

Library Database System

Course Title:	Database Systems I
Course Number:	CPS 510
Semester/Year (e.g. F2016)	W2023
Instructor	Dr. Abdoreza Abhari

Assignment	Lab Assignment 10
Assignment Title	FINAL REPORT

Submission Date:	Nov. 30, 2023,
Due Date:	Nov. 30, 2023,

Student Last Name	Student First Name	Student Number	Section	Signature*
Lee	Woojin	501031919	03	W. L.
Jiang	Talon	501125811	03	T.J
Minh	Tran	501132173	03	M.T

Table of Contents

ASSIGNMENT 1:.....	2
ASSIGNMENT 2: ER_Model Library system:.....	7
ASSIGNMENT 3: RELATIONAL MODEL AND SQL STATEMENTS.....	9
ASSIGNMENT 4: SAMPLE SELECT AND QUERIES IN SQL.....	15
ASSIGNMENT 4 PART II:.....	43
ASSIGNMENT 5.....	46
ASSIGNMENT 6.....	53
ASSIGNMENT 8:.....	64
ASSIGNMENT 9:.....	72
ASSIGNMENT 10:.....	75

ASSIGNMENT 1: Logical Database Design

Introduction

In this following report, we will describe our library database system, which manages and controls the essential functions of the typical library system. As mentioned, the report will give the reader a general overview of how the library data is processed, and how the library system is to be used from a user standpoint.

General Description of the Library Database System

The purpose of the system is to emulate a library that uses a database to store information about members, staff, and other forms of media. For this system, four main entities will be considered: The librarians as administrators who manage the data of books, and the members. The media entity consists of the content and properties of all media. Furthermore, the member entity can request a viewing of the books and other information about the library. Finally, the library entity is the control system that keeps track of all the staff, media, and members. In the future, there will be some adjustments depending on the system we construct; hence, there will be some differences or some functions that we can add to our library to create an easily accessible and friendly environment for users. We will update our process along with those changes in the next reports if needed.

Usage of the System

The system described in this report is run via Jar File with Java versions 16 and 21, and includes all necessary dependencies in an Uber Jar format. The system makes use of the OJDBC Library to read and write to an Oracle 11g database system, alongside JavaFX and Apache POI to handle data and front-end interactions.

To date, we have compiled two different versions of the application, an Alpha version described in Assignment 9, which makes use of simple console interactions, and a more complete version described in Assignment 10, which makes use of a login system and graphical user interfaces.

Using the Console Application

The Library System in Assignment 9 consists of a terminal based selection program, which creates interactions with the database depending on the option selected by the user.

OpenVPN connections to the Toronto Metropolitan network are required to use the system properly, as the interactions with the database will not execute successfully otherwise.

Options A-C involve the manipulation of tables in the database; To use the tables, they first must be created and populated. If there is no data in the table when the queries are run, then there will be an empty result.

Options from D-I correspond with queries in the database system, where each query performs a different action on the Database System. The output is formatted and printed to the console in each instance.

Inputting option J will exit the System and save the state of the database.

Using the Graphical Application

The graphical user interface system described in Assignment 10 is a more completed version of the project, however not all functionality has been implemented due to time constraints in the project timeline. Future implementations of these functions are planned.

The graphical user interface currently consists of a login page, a query page, and a table modification area which allows the user to add new entries to the database system. The application is configured also to be an Uber jar containing all necessary dependencies for system to function, alongside resources which it uses to interact with tables.

When first running the system, the application will create several files which contain commands necessary for running the program. These files are then read by the program, of which the data inside is used to create, drop, and populate tables. Like the version described in Assignment 9, this version of the program also requires the user to connect to the Toronto Metropolitan network via VPN in order to execute queries successfully.

To use the system, the user must first log in as an existing user to the system. Functionality is planned to allow administrators to authenticate new users and create user accounts.

When a user logs in, they are presented by a panel which allows them to make specific queries, as well as search for media by name. An output log below shows the result of each interaction.

If the user chooses to add or delete data, they will be presented with a screen which allows them to add and delete media and member accounts. It was chosen to not allow the user to modify the history of transactions due to the possibility that they may tamper with the data. Functionality is planned to allow librarians to interact and view transactions and feedback tables.

Functions of the Database System

Entity	Functions
Media (Books, CDs, Articles, etc.)	Holds information about the content of the media Can be borrowed and reserved by members
Librarians	Manages the members in the library system Allow users to reset their password Receive feedback from users Manages the media in the database system
Members	Send requests to view media Borrow and reserve media Create accounts to access the system Send feedback to librarians
Feedback	Keep track of the librarians and member list Keep track of the media in the library Sorts(searches) for the media depending on the member's request
Transaction details	Keeps track of orders (Reservations, borrows, returns) in the system Sends overdue transactions to penalties

Penalties	Keeps track of penalties for members Applies penalties to members
-----------	--

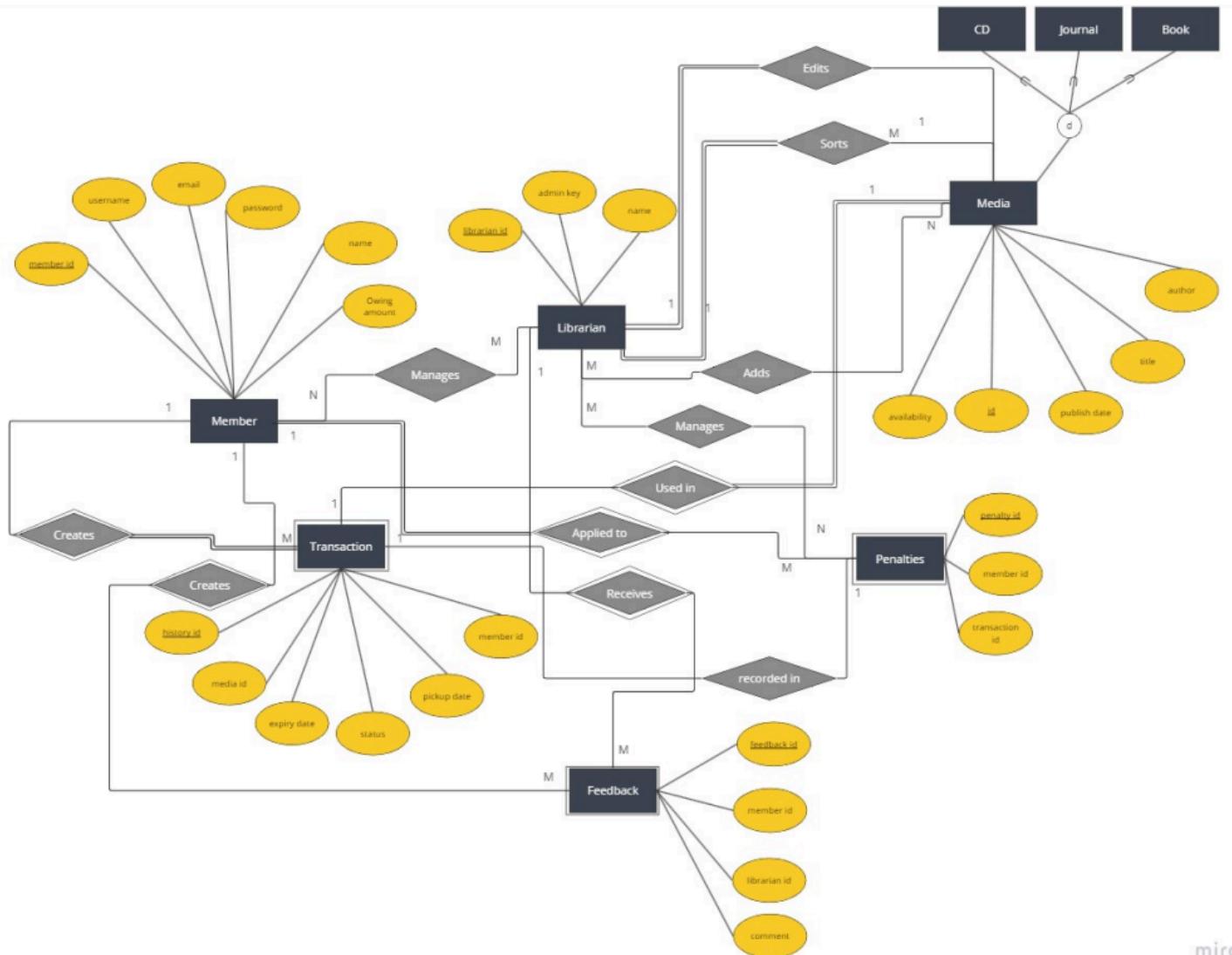
Relationships between entities

Entity	Relationships
Media	<ul style="list-style-type: none"> - Relationship between the librarian and itself for when new content is added to the system. It also stores the system. (Many : Many) - Relationship between itself to transaction history for when it gets borrowed or reserved, the media can be tracked down. (1 : Many)
Librarian	<ul style="list-style-type: none"> - Relationship between itself and the members as the Librarian is able to View, edit the information of the members if necessary. (1:Many) - Relationship between itself and the media to edit, add or sort the data. (1:Many) - Relationship between itself and the feedback as it receives feedback from users, which is inputted into the system. (1 : many)
Members	<ul style="list-style-type: none"> - Relationship between itself and the librarian when a member account is created and is stored into the

	<p>database (Many : Many)</p> <ul style="list-style-type: none"> - Relationship between itself and the transaction history as when the user wants to borrow or reserve the media, the information gets sent in. (1 : many) - Relationship between itself and the penalty as if the member was unsuccessful in returning media on time, some sort of penalty will be applied to the member. (1 : many)
Transaction details	<ul style="list-style-type: none"> ● Relationship between itself and members as when members make order, its id gets recorded (many : 1) ● Relationship between itself and books as when the transaction occurs, that book is now borrowed/reserved to a member (1 to 1) ● Relationship between itself and the penalty system as if the member fails to return the media on time, it gets sent to the feedback system. (1 : 1)
Penalty	<ul style="list-style-type: none"> ● Relationship between itself and the Transaction details as all the overdue information gets transferred over to the feedback section. (1 : 1) ● Relationship between itself and the librarian as it notifies the librarians about the overdue books (1 : 1) ● Relationship between itself and the members to apply

	penalties if they have overdue media. (Many : 1)
Feedback	<ul style="list-style-type: none"> ● Relationship between itself and the members to receive the penalty. () ● Relationship between itself and the Librarians to deliver the feedback to the librarian. ()

ASSIGNMENT 2: ER MODEL OF THE SYSTEM:



miro

Figure 1: ER model of library system

Entities in the System

Media

Attribute	Attribute type
author	varchar
title	varchar
publish date	Date
id	Int (primary key)
availability	Int

Transactiondetails

Attribute	Attribute type
history id	Int (primary key)
media id	Int (Foreign key)
expiry date	Date
status	Int
pickup date	Date
member id	Int (Foreign key)

Penalties

Attribute	Attribute type
penalty id	Int (Primary key)
member id	Int (Foreign key)
transaction id	Int (Foreign key)

Members

Attribute	Attribute type
MemID	Int (Primary key)
Username	varchar
Pass	varchar
Email	varchar
Name	varchar
AmountOwed	Double

Feedback

Attribute	Attribute type
feedback id	Int (Primary key)
member id	Int (Foreign key)
librarian id	Int (Foreign key)
comment	String
Stars	Int

Librarian

Attribute	Attribute type
LibID	Int (Primary key)
AdminKey	Int (Primary key)
Name	varchar

In Figure 1, our team has identified six main entities that play integral roles in our library system:

- **Members:** MEMBERS represent registered users who have access to library services. They are identified by a unique member ID and use usernames and passwords for authentication within the system. This entity is fundamental to user management and access control.
- **Media:** This entity serves as a representation of various types of library content, encompassing items such as books, CDs, and journal articles. It includes the author's name, title, publish date, and availability. Media plays a pivotal role in the library system, enabling users to borrow, reserve, and interact with the available resources.
- **Transaction:** The Transaction entity records the various transactions within the library system, including borrowing, reservation, and return activities. It maintains essential information such as transaction history ID, media ID, expiry date, status, pickup date, and member ID. These records help in tracking the interactions between members and library resources.
- **Penalties:** The Penalties entity manages penalties imposed on library members for overdue items. It maintains a record of penalties associated with specific members and transactions. This entity plays a crucial role in ensuring the timely return of borrowed items.
- **Feedback:** The Feedback entity facilitates communication between library users and librarians. It receives input from members, allowing them to express their thoughts and concerns. This feedback is associated with specific members and librarians, enabling effective communication and response.
- **Librarians:** Librarians are the administrative backbone of the library system. They manage member accounts, facilitate password resets, handle user feedback, and maintain the library's media inventory. They have one-to-many relationships with members, media, and feedback, allowing them to oversee these critical aspects of the system efficiently.

ASSIGNMENT 3: RELATIONAL MODEL AND SQL STATEMENTS

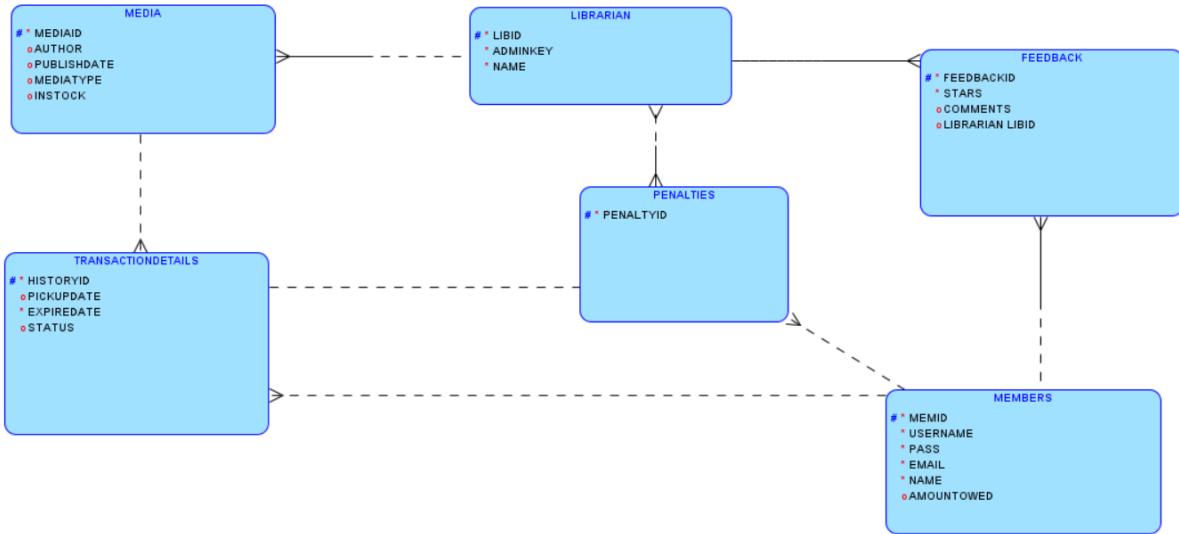


Figure 2: Logical model of library management

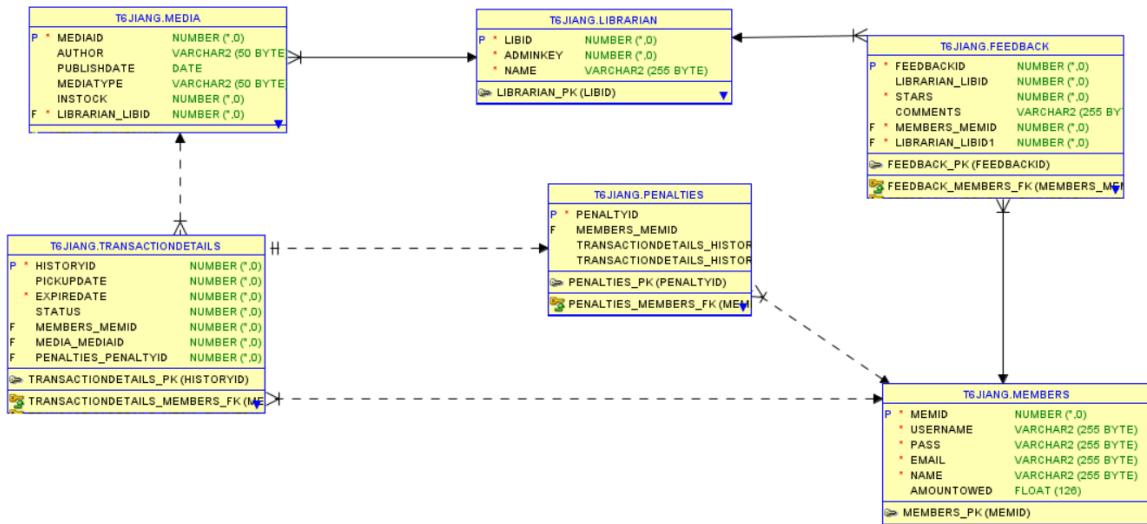


Figure 3: Relational model of library management

SQL TABLE CREATION STATEMENTS

<pre>CREATE TABLE Media(MediaID INT PRIMARY KEY, author VARCHAR(50) DEFAULT 'Unknown', MediaTitle VARCHAR(50) NOT NULL, publishDate DATE DEFAULT '1900-01-01', MediaType VARCHAR(50) DEFAULT 'undefined', inStock int DEFAULT 1);</pre>	<pre>CREATE TABLE Members (MemID int PRIMARY KEY, Username varchar(255) NOT NULL, Pass varchar(255) NOT NULL, Email varchar(255) NOT NULL, Name varchar(255) NOT NULL, AmountOwed double PRECISION Default 0.0);</pre>
<pre>CREATE TABLE Librarian (LibID int PRIMARY KEY, AdminKey int NOT NULL, Name varchar(255) NOT NULL);</pre>	<pre>CREATE TABLE TransactionDetails (HistoryID int PRIMARY KEY, PickupDate Date DEFAULT '1900-01-01', ExpireDate Date NOT NULL, Status int Default 0, MemID int, MediaID int, FOREIGN KEY(MemID) REFERENCES Members(MemID), FOREIGN KEY(MediaID) REFERENCES Media (MediaID), CHECK (Status >= 0 AND Status <= 4));</pre>
<pre>CREATE TABLE Penalties(PenaltyID int PRIMARY KEY, MemID int, HistoryID int, FOREIGN KEY(MemID) REFERENCES Members(MemID), FOREIGN KEY(HistoryID) REFERENCES TransactionDetails (HistoryID));</pre>	<pre>CREATE TABLE Feedback (FeedbackID int PRIMARY KEY, MemID int, LibID int, Stars int NOT Null, Comments varChar(255) Default 'No Feedback', FOREIGN KEY(MemID) REFERENCES Members(MemID), FOREIGN KEY(LibID) REFERENCES Librarian (LibID), CHECK (Stars >= 0 AND Stars <= 5));</pre>

Figure 4: SQL code for the Library System

TransactionDetails

The screenshot shows a database management interface with a query builder and a results grid.

Query Builder:

```

53: HistoryID int,
54: FOREIGN KEY (MemID) REFERENCES Members(MemID),
55: FOREIGN KEY(HistoryID) REFERENCES TransactionDetails (HistoryID)
56: ;
57:
58: select * from media;
59:
60: SELECT * FROM members;
61: SELECT * FROM Librarian;
62: SELECT * FROM transactiondetails;
63:
64:
65:
66:
67: drop table mediapublishdate;
68: drop table reg_member;

```

Results Grid:

HISTORYID	PICKUPDATE	EXPIREDATE	STATUS	MEMID	MEDIAID
1	1	15031900	15033400	1	1
2	2	30052003	30053503	0	2
3	3	45072106	45073606	1	3
4	4	60092209	60093709	0	4
5	5	75112312	75113812	0	5
6	6	90132415	90133915	1	6
7	7	105152518	105154018	1	7
8	8	120172621	120174121	0	8
9	9	135192724	135194224	1	9
10	10	150212827	150214327	0	10

Members

The screenshot shows a database management interface with a data preview tool and a query builder.

