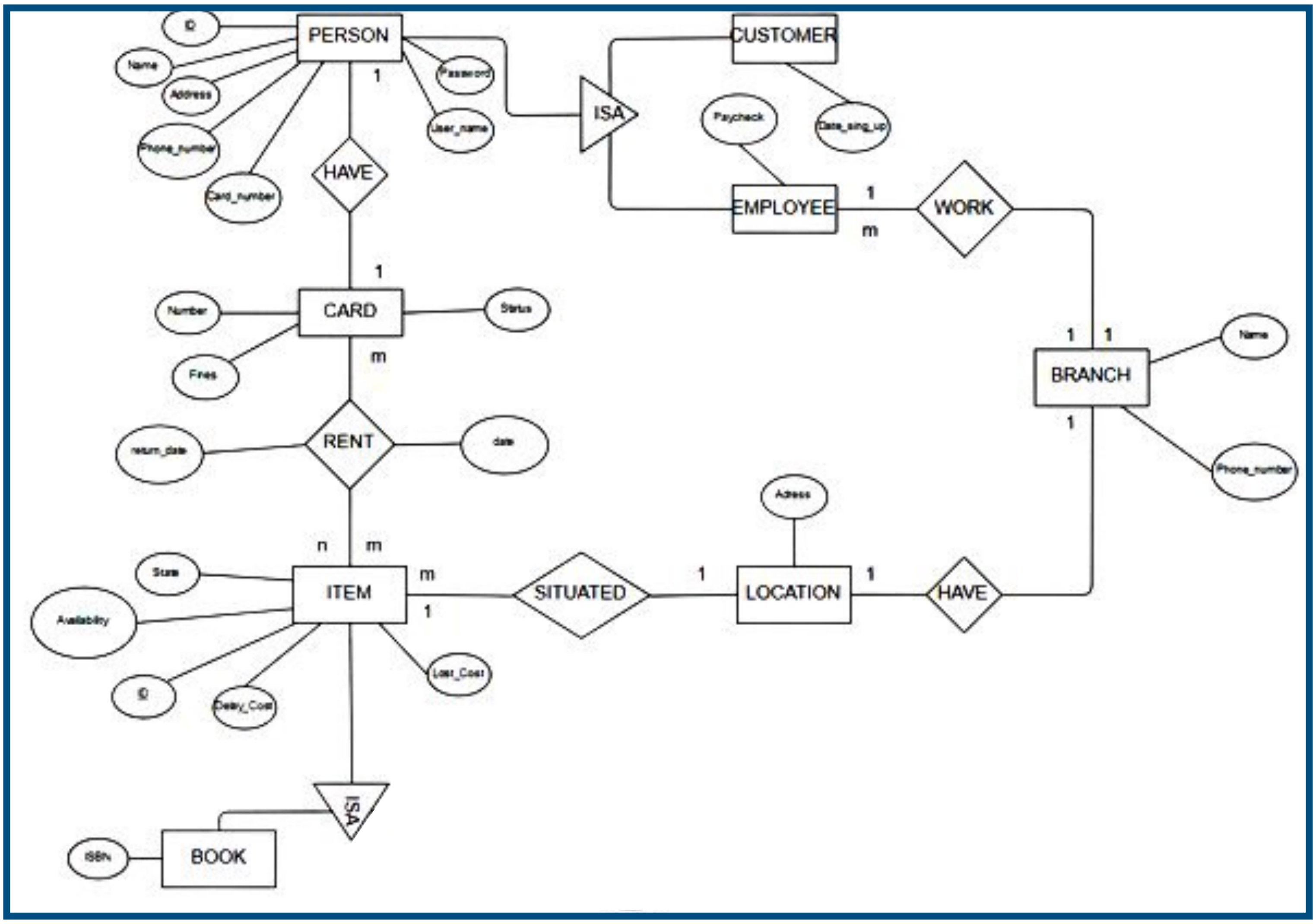
This application uses SQL and PL/SQL for creating the database.

The following pages, first demonstrate the design of the store using a detailed ER Diagram. It clearly shows all the tables that are present along with their inter-relationship through foreign & primary keys.

This is followed by normalisation of the tables until every table reaches the Boyce-Codd Normalised Form(BCNF).

This is followed by the queries that have been shown that demonstrate the working of the database.

E-R Diagram


PERSON
ID
Name
Address
Phone_number
Card_number
Password
User_name
1
HAVE
1
ISA
CUSTOMER
Paycheck
Date_sing_up
EMPLOYEE
1
m
WORK
CARD
Number
Fines
Status
m
RENT
return_date
date
1
1
BRANCH
Name
1
Phone_number
Adress
n
m
ITEM
State
Availability
ID
Delay_Cost
Lost_Cost
m
1
SITUATED
1
LOCATION
1
HAVE
ISA
BOOK
ISBN

Normalisation

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, Date, Return_date)

BOOK (ISBN, ID, State, Avalability, Deby_cost, Lost_cost, Address [References LOCATION(Address)])

SQL Queries

PART 1: Creating

Step 0

Drop tables

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

Step 1

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 (itemID));

CREATE TABLE Book(
ISBN VARCHAR2(4),
bookID 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 (bookID));
```

Step 2

Add 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 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(bookID);
```