Data Preview:

Source: Local File
File: C:\Users\jang\Downloads\MEDIA.xlsx

File Format:
Header
Format: excel 95-2003 (xls)
Skip Rows: 0
Preview Row Limit: 100
Worksheet: Members

Query Builder:

```

53: HistoryID int,
54: FOREIGN KEY (MemID) REFERENCES Members(MemID),
55: FOREIGN KEY(HistoryID) REFERENCES TransactionDetails (HistoryID)
56: ;
57:
58: |-----|-----|
59: |-----|-----|
60: |-----|-----|
61: |-----|-----|
62: |-----|-----|
63: |-----|-----|
64: |-----|-----|
65: |-----|-----|
66: |-----|-----|
67: |-----|-----|
68: |-----|-----|

```

Results Grid:

MEMID	USERNAME	PASS	EMAIL	NAME	AMOUNTOWED
1	MinhTran	CPS_510 co...	mttran@tor...	Minh	69
2	Woojin Lee	ELE_532	Wjlee@toro...	Woojin	85
3	Talon	CPS_510 co...	mttran@tor...	Talon	79
4	MinhTran	ELE_533	Wjlee@toro...	Minh	20
5	Woojin Lee	CPS_510 co...	mttran@tor...	Woojin	6
6	Talon	ELE_534	Wjlee@toro...	Talon	72
7	MinhTran	CPS_510 co...	mttran@tor...	Minh	58
8	Woojin Lee	ELE_535	Wjlee@toro...	Woojin	16
9	Talon	CPS_510 co...	mttran@tor...	Talon	29
10	MinhTran	ELE_536	Wjlee@toro...	Minh	79

Feedback

The screenshot shows the 'Data Preview' step of the Data Import Wizard. The 'Source' is set to 'Local File' with the file path 'C:\Users\jiang\Downloads\MEDIA.xlsx'. The 'Format' is 'excel 95-2003 (.xls)'. The 'Worksheet' is set to 'Feedback'. The 'File Contents' table shows the following data:

FEEDBACKID	MEMID	LIBID	STARS	COMMENTS
1	1	1	1	no good
2	2	2	1	good
3	3	3	3	no good
4	4	4	4	good
5	5	5	2	no good
6	6	6	5	good
7	7	7	1	no good
8	8	8	5	good
9	9	9	4	no good
10	10	10	3	good

The screenshot shows the 'Query Builder' tab of the SQL Worksheet. The query is:

```

57   select * from media;
58
59   SELECT * FROM members;
60   SELECT * FROM Librarian;
61   SELECT * FROM transactiondetails;
62   SELECT * FROM penalties;
63
64   SELECT * FROM feedback;
65
66
67
68
69   drop table mediapublishdate;
70   drop table reg_member;
71   drop table librarian_member;

```

The 'Query Result' tab shows the data from the 'feedback' table:

FEEDBACKID	MEMID	LIBID	STARS	COMMENTS
1	1	1	1	no good
2	2	2	1	good
3	3	3	3	no good
4	4	4	4	good
5	5	5	5	2 no good
6	6	6	6	5 good
7	7	7	7	1 no good
8	8	8	8	5 good
9	9	9	9	4 no good
10	10	10	10	3 good

Librarian

The screenshot shows the 'Data Preview' step of the Data Import Wizard. The 'Source' is set to 'Local File' with the file path 'C:\Users\jiang\Downloads\MEDIA.xlsx'. The 'Format' is 'excel 95-2003 (.xls)'. The 'Worksheet' is set to 'Librarian'. The 'File Contents' table shows the following data:

LIBID	ADMINKEY	NAME
1	325	MinhTran
2	324	Lee
3	323	Man
4	322	MinhTran
5	321	Lee
6	320	Man
7	319	MinhTran
8	318	Lee
9	317	Man
10	316	MinhTran

The screenshot shows the 'Query Builder' tab of the SQL Worksheet. The query is:

```

53   HistoryID int,
54   FOREIGN KEY (MemID) REFERENCES Members (MemID),
55   FOREIGN KEY (HistoryID) REFERENCES TransactionDetails (HistoryID)
56 );
57
58   select * from media;
59
60   SELECT * FROM members;
61   SELECT * FROM Librarian;
62
63
64
65
66   drop table mediapublishdate;
67   drop table reg_member;
68   drop table librarian_member;

```

The 'Query Result' tab shows the data from the 'Librarian' table:

LIBID	ADMINKEY	NAME
1	1	325 MinhTran
2	2	324 Lee
3	3	323 Man
4	4	322 MinhTran
5	5	321 Lee
6	6	320 Man
7	7	319 MinhTran
8	8	318 Lee
9	9	317 Man
10	10	316 MinhTran

Media

Data Preview

Source: Local File ▾
File: C:\Users\jiang\Downloads\MEDIA.xlsx

File Format
 Header Skip Rows:
Format: excel 95-2003 (xls) Preview F

Worksheet: Media

File Contents

MEDIAID	AUTHOR	PUBLISHDATE	MEDIA_TYPE	INSTOCK
1	author 1	03/02/2003	Ebooks	0
2	author 2	03/03/2003	Books	1
3	author 3	03/04/2003	Ebooks	1
4	author 4	03/05/2003	Books	0
5	author 5	03/06/2003	Ebooks	1
6	author 6	03/07/2003	Books	1
7	author 7	03/08/2003	Ebooks	0
8	author 8	03/09/2003	Books	1
9	author 9	03/10/2003	Ebooks	1
10	author 10	03/11/2003	Books	0

ImportsRyScript.sql

```

--> SQLWorkload RyScript.qd [My] [MEDIA] Import-MEDIA.xlsx-bad_2023.09.29-20.35.40.sql
--> SQL Worksheet History
--> Query Builder
--> View: CTE/Recursive Results[1]
S1:    PenaltyID int      PRIMARY KEY,
S2:    MemberID int,
S3:    HistoryID int,
S4:    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
S5:    FOREIGN KEY (HistoryID) REFERENCES TransactionDetails(HistoryID)
S6:  ;
S7:
S8:  SELECT * FROM media
S9:  INSERT INTO media (mediaid, author, publishdate, mediatype, instock) VALUES (20, 'author_name', '2001-03-01', 'book', 1);
S10: SELECT mediaid, username FROM media WHERE mediaid > 0;
S11:
S12:
S13:
S14: INSERT INTO members (memrid, username, pass, email, name, amountowed) VALUES (18, 'username1', 'password', 'user1@gmail.com', 'name1',
S15: SELECT memrid, username FROM members WHERE memrid = 5;
S16:
S17:
S18:
S19:
S20:
S21:
S22:
S23:
S24:
S25:
S26:
S27:
S28:
S29:
S30:
S31:
S32:
S33:
S34:
S35:
S36:
S37:
S38:
S39:
S40:
S41:
S42:
S43:
S44:
S45:
S46:
S47:
S48:
S49:
S50:
S51:
S52:
S53:
S54:
S55:
S56:
S57:
S58:
S59:
S60:
S61:
S62:
S63:
S64:
S65:
S66:
S67:
S68:
S69:
S70:
S71:
S72:
S73:
S74:
S75:
S76:
S77:
S78:
S79:
S80:
S81:
S82:
S83:
S84:
S85:
S86:
S87:
S88:
S89:
S90:
S91:
S92:
S93:
S94:
S95:
S96:
S97:
S98:
S99:
S100:
S101:
S102:
S103:
S104:
S105:
S106:
S107:
S108:
S109:
S110:
S111:
S112:
S113:
S114:
S115:
S116:
S117:
S118:
S119:
S120:
S121:
S122:
S123:
S124:
S125:
S126:
S127:
S128:
S129:
S130:
S131:
S132:
S133:
S134:
S135:
S136:
S137:
S138:
S139:
S140:
S141:
S142:
S143:
S144:
S145:
S146:
S147:
S148:
S149:
S150:
S151:
S152:
S153:
S154:
S155:
S156:
S157:
S158:
S159:
S160:
S161:
S162:
S163:
S164:
S165:
S166:
S167:
S168:
S169:
S170:
S171:
S172:
S173:
S174:
S175:
S176:
S177:
S178:
S179:
S180:
S181:
S182:
S183:
S184:
S185:
S186:
S187:
S188:
S189:
S190:
S191:
S192:
S193:
S194:
S195:
S196:
S197:
S198:
S199:
S200:
S201:
S202:
S203:
S204:
S205:
S206:
S207:
S208:
S209:
S210:
S211:
S212:
S213:
S214:
S215:
S216:
S217:
S218:
S219:
S220:
S221:
S222:
S223:
S224:
S225:
S226:
S227:
S228:
S229:
S230:
S231:
S232:
S233:
S234:
S235:
S236:
S237:
S238:
S239:
S240:
S241:
S242:
S243:
S244:
S245:
S246:
S247:
S248:
S249:
S250:
S251:
S252:
S253:
S254:
S255:
S256:
S257:
S258:
S259:
S259:
S260:
S261:
S262:
S263:
S264:
S265:
S266:
S267:
S268:
S269:
S270:
S271:
S272:
S273:
S274:
S275:
S276:
S277:
S278:
S279:
S280:
S281:
S282:
S283:
S284:
S285:
S286:
S287:
S288:
S289:
S289:
S290:
S291:
S292:
S293:
S294:
S295:
S296:
S297:
S298:
S299:
S299:
S300:
S301:
S302:
S303:
S304:
S305:
S306:
S307:
S308:
S309:
S309:
S310:
S311:
S312:
S313:
S314:
S315:
S316:
S317:
S318:
S319:
S319:
S320:
S321:
S322:
S323:
S324:
S325:
S326:
S327:
S328:
S329:
S329:
S330:
S331:
S332:
S333:
S334:
S335:
S336:
S337:
S338:
S339:
S339:
S340:
S341:
S342:
S343:
S344:
S345:
S346:
S347:
S348:
S349:
S349:
S350:
S351:
S352:
S353:
S354:
S355:
S356:
S357:
S358:
S359:
S359:
S360:
S361:
S362:
S363:
S364:
S365:
S366:
S367:
S368:
S369:
S369:
S370:
S371:
S372:
S373:
S374:
S375:
S376:
S377:
S378:
S379:
S379:
S380:
S381:
S382:
S383:
S384:
S385:
S386:
S387:
S388:
S389:
S389:
S390:
S391:
S392:
S393:
S394:
S395:
S396:
S397:
S398:
S399:
S399:
S400:
S401:
S402:
S403:
S404:
S405:
S406:
S407:
S408:
S409:
S409:
S410:
S411:
S412:
S413:
S414:
S415:
S416:
S417:
S418:
S419:
S419:
S420:
S421:
S422:
S423:
S424:
S425:
S426:
S427:
S428:
S429:
S429:
S430:
S431:
S432:
S433:
S434:
S435:
S436:
S437:
S438:
S439:
S439:
S440:
S441:
S442:
S443:
S444:
S445:
S446:
S447:
S448:
S449:
S449:
S450:
S451:
S452:
S453:
S454:
S455:
S456:
S457:
S458:
S459:
S459:
S460:
S461:
S462:
S463:
S464:
S465:
S466:
S467:
S468:
S469:
S469:
S470:
S471:
S472:
S473:
S474:
S475:
S476:
S477:
S478:
S479:
S479:
S480:
S481:
S482:
S483:
S484:
S485:
S486:
S487:
S488:
S489:
S489:
S490:
S491:
S492:
S493:
S494:
S495:
S496:
S497:
S498:
S499:
S499:
S500:
S501:
S502:
S503:
S504:
S505:
S506:
S507:
S508:
S509:
S509:
S510:
S511:
S512:
S513:
S514:
S515:
S516:
S517:
S518:
S519:
S519:
S520:
S521:
S522:
S523:
S524:
S525:
S526:
S527:
S528:
S529:
S529:
S530:
S531:
S532:
S533:
S534:
S535:
S536:
S537:
S538:
S539:
S539:
S540:
S541:
S542:
S543:
S544:
S545:
S546:
S547:
S548:
S549:
S549:
S550:
S551:
S552:
S553:
S554:
S555:
S556:
S557:
S558:
S559:
S559:
S560:
S561:
S562:
S563:
S564:
S565:
S566:
S567:
S568:
S569:
S569:
S570:
S571:
S572:
S573:
S574:
S575:
S576:
S577:
S578:
S579:
S579:
S580:
S581:
S582:
S583:
S584:
S585:
S586:
S587:
S588:
S589:
S589:
S590:
S591:
S592:
S593:
S594:
S595:
S596:
S597:
S598:
S599:
S599:
S600:
S601:
S602:
S603:
S604:
S605:
S606:
S607:
S608:
S609:
S609:
S610:
S611:
S612:
S613:
S614:
S615:
S616:
S617:
S618:
S619:
S619:
S620:
S621:
S622:
S623:
S624:
S625:
S626:
S627:
S628:
S629:
S629:
S630:
S631:
S632:
S633:
S634:
S635:
S636:
S637:
S638:
S639:
S639:
S640:
S641:
S642:
S643:
S644:
S645:
S646:
S647:
S648:
S649:
S649:
S650:
S651:
S652:
S653:
S654:
S655:
S656:
S657:
S658:
S659:
S659:
S660:
S661:
S662:
S663:
S664:
S665:
S666:
S667:
S668:
S669:
S669:
S670:
S671:
S672:
S673:
S674:
S675:
S676:
S677:
S678:
S679:
S679:
S680:
S681:
S682:
S683:
S684:
S685:
S686:
S687:
S688:
S689:
S689:
S690:
S691:
S692:
S693:
S694:
S695:
S696:
S697:
S698:
S699:
S699:
S700:
S701:
S702:
S703:
S704:
S705:
S706:
S707:
S708:
S709:
S709:
S710:
S711:
S712:
S713:
S714:
S715:
S716:
S717:
S718:
S719:
S719:
S720:
S721:
S722:
S723:
S724:
S725:
S726:
S727:
S728:
S729:
S729:
S730:
S731:
S732:
S733:
S734:
S735:
S736:
S737:
S738:
S739:
S739:
S740:
S741:
S742:
S743:
S744:
S745:
S746:
S747:
S748:
S749:
S749:
S750:
S751:
S752:
S753:
S754:
S755:
S756:
S757:
S758:
S759:
S759:
S760:
S761:
S762:
S763:
S764:
S765:
S766:
S767:
S768:
S769:
S769:
S770:
S771:
S772:
S773:
S774:
S775:
S776:
S777:
S778:
S779:
S779:
S780:
S781:
S782:
S783:
S784:
S785:
S786:
S787:
S788:
S789:
S789:
S790:
S791:
S792:
S793:
S794:
S795:
S796:
S797:
S798:
S799:
S799:
S800:
S801:
S802:
S803:
S804:
S805:
S806:
S807:
S808:
S809:
S809:
S810:
S811:
S812:
S813:
S814:
S815:
S816:
S817:
S818:
S819:
S819:
S820:
S821:
S822:
S823:
S824:
S825:
S826:
S827:
S828:
S829:
S829:
S830:
S831:
S832:
S833:
S834:
S835:
S836:
S837:
S838:
S839:
S839:
S840:
S841:
S842:
S843:
S844:
S845:
S846:
S847:
S848:
S849:
S849:
S850:
S851:
S852:
S853:
S854:
S855:
S856:
S857:
S858:
S859:
S859:
S860:
S861:
S862:
S863:
S864:
S865:
S866:
S867:
S868:
S869:
S869:
S870:
S871:
S872:
S873:
S874:
S875:
S876:
S877:
S878:
S879:
S879:
S880:
S881:
S882:
S883:
S884:
S885:
S886:
S887:
S888:
S889:
S889:
S890:
S891:
S892:
S893:
S894:
S895:
S896:
S897:
S898:
S899:
S899:
S900:
S901:
S902:
S903:
S904:
S905:
S906:
S907:
S908:
S909:
S909:
S910:
S911:
S912:
S913:
S914:
S915:
S916:
S917:
S918:
S919:
S919:
S920:
S921:
S922:
S923:
S924:
S925:
S926:
S927:
S928:
S929:
S929:
S930:
S931:
S932:
S933:
S934:
S935:
S936:
S937:
S938:
S939:
S939:
S940:
S941:
S942:
S943:
S944:
S945:
S946:
S947:
S948:
S949:
S949:
S950:
S951:
S952:
S953:
S954:
S955:
S956:
S957:
S958:
S959:
S959:
S960:
S961:
S962:
S963:
S964:
S965:
S966:
S967:
S968:
S969:
S969:
S970:
S971:
S972:
S973:
S974:
S975:
S976:
S977:
S978:
S979:
S979:
S980:
S981:
S982:
S983:
S984:
S985:
S986:
S987:
S988:
S989:
S989:
S990:
S991:
S992:
S993:
S994:
S995:
S996:
S997:
S998:
S999:
S999:
S1000:
S1001:
S1002:
S1003:
S1004:
S1005:
S1006:
S1007:
S1008:
S1009:
S1009:
S1010:
S1011:
S1012:
S1013:
S1014:
S1015:
S1016:
S1017:
S1018:
S1019:
S1019:
S1020:
S1021:
S1022:
S1023:
S1024:
S1025:
S1026:
S1027:
S1028:
S1029:
S1029:
S1030:
S1031:
S1032:
S1033:
S1034:
S1035:
S1036:
S1037:
S1038:
S1039:
S1039:
S1040:
S1041:
S1042:
S1043:
S1044:
S1045:
S1046:
S1047:
S1048:
S1049:
S1049:
S1050:
S1051:
S1052:
S1053:
S1054:
S1055:
S1056:
S1057:
S1058:
S1059:
S1059:
S1060:
S1061:
S1062:
S1063:
S1064:
S1065:
S1066:
S1067:
S1068:
S1069:
S1069:
S1070:
S1071:
S1072:
S1073:
S1074:
S1075:
S1076:
S1077:
S1078:
S1079:
S1079:
S1080:
S1081:
S1082:
S1083:
S1084:
S1085:
S1086:
S1087:
S1088:
S1089:
S1089:
S1090:
S1091:
S1092:
S1093:
S1094:
S1095:
S1096:
S1097:
S1098:
S1099:
S1099:
S1100:
S1101:
S1102:
S1103:
S1104:
S1105:
S1106:
S1107:
S1108:
S1109:
S1109:
S1110:
S1111:
S1112:
S1113:
S1114:
S1115:
S1116:
S1117:
S1118:
S1119:
S1119:
S1120:
S1121:
S1122:
S1123:
S1124:
S1125:
S1126:
S1127:
S1128:
S1129:
S1129:
S1130:
S1131:
S1132:
S1133:
S1134:
S1135:
S1136:
S1137:
S1138:
S1139:
S1139:
S1140:
S1141:
S1142:
S1143:
S1144:
S1145:
S1146:
S1147:
S1148:
S1149:
S1149:
S1150:
S1151:
S1152:
S1153:
S1154:
S1155:
S1156:
S1157:
S1158:
S1159:
S1159:
S1160:
S1161:
S1162:
S1163:
S1164:
S1165:
S1166:
S1167:
S1168:
S1169:
S1169:
S1170:
S1171:
S1172:
S1173:
S1174:
S1175:
S1176:
S1177:
S1178:
S1179:
S1179:
S1180:
S1181:
S1182:
S1183:
S1184:
S1185:
S1186:
S1187:
S1188:
S1189:
S1189:
S1190:
S1191:
S1192:
S1193:
S1194:
S1195:
S1196:
S1197:
S1198:
S1199:
S1199:
S1200:
S1201:
S1202:
S1203:
S1204:
S1205:
S1206:
S1207:
S1208:
S1209:
S1209:
S1210:
S1211:
S1212:
S1213:
S1214:
S1215:
S1216:
S1217:
S1218:
S1219:
S1219:
S1220:
S1221:
S1222:
S1223:
S1224:
S1225:
S1226:
S1227:
S1228:
S1229:
S1229:
S1230:
S1231:
S1232:
S1233:
S1234:
S1235:
S1236:
S1237:
S1238:
S1239:
S1239:
S1240:
S1241:
S1242:
S1243:
S1244:
S1245:
S1246:
S1247:
S1248:
S1249:
S1249:
S1250:
S1251:
S1252:
S1253:
S1254:
S1255:
S1256:
S1257:
S1258:
S1259:
S1259:
S1260:
S1261:
S1262:
S1263:
S1264:
S1265:
S1266:
S1267:
S1268:
S1269:
S1269:
S1270:
S1271:
S1272:
S1273:
S1274:
S1275:
S1276:
S1277:
S1278:
S1279:
S1279:
S1280:
S1281:
S1282:
S1283:
S1284:
S1285:
S1286:
S1287:
S1288:
S1289:
S1289:
S1290:
S1291:
S1292:
S1293:
S1294:
S1295:
S1296:
S1297:
S1298:
S1299:
S1299:
S1300:
S1301:
S1302:
S1303:
S1304:
S1305:
S1306:
S1307:
S1308:
S1309:
S1309:
S1310:
S1311:
S1312:
S1313:
S1314:
S1315:
S1316:
S1317:
S1318:
S1319:
S1319:
S1320:
S1321:
S1322:
S1323:
S1324:
S1325:
S1326:
S1327:
S1328:
S1329:
S1329:
S1330:
S1331:
S1332:
S1333:
S1334:
S1335:
S1336:
S1337:
S1338:
S1339:
S1339:
S1340:
S1341:
S1342:
S1343:
S1344:
S1345:
S1346:
S1347:
S1348:
S1349:
S1349:
S1350:
S1351:
S1352:
S1353:
S1354:
S1355:
S1356:
S1357:
S1358:
S1359:
S1359:
S1360:
S1361:
S1362:
S1363:
S1364:
S1365:
S1366:
S1367:
S1368:
S1369:
S1369:
S1370:
S1371:
S1372:
S1373:
S1374:
S1375:
S1376:
S1377:
S1378:
S1379:
S1379:
S1380:
S1381:
S1382:
S1383:
S1384:
S1385:
S1386:
S1387:
S1388:
S1388:
S1389:
S1390:
S1391:
S1392:
S1393:
S1394:
S1395:
S1396:
S1397:
S1398:
S1398:
S1399:
S1400:
S1401:
S1402:
S1403:
S1404:
S1405:
S1406:
S1407:
S1408:
S1409:
S1409:
S1410:
S1411:
S1412:
S1413:
S1414:
S1415:
S1416:
S1417:
S1418:
S1419:
S1419:
S1420:
S1421:
S1422:
S1423:
S1424:
S1425:
S1426:
S1427:
S1428:
S1429:
S1429:
S1430:
S1431:
S1432:
S1433:
S1434:
S1435:
S1436:
S1437:
S1438:
S1439:
S1439:
S1440:
S1441:
S1442:
S1443:
S1444:
S1445:
S1446:
S1447:
S1448:
S1449:
S1449:
S1450:
S1451:
S1452:
S1453:
S1454:
S1455:
S1456:
S1457:
S1458:
S1459:
S1459:
S1460:
S1461:
S1462:
S1463:
S1464:
S1465:
S1466:
S1467:
S1468:
S1469:
S1469:
S1470:
S1471:
S1472:
S1473:
S1474:
S1475:
S1476:
S1477:
S1478:
S1479:
S1479:
S1480:
S1481:
S1482:
S1483:
S1484:
S1485:
S1486:
S1487:
S1488:
S1489:
S1489:
S1490:
S1491:
S1492:
S1493:
S1494:
S1495:
S1496:
S1497:
S1498:
S1499:
S1499:
S1500:
S1501:
S1502:
S1503:
S1504:
S1505:
S1506:
S1507:
S1508:
S1509:
S1509:
S1510:
S1511:
S1512:
S1513:
S1514:
S1515:
S1516:
S1517:
S1518:
S1519:
S1519:
S1520:
S1521:
S1522:
S1523:
S1524:
S1525:
S1526:
S1527:
S1528:
S1529:
S1529:
S1530:
S1531:
S1532:
S1533:
S1534:
S1535:
S1536:
S1537:
S1538:
S1539:
S1539:
S1540:
S1541:
S1542:
S1543:
S1544:
S1545:
S1546:
S1547:
S1548:
S1549:
S1549:
S1550:
S1551:
S1552:
S1553:
S1554:
S1555:
S1556:
S1557:
S1558:
S1559:
S1559:
S1560:
S1561:
S1562:
S1563:
S1564:
S1565:
S1566:
S1567:
S1568:
S1569:
S1569:
S1570:
S1571:
S1572:
S1573:
S1574:
S1575:
S1576:
S1577:
S1578:
S1579:
S1579:
S1580:
S1581:
S1582:
S1583:
S1584:
S1585:
S1586:
S1587:
S1588:
S1589:
S1589:
S1590:
S1591:
S1592:
S1593:
S1594:
S1595:
S1596:
S1597:
S1598:
S1599:
S1599:
S1600:
S1601:
S1602:
S1603:
S1604:
S1605:
S1606:
S1607:
S1608:
S1609:
S1609:
S1610:
S1611:
S1612:
S1613:
S1614:
S1615:
S1616:
S1617:
S1618:
S1619:
S1619:
S1620:
S1621:
S1622:
S1623:
S1624:
S1625:
S1626:
S1627:
S1628:
S1629:
S1629:
S1630:
S1631:
S1632:
S1633:
S1634:
S1635:
S1636:
S1637:
S1638:
S1639:
S1639:
S1640:
S1641:
S1642:
S1643:
S1644:
S1645:
S1646:
S1647:
S1648:
S1649:
S1649:
S1650:
S1651:
S1652:
S1653:
S1654:
S1655:
S1656:
S1657:
S1658:
S1659:
S1659:
S1660:
S1661:
S1662:
S1663:
S1664:
S1665:
S1666:
S1667:
S1668:
S1669:
S1669:
S1670:
S1671:
S1672:
S1673:
S1674:
S1675:
S1676:
S1677:
S1678:
S1679:
S1679:
S1680:
S1681:
S1682:
S1683:
S1684:
S1685:
S1686:
S1687:
S1688:
S1689:
S1689:
S1690:
S1691:
S1692:
S1693:
S1694:
S1695:
S1696:
S1697:
S1698:
S1699:
S1699:
S1700:
S1701:
S1702:
S1703:
S1704:
S1705:
S1706:
S1707:
S1708:
S1709:
S1709:
S1710:
S1711:
S1712:
S1713:
S1714:
S1715:
S1716:
S1717:
S1718:
S1719:
S1719:
S1720:
S1721:
S1722:
S1723:
S1724:
S1725:
S1726:
S1727:
S1728:
S1729:
S1729:
S1730:
S1731:
S1732:
S1733:
S1734:
S1735:
S1736:
S1737:
S1738:
S1739:
S1739:
S1740:
S1741:
S1742:
S1743:
S1744:
S1745:
S1746:
S1747:
S1748:
S1749:
S1749:
S1750:
S1751:
S1752:
S1753:
S1754:
S1755:
S1756:
S1757:
S1758:
S1759:
S1759:
S1760:
S1761:
S1762:
S1763:
S1764:
S1765:
S1766:
S1767:
S1768:
S1769:
S1769:
S1770:
S1771:
S1772:
S1773:
S1774:
S1775:
S1776:
S1777:
S1778:
S1779:
S177
```

ASSIGNMENT 4: SAMPLE SELECT AND QUERIES IN SQL

Sample select queries for table Media

Select * FROM Media ORDER BY INSTOCK ASC;

// If the media is INSTOCK, it gets ordered on the top.

MEDIAID	MEDIATITLE	AUTHOR	PUBLISHDATE	MEDIATYPE	INSTOCK
1	1 The Science of Snails	Charles Hadwell	23/02/01	Book	0
2	18 Lisp for 1990	Moris Lisp	03/12/31	Book	0
3	17 How to build a nuclear reactor	Charles Hadwell	00/04/30	CD	0
4	7 NEAR API	Matt Shell	21/11/08	Book	0
5	10 21st Century Fashion	Sarah Crown	13/03/26	Book	0
6	4 Electromagnetic waves	Marty small	11/08/19	Journal	0
7	13 Atomic Science	Kelly Spencer	90/02/07	Book	0
8	12 Iceberg melting	Lawrence Walker	90/03/07	Journal	1
9	14 Lisp is Best	Kevin Spencer	90/11/25	Book	1
10	15 Functional Programming with Lisp	Mark Gotet	00/06/17	Book	1
11	16 C Language History	Kyle Wang	00/05/18	Book	1
12	19 Christmas tales	Mark Rober	20/12/31	CD	1
13	11 Study of Slang	Kevin Spencer	21/05/01	Journal	1
14	9 Studio C	Matt Meese	18/05/01	CD	1
15	8 Fall tunes	Matt Meese	06/03/09	CD	1
16	6 Anthropology	Venice Hare	08/03/02	Journal	1
17	5 Air currents	Marty small	00/04/07	Journal	1
18	3 C programming	Alex Ufkes	10/01/04	Book	1
19	20 Exploding cucumbers	Sarah Crown	14/05/08	Book	1
20	2 How to grow blueberries	Charles Hadwell	98/07/21	Book	1

1. Select * FROM Media ORDER BY INSTOCK ASC;

Comment : //If the media is INSTOCK, it gets ordered on the top.

Select * FROM Media ORDER BY MEDIATITLE DESC;

// Order Alphabetically (Descending) According to mediaTitle.

MEDIAID	MEDIATITLE	AUTHOR	PUBLISHDATE	MEDIATYPE	INSTOCK
1	1 The Science of Snails	Charles Hadwell	23/02/01	Book	0
2	11 Study of Slang	Kevin Spencer	21/05/01	Journal	1
3	9 Studio C	Matt Meese	18/05/01	CD	1
4	7 NEAR API	Matt Shell	21/11/08	Book	0
5	14 Lisp is Best	Kevin Spencer	90/11/25	Book	1
6	18 Lisp for 1990	Moris Lisp	03/12/31	Book	0
7	12 Iceberg melting	Lawrence Walker	90/03/07	Journal	1
8	2 How to grow blueberries	Charles Hadwell	98/07/21	Book	1
9	17 How to build a nuclear reactor	Charles Hadwell	00/04/30	CD	0
10	15 Functional Programming with Lisp	Mark Gotet	00/06/17	Book	1
11	8 Fall tunes	Matt Meese	06/03/09	CD	1
12	20 Exploding cucumbers	Sarah Crown	14/05/08	Book	1
13	4 Electromagnetic waves	Marty small	11/08/19	Journal	0
14	19 Christmas tales	Mark Rober	20/12/31	CD	1
15	3 C programming	Alex Ufkes	10/01/04	Book	1
16	16 C Language History	Kyle Wang	00/05/18	Book	1
17	13 Atomic Science	Kelly Spencer	90/02/07	Book	0
18	6 Anthropology	Venice Hare	08/03/02	Journal	1
19	5 Air currents	Marty small	00/04/07	Journal	1
20	10 21st Century Fashion	Sarah Crown	13/03/26	Book	0

2. Select * FROM Media ORDER BY MEDIATITLE DESC;

Comment : // Order Alphabetically (Descending) According to mediaTitle.

Select * FROM MEDIA ORDER BY MEDIATYPE, MEDIATITLE; // Order by Type first, and within the type, order by title.

SQL | 인출된 모든 행: 20(0.026초)

MEDIAID	MEDIATITLE	AUTHOR	PUBLISHDATE	MEDIATYPE	INSTOCK
1	10 21st Century Fashion	Sarah Crown	13/03/26	Book	0
2	13 Atomic Science	Kelly Spencer	90/02/07	Book	0
3	16C Language History	Kyle Wang	00/05/18	Book	1
4	3 C programming	Alex Ufkes	10/01/04	Book	1
5	20 Exploding cucumbers	Sarah Crown	14/05/08	Book	1
6	15 Functional Programming with Lisp	Mark Gotte	00/06/17	Book	1
7	2 How to grow blueberries	Charles Hadwell	98/07/21	Book	1
8	18 Lisp for 1990	Moris Lisp	03/12/31	Book	0
9	14 Lisp is Best	Kevin Spencer	90/11/25	Book	1
10	7 NEAR API	Matt Shell	21/11/08	Book	0
11	1 The Science of Snails	Charles Hadwell	23/02/01	Book	0
12	19 Christmas tales	Mark Rober	20/12/31	CD	1
13	8 Fall tunes	Matt Meese	06/03/09	CD	1
14	17 How to build a nuclear reactor	Charles Hadwell	00/04/30	CD	0
15	9 Studio C	Matt Meese	18/05/01	CD	1
16	5 Air currents	Marty small	00/04/07	Journal	1
17	6 Anthropology	Venice Hare	08/03/02	Journal	1
18	4 Electromagnetic waves	Marty small	11/08/19	Journal	0
19	12 Iceberg melting	Lawrence Walker	90/03/07	Journal	1
20	11 Study of Slang	Kevin Spencer	21/05/01	Journal	1

3. Select * FROM MEDIA ORDER BY MEDIATYPE, MEDIATITLE;

Comment : // Order by Type first, and within the type, order by title.

Select DISTINCT Author FROM MEDIA; // Display Unique Authors within the media database.

SQL | 인출된 모든 행: 14(0.019초)

AUTHOR
1 Marty small
2 Sarah Crown
3 Mark Gotte
4 Venice Hare
5 Lawrence Walker
6 Matt Meese
7 Kevin Spencer
8 Matt Shell
9 Mark Rober
10 Alex Ufkes
11 Kyle Wang
12 Moris Lisp
13 Charles Hadwell
14 Kelly Spencer

4. Select DISTINCT Author FROM MEDIA;

Comment // Display Unique Authors within the media database.

```
SELECT DISTINCT AUTHOR FROM MEDIA ORDER BY AUTHOR DESC; // Display Unique Authors within the media database, and sort them by descending alphabetical order.
```

스크립트 출력 x 질의 결과 x

SQL | 인출됨 모든 행: 14(0.02초)

AUTHOR
1 Venice Hare
2 Sarah Crown
3 Morris Lisp
4 Matt Shell
5 Matt Meese
6 Marty small
7Mark Rober
8Mark Gotet
9 Lawrence Walker
10 Kyle Wang
11 Kevin Spencer
12 Kelly Spencer
13 Charles Hadwell
14 Alex Urkes

5. SELECT DISTINCT AUTHOR FROM MEDIA ORDER BY AUTHOR DESC;

Comment // Display Unique Authors within the media database, and sort them by descending alphabetical order.

```
SELECT MEDIATYPE, COUNT(*) AS TotalCount FROM MEDIA GROUP BY MEDIATYPE; // Display Only the media type, and count how many duplicated media types there are.
```

스크립트 출력 x 질의 결과 x

SQL | 인출됨 모든 행: 3(0.016초)

MEDIATYPE	TOTALCOUNT
1 CD	4
2 Journal	5
3 Book	11

6. SELECT MEDIATYPE as "Media Type" , COUNT(*) AS "Total Count" FROM MEDIA GROUP BY MEDIATYPE;

Comment // Display Only the media type, and count how many duplicated media types there are.

The screenshot shows a database query execution window. The SQL query is:

```
SELECT mediatitle,instock FROM MEDIA Order By mediatitle; // Display mediatitle and instock and order them by media title alphabetically.
```

The results are displayed in a table with two columns: MEDIATITLE and INSTOCK. The data consists of 20 rows, each containing a media title and its corresponding instock value. The titles are listed in alphabetical order.

MEDIATITLE	INSTOCK
1 21st Century Fashion	0
2 Air currents	1
3 Anthropology	1
4 Atomic Science	0
5 C Language History	1
6 C programming	1
7 Christmas tales	1
8 Electromagnetic waves	0
9 Exploding cucumbers	1
10 Fall tunes	1
11 Functional Programming with Lisp	1
12 How to build a nuclear reactor	0
13 How to grow blueberries	1
14 Iceberg melting	1
15 Lisp for 1990	0
16 Lisp is Best	1
17 NEAR API	0
18 Studio C	1
19 Study of Slang	1
20 The Science of Snails	0

7. SELECT mediatitle,instock FROM MEDIA Order By mediatitle;

Comment // Display mediatitle and instock and order them by media title alphabetically.

Sample select statements for Members

--Selecting all users where the amount owed is greater than the average

```
SELECT amountowed, memid FROM Members WHERE amountowed >
    (SELECT AVG(amountowed) FROM Members) ORDER BY amountowed DESC;
```

The screenshot shows a SQL query editor interface. The code area contains the following SQL statement:

```
9
10 SELECT amountowed, memid FROM Members WHERE amountowed >
11     (SELECT AVG(amountowed) FROM Members) ORDER BY amountowed DESC;
12
13
14
15
16
17 select count(mediaid) from media;
18 select * from librarian;
19 select count(memid) from members;
20 select * from feedback.
```

The results pane shows a table titled "Query Result" with the following data:

AMOUNTOWED	MEMID
49	20
48	14
47	19
46	13
44	8
43	6
43	17
42	10
40	16
37	1

Below the table, the status bar indicates "All Rows Fetched: 10 in 0.039 seconds".

-- Selecting all users using gmail for their email

```
SELECT username, email, name FROM members WHERE
    email LIKE '%gmail.com';
```

```

9
10 --selecting all users using gmail
11 SELECT username, email, name FROM members WHERE
12   email LIKE '%gmail.com';
13
14
15 --finding someones password
16 SELECT distinct name, email, pass as Password FROM members WHERE email LIKE 'wwiend%'
17   AND name LIKE 'John%' ORDER BY name DESC;
18
19
20 --getting the guys with penalties
21 SELECT DISTINCT members.memid, members.username FROM members
22   INNER JOIN penalties ON penalties.memid = members.memid ORDER BY
23     members.username ASC;

```

Script Output x | Query Result x

SQL | All Rows Fetched: 18 in 0.043 seconds

USERNAME	EMAIL	NAME
1 wwiegand	wwiend@gmail.com	John
2 schinner.gladyc	schice@gmail.com	Schinner
3 orie.armstrong	orieng@gmail.com	Orie
4 jwalsh	jwalsh@gmail.com	Josh
5 ljacobson	ljacon@gmail.com	Jacob
6 andy.gerlach	andych@gmail.com	Andy
7 rdubuque	rdubue@gmail.com	Ruby
8 brekke.robb	brekbb@gmail.com	Brekke
9 bbaroletti	imbaroque@gmail.com	Bart
10 little.noel	noel21@gmail.com	Noel

– Finding the password for a member

```

SELECT distinct name, email, pass FROM members WHERE email LIKE 'wwiend%'
  AND name LIKE 'John%' ORDER BY name DESC;

```

```

13
14
15 --finding someones password
16 SELECT distinct name, email, pass as Password FROM members WHERE email LIKE 'wwiend%'
17   AND name LIKE 'John%' ORDER BY name DESC;
18
19
20 --getting the guys with penalties
21 SELECT DISTINCT members.memid, members.username FROM members
22   INNER JOIN penalties ON penalties.memid = members.memid ORDER BY
23     members.username ASC;
24
25
26 --all members who have borrowed something

```

Script Output x | Query Result x | Query Result 1 x

SQL | All Rows Fetched: 1 in 0.123 seconds

NAME	EMAIL	PASSWORD
1 John	wwiend@gmail.com	FCFLRKUI

-- Finding all members who have an entry in the penalties table.

```
SELECT DISTINCT members.memid, members.username FROM members
INNER JOIN penalties ON penalties.memid = members.memid ORDER BY
members.username ASC;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
19 --getting the guys with penalties
20
21 SELECT DISTINCT members.memid, members.username FROM members
22 INNER JOIN penalties ON penalties.memid = members.memid ORDER BY
23     members.username ASC;
24
25
26 --all members who have borrowed something
27 SELECT DISTINCT members.memid AS "Member Id", members.username AS "Username", members.email AS "Member email"
28 FROM members
29 INNER JOIN transactiondetails ON members.memid = transactiondetails.memid
```

The results pane shows a table with columns 'MEMID' and 'USERNAME'. The data is:

MEMID	USERNAME
1	fandy-gerlach
2	9bbartoletti
3	8brekke.robb
4	11icie25
5	12ipagac
6	17jenkins.fritz
7	5ljacobson
8	7rdubuque
9	15rosalia56
10	2schinner.gladysce

- Finding all members who have created a transaction

```
SELECT DISTINCT members.memid AS "Member Id", members.username AS "Username", members.email AS
"Member email"
FROM members
INNER JOIN transactiondetails ON members.memid = transactiondetails.memid
ORDER BY members.username DESC;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
23     members.username DESC;
24
25
26 --all members who have borrowed something
27 SELECT DISTINCT members.memid AS "Member Id", members.username AS "Username", members.email AS "Member email"
28 FROM members
29 INNER JOIN transactiondetails ON members.memid = transactiondetails.memid
30     ORDER BY members.username DESC;
31
32
```

The results pane shows a table with columns 'Member Id', 'Username', and 'Member email'. The data is:

Member Id	Username	Member email
1	1wriegand	wriegand@gmail.com
2	18wriegand	Williamgreat2000@gmail.com
3	19ucronin	ucronin100@gmail.com
4	2schinner.gladysce	schinner.gladysce@gmail.com
5	15rosalia56	rosalia5656@gmail.com
6	20retha.wyman	Retha.wyman@gmail.com
7	7rdubuque	rdubuque@gmail.com
8	3orie.armstrong	orie.armstrong@gmail.com
9	5ljacobson	ljacobson@gmail.com
10	10little.noel	noel21@gmail.com

– Finding all members with penalties who owe an amount

```
SELECT DISTINCT members.memid AS "Member id",
    members.username AS "Username",
    members.email AS "Member email",
    members.amountowed AS "Amount Owed"
FROM members INNER JOIN penalties ON members.memid = penalties.memid
WHERE members.amountowed > 0 ORDER BY amountowed ASC;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query for selecting members with amounts owed. The results pane shows a table with columns: Member Id, Username, Member email, and Amount Owed. The data includes 10 rows of member information.

Member Id	Username	Member email	Amount Owed
1	9bbartoletti	imbaroque@gmail.com	9
2	2schinner.gladyc	schice@gmail.com	16
3	5ljacobson	ljacon@gmail.com	27
4	15rosalia56	rosalia5656@gmail.com	29
5	11icie25	iceeyou@yahoo.ca	30
6	7rdubuque	rdubue@gmail.com	31
7	12ipagac	gataip@hotmail.com	32
8	6andy.gerlach	andych@gmail.com	43
9	17jenkins.fritz	jenkins.fritz@gmail.com	43
10	8brekke.robb	brekko@gmail.com	44

– Finding all members whose username starts with A.

```
SELECT members.memid AS "Member Id", members.username AS "Username", members.email AS "Email" FROM members
WHERE members.username LIKE 'a%' ORDER BY members.username;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query for selecting members whose usernames start with 'A'. The results pane shows a table with columns: Member Id, Username, and Email. The data includes 2 rows of member information.

Member Id	Username	Email
1	13allans54	allanturing@gmail.com
2	6andy.gerlach	andych@gmail.com

Penalties

/* 1. Search Penalties with a specific Member ID*/

```
select *from Penalties Where MemID = 5;
```

```
--  
102 | select * from Penalties;  
103 | /* 1. Search Penalties with a specific Member ID*/  
104 | select *from Penalties Where MemID = 5;  
105 |  
106 |/* 2. */
```

Query Result x | Script Output x | Query Result 1 x

SQL | All Rows Fetched: 2 in 0.02 seconds

	PENALTYID	MEMID	HISTORYID
1	6	5	6
2	7	5	11

2. List members with Penalties by Join statement

```
select Members.*, Penalties.*
```

```
From Members
```

```
Join Penalties ON Members.MemID = Penalties.MemID;
```

```

/* 2. List member with Penalties by Join statement*/
select Members.*, Penalties.*
From Members
Join Penalties ON Members.MemID = Penalties.MemID;

/* 3. */
/* 4. */
/* 5. */

```

Query Result | Script Output | Query Result 1 | All Rows Fetched: 20 in 0.092 seconds

MEMID	USERNAME	PASS	EMAIL	NAME	AMOUNTOWED	PENALTYID	MEMID_1	HISTORYID
1	3 orie.armstrong	KYROYOZJ	orieng@gmail.com	Orie	16	5	3	15
2	4 jwalsh	SIQMRMSK	jwalsh@gmail.com	Josh	26	17	4	7
3	4 jwalsh	SIQMRMSK	jwalsh@gmail.com	Josh	26	1	4	2
4	5 ljacobsen	OBCUDFZN	ljacon@gmail.com	Jacob	35	7	5	11
5	5 ljacobsen	OBCUDFZN	ljacon@gmail.com	Jacob	35	6	5	6
6	7 rdubuque	LGUCVQBX	rdubue@gmail.com	Ruby	11	13	7	19
7	8 brekke.robb	DXKZJZAK	brekbb@gmail.com	Brekke	0	20	8	20
8	9 bbartoletti	LFERCCLJ	imbaroque@gmail.com	Bart	9	12	9	17
9	9 bbartoletti	LFERCCLJ	imbaroque@gmail.com	Bart	9	9	9	8
0	9 bbartoletti	LFERCCLJ	imbaroque@gmail.com	Bart	9	2	9	4
1	10 little.noel	SJHLJWOT	noel21@gmail.com	Noel	0	18	10	18
2	13 allan54	ZAXEWAGN	allanturing@gmail.com	Allan	17	15	13	13

3. retrieving distinct HistoryID values associated with penalties

SELECT DISTINCT HistoryID FROM Penalties;

```

111  /*retrieving distinct HistoryID values associated with penalties */
112  SELECT DISTINCT HistoryID FROM Penalties;
113
114
115  /* 4. Penalties for Transaction according to the date range*/
116  SELECT Penalties.*;

```

Query Result | Script Output | Query Result 1 | All Rows Fetched: 20 in 0.041 seconds

HISTORYID	
1	1
2	6
3	11
4	13
5	2
6	14
7	20
8	4
9	5
10	8
11	17
12	3

4. Penalties for Transaction according to the date range

```
SELECT Penalties.*  
FROM Penalties  
JOIN TransactionDetails ON Penalties.HistoryID = TransactionDetails.HistoryID  
WHERE TransactionDetails.PickupDate BETWEEN '01-JAN-23' AND '05-JAN-23';
```

```
/* 4. Penalties for Transaction according to the date range*/
```

```
SELECT Penalties.*  
FROM Penalties  
JOIN TransactionDetails ON Penalties.HistoryID = TransactionDetails.HistoryID  
WHERE TransactionDetails.PickupDate BETWEEN '01-JAN-23' AND '05-JAN-23';
```

```
/* 5. */  
/* 6. */  
/* 7. */
```

PENALTYID	MEMID	HISTORYID
1	1	4
2	2	9
3	3	15
4	4	13
		12

5 Count the Number of Penalties per Member:

```
SELECT MemID, COUNT(*) AS PenaltyCount  
FROM Penalties  
GROUP BY MemID;
```

```

121 /* 5 Count the Number of Penalties per Member:
122
123 SELECT MemID, COUNT(*) AS PenaltyCount
124 FROM Penalties
125 GROUP BY MemID;
126
127 /* 6. */
128 /* 7. */

```

Query Result | Script Output | Query Result 1

SQL | All Rows Fetched: 12 in 0.022 seconds

MEMID	PENALTYCOUNT
1	13
2	4
3	5
4	17
5	8
6	3
7	18
8	7
9	9
10	15
11	19
12	10

6. retrieving all penalties from the "Penalties" table and order them by the "PenaltyID" in ascending order.