Step 3

Insert values

```
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 Location VALUES ('ARCHEOLOGY ROAD');
INSERT INTO Location VALUES ('CHEMISTRY ROAD');
INSERT INTO Location VALUES ('COMPUTING ROAD');
INSERT INTO Location VALUES ('PHYSICS ROAD');

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-May-2018', 101);
INSERT INTO Customer VALUES (2, 'JAMES', 'DOWNTOWN ABBEY', 659659659, 'james123',
'ja2', '10-May-2018', 102);
INSERT INTO Customer VALUES (3, 'GEORGE', 'DETROIT CITY', 654654654, 'george123',
'ge3', '21-June-2017', 103);
INSERT INTO Customer VALUES (4, 'TOM', 'WASHINGTON DC.', 658658658, 'tom123', 'tom4',
'05-Dec-2016', 104);
INSERT INTO Customer VALUES (5, 'PETER', 'CASTERLY ROCK', 652652652, 'peter123', 'pe5',
'09-Aug-2016', 105);
INSERT INTO Customer VALUES (6, 'JENNY', 'TERRAKOTA', 651651651, 'jenny123', 'je6', '30-
April-2017', 106);
```

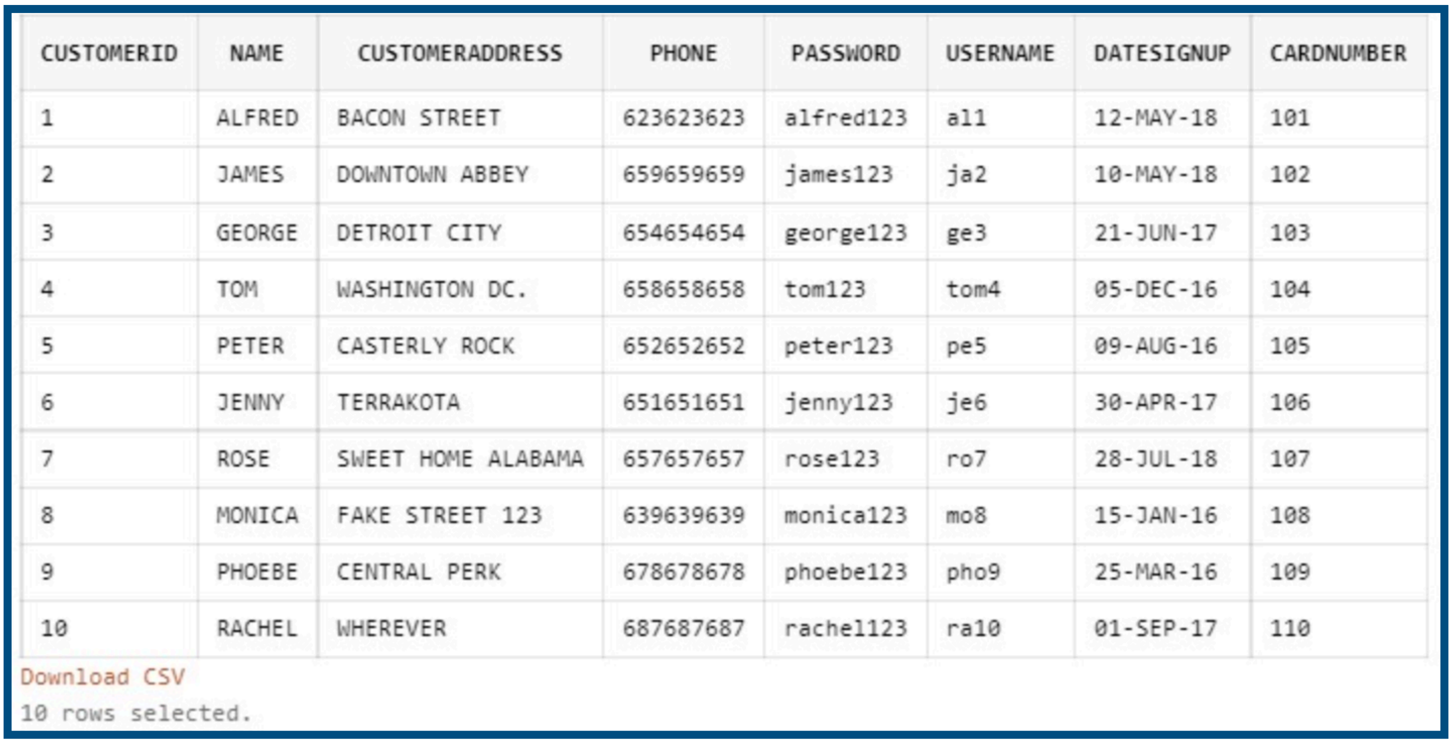
```sql
INSERT INTO Customer VALUES (7, 'ROSE', 'SWEET HOME ALABAMA', 657657657, 'rose123',
'ro7', '28-July-2018', 107);
INSERT INTO Customer VALUES (8, 'MONICA', 'FAKE STREET 123', 639639639, 'monica123',
'mo8', '15-Jan-2016', 108);
INSERT INTO Customer VALUES (9, 'PHOEBE', 'CENTRAL PERK', 678678678, 'phoebe123',
'pho9', '25-MAr-2016', 109);
INSERT INTO Customer VALUES (10, 'RACHEL', 'WHEREVER', 687687687, 'rachel123', 'ra10',
'01-September-2017', 110);

INSERT INTO Employee VALUES (211, 'ROSS', 'HIS HOUSE', 671671671, 'ross123', 'ro11',
1200, 'ARCHEOLOGY', 151);
INSERT INTO Employee VALUES (212, 'CHANDLER', 'OUR HEARTHS', 688688688,
'chandler123', 'chand12', 1150.50, 'ARCHEOLOGY', 152);
INSERT INTO Employee VALUES (213, 'JOEY', 'LITTLE ITAYLY', 628628628, 'joey123', 'jo13',
975.75, 'ARCHEOLOGY', 153);
INSERT INTO Employee VALUES (214, 'VICTOR', 'SANTA FE', 654321987, 'victor123', 'vic14',
2200, 'COMPUTING', 154);
INSERT INTO Employee VALUES (215, 'JAIRO', 'ARMILLA', 698754321, 'jairo123', 'ja15',
2200.50, 'CHEMISTRY', 155);

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 Rent VALUES (101, 'B2A123', '10-May-2018', '20-May-2018');
INSERT INTO Rent VALUES (102, 'B1Z123', '10-May-2018', '25-May-2018');
INSERT INTO Rent VALUES (154, 'B1L321', '04-May-2018', '26-May-2018');
```