SELECT * FROM Penalties
ORDER BY PenaltyID ASC;

```
126 /* 6. retrieving all penalties from the "Penalties" table and orders them by the "PenaltyID" in ascending order.*/
127 SELECT * FROM Penalties
128 ORDER BY PenaltyID ASC;
```

Query Result x | Script Output x | Query Result 1 x | Query Result 2 x | Query Result 3 x

SQL | All Rows Fetched: 20 in 0.028 seconds

PENALTYID	MEMID	HISTORYID
1	1	4
2	2	9
3	3	15
4	4	13
5	5	3
6	6	5
7	7	5
8	8	18
9	9	9
10	10	19
11	11	18
12	12	9
13	13	1
14	14	17
15	15	12
16	16	10
17	17	11
18	18	15
19	19	6
20	20	13

7. Retrieving penalties for a specific transaction (specified by HistoryID).

```
SELECT * FROM Penalties WHERE HistoryID = 12;
```

```
128 /* 7. Retrieving penalties for a specific transaction (specified by HistoryID).*/
129 SELECT * FROM Penalties WHERE HistoryID = 12;
130
131
132
```

Query Result x | Script Output x | Query Result 1 x

SQL | All Rows Fetched: 1 in 0.029 seconds

PENALTYID	MEMID	HISTORYID
1	4	13
2	13	12

Librarian

1. Select librarians with names starting with a specific letter

```
SELECT * FROM Librarian WHERE Name LIKE 'M%';
```

```
/* 1 Select librarians with names starting with a specific letter*/
SELECT * FROM Librarian WHERE Name LIKE 'M%';
```

```
/* 2 Count the total number of librarians */
```

Query Result x | Query Result 1 x | Query Result 2 x | Query Result 3 x | Query Result
SQL | All Rows Fetched: 2 in 0.023 seconds

LIBID	ADMINKEY	NAME
1	1	801 Minh Tran
2	5	321 Morgan Freeman

2. Count the total number of librarians

```
SELECT COUNT(*) FROM Librarian;
```

```
76
77 /* 2 Count the total number of librarians */
78 SELECT COUNT(*) FROM Librarian;
79
80 /*3 Select librarians with distinct names ordered by name*/
```

Query Result x | Query Result 1 x | Query Result 2 x | Query Result 3 x | Query Result
SQL | All Rows Fetched: 1 in 0.035 seconds

COUNT(*)
1 5

3. Select librarians with distinct names ordered by AdminKey

```
select COUNT(*) FROM ( select distinct AdminKey FROM Librarian);
```

```
80  /*3 Select librarians with distinct names ordered by AdminKey*/
81  select COUNT(*) FROM ( select distinct AdminKey FROM Librarian);
82
83  /* 4 Select all librarians sorted by name in ascending order*/
```

Query Result x | Query Result 1 x | Query Result 2 x | Query Result 3 x

SQL | All Rows Fetched: 1 in 0.038 seconds

Count(*)
1 5

4. Select all librarians sorted by name in ascending order

```
select * from Librarian order by name ASC;
```

```
83  /* 4 Select all librarians sorted by name in ascending order*/
84  select * from Librarian order by name ASC;
85
86  /* 5 Count the number of librarians with lowest AdminKeys*/
```

Query Result x | Query Result 1 x | Query Result 2 x

SQL | All Rows Fetched: 5 in 0.018 seconds

LIBID	ADMINKEY	NAME
1	3	323 Ho Kim Lee
2	4	322 Kevin Jones
3	2	324 Kimberly Delamere
4	1	801 Minh Tran
5	5	321 Morgan Freeman

5. Count the number of librarians with lowest AdminKeys

```
select * from Librarian where Adminkey = (select min(Adminkey) FROM Librarian);
```

```
86 /* 5 Count the number of librarians with lowest AdminKeys*/
87 select * from Librarian where Adminkey = (select min(Adminkey) FROM Librarian);
88
89 /*6 Show information of Librarian according to LibID */
```

Query Result X | Query Result 1 X

SQL | All Rows Fetched: 1 in 0.024 seconds

LIBID	ADMINKEY	NAME
1	5	321 Morgan Freeman

6. Show information of Librarian according to LibID

```
select LibID, Name From Librarian;
```

```
89 /*6 Show information of Librarian according to LibID */
90 select LibID, Name From Librarian;
91
92 /* 7 Searching for specific Librarian name */
```

Query Result X

SQL | All Rows Fetched: 5 in 0.043 seconds

LIBID	NAME
1	1 Minh Tran
2	2 Kimberly Delamere
3	3 Ho Kim Lee
4	4 Kevin Jones
5	5 Morgan Freeman

7. Searching for specific Librarian name.

```
select * from Librarian Where NAME LIKE 'Minh Tran' ;
```

```
/* 7 Searching for specific Librarian name */
```

```
select * from Librarian Where NAME LIKE 'Minh Tran' ;
```

My Result		
LIBID	ADMINKEY	NAME
1	801	Minh Tran

Transaction details

```
SELECT * FROM TransactionDetails Order By PickupDate Asc; // sort by earliest pickup dates
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 20(0.018초)

HISTORYID	PICKUPDATE	EXPIREDATE	STATUS	MEMID	MEDIAID
1	23/01/02	23/01/09	1	1	13
2	4 23/01/03	23/01/10	0	19	14
3	5 23/01/04	23/01/11	1	12	15
4	12 23/01/05	23/01/09	0	16	20
5	15 23/01/06	23/01/14	0	6	1
6	6 23/01/07	23/01/19	1	2	16
7	11 23/01/08	23/01/09	1	15	17
8	10 23/01/09	23/01/19	0	13	5
9	8 23/01/10	23/01/22	1	5	6
10	9 23/01/11	23/01/19	0	17	12
11	1 23/01/12	23/01/20	0	11	19
12	17 23/01/13	23/01/21	1	20	10
13	19 23/01/14	23/01/22	1	9	11
14	3 23/01/15	23/01/23	0	4	7
15	13 23/01/16	23/01/24	1	14	8
16	14 23/01/17	23/01/25	1	3	9
17	7 23/01/18	23/01/26	1	8	18
18	18 23/01/19	23/01/27	0	10	2
19	16 23/01/20	23/01/28	1	18	3
20	20 23/01/21	23/01/29	1	7	4

1. SELECT * FROM TransactionDetails Order By PickupDate Asc;

Comment// sort by earliest pickup dates.

```
SELECT Status, COUNT(*) AS TotalReturned FROM TransactionDetails GROUP BY Status; // Count how many people have returned the book (1 is returned, and 0 is not)
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 2(0.024초)

STATUS	TOTALRETURNED
1	12
0	8

2. SELECT Status, COUNT(*) AS TotalReturned FROM TransactionDetails GROUP BY Status; Comment //

Count how many people have returned the book (1 is returned, and 0 is not)

```
SELECT * From transactionDetails Where expiredate > '2023-01-20' ORDER by ExpireDate Desc; //find the expired book starting of Jan 20.
```

스크립트 출력 x | 질의 결과 2 x

SQL | 인출된 모든 행: 10(0.022초)

HISTORYID	PICKUPDATE	EXPIREDATE	STATUS	MEMID	MEDIAID
1	20 23/01/21	23/01/29	1	7	4
2	16 23/01/20	23/01/28	1	18	3
3	18 23/01/19	23/01/27	0	10	2
4	7 23/01/18	23/01/26	1	8	18
5	14 23/01/17	23/01/25	1	3	9
6	13 23/01/16	23/01/24	1	14	8
7	3 23/01/15	23/01/23	0	4	7
8	19 23/01/14	23/01/22	1	9	11
9	8 23/01/10	23/01/22	1	5	6
10	17 23/01/13	23/01/21	1	20	10

3. SELECT * From transactionDetails Where expiredate > '2023-01-20' ORDER by ExpireDate Desc;
 Comment //find the expired book starting Jan 20.

```
SELECT expireDate,MemID From TransactionDetails Where Status = 0 AND expiredate < SYSDATE; // Find a member with unreturned book that will be expired
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 8(0.02초)

EXPIREDATE	MEMID
1 23/01/10	19
2 23/01/09	16
3 23/01/14	6
4 23/01/19	13
5 23/01/19	17
6 23/01/20	11
7 23/01/23	4
8 23/01/27	10

4. SELECT expireDate,MemID From TransactionDetails Where Status = 0 AND expiredate < SYSDATE;
 Comment // Find a member with unreturned book that will be expired

```
SELECT DISTINCT pickupdate From transactionDetails; // Find unique pickupdate (ALL of the variable have distinct values)
```

스크립트 출력 x 질의 결과 5 x
SQL | 인출된 모든 행: 20(0,018초)

PICKUPDATE
1 23/01/10
2 23/01/12
3 23/01/13
4 23/01/15
5 23/01/20
6 23/01/03
7 23/01/21
8 23/01/05
9 23/01/16
10 23/01/02
11 23/01/08
12 23/01/09
13 23/01/07
14 23/01/11
15 23/01/14
16 23/01/17
17 23/01/18
18 23/01/19
19 23/01/04
20 23/01/06

5. SELECT DISTINCT pickupdate From transactionDetails;

Comment // Find unique pickupdate (ALL of the variable have distinct values)

```
SELECT * From Transactiondetails WHERE Pickupdate BETWEEN '2023-01-10' AND '2023-01-15'; //Pickupdate between jan 10 to jan 15.
```

스크립트 출력 x 질의 결과 2 x
SQL | 인출된 모든 행: 6(0,018초)

HISTORYID	PICKUPDATE	EXPIREDATE	STATUS	MEMID	MEDIAID
1	8 23/01/10	23/01/22	1	5	6
2	9 23/01/11	23/01/19	0	17	12
3	1 23/01/12	23/01/20	0	11	19
4	17 23/01/13	23/01/21	1	20	10
5	19 23/01/14	23/01/22	1	9	11
6	3 23/01/15	23/01/23	0	4	7

6. SELECT * From Transactiondetails WHERE Pickupdate BETWEEN '2023-01-10' AND '2023-01-15';

Comment //Pickupdate between Jan 10 to Jan 15.

```
SELECT * From Transactiondetails where (Status = 0) And (ExpireDate Between'2023-01-22' AND '2023-01-27'); //Find an unreturned book that is within that expiredate timeframe.
```

스크립트 출력 x 질의 결과 3 x

SQL | 인출된 모든 행: 2(0.019초)

HISTORYID	PICKUPDATE	EXPIREDATE	STATUS	MEMID	MEDIAID
1	23/01/15	23/01/23	0	4	7
2	23/01/19	23/01/27	0	10	2

7. SELECT * From Transaction details where (Status = 0) And (ExpireDate Between 2023-01-22' AND '2023-01-27');

Comment //Find an unreturned book that is within that expiredate time frame.

Feedback

1. number of comments per librarian for each member

```
SELECT COUNT(libid), memid FROM feedback  
GROUP BY memid;
```

```
52  
53 --number of comments per librarian for each member  
54 SELECT COUNT(libid), memid FROM feedback  
55 GROUP BY memid;  
56  
57 --getting reviews better than the average  
58 SELECT libid, stars, comments FROM feedback WHERE  
59 stars >= (SELECT AVG(stars) FROM feedback) ORDER BY  
60 stars DESC;  
61  
62  
63 --get all reviews for a librarian  
64 SELECT DISTINCT librarian.libid, librarian.name, feedback.stars,  
65 librarian ON librarian.libid = feedback.libid WHERE libraria
```

Script Output x Query Result x

SQL | All Rows Fetched: 13 in 0.042 seconds

COUNT(LIBID)	MEMID
1	13
2	11
3	6

2. getting reviews better than the average

```
SELECT libid, stars, comments FROM feedback WHERE  
stars >= (SELECT AVG(stars) FROM feedback) ORDER BY  
stars DESC;
```

```
57  --getting reviews better than the average  
58  SELECT libid, stars, comments FROM feedback WHERE  
59      stars >= (SELECT AVG(stars) FROM feedback) ORDER BY  
60          stars DESC;  
61  
62  
63  --get all reviews for a librarian  
64  SELECT DISTINCT librarian.libid, librarian.name, feedback.stars, feedback.comments  
       FROM librarian INNER JOIN feedback  
       ON librarian.libid = feedback.libid WHERE librarian.name = 'Ho  
65  
66  
67  
68  -- getting all reviews for each member with name  
69  SELECT members.name, feedback.memid, feedback.comments FROM feedback INNER JOIN  
70      members ON members.memid = feedback.memid ORDER BY members.memid ASC;  
71
```

LIBID	STARS	COMMENTS
1	2	5 They were my highschool teacher
2	4	5 They're ok
3	2	5 Best librarian ever but they only come on tuesdays
4	4	4 Siuuu that's nice library for student

3. get all reviews for a librarian

```
SELECT DISTINCT librarian.libid, librarian.name, feedback.stars, feedback.comments  
FROM feedback INNER JOIN  
librarian ON librarian.libid = feedback.libid WHERE librarian.name = 'Ho Kim Lee';
```

LIBID	NAME	STARS	COMMENTS
1	3 Ho Kim Lee	2	They are very quiet but otherwise good
2	3 Ho Kim Lee	1	Need update of reading offline
3	3 Ho Kim Lee	1	They should consider a career change.

4. getting all reviews for each member with name

```
SELECT members.name, feedback.memid, feedback.comments FROM feedback INNER  
JOIN members ON  
members.memid = feedback.memid ORDER BY members.memid ASC;
```

The screenshot shows a SQL query editor interface. The code area contains several SQL statements. The statement at line 68 is highlighted in blue. The results pane shows a table with three columns: NAME, MEMID, and COMMENTS. The data in the table is as follows:

NAME	MEMID	COMMENTS
1 Orie	3	I wish they followed the marking rubric
2 Orie	3	They were my highschool teacher
3 Josh	4	She stole my heart :(
4 Andy	6	They need therapy
5 Bart	9	decent
6 Bart	9	THE BEST!!!

5. -- getting all reviews for specific member SELECT members.name, feedback.memid, feedback.comments FROM feedback INNER JOIN members ON members.memid = feedback.memid where members.memid = 3 ORDER BY comments;

The screenshot shows a SQL query editor interface. The code area contains several SQL statements. The statement at line 74 is highlighted in blue. The results pane shows a table with three columns: NAME, MEMID, and COMMENTS. The data in the table is as follows:

NAME	MEMID	COMMENTS
1 Orie	3	I wish they followed the marking rubric
2 Orie	3	They were my highschool teacher

6. select feedback with keywords

```
SELECT feedback.memid, feedback.comments FROM  
feedback WHERE feedback.comments LIKE '%highschool%';
```

The screenshot shows the Oracle SQL Developer interface. On the left is the code editor with the following SQL script:

```
77 --select feedback with keywords  
78 SELECT feedback.memid, feedback.comments FROM  
79     feedback WHERE feedback.comments LIKE '%highschool%';  
80  
81  
82 --select all comments with a given rating  
83 SELECT feedback.memid, feedback.stars,  
84     feedback.comments FROM feedback WHERE feedback.stars = 5;  
85  
86
```

On the right is the 'Query Result' tab, which displays the output of the query:

	MEMID	COMMENTS
1	3	They were my highschool teacher

7. select all comments with a given rating

```
SELECT feedback.memid, feedback.stars,  
    feedback.comments FROM feedback WHERE feedback.stars = 5;
```

The screenshot shows the Oracle SQL Developer interface. On the left is the code editor with the following SQL script:

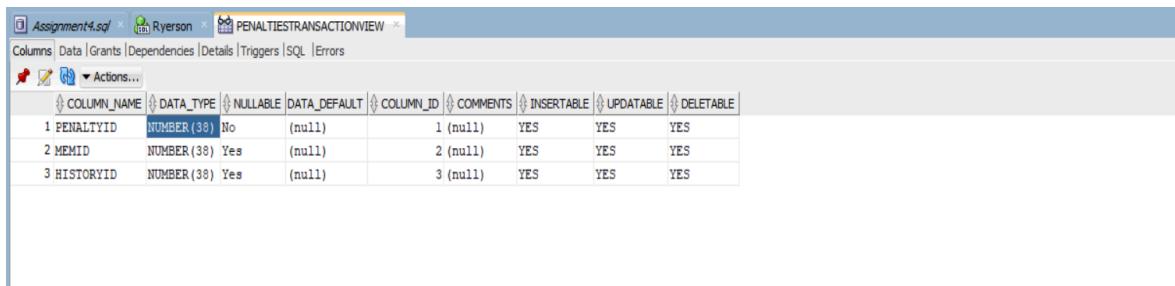
```
81  
82 --select all comments with a given rating  
83 SELECT feedback.memid, feedback.stars,  
84     feedback.comments FROM feedback WHERE feedback.stars = 5;  
85  
86  
87  
88  
89 insert into testing (id, country) values (1,'a');  
90  
91 select * from testing order by id asc;  
92  
93 select distinct id, country from testing where  
94     id > 0 order by id asc;  
95
```

On the right is the 'Query Result' tab, which displays the output of the query:

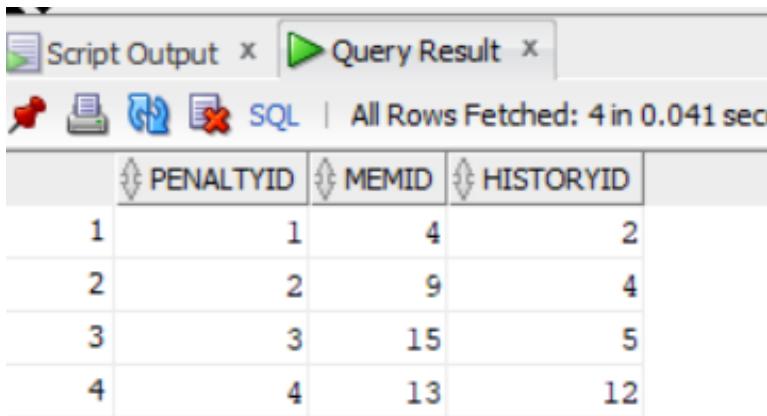
	MEMID	STARS	COMMENTS
1	3	5	They were my highschool teacher
2	19	5	They're ok
3	20	5	Best librarian ever but they only come on tuesdays

ASSIGNMENT 4 PART II:

```
Create view PenaltiesTransactionView AS  
SELECT Penalties.*  
FROM Penalties  
JOIN TransactionDetails ON Penalties.HistoryID = TransactionDetails.HistoryID  
WHERE TransactionDetails.PickupDate BETWEEN '01-JAN-23' AND '05-JAN-23';  
  
select * from PenaltiesTransactionView;
```



COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS	INSERTABLE	UPDATABLE	DELETABLE
1 PENALTYID	NUMBER(38)	No	(null)	1 (null)	YES	YES	YES	
2 MEMID	NUMBER(38)	Yes	(null)	2 (null)	YES	YES	YES	
3 HISTORYID	NUMBER(38)	Yes	(null)	3 (null)	YES	YES	YES	



PENALTYID	MEMID	HISTORYID	
1	1	4	2
2	2	9	4
3	3	15	5
4	4	13	12

```

CREATE VIEW mediaInStock AS
SELECT mediaid AS "Id",
mediatitle AS "Title",
author AS "Author",
mediatype AS "Type"
FROM media WHERE media.instock = 1 ORDER BY mediatitle ASC;

```

	Id	Title	Author	Type
1	5	Air currents	Marty small	Journal
2	6	Anthropology	Venice Hare	Journal
3	16	C Language History	Kyle Wang	Book
4	3	C programming	Alex Ufkes	Book
5	19	Christmas tales	Mark Rober	CD
6	20	Exploding cucumbers	Sarah Crown	Book
7	8	Fall tunes	Matt Meese	CD
8	15	Functional Programming with Lisp	Mark Gotet	Book
9	2	How to grow blueberries	Charles Hadwell	Book
10	12	Iceberg melting	Lawrence Walker	Journal
11	14	Lisp is Best	Kevin Spencer	Book
12	9	Studio C	Matt Meese	CD
13	11	Study of Slang	Kevin Spencer	Journal

```
create or replace view MEMBER_Credibility as  
select members.memid as "ID",  
members.name as "Name",  
members.Amountowed as "Charges" from members  
where members.amountowed < 20.0  
Order by memID asc;
```

```
select * from MEMBER_Credibility;
```

	ID	Name	Charges
1	2	Schinner	8
2	4	Josh	6
3	5	Jacob	13
4	7	Ruby	5
5	9	Bart	1
6	10	Noel	14
7	13	Allan	11
8	15	Rose	7
9	18	William	6
10	19	Allan	12

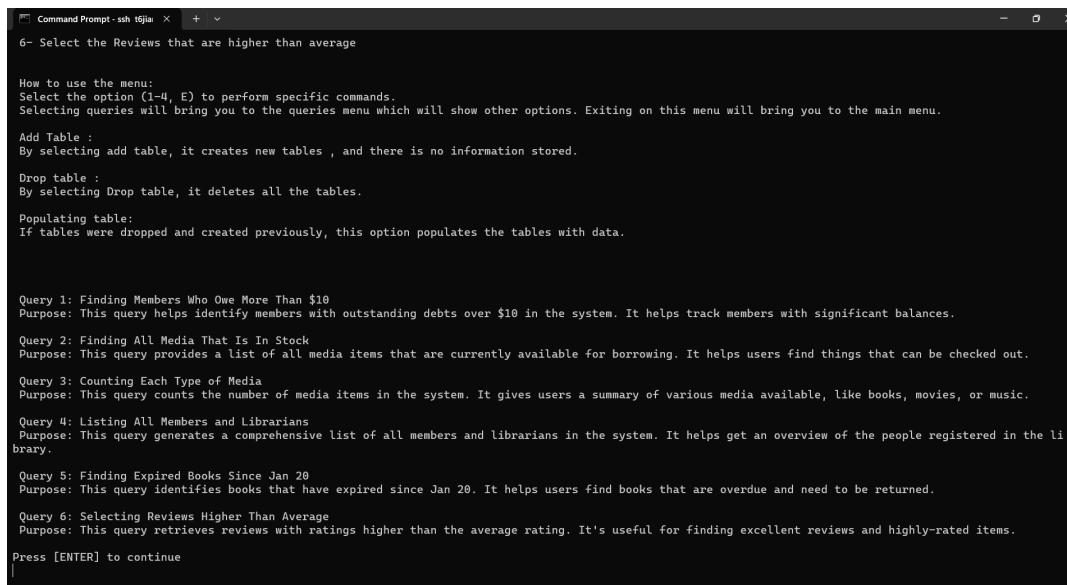
ASSIGNMENT 5

Menu program:



```
tjiang@metis:~/cps510$ ./menu.sh
=====
| Oracle All Inclusive Tool |
| Main Menu - Select Desired Operation(s): |
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|
=====
M) View Manual
1) Drop Tables
2) Create Tables
3) Populate Tables
4) Query Tables
E) End/Exit
Choose:
```

Manual:



```
6- Select the Reviews that are higher than average

How to use the menu:
Select the option (1-4, E) to perform specific commands.
Selecting queries will bring you to the queries menu which will show other options. Exiting on this menu will bring you to the main menu.