Step 4

Display current content in each table

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

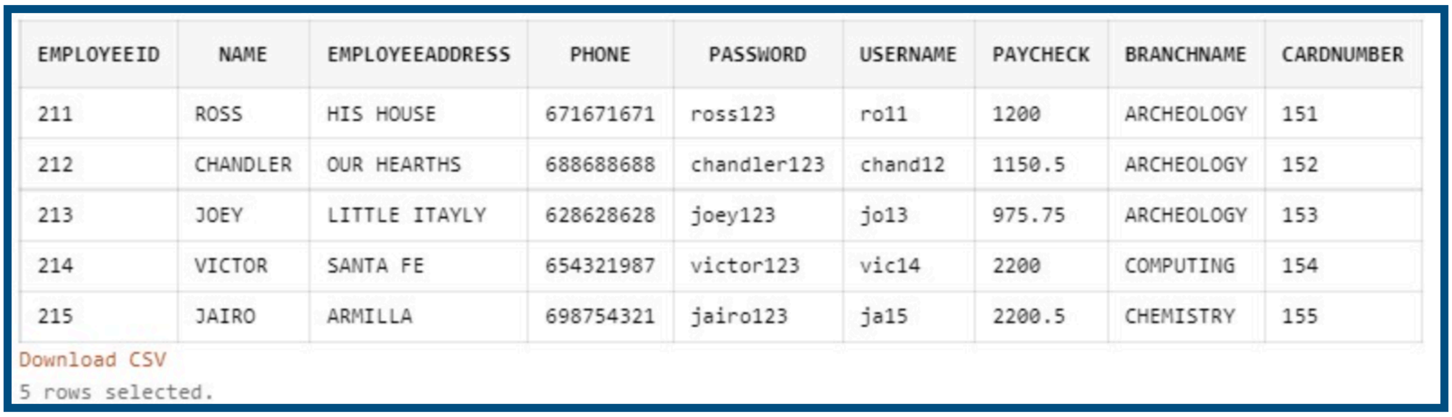
Snapshots:



CUSTOMERID	NAME	CUSTOMERADDRESS	PHONE	PASSWORD	USERNAME	DATESIGNUP	CARDNUMBER
1	ALFRED	BACON STREET	623623623	alfred123	al1	12-MAY-18	101
2	JAMES	DOWNTOWN ABBEY	659659659	james123	ja2	10-MAY-18	102
3	GEORGE	DETROIT CITY	654654654	george123	ge3	21-JUN-17	103
4	TOM	WASHINGTON DC.	658658658	tom123	tom4	05-DEC-16	104
5	PETER	CASTERLY ROCK	652652652	peter123	pe5	09-AUG-16	105
6	JENNY	TERRAKOTA	651651651	jenny123	je6	30-APR-17	106
7	ROSE	SWEET HOME ALABAMA	657657657	rose123	ro7	28-JUL-18	107
8	MONICA	FAKE STREET 123	639639639	monica123	mo8	15-JAN-16	108
9	PHOEBE	CENTRAL PERK	678678678	phoebe123	pho9	25-MAR-16	109
10	RACHEL	WHEREVER	687687687	rachel123	ra10	01-SEP-17	110