Add Table :
By selecting add table, it creates new tables , and there is no information stored.

Drop table :
By selecting Drop table, it deletes all the tables.

Populating table:
If tables were dropped and created previously, this option populates the tables with data.

Query 1: Finding Members Who Owe More Than $10
Purpose: This query helps identify members with outstanding debts over $10 in the system. It helps track members with significant balances.

Query 2: Finding All Media That Is In Stock
Purpose: This query provides a list of all media items that are currently available for borrowing. It helps users find things that can be checked out.

Query 3: Counting Each Type Of Media
Purpose: This query counts the number of media items in the system. It gives users a summary of various media available, like books, movies, or music.

Query 4: Listing All Members and Librarians
Purpose: This query generates a comprehensive list of all members and librarians in the system. It helps get an overview of the people registered in the library.

Query 5: Finding Expired Books Since Jan 20
Purpose: This query identifies books that have expired since Jan 20. It helps users find books that are overdue and need to be returned.

Query 6: Selecting Reviews Higher Than Average
Purpose: This query retrieves reviews with ratings higher than the average rating. It's useful for finding excellent reviews and highly-rated items.

Press [ENTER] to continue
```

Drop tables:

```
Command Prompt - ssh t6jia* + ▾
1) Drop Tables
2) Create Tables
3) Populate Tables
4) Query Tables
E) End/Exit
Choose:
1
./functions/droptables.sh: line 14: warning: here-document at line 4 delimited by end-of-file (wanted 'EOF')
SQL*Plus: Release 12.1.0.2.0 Production on Tue Oct 24 18:32:41 2023
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>
Table dropped.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
```

Create tables:

```
Command Prompt - ssh t6jia* + ▾
1) Drop Tables
2) Create Tables
3) Populate Tables
4) Query Tables
E) End/Exit
Choose:
2
./functions/createtables.sh: line 69: warning: here-document at line 4 delimited by end-of-file (wanted 'EOF')
SQL*Plus: Release 12.1.0.2.0 Production on Tue Oct 24 18:33:01 2023
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL> 2 3 4 5 6 7
Table created.

SQL> SQL> SQL> 2 3 4 5 6 7 8
Table created.

SQL> SQL> 2 3 4 5
Table created.

SQL> SQL> SQL> 2 3 4 5 6 7 8 9 10 11
Table created.

SQL> SQL> SQL> 2 3 4 5 6 7 8 9 10
Table created.

SQL> SQL> SQL> 2 3 4 5 6 7
Table created.

SQL> SQL> SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
```

Populate tables:

```
Command Prompt - ssh t6jia  x  +  ▾
SQL> SQL> 2
1 row created.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
|
```

Query menu:

```
Command Prompt - ssh t6jia  x  +  ▾
SQL> SQL> 2
1 row created.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue

M) View Manual
1) Drop Tables
2) Create Tables
3) Populate Tables
4) Query Tables
E) End/Exit
Choose:
4
1) List all members owing more than ten dollars
2) List all media in stock
3) List a count of all types of media in the database
4) List all librarians and members in the library
5) List all expired books starting from Jan 20
6) List all reviews for librarians better than average
E) Return to main menu
Choose:
|
```

List all members owing more than 10 dollars:

```
19
ucronin
ucornl00@gmail.com
38

Member id
-----
Username
-----
Member email
-----
Amount Owed
-----
16
ehuel
ehu0098@gmail.com
42

Member id
-----
Username
-----
Member email
-----
Amount Owed
-----
4
jwalsh
jwalsh@gmail.com
43

10 rows selected.

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
```

List all media in stock:

```
Type
-----
12 Iceberg melting
Lawrence Walker
Journal

14 Lisp is Best
Kevin Spencer
Id Title
Author
Type
Book

9 Studio C
Matt Meese
CD

11 Study of Slang
Id Title
Author
Type
Kevin Spencer
Journal

13 rows selected.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
```

List a count of all media in the library:

```
Command Prompt - ssh t6jai + x - o x
Journal

13 rows selected.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue

1) List all members owing more than ten dollars
2) List all media in stock
3) List a count of all types of media in the database
4) List all librarians and members in the library
5) List all expired books starting from Jan 20
6) List all reviews for librarians better than average
E) Return to main menu
Choose:
3
./queries/mediacount.sh: line 11: warning: here-document at line 7 delimited by end-of-file (wanted `EOF')
SQL*Plus: Release 12.1.0.2.0 Production on Tue Oct 24 18:34:39 2023
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>
MEDIA_TYPE          TOTALCOUNT
-----  -----
CD                  4
Journal             5
Book                11

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
```

List all members and librarians in the library:

```
Command Prompt - ssh t6jai + x - o x
PERSONNEL
-----
Ruby
rdubue@gmail.com
Member

Schinner
schice@gmail.com
Member

PERSONNELNAME
-----
CONTACTINFO
-----
PERSONNEL
-----
Susan
iceeyou@yahoo.ca
Member

William
Williamgreat2000@gmail.com

PERSONNELNAME
-----
CONTACTINFO
-----
PERSONNEL
-----
Member

25 rows selected.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
```

List all books expired starting from Jan 20.

```
Command Prompt - ssh t6jia x + v
1) List all members owing more than ten dollars
2) List all media in stock
3) List a count of all types of media in the database
4) List all librarians and members in the library
5) List all expired books starting from Jan 20
6) List all reviews for librarians better than average
E) Return to main menu
Choose: 5
./queries/expiredjan20.sh: line 9: warning: here-document at line 4 delimited by end-of-file (wanted 'EOF')
SQL*Plus: Release 12.1.0.2.0 Production on Tue Oct 24 18:35:24 2023
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>
HISTORYID PICKUPDAT EXPIREDAT      STATUS      MEMID      MEDIAID
-----  -----  -----  -----
28 21-JAN-23 29-JAN-23      1          7          4
16 20-JAN-23 28-JAN-23      1          18         3
18 19-JAN-23 27-JAN-23      0          10         2
7 18-JAN-23 26-JAN-23      1          8          18
14 17-JAN-23 25-JAN-23      1          3          9
13 16-JAN-23 24-JAN-23      1          14         8
3 15-JAN-23 23-JAN-23      0          4          7
19 14-JAN-23 22-JAN-23      1          9          11
8 10-JAN-23 22-JAN-23      1          5          6
17 13-JAN-23 21-JAN-23      1          20         10
10 rows selected.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
|
```

List all librarians with higher than average reviews:

```
Command Prompt - ssh t6jia x + v
They are very quiet but otherwise good
5      5
My dog wasn't allowed inside so they had to stay outside
3      4
Siuuu that's nice library for student

LIBID      STARS
-----  -----
COMMENTS
-----  -----
2          4
I wish they followed the marking rubric
2          3
Best librarian ever but they only come on tuesdays
5          3
They're ok

LIBID      STARS
-----  -----
COMMENTS
-----  -----
4          3
Pretty good overall. Library seems understaffed.
2          3
Other librarians are better
8 rows selected.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue
|
```

Returning to main menu:

```
Command Prompt - ssh t6jiang | + x - o x
5      3
They're ok

LIBID      STARS
-----  
COMMENTS
-----  
4      3
Pretty good overall. Library seems understaffed.

2      3
Other librarians are better

8 rows selected.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue

1) List all members owing more than ten dollars
2) List all media in stock
3) List a count of all types of media in the database
4) List all librarians and members in the library
5) List all expired books starting from Jan 20
6) List all reviews for librarians better than average
E) Return to main menu
Choose:
E
Press [ENTER] to continue

M) View Manual
1) Drop Tables
2) Create Tables
3) Populate Tables
4) Query Tables
E) End/Exit
Choose:
|
```

Quitting the menu program:

```
Command Prompt - ssh t6jiang | + x - o x
They're ok

LIBID      STARS
-----  
COMMENTS
-----  
4      3
Pretty good overall. Library seems understaffed.

2      3
Other librarians are better

8 rows selected.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press [ENTER] to continue

1) List all members owing more than ten dollars
2) List all media in stock
3) List a count of all types of media in the database
4) List all librarians and members in the library
5) List all expired books starting from Jan 20
6) List all reviews for librarians better than average
E) Return to main menu
Choose:
E
Press [ENTER] to continue

M) View Manual
1) Drop Tables
2) Create Tables
3) Populate Tables
4) Query Tables
E) End/Exit
Choose:
E
t6jiang@metis:~/cps510$ |
```

ASSIGNMENT 6

Media

Attribute	Attribute type
author	varchar
title	varchar
publish date	Date
<u>ID</u>	Int (primary key)
availability	Int

$\text{id} \rightarrow \{\text{author, title, publish date, availability}\}$

$\text{id} \rightarrow \{\text{author}\}$ $\text{id} \rightarrow \{\text{title}\}$
 $\text{id} \rightarrow \{\text{publish date}\}$ $\text{id} \rightarrow \{\text{availability}\}$

=====

$\text{Id} \rightarrow \{\text{title, availability}\}$
 $\text{title} \rightarrow \{\text{publish date, author}\}$

Penalties

Attribute	Attribute type
penalty id	Int (Primary key)
member id	Int (Foreign key)
transaction id	Int (Foreign key)

penalty id → { member id, transaction id }

penalty id → { member id }

penalty id → {transaction id }

Members

Attribute	Attribute type
MemID	Int (Primary key)
Username	varchar
Pass	varchar
Email	varchar
Name	varchar
AmountOwed	Double

memID → {username, pass, email, name, amount owed}

memID → {username}

memID → {pass}

memID → {name}

memID → {amount owed}

Transaction Details

Attribute	Attribute type
<u>history id</u>	Int (primary key)
media id	Int (Foreign key)
expiry date	Date
status	Int
pickup date	Date
member id	Int (Foreign key)

history id → { media id, expiry date, status, pickup date, member id }

history id → { media id }

history id → { expiry date }

history id → { status }

history id → { pickup date }

history id → { member id }

Librarian

Attribute	Attribute type
<u>LibID</u>	Int (Primary key)
AdminKey	Int (Primary key)
Name	varchar

{LibID, AdminKey} → Name

Feedback

Attribute	Attribute type
<u>feedback id</u>	Int (Primary key)
member id	Int (Foreign key)
librarian id	Int (Foreign key)
comment	String
Stars	Int

Feedback ID → {memberId, librarianId, comment , Stars}

Feedback ID → {memberID}

Feedback ID → {librarianID}

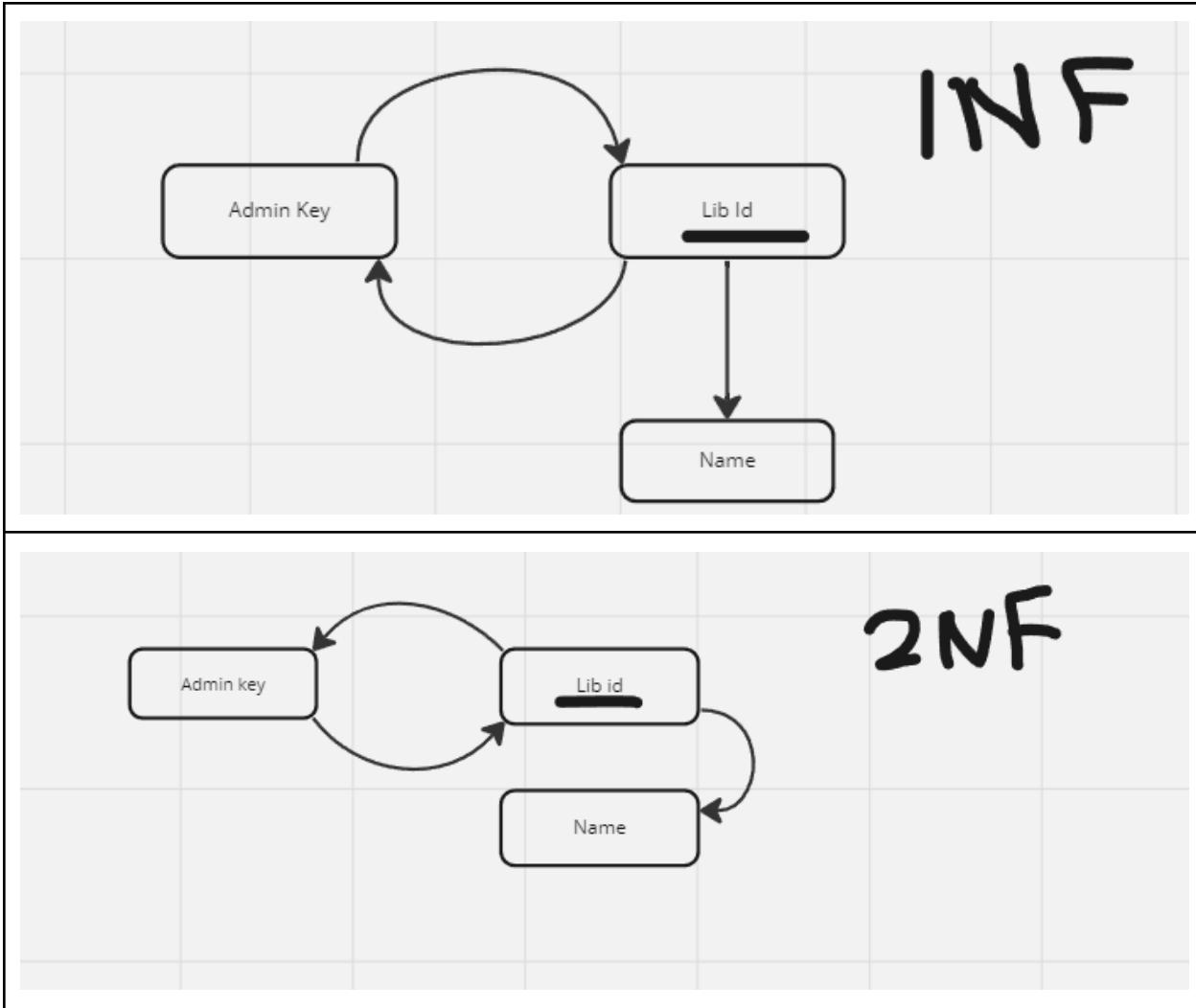
Feedback ID → {comment}

Feedback ID → {Stars}

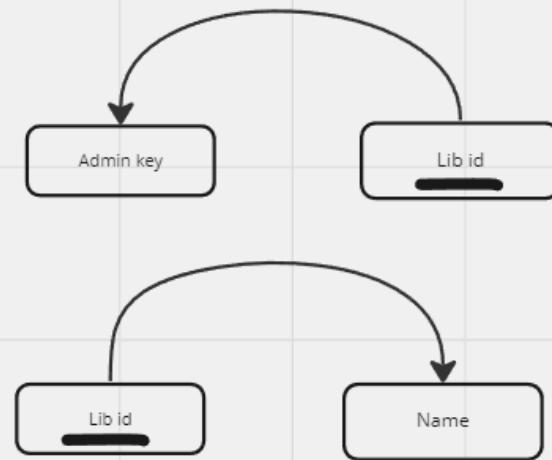
From observing all of the tables we have currently, we have been able to determine that the attributes are atomic, which means that the tables are in 1NF. By further observation, We also observe many of these attributes are functionally dependent on their key values and that some do not have any transitive dependencies on one another. Due to this, we can conclude that the tables some of the tables may already be in 3NF, while the others may be in a combination of 1 and 2

ASSIGNMENT 7:

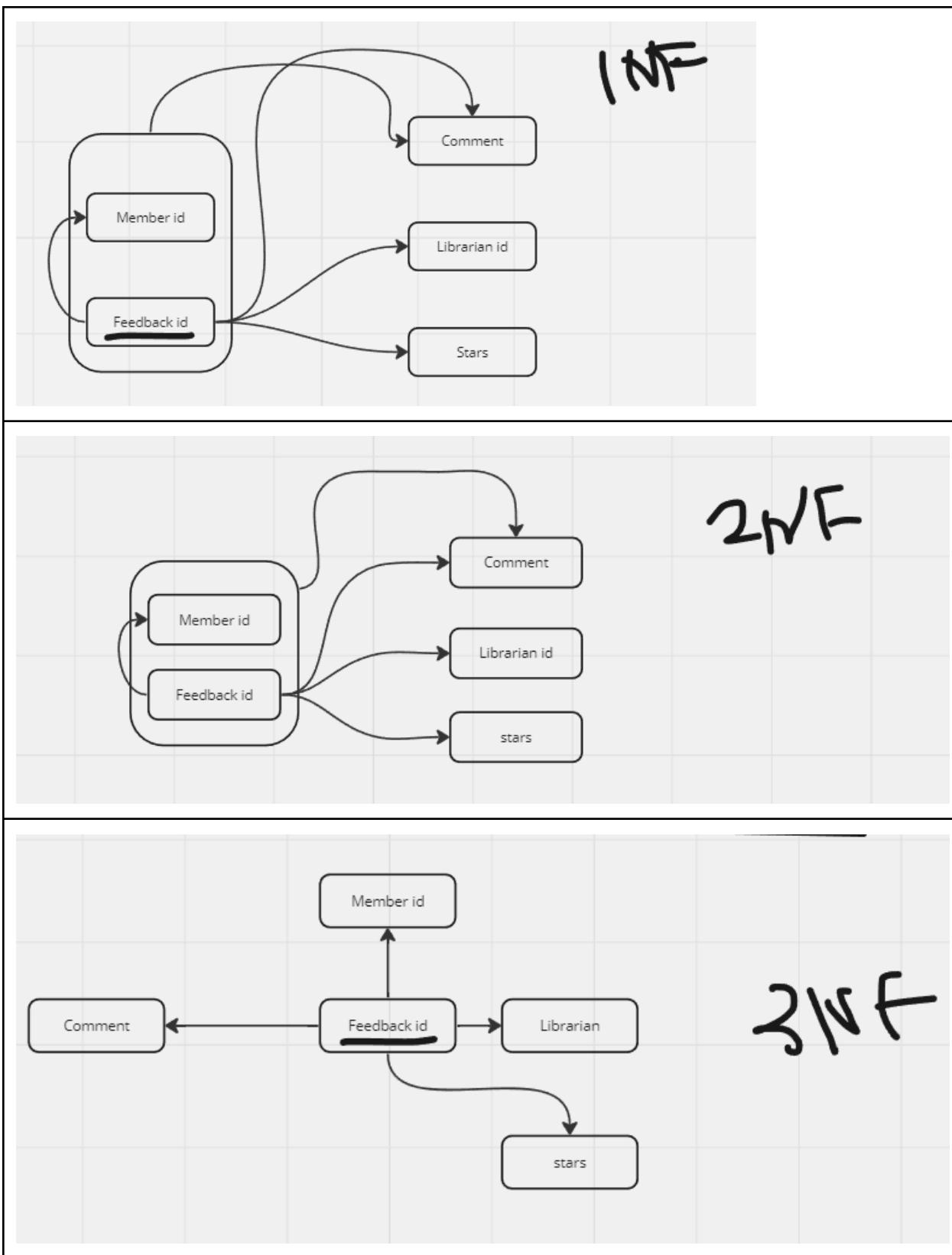
Librarian



3NF

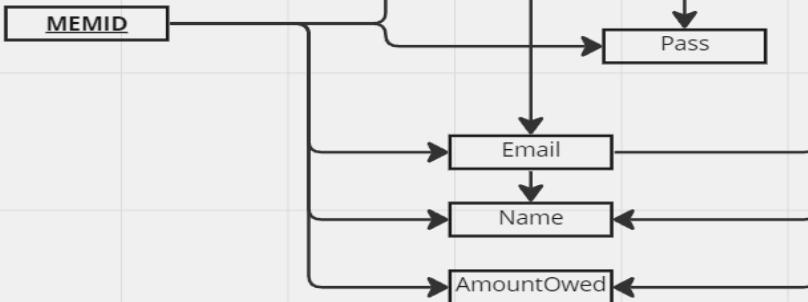


Feedback

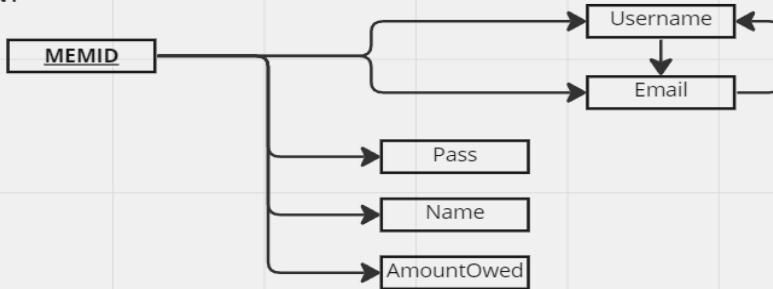


Members

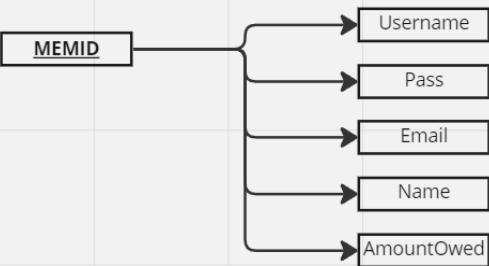
1NF



2NF

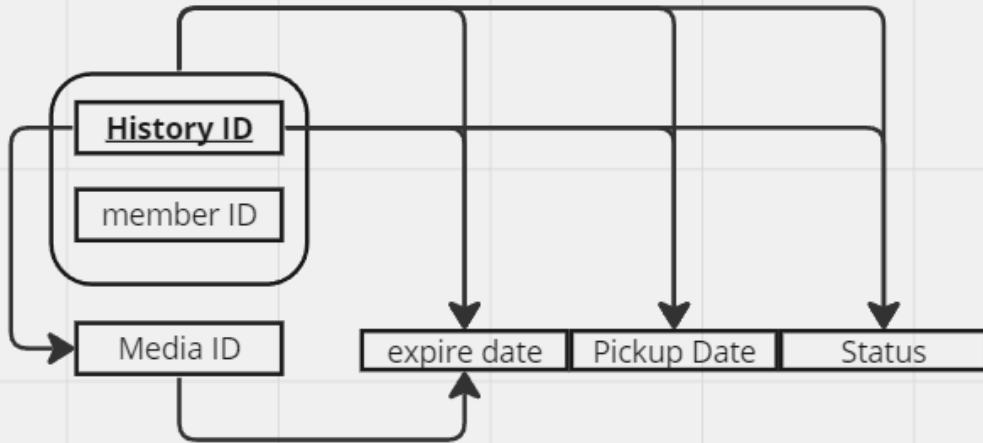


3NF

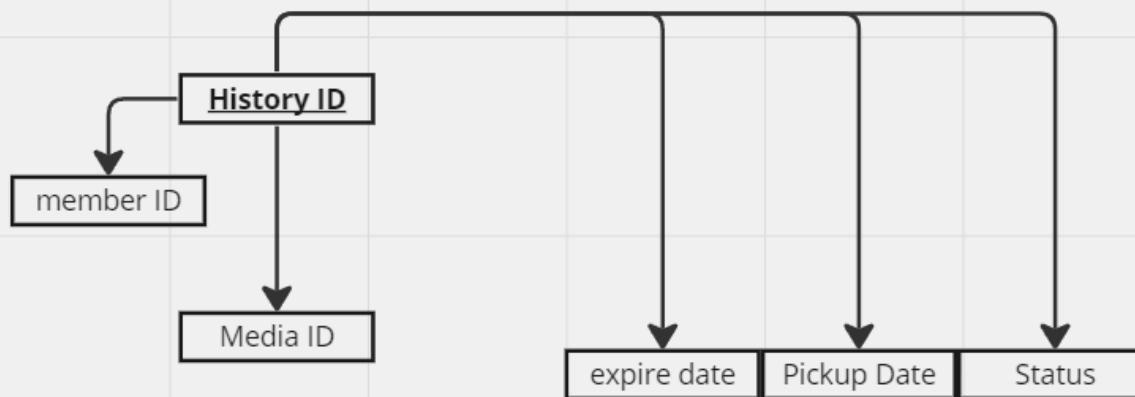


Transaction Details

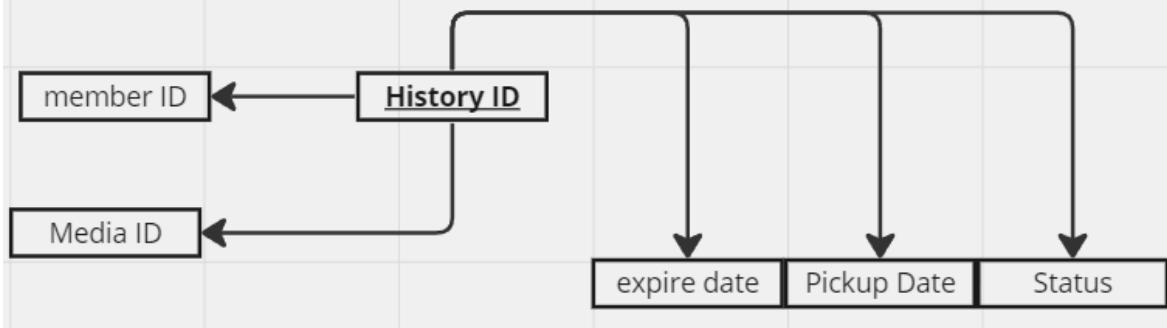
1NF



2NF

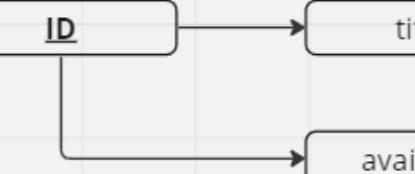


3NF



Media

1NF



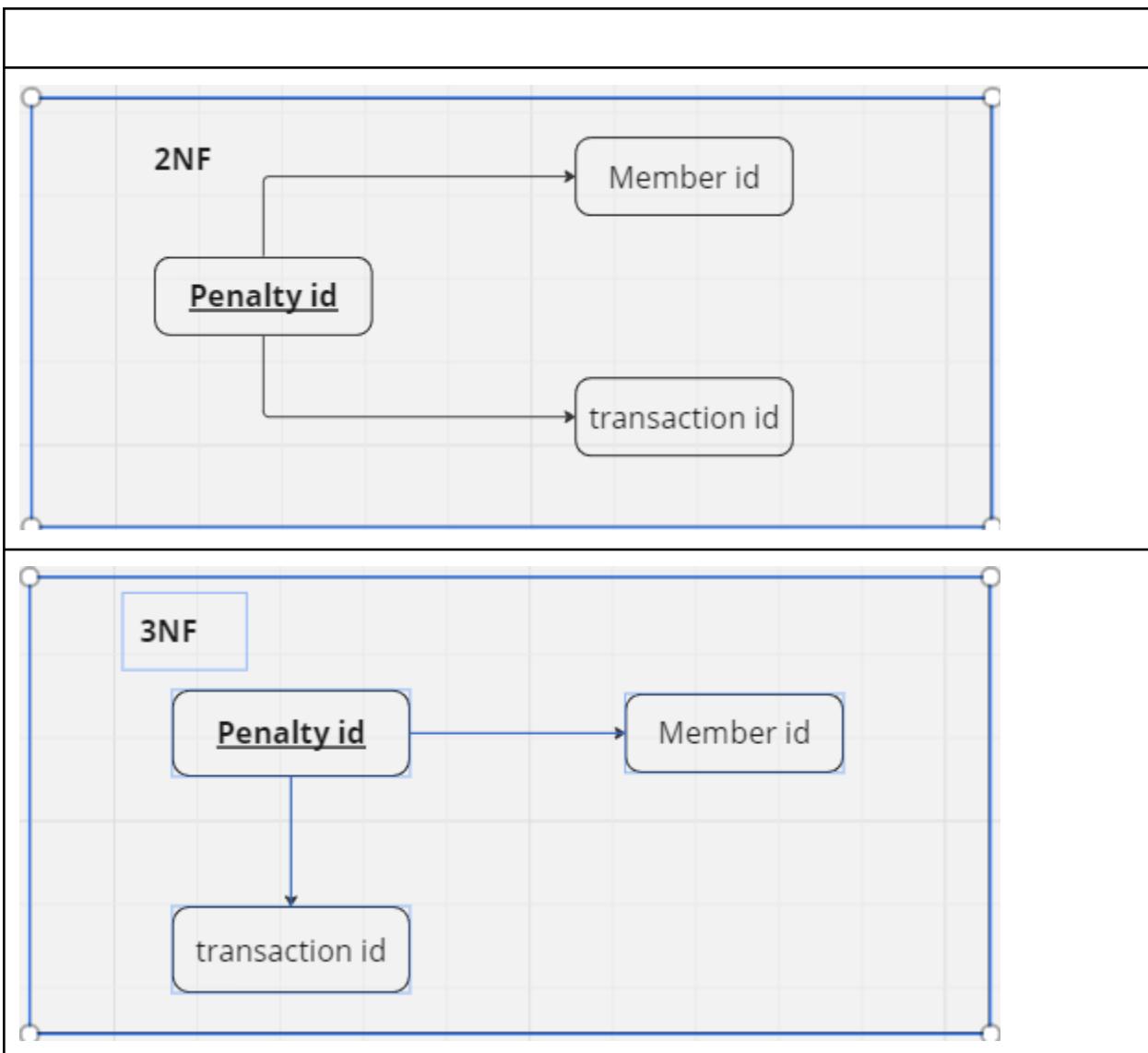
2NF



3NF



Penalties

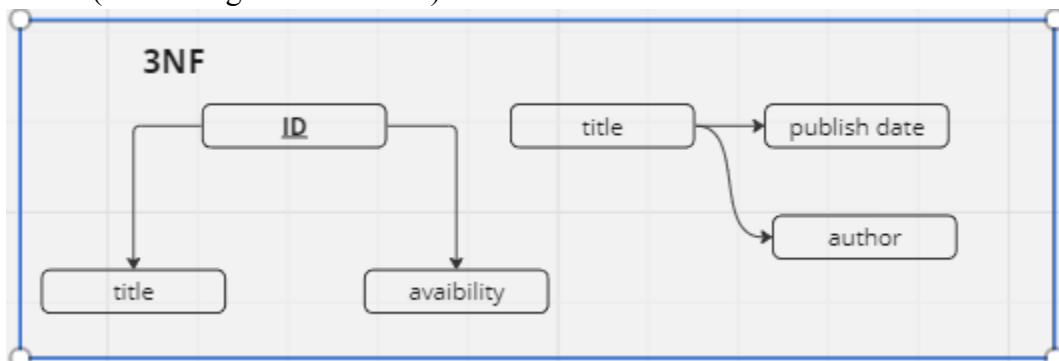


ASSIGNMENT 8:

DISCUSSION:

The tables above present the BCNF form. In the context of Boyce-Codd Normal Form (BCNF) in database normalization, they must be in their 3NF form, and they ensure that for every non-trivial functional dependency in a relation, the set of attributes on the left side of the dependency constitutes a superkey. In simpler terms, if a non-key determines a key or a key determines a key, it is necessary to break down the table into smaller tables to meet the BCNF criteria, thereby avoiding anomalies and maintaining data integrity. Our tables are in BCNF form based on the criteria.

Media (BCNF Algorithm is Used)



Functional dependencies

$\text{id} \rightarrow \{\text{author}, \text{title}, \text{publish date}, \text{availability}\}$

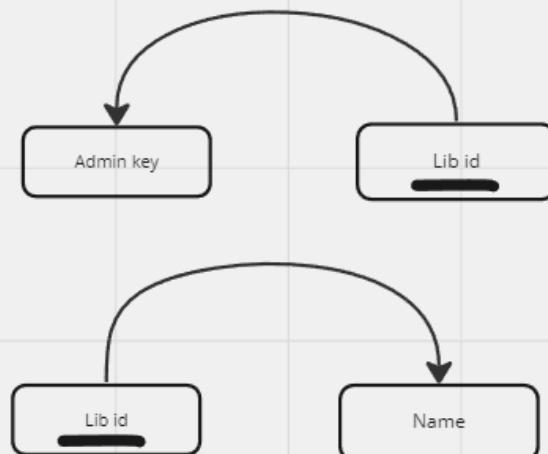
$\text{id} \rightarrow \{\text{author}\}$ $\text{id} \rightarrow \{\text{title}\}$
 $\text{id} \rightarrow \{\text{publish date}\}$ $\text{id} \rightarrow \{\text{availability}\}$

====

$\text{id} \rightarrow \{\text{title}, \text{availability}\}$
 $\text{title} \rightarrow \{\text{publish date}, \text{author}\}$

Librarian

3NF

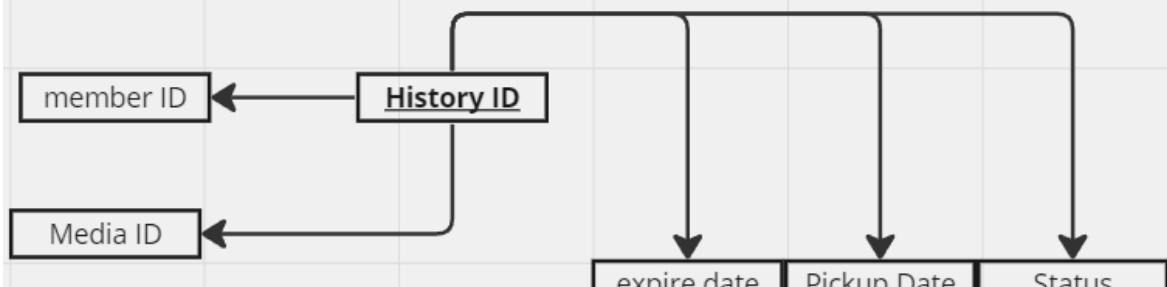


Functional dependencies

$\text{LibId} \rightarrow \text{Admin key}$
 $\text{LibId} \rightarrow \text{Name}$

Transaction Details

3NF

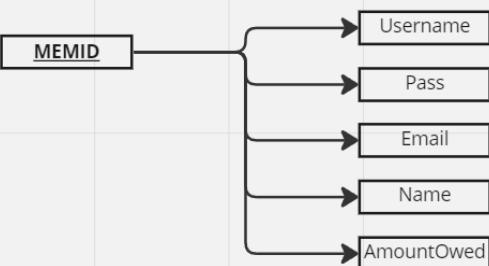


Functional Dependencies:

$\text{HistoryId} \rightarrow \text{memberId}$
 $\text{HistoryId} \rightarrow \text{MediaId}$
 $\text{HistoryId} \rightarrow \text{expireDate}$
 $\text{HistoryId} \rightarrow \text{pickupDate}$
 $\text{HistoryId} \rightarrow \text{status}$

Members

3NF



Functional Dependencies

$\text{MEMID} \rightarrow \text{Username}$

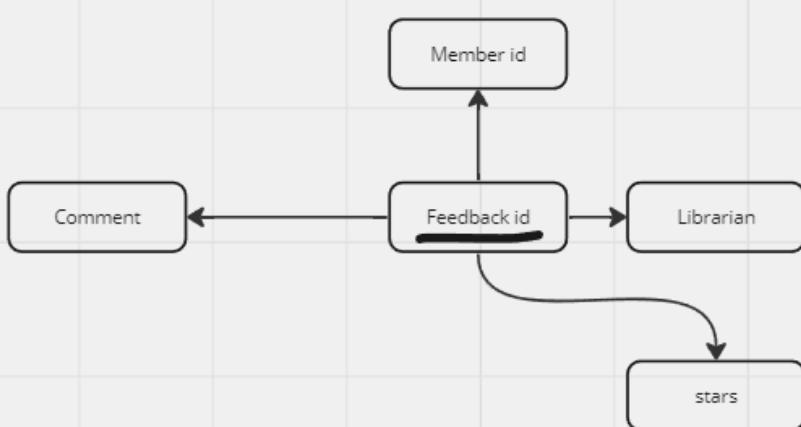
$\text{MEMID} \rightarrow \text{Pass}$

$\text{MEMID} \rightarrow \text{Email}$

$\text{MEMID} \rightarrow \text{Name}$

$\text{MEMID} \rightarrow \text{AmountOwed}$

Feedback



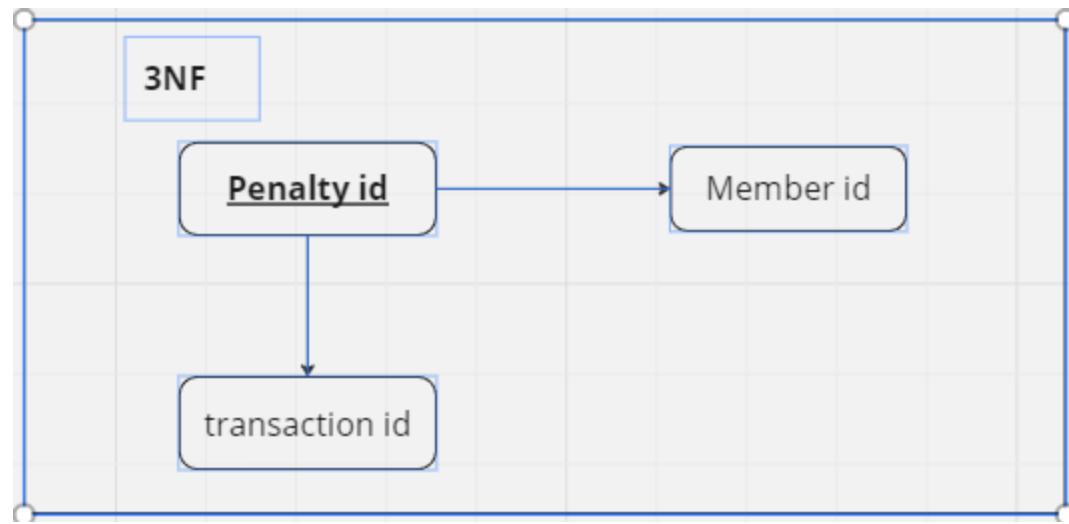
$\text{FeedbackId} \rightarrow \text{MemberId}$

$\text{FeedbackId} \rightarrow \text{Librarian}$

$\text{FeedbackId} \rightarrow \text{Stars}$

$\text{FeedbackId} \rightarrow \text{Comment}$

Penalty



Function Dependencies:

Penalty ID -> Member ID

Penalty ID -> Transaction ID

Berstein's Algorithm

Step 1:

Determine attributes, FDs about the real world.
From our application, we determined that we require the following attributes for Table Media.

Type refers to the media being digital, paperback, etc
Category refers to Novel, manga, journal, video, etc

TAKE MEDIA AS A MODEL EXAMPLE:

ID, Type, Title, Author, Publisher, Category, Language, Reviews, OnSale

ID → {Title, Publisher, Author, Reviews}
Title → {Language, Category, Reviews}
Publisher → {onSale, Type}
Author → {Language}

Step 2a:

2a : Remove the redundant dependencies. From our table, we determined that we could get the following functional dependencies after removal of the redundancies:

Attribute closures of the attributes

ID⁺ = {ID, Title, Language, Publisher, onSale, Type, Author, Reviews, Category}

Title⁺ = {Language, Category}

Publisher* = {Publisher, onSale, Type}

Functional Dependencies after removing redundancies:

ID → {Title, Publisher, Author, Reviews}
Title → {Language, Category}
Publisher → {OnSale, Type}

Step 2b:

2b : Find and remove partial dependencies. From our table, these following partial dependencies will be removed to complete this step.

This process will make this table into a 2NF.

1NF:

Since all of the attributes in our table are already atomic, the table is already in 1NF.

2NF:

After looking at the table, we have determined that decomposition is needed to bring the relation to 2NF.

ID, Type, Title, Author, Publisher, Category, Language, Reviews, OnSale

Original table:

R(ID, Title, Language, Publisher, onSale, Type, Author, Reviews, Category)

Decomposed tables:

R1(ID, Title, Publisher, Author, Reviews)

R2(Publisher, Onsale, Type)

R3(Title, Language, Category)

Step 3:

Now that your diagram is in 2nf, you can:

find the keys of your diagram.

This requires you to find closures of all your attributes and combine them with all other attributes to determine if they're keys.

This is computationally expensive.

Functional dependencies:

ID → {Title, Publisher, Author, Reviews}

Title → {Language, Category}

Publisher → {OnSale, Type}

Decomposed Tables:

R1(ID, Title, Publisher, Author, Reviews)

R2(Publisher, Onsale, Type)

R3(Title, Language, Category)

Attribute closures:

```
ID+ = {ID, Title, Publisher, Author, Reviews, Language, Category, Onsale, Type}
Publisher+ = {Publisher, OnSale, Type}
Title+ = {Title, Language, Category}
Type+ = {Type}
Author+ = {Author}
Reviews+ = {Reviews}
OnSale+ = {OnSale}
Category = {Category}
```

From the attribute closure analysis, We have determined that the key should be ID, Publisher, Title, because combining all the tables, it will enclose all the attributes that are part of media.

Step 4:

From step 3, we have determined that the relational schema from above should satisfy the requirements for the keys.

R1(ID, Title, Publisher, Author, Reviews)

R2(Publisher, Onsale, Type)

R3(Title, Language, Category)

Key Relations:

Reviewing the three decomposed tables, all of the tables have key relations, therefore step 4 is complete and the table is in 3NF that is lossless and dependency preserving.

Counter Example of a BCNF:

An Example of a table that is not in BCNF is an example such as:

R(MediaId, ISBN, Title, Category, PublishDate)

FD:

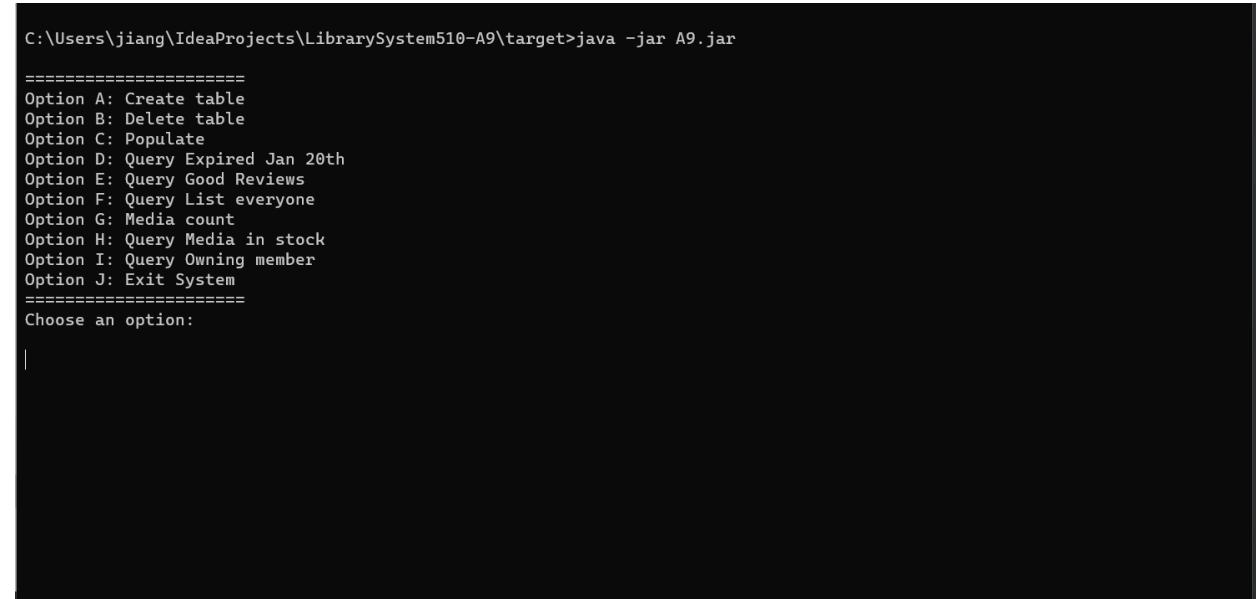
MediaID → {ISBN}

MediaID -> {Title, Category, PublishDate}

This table is not in BCNF because there is a key attribute (MediaId), which determines another key attribute (ISBN). By definition of BCNF, all keys should be on the left side of the title:relation, however in this case, ISBN is not.

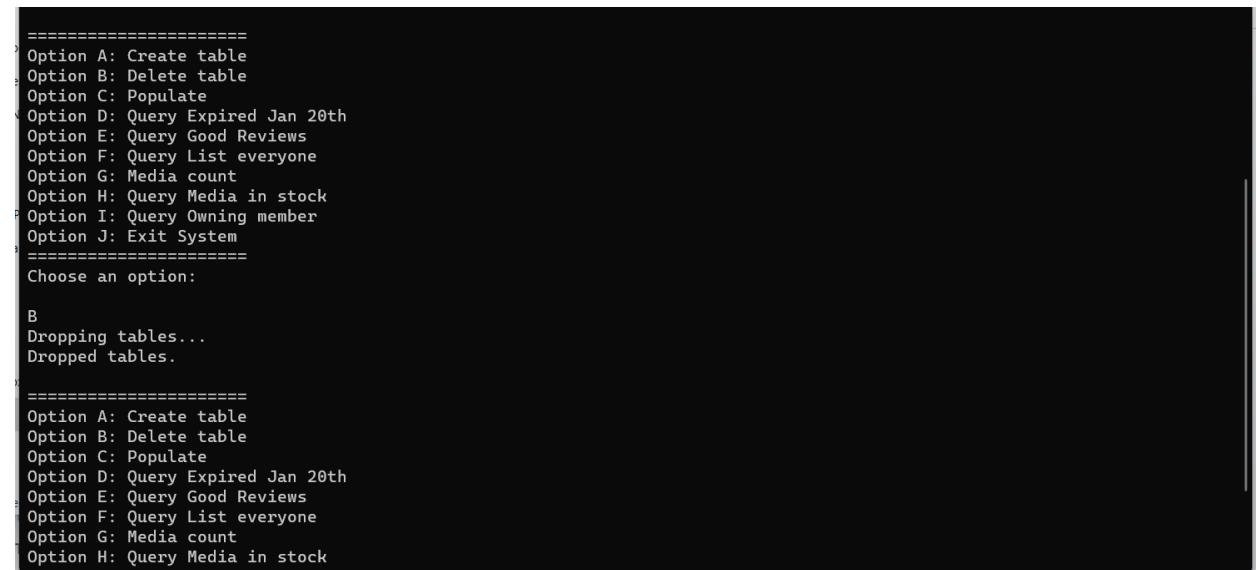
ASSIGNMENT 9:

Screenshots of the system:



```
C:\Users\jiang\IdeaProjects\LibrarySystem510-A9\target>java -jar A9.jar
=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:
|
```

Dropping tables:



```
=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:
B
Dropping tables...
Dropped tables.