Download CSV

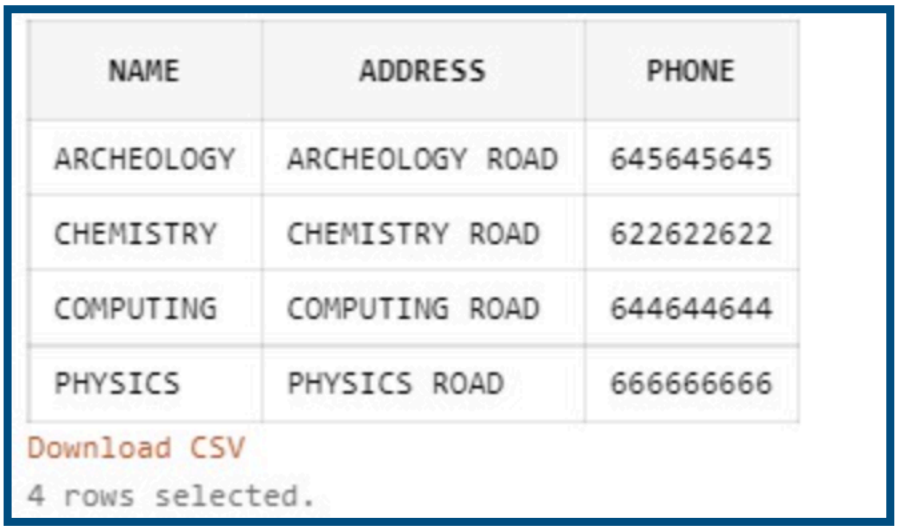
10 rows selected.



EMPLOYEEID	NAME	EMPLOYEEADDRESS	PHONE	PASSWORD	USERNAME	PAYCHECK	BRANCHNAME	CARDNUMBER
211	ROSS	HIS HOUSE	671671671	ross123	ro11	1200	ARCHEOLOGY	151
212	CHANDLER	OUR HEARTHS	688688688	chandler123	chand12	1150.5	ARCHEOLOGY	152
213	JOEY	LITTLE ITAYLY	628628628	joey123	jo13	975.75	ARCHEOLOGY	153
214	VICTOR	SANTA FE	654321987	victor123	vic14	2200	COMPUTING	154
215	JAIRO	ARMILLA	698754321	jairo123	ja15	2200.5	CHEMISTRY	155

Download CSV

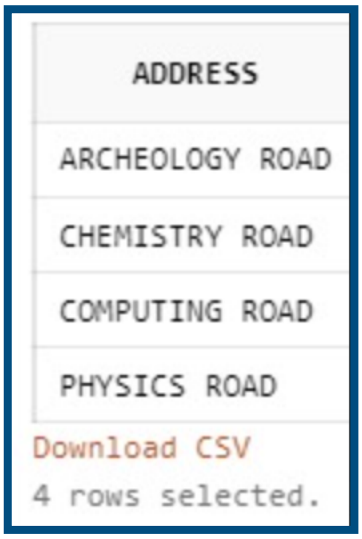
5 rows selected.



NAME	ADDRESS	PHONE
ARCHEOLOGY	ARCHEOLOGY ROAD	645645645
CHEMISTRY	CHEMISTRY ROAD	622622622
COMPUTING	COMPUTING ROAD	644644644
PHYSICS	PHYSICS ROAD	666666666

Download CSV

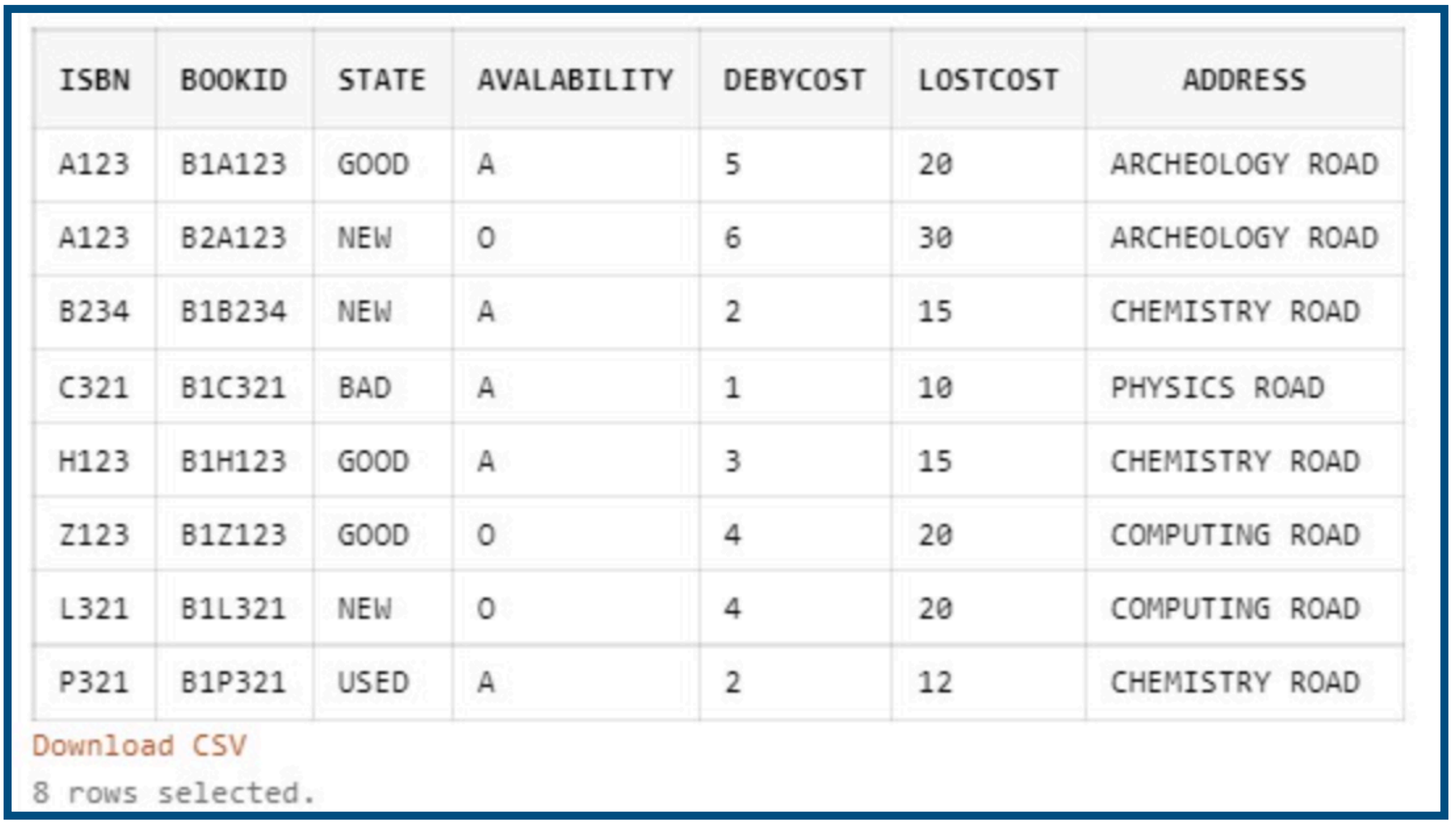
4 rows selected.



ADDRESS
ARCHEOLOGY ROAD
CHEMISTRY ROAD
COMPUTING ROAD
PHYSICS ROAD

Download CSV

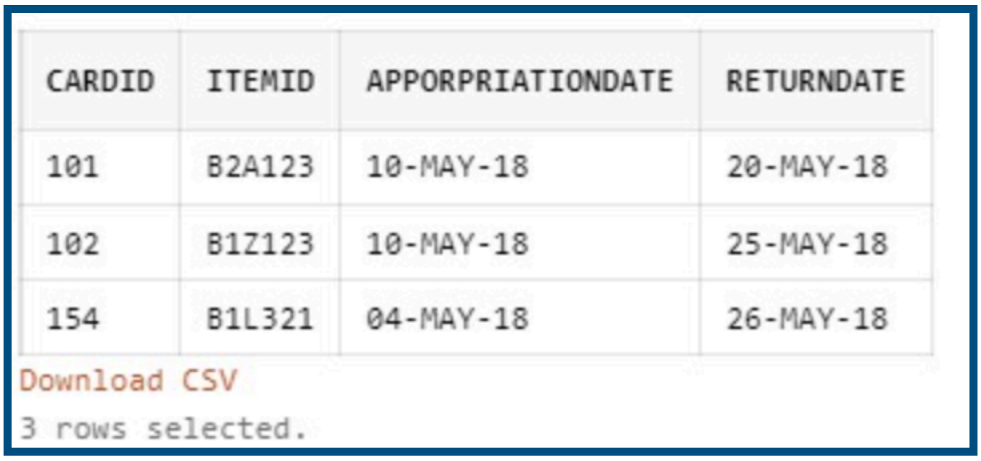
4 rows selected.



ISBN	BOOKID	STATE	AVALABILITY	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
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

Download CSV

8 rows selected.



CARDID	ITEMID	APPORPRIATIONDATE	RETURNDATE
101	B2A123	10-MAY-18	20-MAY-18
102	B1Z123	10-MAY-18	25-MAY-18
154	B1L321	04-MAY-18	26-MAY-18

Download CSV

3 rows selected.

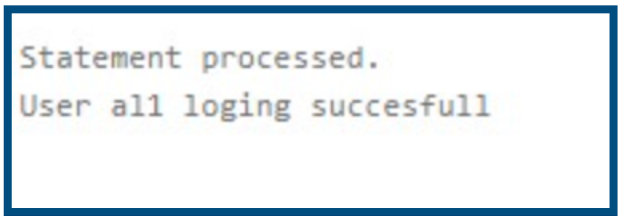
PART 2: Procedures, cursors and triggers

Step 1:

a. Login Employee

```
-- Login for Employee
DECLARE
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 ' II user II ' 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;
BEGIN
loginEmployee_library('ro11','ross123');
END;
```

Snapshots

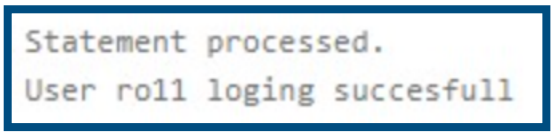


```
Statement processed.
User al1 loging succesfull
```

b. Login Customer

```
-- Login for customer
DECLARE
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 ' II user II ' 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;
BEGIN
loginCustomer_library('al1','alfred123');
END;
```