=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
```

Creating tables:

```
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:
```

```
A
Creating tables...
Created tables.
```

```
=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:
```

Populating Tables:

```
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:
```

```
C
Populating tables...
Populated table Media...
Populated table Members...
Populated table Librarian...
Populated table Transaction Details...
Populated table Feedback...
Populated table Penalties...
Populated all tables.
```

```
=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
```

Query 1 : Query “Expired Jan 20th”

```
=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:

D
History id: 20 | Media Id: 4 | status: 1 | Member ID: 7 | Pick up date: 2023-01-21|Expired Day: 2023-01-29
History id: 16 | Media Id: 3 | status: 1 | Member ID: 18 | Pick up date: 2023-01-20|Expired Day: 2023-01-28
History id: 18 | Media Id: 2 | status: 0 | Member ID: 10 | Pick up date: 2023-01-19|Expired Day: 2023-01-27
History id: 7 | Media Id: 18 | status: 1 | Member ID: 8 | Pick up date: 2023-01-18|Expired Day: 2023-01-26
History id: 14 | Media Id: 9 | status: 1 | Member ID: 3 | Pick up date: 2023-01-17|Expired Day: 2023-01-25
History id: 13 | Media Id: 8 | status: 1 | Member ID: 14 | Pick up date: 2023-01-16|Expired Day: 2023-01-24
History id: 3 | Media Id: 7 | status: 0 | Member ID: 4 | Pick up date: 2023-01-15|Expired Day: 2023-01-23
History id: 8 | Media Id: 6 | status: 1 | Member ID: 5 | Pick up date: 2023-01-10|Expired Day: 2023-01-22
History id: 19 | Media Id: 11 | status: 1 | Member ID: 9 | Pick up date: 2023-01-14|Expired Day: 2023-01-22
History id: 17 | Media Id: 10 | status: 1 | Member ID: 20 | Pick up date: 2023-01-13|Expired Day: 2023-01-21
```

Query 2 : Query “Good Reviews”

```
=====
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:

E
5 5 My dog wasn't allowed inside so they had to stay outside
1 5 They are very quiet but otherwise good
3 4 Siuuu that's nice library for student
2 4 I wish they followed the marking rubric
5 3 They're ok
4 3 Pretty good overall. Library seems understaffed.
2 3 Other librarians are better
2 3 Best librarian ever but they only come on tuesdays
=====
Option A: Create table
```

Query 3 : Query “Query List Everyone”

```
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:

F
Allan allanturing@gmail.com Member
Allan ucorn100@gmail.com Member
Andy andych@gmail.com Member
Bart imbaroque@gmail.com Member
Brekke brekbb@gmail.com Member
Ellen ehu0098@gmail.com Member
Emerson Emerson.watsica@gmail.com Member
Ho Kim Lee null Librarian
Jacob ljacon@gmail.com Member
Jenkins jenkins.fritz@gmail.com Member
John wwiend@gmail.com Member
Josh jwalsh@gmail.com Member
Kevin Jones null Librarian
Kimberly Delamere null Librarian
Minh Tran null Librarian
Morgan Freeman null Librarian
Noel noel21@gmail.com Member
Orie orieng@gmail.com Member
Pam gatdaip@hotmail.com Member
Retha Rethh1029@gmail.com Member
Rose rosalia5656@gmail.com Member
Ruby rdubue@gmail.com Member
Schinner schice@gmail.com Member
```

Query 4 : Query “Media Count”

```
Retha Rethh1029@gmail.com Member
Rose rosalia5656@gmail.com Member
Ruby rdubue@gmail.com Member
Schinner schice@gmail.com Member
Susan iceeyou@yahoo.ca Member
William Williamgreat2000@gmail.com Member

=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:

G
Type: CD Amount:4
Type: Book Amount:11
Type: Journal Amount:5
```

Query 5 : Query Media InStock

```
=====
Choose an option:

H
Id: 2 Title: How to grow blueberries Author: Charles Hadwell Type: Book
Id: 3 Title: C programming Author: Alex Ufkes Type: Book
Id: 5 Title: Air currents Author: Marty small Type: Journal
Id: 6 Title: Anthropology Author: Venice Hare Type: Journal
Id: 8 Title: Fall tunes Author: Matt Meese Type: CD
Id: 9 Title: Studio C Author: Matt Meese Type: CD
Id: 11 Title: Study of Slang Author: Kevin Spencer Type: Journal
Id: 12 Title: Iceberg melting Author: Lawrence Walker Type: Journal
Id: 14 Title: Lisp is Best Author: Kevin Spencer Type: Book
Id: 15 Title: Functional Programming with Lisp Author: Mark Gotet Type: Book
Id: 16 Title: C Language History Author: Kyle Wang Type: Book
Id: 19 Title: Christmas tales Author: Mark Rober Type: CD
Id: 20 Title: Exploding cucumbers Author: Sarah Crown Type: Book

=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
```

Query 6 : Query Owning Member

```
Option I: Query Owning member
Option J: Exit System
=====
Choose an option:

I
Member id: 11 username: icie25 email:iceeyou@yahoo.ca amount owed: 13.0
Member id: 2 username: schinner.gladycce email:schice@gmail.com amount owed: 15.0
Member id: 20 username: retha.wyman email:Rethh1029@gmail.com amount owed: 20.0
Member id: 8 username: brekke.robb email:brekbb@gmail.com amount owed: 21.0
Member id: 6 username: andy.gerlach email:andych@gmail.com amount owed: 22.0
Member id: 14 username: emerson.watsica email:Emerson.watsica@gmail.com amount owed: 22.0
Member id: 9 username: bbartoletti email:imbaroque@gmail.com amount owed: 23.0
Member id: 19 username: ucronin email:ucorn100@gmail.com amount owed: 38.0
Member id: 16 username: ehuemail:ehu0098@gmail.com amount owed: 42.0
Member id: 4 username: jwalsh email:jwalsh@gmail.com amount owed: 43.0

=====
Option A: Create table
Option B: Delete table
Option C: Populate
Option D: Query Expired Jan 20th
Option E: Query Good Reviews
Option F: Query List everyone
Option G: Media count
Option H: Query Media in stock
Option I: Query Owning member
Option J: Exit System
=====
```

Exiting the System:

```
Choose an option:  
E  
5 5 My dog wasn't allowed inside so they had to stay outside  
1 5 They are very quiet but otherwise good  
3 4 Siuuu that's nice library for student  
2 4 I wish they followed the marking rubric  
5 3 They're ok  
4 3 Pretty good overall. Library seems understaffed.  
2 3 Other librarians are better  
2 3 Best librarian ever but they only come on tuesdays  
=====  
Option A: Create table  
Option B: Delete table  
Option C: Populate  
Option D: Query Expired Jan 20th  
Option E: Query Good Reviews  
Option F: Query List everyone  
Option G: Media count  
Option H: Query Media in stock  
Option I: Query Owning member  
Option J: Exit System  
=====  
Choose an option:  
J  
Press any key to continue . . . |
```

ASSIGNMENT 10:

SQL to Relational Algebra:

```
"SELECT mediaid AS \"Id\"," + " mediatitle AS \"Title\"," + " author AS \"Author\"," + "  
mediatype AS \"Type\\"" + " FROM media WHERE media.instock = 1 ORDER BY  
mediatitle ASC
```

Relational Algebra:

```
 $\pi \leftarrow (\sigma_{\text{expiredate} > \text{TO\_DATE('2023-01-20', 'RRRR-MM-DD')}) \text{ (transactionDetails)}}$ 
```

```
String query = "SELECT libid, stars, comments FROM feedback WHERE stars >= (SELECT  
AVG(stars) FROM feedback) ORDER BY stars DESC" ;
```

Relational Algebra:

```
 $\pi \leftarrow (\text{libid}, \text{stars}, \text{comments})$ 
```

```
 $\sigma \leftarrow (\text{stars} \geq \rho_{\text{feedback}}(\text{AVG(stars)}) \text{ AVG_stars}) \text{ (feedback) } \bowtie \text{ true})$ 
```

```
SELECT DISTINCT Name AS PersonnelName, Email AS ContactInfo, 'Member' AS
PersonnelType FROM Members" + " UNION " + "SELECT DISTINCT Name AS
PersonnelName, NULL AS ContactInfo, 'Librarian' AS PersonnelType FROM Librarian
```

Relational Algebra:

$$\pi \leftarrow (\text{Name, Email, 'Member' as PersonnelType}) \text{ (Members)} \cup \pi \leftarrow (\text{Name, null as ContactInfo, 'Librarian' as PersonnelType}) \text{ (Librarian)}$$

```
SELECT MEDIATYPE, COUNT(*) AS TotalCount FROM MEDIA GROUP BY
MEDIATYPE
```

Relational Algebra:

$$\pi \leftarrow (\text{MEDIATYPE, COUNT(*) as TotalCount}) \text{ (MEDIA)} \div \text{MEDIATYPE}$$

```
SELECT mediaid AS \"Id\", " + " mediatitle AS \"Title\", " + " author AS \"Author\", " + "
mediatype AS \"Type\"" + " FROM media WHERE media.instock = 1 ORDER BY
mediatitle ASC
```

Relational Algebra:

$$\begin{aligned} \pi &\leftarrow (\text{mediaid as Id, mediatitle as Title, author as Author, medatype as Type}) \\ (\sigma &\leftarrow (\text{instock} = 1)) \text{ (media)} \end{aligned}$$

```
"SELECT DISTINCT members.memid AS \"Member id\", " +
"members.username AS \"Username\", " +
"members.email AS \"Member email\", "
```

```
"members.amountowed AS \"Amount Owed\" " +
"FROM members INNER JOIN penalties ON members.memid =
penalties.memid" +
"WHERE members.amountowed > 10 ORDER BY amountowed ASC";
```

$\pi \leftarrow (\text{memid as Member_id, username as Username, email as Member_email, amountowed as Amount_Owed})$
 $(\sigma \leftarrow (\text{amountowed} > 10))$
 $(\text{Members} \bowtie (\text{memid} = \text{memid}) \text{ penalties}))$

```
Select * FROM Media ORDER BY INSTOCK ASC;
```

Sort $\leftarrow \sigma(\text{Media})$
Result $\leftarrow \tau_{\text{INSTOCK ASC}}(\text{Sort})$

```
Select * FROM Media ORDER BY MEDIATITLE DESC;
```

Sort $\leftarrow \sigma(\text{Media})$
Result $\leftarrow \tau_{\text{MEDIATITLE DESC}}(\text{Sort})$

```
Select * FROM MEDIA ORDER BY MEDIATYPE, MEDIATITLE;
```

Sort $\leftarrow \sigma(\text{Media})$
Result $\leftarrow \tau_{\text{MEDIATYPE, MEDIATITLE DESC}}(\text{Sort})$

```
Select DISTINCT Author FROM MEDIA;
```

///The tuples returned from RA operations are ALWAYS distinct, so No Distinct keyword needed

```
Proj ←  $\pi_{\text{Author}}(\text{MEDIA})$ 
Result ←  $\sigma(\text{Proj})$ 
```

```
SELECT DISTINCT AUTHOR FROM MEDIA ORDER BY AUTHOR DESC;
```

```
Proj ←  $\pi_{\text{Author}}(\text{MEDIA})$ 
Sort ←  $\sigma(\text{Proj})$ 
Result ←  $\tau_{\text{MEDIATYPE}, \text{MEDIATITLE DESC}}(\text{Proj})$ 
```

```
SELECT MEDIATYPE as "Media Type", COUNT(*) AS "Total Count" FROM MEDIA
GROUP BY MEDIATYPE;
```

```
Temp ←  $\Pi_{\text{MediaType}}(\text{COUNT}(*))$ (Media)
Temp2 ←  $\rho_{(\text{Media Type}, \text{Total Count})}$  Temp
Result ←  $\gamma_{\text{MediaType}, \text{Total Count}}(\text{Temp2})$ 
```

```
SELECT mediatitle,instock FROM MEDIA Order By mediatitle;
```

```
Proj ←  $\Pi_{\text{mediatitle,instock}}(\text{Media})$ 
Result ←  $\tau_{\text{mediatitle}}(\text{Proj})$ 
```

```
SELECT amountowed, memid FROM Members WHERE amountowed >
(SELECT AVG(amountowed) FROM Members) ORDER BY amountowed DESC;
```

// γ <, grouping / aggregation
// δ duplicate elimination
// τ sorting

```
Mems ←  $\pi_{\text{MEMID, AMOUNTOWED}}(\text{Members})$ 
Ave ← AVG(AmountOwed)
Filter ←  $\sigma_{\text{AmountOwed} > \text{Ave}}(\text{Mems})$ 
Result ←  $\tau_{\text{amountowed DESC}}(\text{Filter})$ 
```

```
SELECT username, email, name FROM members WHERE
```

```
email LIKE '%gmail.com';
```

```
Temp =  $\pi_{\text{USERNAME}, \text{EMAIL}, \text{NAME}}$ 
```

```
Result  $\leftarrow \sigma_{\text{email Like } (\% \text{gmail.com})}(\text{Temp})$ 
```

```
SELECT distinct name, email, pass FROM members WHERE email LIKE 'wwiend%'  
AND name LIKE 'John%' ORDER BY name DESC;
```

```
Temp  $\leftarrow \pi_{\text{NAME}, \text{EMAIL}, \text{PASS}}$ 
```

```
Filter  $\leftarrow \sigma_{(\text{email Like } (\% \text{wwiend})) \cap (\text{Name Like } (\% \text{John }))}(\text{Temp})$ 
```

```
Result  $\leftarrow \tau_{\text{name DESC}}(\text{Filter})$ 
```

```
SELECT DISTINCT members.memid, members.username FROM members  
INNER JOIN penalties ON penalties.memid = members.memid ORDER BY  
members.username ASC;
```

```
Cross  $\leftarrow \text{Penalties} \times \text{Members}$ 
```

```
Filter  $\leftarrow \delta_{\text{Penalties.memId} = \text{Members.memId}}(\text{Cross})$ 
```

```
Ordered  $\leftarrow \tau_{\text{Members.Username ASC}}(\text{Filter})$ 
```

```
SELECT DISTINCT members.memid AS "Member Id", members.username AS "Username",  
members.email AS "Member email"  
FROM members  
INNER JOIN transactiondetails ON members.memid = transactiondetails.memid  
ORDER BY members.username DESC;
```

```
Crossed  $\leftarrow \text{TransactionDetails} \times \text{Members}$ 
```

```
Filtered  $\leftarrow \delta_{\text{Members.MemId} = \text{TransactionDetails.memId}}(\text{Crossed})$ 
```

```
Renamed  $\leftarrow p_{(\text{Member Id}, \text{Username}, \text{Member Email})}(\pi_{\text{Members.MemId}, \text{Members.Username}, \text{Members.email}}(\text{Filtered}))$ 
```

```
Ordered  $\leftarrow \tau_{\text{Members.Username DESC}}(\text{Filtered})$ 
```

```

SELECT DISTINCT members.memid AS "Member id",
    members.username AS "Username",
    members.email AS "Member email",
    members.amountowed AS "Amount Owed"
FROM members INNER JOIN penalties ON members.memid = penalties.memid
WHERE members.amountowed > 0 ORDER BY amountowed ASC;

```

$\text{Join} \leftarrow \text{Members} \bowtie_{\text{Members.memId} = \text{Penalties.MemId}} (\text{Penalties})$
 $\text{Distinct} \leftarrow P_{(\text{Member Id}, \text{Username}, \text{Member Email}, \text{Amount Owed})} \pi_{\text{Members.memId}, \text{members.username}, \text{members.email}, \text{members.amountowed}} (\text{Join})$
 $\tau_{\text{Members.amountOwed ASC}} (\delta_{\text{Amount owed} > 0} (\text{Distinct}))$

```

SELECT members.memid AS "Member Id", members.username AS "Username",
members.email AS "Email" FROM members where
members.username LIKE 'a%' ORDER BY members.username;

```

$\delta_{\text{members.username LIKE 'a%'}} (\text{Members})$
 $\text{filtered} \leftarrow \pi_{\text{members.memId}, \text{members.username}, \text{members.email}} (\text{Members})$
 $\tau_{\text{members.username}} (p_{(\text{Member Id}, \text{Username}, \text{Email})} (\text{filtered}))$

```
select * from Penalties Where MemID = 5;
```

$\delta_{\text{MemId} = 5} (\text{Penalties})$

```

select Members.*, Penalties.*
From Members
Join Penalties ON Members.MemID = Penalties.MemID;

 $\sigma(\text{Members} \bowtie_{\text{members.memId} = \text{Penalties.memId}} (\text{Penalties}))$ 

```

```
SELECT DISTINCT HistoryID FROM Penalties;
```

```
 $\delta_{HistoryId}(Penalties)$ 
```

```
SELECT Penalties.*
```

```
FROM Penalties
```

```
JOIN TransactionDetails ON Penalties.HistoryID = TransactionDetails.HistoryID  
WHERE TransactionDetails.PickupDate BETWEEN '01-JAN-23' AND '05-JAN-23';
```

```
join  $\leftarrow$  TransactionDetails  $\bowtie_{Penalties.HistoryId = TransactionDetails.HistoryId}$  (Penalties)
```

```
 $\delta_{TransactionDetails.PickupDate \text{ BETWEEN } '01-JAN-23' \text{ AND } '05-JAN-23'}(join)$ 
```

```
SELECT MemID, COUNT(*) AS PenaltyCount
```

```
FROM Penalties
```

```
GROUP BY MemID;
```

```
filtered  $\leftarrow \pi_{MemId, Count(*)}(Penalties)$ 
```

```
 $\gamma_{MemId}(\text{filtered})$ 
```

```
SELECT * FROM Penalties
```

```
ORDER BY PenaltyID ASC;
```

```
 $\tau_{PenaltyId \text{ ASC}}(\sigma_*(Penalties))$ 
```

```
SELECT * FROM Penalties WHERE HistoryID = 12;
```

```
 $\sigma_{HistoryId = 12}(Penalties)$ 
```

```
SELECT * FROM Librarian WHERE Name LIKE 'M%';
```

$$\sigma_{Name \text{ LIKE } 'M\%'}(\text{Librarian})$$

```
SELECT COUNT(*) FROM Librarian;
```

$$\sigma_{COUNT(*)}(\text{Librarian})$$

```
select COUNT(*) FROM ( select distinct AdminKey FROM Librarian);
```

$$\sigma_{COUNT(*)} (\delta_{AdminKey} (\text{Librarian}))$$

```
select * from Librarian order by name ASC;
```

$$\tau_{name \text{ ASC}}(\sigma_*(\text{Librarian}))$$

```
select * from Librarian where Adminkey = (select min(Adminkey) FROM Librarian);
```

$$\sigma_{AdminKey = (\sigma_{min(AdminKey)}(\text{Librarian}))}$$

```
select LibID, Name From Librarian;
```

$$\sigma_{LibId, Name} (\text{Librarian})$$

```
select * from Librarian Where NAME LIKE 'Minh Tran' ;
```

$$\pi \leftarrow (ID, NAME, \dots)$$
$$(\sigma \leftarrow (NAME='Minh Tran') (\text{Librarian}))$$