Snapshots



```
Statement processed.
User ro11 loging succesfull
```

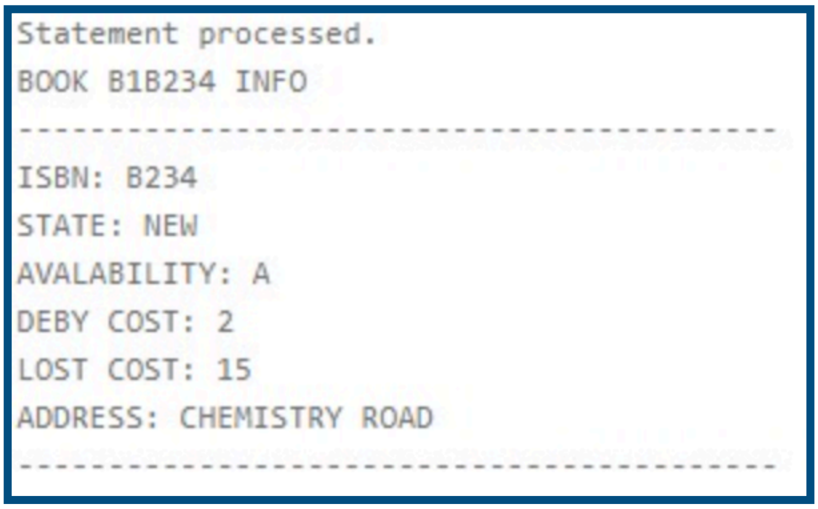
Step 2:
Viewing details

a. book

```
DECLARE
auxItemID VARCHAR2(10);
PROCEDURE viewItem_library(auxItemID IN VARCHAR2)
IS
auxISBN VARCHAR2(4);
auxState VARCHAR2(10);
auxDebyCost NUMBER(10,2);
auxLostCost NUMBER(10,2);
auxAddress VARCHAR2(50);
auxAbala VARCHAR2(1);
auxBook NUMBER;
BEGIN
SELECT COUNT(*) INTO auxBook
FROM book
WHERE bookid LIKE auxItemID;

IF auxBook > 0 THEN
SELECT isbn, state, avalability, 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('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;
BEGIN
auxItemID :='B1B234';
viewItem_library(auxItemID);
END;
```

Snapshots



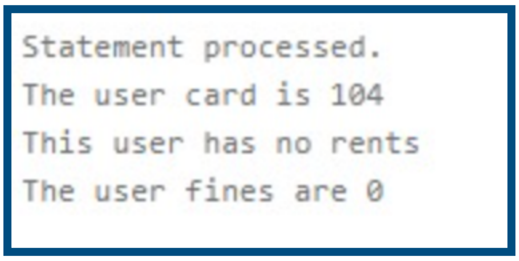
```
Statement processed.
BOOK B1B234 INFO
-------------------------------------------
ISBN: B234
STATE: NEW
AVALABILITY: A
DEBY COST: 2
LOST COST: 15
ADDRESS: CHEMISTRY ROAD
-------------------------------------------
```

b. Customer

```
--CUSTOMER--
DECLARE
custoID customer.customerid%TYPE;
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('NOT DATA FOUND');
END;
BEGIN
custoID := 4;
customerAccount_library(custoID);
END;
```

Snapshots



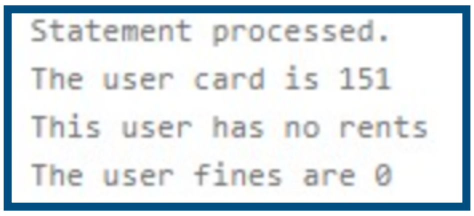
```
Statement processed.
The user card is 104
This user has no rents
The user fines are 0
```

c. Employee

```
--EMPLOYEE--
DECLARE
emploID employee.employeeid%TYPE;
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('NOT DATA FOUND');
END;
BEGIN
emploID := 211;
employeeAccount_library(emploID);
END;
```

Snapshots



```
Statement processed.
The user card is 151
This user has no rents
The user fines are 0
```

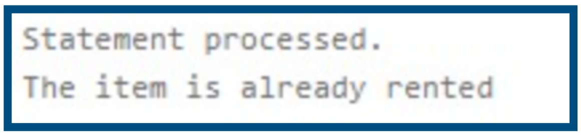
Step 3:

a. Renting books

```
DECLARE
auxCard NUMBER;
auxItemID VARCHAR2(10);
auxDate DATE;
PROCEDURE rentItem_library(auxCard IN NUMBER, auxItemID 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
SELECT avalability INTO itemStatus
FROM book
WHERE bookid LIKE auxItemID;
IF itemStatus LIKE 'A' THEN
UPDATE book
SET avalability = '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;
ELSE
DBMS_OUTPUT.PUT_LINE('The user is blocked');
END IF;
END;
BEGIN
auxCard := 101;
auxItemID := 'B2A123';
auxDate := '20-May-2018';
```

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

Snapshots



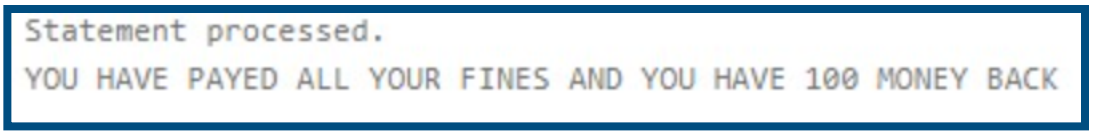
```
Statement processed.
The item is already rented
```

b. Managing customer fines

```
DECLARE
auxCard card.cardid%TYPE;
money NUMBER;
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 HAVE PAYED 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;
BEGIN
auxCard := 101;
money := 100;
payFines_library(auxCard,money);
END;
```

Snapshots



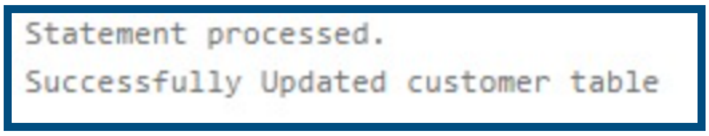
```
Statement processed.
YOU HAVE PAYED ALL YOUR FINES AND YOU HAVE 100 MONEY BACK
```

Step 4: Updating the database

a. Customer

```
-- This is used to update the information of the customers if required
DECLARE
auxCustomer customer.customerid%TYPE;
pNumber NUMBER;
address VARCHAR(20);
newPass VARCHAR(20);
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;
DBMS_OUTPUT.PUT_LINE('Successfully Updated customer table');
END;
BEGIN
auxCustomer := 4;
pNumber := 623623623;
address := 'WASHINGTON DC.';
newPass := 'tom123';
updateInfoCusto_library(auxCustomer,pNumber,address,newPass);
END;
```

Snapshots

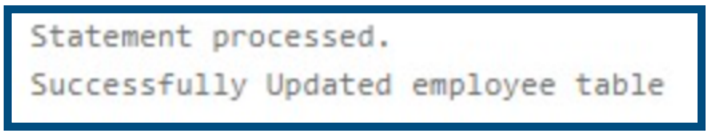


```
Statement processed.
Successfully Updated customer table
```

b. Employee

```
-- This is used to update the information of the employee if required
DECLARE
auxEmployee employee.employeeid%TYPE;
pNumber NUMBER;
address VARCHAR(45);
newPass VARCHAR(45);
newPayCheck NUMBER;
newBranch VARCHAR(45);
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 = auxEmployee, branchname = newBranch
WHERE employeeid = auxEmployee;
DBMS_OUTPUT.PUT_LINE('Successfully Updated employee table');
END;
BEGIN
auxEmployee := 211;
pNumber := 623623623;
address := 'HIS HOUSE';
newPass := 'ross123';
newPayCheck := 1300;
newBranch := 'COMPUTING';
updateInfoEmp_library(auxEmployee,pNumber,address,newPass,newPayCheck,newB
ranch);
END;
```

Snapshots



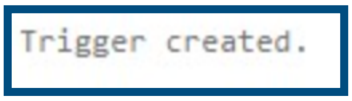
```
Statement processed.
Successfully Updated employee table
```

Step 5: Adding new data

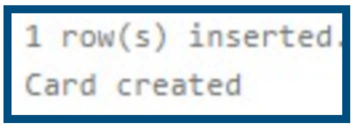
a. <u>Trigger When adding a new employee</u>

```
-- To maintain the referential integrity of the tables we need this trigger to make a card
after there has been insertion
-- in the employee table
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;
-- DML statement to test the trigger
INSERT INTO employee
VALUES (11,'MARI
CARMEN','CORDOBA',645892456,'maricarmen123','ma11',1200,'CHEMISTRY',111);
```

<u>Snapshots</u>



```
Trigger created.
```

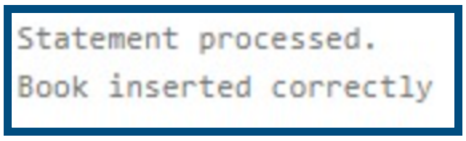


```
1 row(s) inserted.
Card created
```

b. Adding new book

```
--this procedure is used to add new books in the library database--
DECLARE
auxISBN VARCHAR2(4);
auxItemID VARCHAR2(6);
auxState VARCHAR2(10);
auxDebyCost NUMBER(10,2);
auxLostCost NUMBER(10,2);
auxAddress VARCHAR2(50);
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;
BEGIN
auxISBN := 'D123';
auxItemID := 'B2B234';
auxState := 'NEW';
auxDebyCost := 5;
auxLostCost := 15;
auxAddress := 'CHEMISTRY ROAD';
addBook_library(auxISBN, auxItemID, auxState, auxDebyCost, auxLostCost,
auxAddress);
END;
```

Snapshots



```
Statement processed.
Book inserted correctly
```

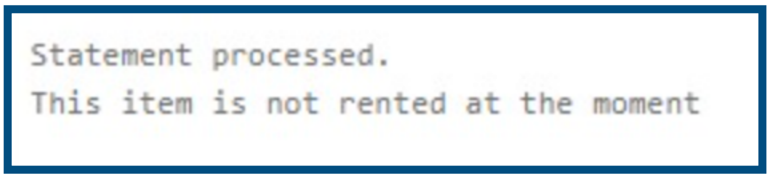
Step 6: Returning a book

a. Handling returns

```
-- This function is used to handle the return of items and modify the status of the given
item in all the tables
--affected by it.
DECLARE
auxItemID VARCHAR(10);
PROCEDURE handleReturns_library(auxItemID IN VARCHAR)
IS
auxRented NUMBER;
auxBook NUMBER;
BEGIN
SELECT COUNT(*) INTO auxRented
FROM rent
WHERE itemid LIKE auxItemID;

SELECT COUNT(*) INTO auxBook
FROM book
WHERE bookid LIKE auxItemID;

IF auxRented > 0 THEN
DELETE FROM rent
WHERE itemid = auxItemID;
IF auxBook > 0 THEN
UPDATE book
SET avalability = 'A'
WHERE bookid LIKE auxItemID;
DBMS_OUTPUT.PUT_LINE('The book ' || auxItemID || ' is now avaible.');
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;
```

```
BEGIN
auxItemID := 'B1A123';
handleReturns_library(auxItemID);
END;
```

Snapshots



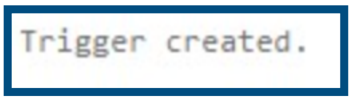
```
Statement processed.
This item is not rented at the moment
```

b. Trigger on updating the rent

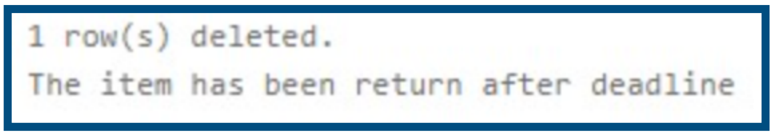
```
-- This trigger has been made to maintain refential integrity of the tables when there is
any deletion in the rent table
CREATE OR REPLACE TRIGGER modifyFines_library
AFTER DELETE
ON rent
FOR EACH ROW
DECLARE
auxCardID NUMBER;
auxItemID VARCHAR(6);
auxBook NUMBER;
auxDeby NUMBER;
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
SELECT cardid, itemid INTO auxCardID, auxItemID FROM rent WHERE cardid
LIKE :old.cardid;

SELECT COUNT(*) INTO auxBook FROM book WHERE bookid LIKE auxItemID;

IF sysdate > :old.returndate THEN
IF auxBook > 0 THEN
SELECT debyCost INTO auxDeby
FROM book
WHERE bookid LIKE auxItemID;
```

```
END IF;
UPDATE card
SET status = 'B', fines = (fines + auxDeby)
WHERE cardid LIKE auxCardID;
DBMS_OUTPUT.PUT_LINE('The item has been return after deadline');
ELSE
DBMS_OUTPUT.PUT_LINE('The item has been return before deadline');
END IF;
COMMIT;
END;
```

Snapshots



```
Trigger created.
```

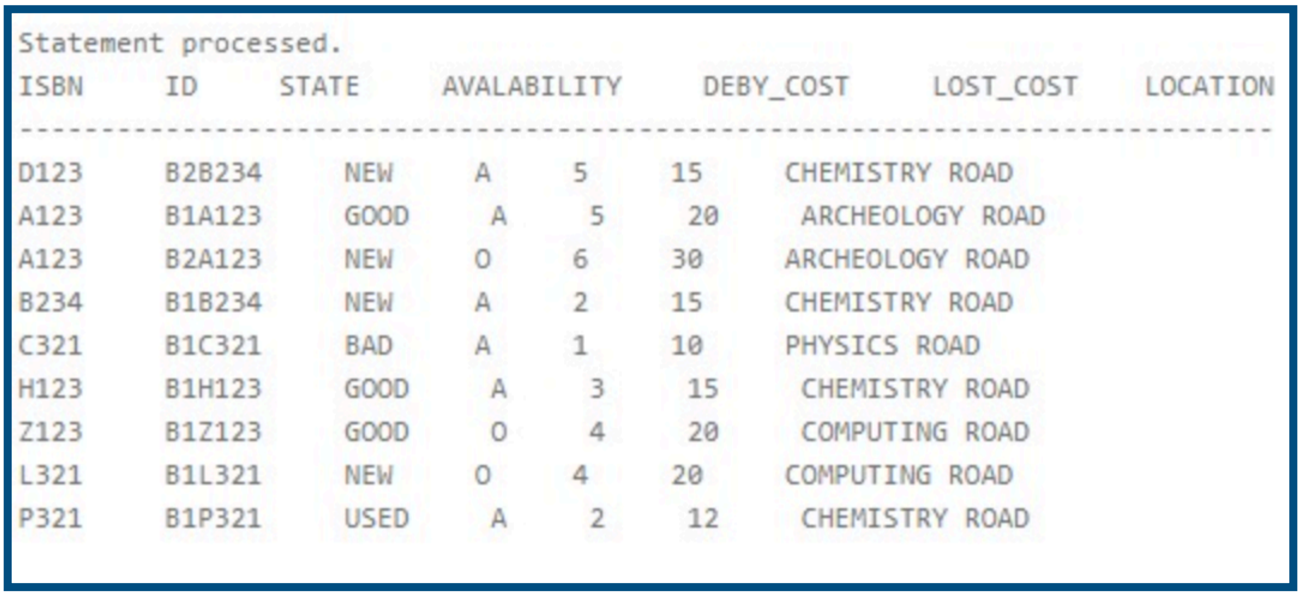


```
1 row(s) deleted.
The item has been return after deadline
```

Step 7: Displaying all books in the library

```
--This Cursor is used to print the details of all the books in the library
DECLARE
CURSOR cBooks IS
select * from book;
xBooks cBooks%ROWTYPE;
BEGIN
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.bookid || '    ' || xBooks.state
|| '    ' || xBooks.avalability || '    ' || xBooks.debycost || '    ' || xBooks.lostcost || '    ' ||
xBooks.address);
END LOOP;
CLOSE cBooks;
END;
```

Snapshot



```
Statement processed.
ISBN     ID      STATE     AVALABILITY     DEBY_COST      LOST_COST     LOCATION
-----------------------------------------------------------------------------------
D123     B2B234     NEW     A     5     15     CHEMISTRY ROAD
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
```