```
SELECT * FROM TransactionDetails Order By PickupDate Asc;
```

$$\tau_{PickupDate \text{ ASC}} (\sigma_*(\text{TransactionDetails}))$$

```

SELECT Status, COUNT(*) AS TotalReturned FROM TransactionDetails GROUP BY Status;

Rename ← p(TotalReturned)(Count(*)(TransactionDetails))
Proj ← πStatus, Rename(TransactionDetails)
Final ← γStatus(Proj)

```

```

SELECT * From transactionDetails Where expiredate > '2023-01-20' ORDER by ExpireDate
Desc;

Filt ← σexpiredate > '2023 - 01 - 20'(TransactionDetails)
Result ← δExpiredate DESC(Filt)

```

```

SELECT expireDate,MemID From TransactionDetails Where Status = 0 AND expiredate <
SYSDATE;

Filt 1 ← σstatus = 0(TransactionDetails)
Filt 2 ← σexpiredate < SYSDATE(TransactionDetails)
Filt ← Filt1 ∩ Filt2
Result ← πexpireDate, MemID(Filt)

```

```

SELECT DISTINCT pickupdate From transactionDetails;

Result ← δPickupdate(transactionDetails)

```

```

SELECT * From Transactiondetails WHERE Pickupdate BETWEEN '2023-01-10' AND
'2023-01-15';

Filt 1 ← σExpiredate > 2023 - 01 - 10(TransactionDetails)
Filt 2 ← σExpiredate < '2023 - 01 - 15'(TransactionDetails)
Result ← Filt1 ∩ Filt2

```

```

SELECT * From TransactionDetails where (Status = 0) And (ExpireDate Between
2023-01-22' AND '2023-01-27');

Filt1 ← σStatus = 0(TransactionDetails)

```

```

Filt2 ←  $\sigma_{\text{Expiredate} > 2023 - 01 - 22}$  (TransactionDetails)
Filt 3 ←  $\sigma_{\text{Expiredate} < '2023 - 01 - 27'}$  (TransactionDetails)
Filt ← Filt2 ∩ Filt3
Result ← Filt1 ∩ Filt

```

```
SELECT COUNT(libid), memid FROM feedback GROUP BY memid;
```

```

Proj ←  $\pi_{\text{COUNT(libid), memid}}$  (feedback)
Result ←  $\gamma_{\text{memid}}$  (Proj)

```

```

SELECT libid, stars, comments FROM feedback WHERE
stars >= (SELECT AVG(stars) FROM feedback) ORDER BY
stars DESC;

```

```

Proj ←  $\pi_{\text{libid, stars, comments}}$  (feedback)
Aver ← AVEstars (Proj)
Filt ←  $\sigma_{\text{stars} \geq \text{Aver}}$  (Proj)
Result ←  $\tau_{\text{members.memid ASC}}$  (Filt)

```

```

SELECT DISTINCT librarian.libid, librarian.name, feedback.stars, feedback.comments
FROM feedback INNER JOIN
librarian ON librarian.libid = feedback.libid WHERE librarian.name = 'Ho Kim Lee';

```

```

Joinedmembers ← librarian.libid ⋙ feedback.libid (Feedback)
Distinct ←  $\delta_{\text{Joinedmembers, librarian.name, feedback.stars, feedback.comments}}$ 
Result ←  $\sigma_{\text{librarian.name} = 'Ho Kim Lee'}$  (Distinct)

```

```

SELECT members.name, feedback.memid, feedback.comments FROM feedback INNER
JOIN members ON members.memid = feedback.memid ORDER BY members.memid ASC;

```

```

Joinedmembers ← members.memid ⋙ feedback.memid (Feedback)
Proj ←  $\pi_{\text{members.name, feedback.memid, feedback.comments}}$  (feedback)
Result ←  $\tau_{\text{members.memid ASC}}$  (Proj)

```

```

SELECT members.name, feedback.memid, feedback.comments FROM feedback INNER
JOIN members ON members.memid = feedback.memid where members.memid = 3 ORDER

```

BY comments;

JoinedMembers \leftarrow members.memid $\bowtie_{\text{members.memid} = 3}$ feedback.memid (feedback)

Proj $\leftarrow \pi_{\text{name, memid, comments}} (\text{JoinedMem})$

Result $\leftarrow \tau_{\text{comments}} (\text{Joined})$

SELECT feedback.memid, feedback.comments FROM
feedback WHERE feedback.comments LIKE '%highschool%';

Proj $\leftarrow \pi_{\text{memid, comments}} (\text{feedback})$

Result $\leftarrow \sigma_{\text{comments Like '%highschool'}} (\text{Proj})$

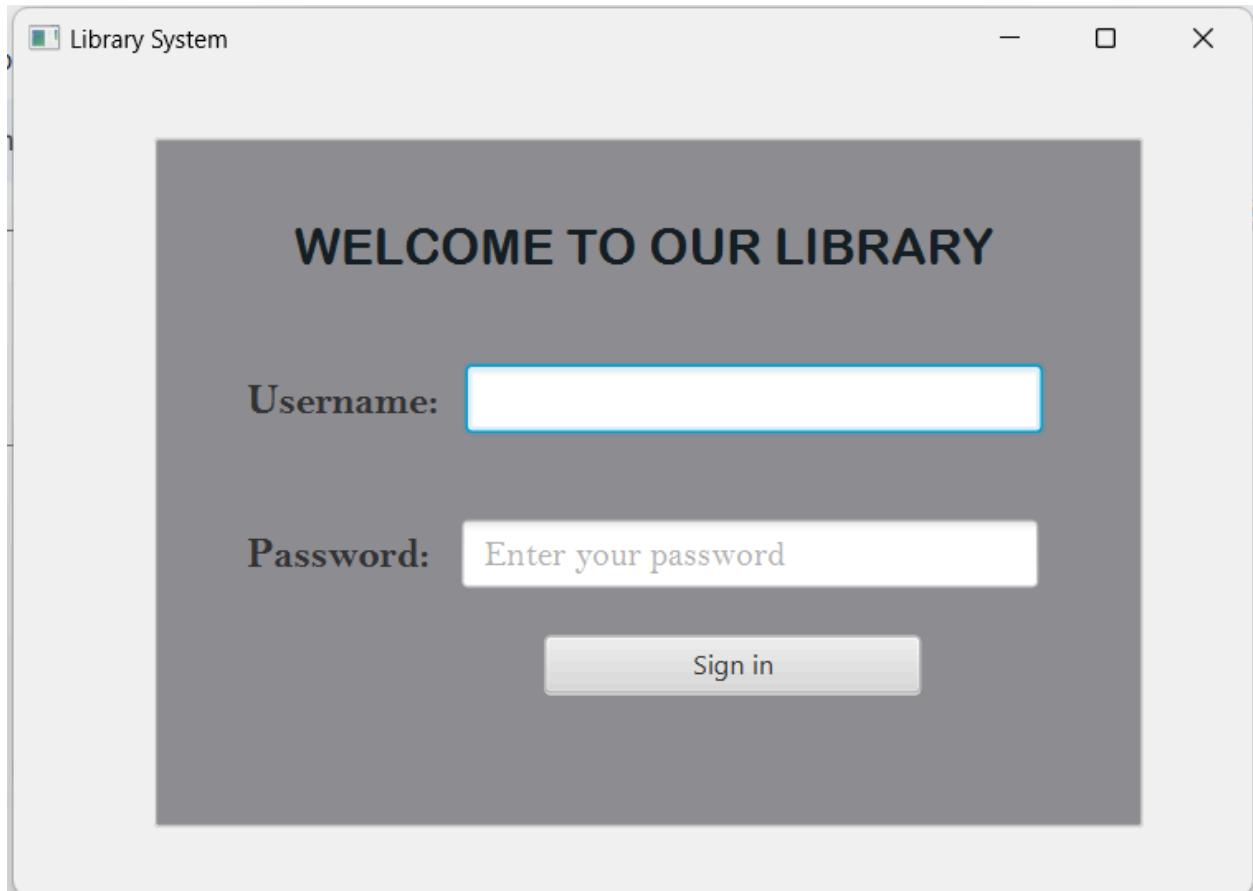
SELECT feedback.memid, feedback.stars,
feedback.comments FROM feedback WHERE feedback.stars = 5;

Proj $\leftarrow \pi_{\text{memid, stars, comments}} (\text{feedback})$

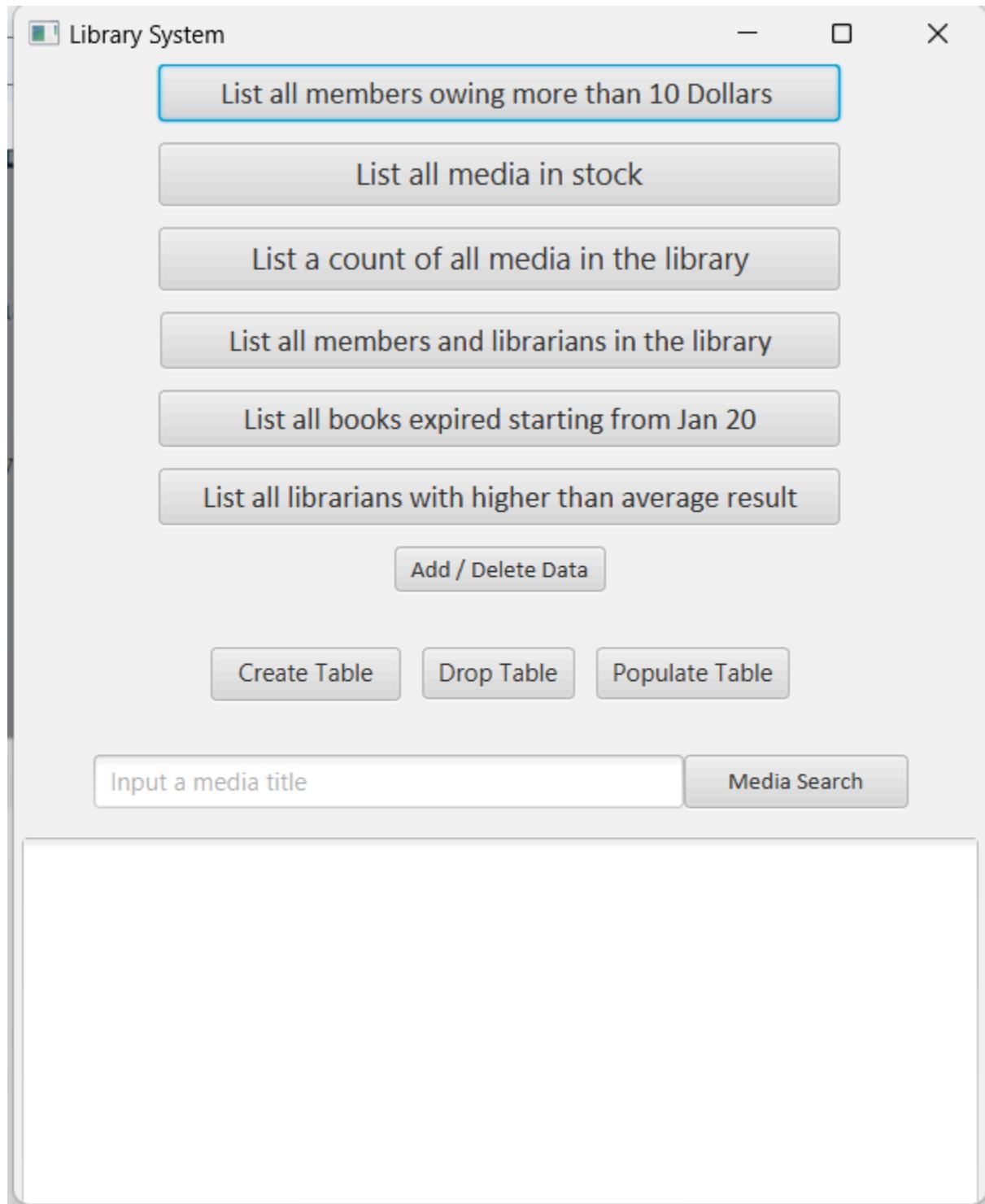
Result $\leftarrow \sigma_{\text{Stars} = 5} (\text{Proj})$

Library System Graphical User Interface

Login page for Members:



Main Library system after login for librarian:



Query 1 : List all members Owing more than 10 dollars on javaFX graphical user interface

The screenshot shows a JavaFX application window titled "Library System". The window contains a list of six buttons, each representing a SQL query:

- List all members owing more than 10 Dollars
- List all media in stock
- List a count of all media in the library
- List all members and librarians in the library
- List all books expired starting from Jan 20
- List all librarians with higher than average result

Below these buttons is a button labeled "Add / Delete Data". Further down are three more buttons: "Create Table", "Drop Table", and "Populate Table". At the bottom of the window is a search bar with the placeholder text "Input a media title" and a "Media Search" button.

The text area at the bottom of the window displays the following output:

```
11 icie25 iceeyou@yahoo.ca 20.0
1 a giand32@gmail.com 45.0
7 rdubuque rdubuque@gmail.com 45.0
3 orie.armstrong oreocookie@gmail.com 46.0
```

Query 2 : List all media in stock on javaFX graphical user interface

The screenshot shows a JavaFX application window titled "Library System". The window contains a list of six query buttons, each enclosed in a rounded rectangle:

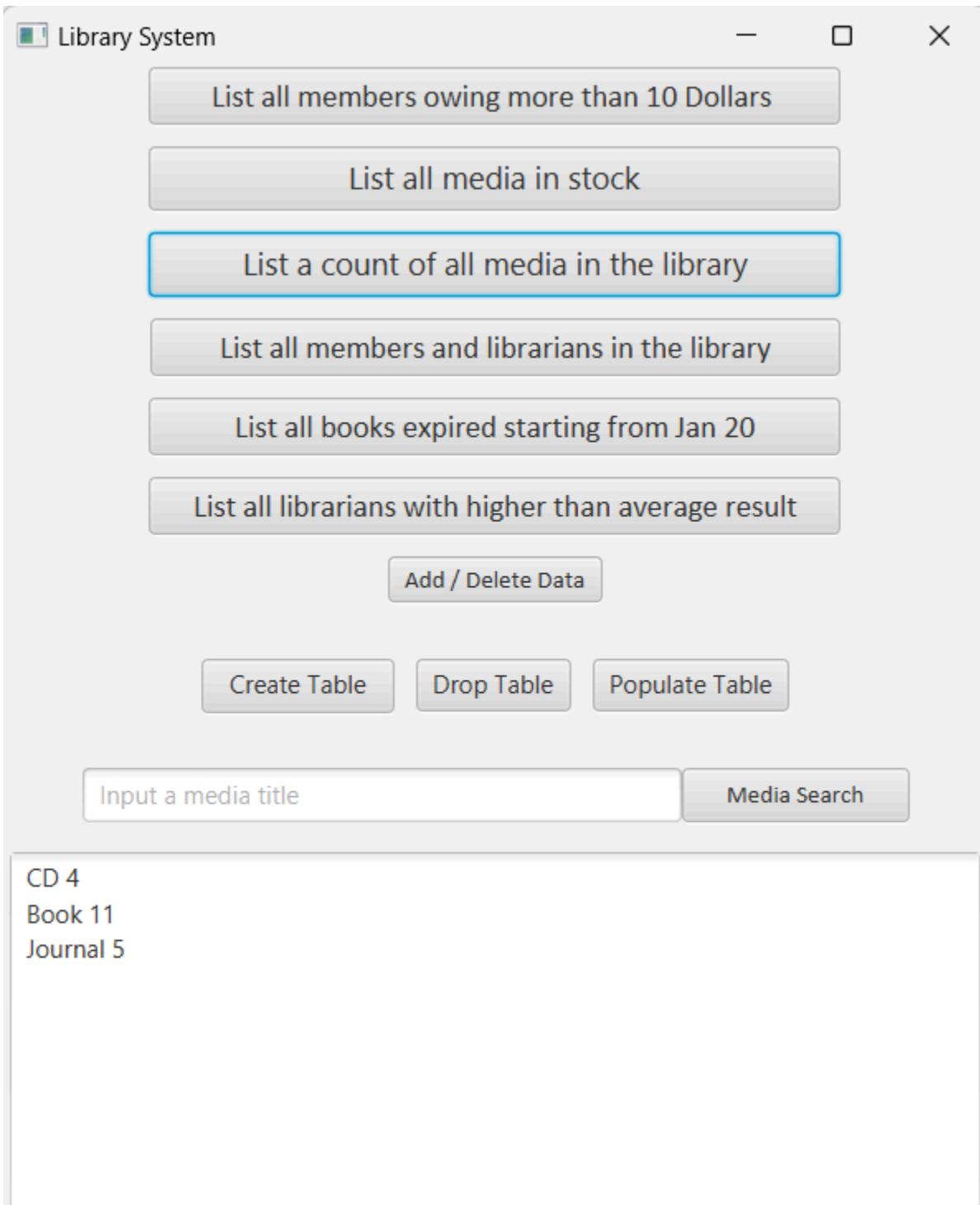
- List all members owing more than 10 Dollars
- List all media in stock** (This button is highlighted with a blue border)
- List a count of all media in the library
- List all members and librarians in the library
- List all books expired starting from Jan 20
- List all librarians with higher than average result

Below the queries is a button labeled "Add / Delete Data". Further down are three more buttons: "Create Table", "Drop Table", and "Populate Table". At the bottom of the window is a search interface consisting of two input fields: "Input a media title" and "Media Search".

The scrollable list area displays the following media entries:

- How to grow blueberries
- Charles Hadwell
- Book
- C programming
- Alex Ufkes
- Book
- Air currents
- Marty small
- Journal
- Anthropology

Query 3 : List a count of all media in the library using FX graphical user interface



Query 4 : List all members and librarians on javaFX graphical user interface

The screenshot shows a JavaFX application window titled "Library System". The window contains a list of six buttons, each representing a SQL query. The fourth button, "List all members and librarians in the library", is highlighted with a blue border. Below the buttons is a "Add / Delete Data" button. Further down are three more buttons: "Create Table", "Drop Table", and "Populate Table". At the bottom of the window is a search bar with the placeholder "Input a media title" and a "Media Search" button. The main area displays a scrollable list of names and email addresses, all labeled as "Member".

Library System

List all members owing more than 10 Dollars

List all media in stock

List a count of all media in the library

List all members and librarians in the library

List all books expired starting from Jan 20

List all librarians with higher than average result

Add / Delete Data

Create Table

Drop Table

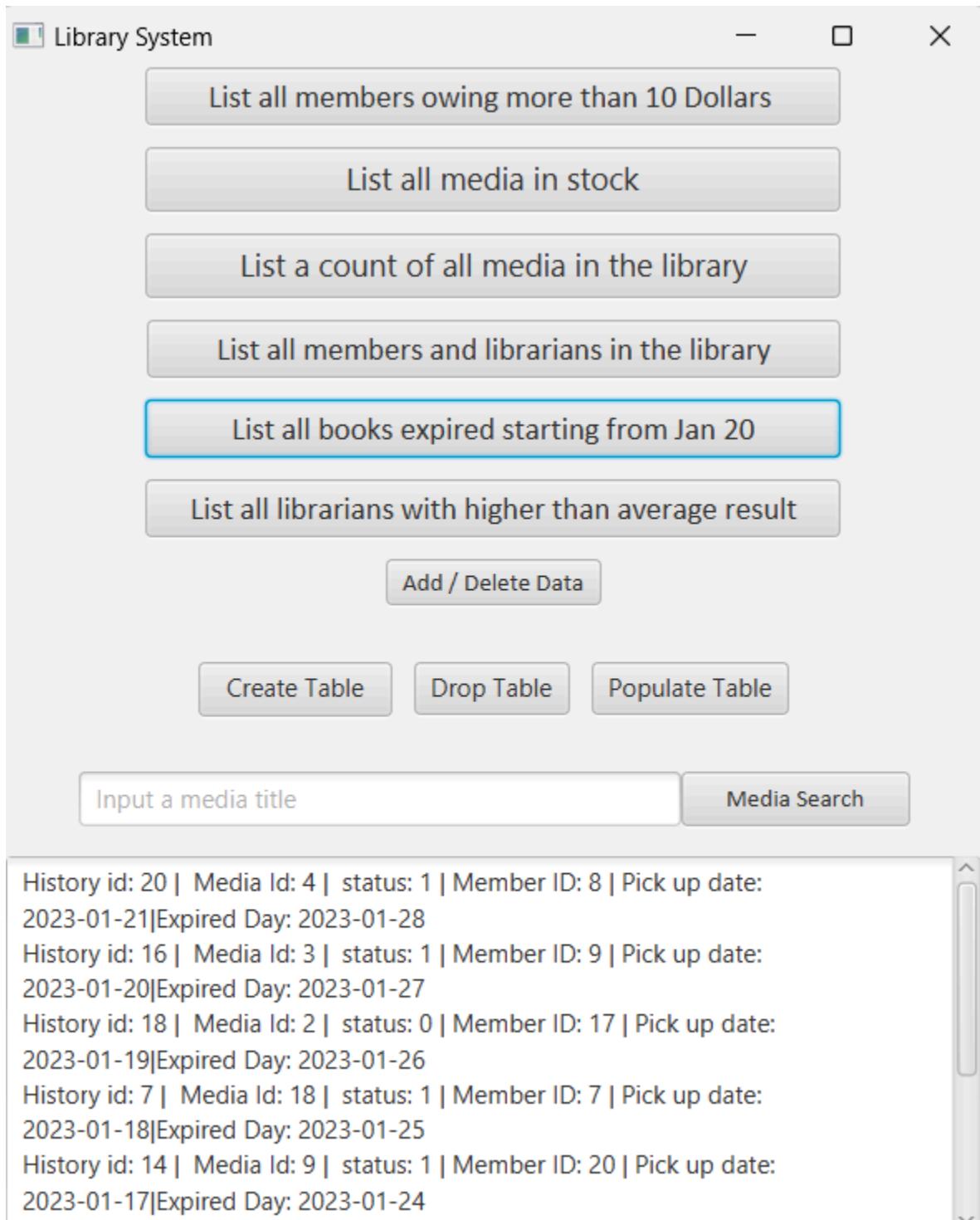
Populate Table

Input a media title

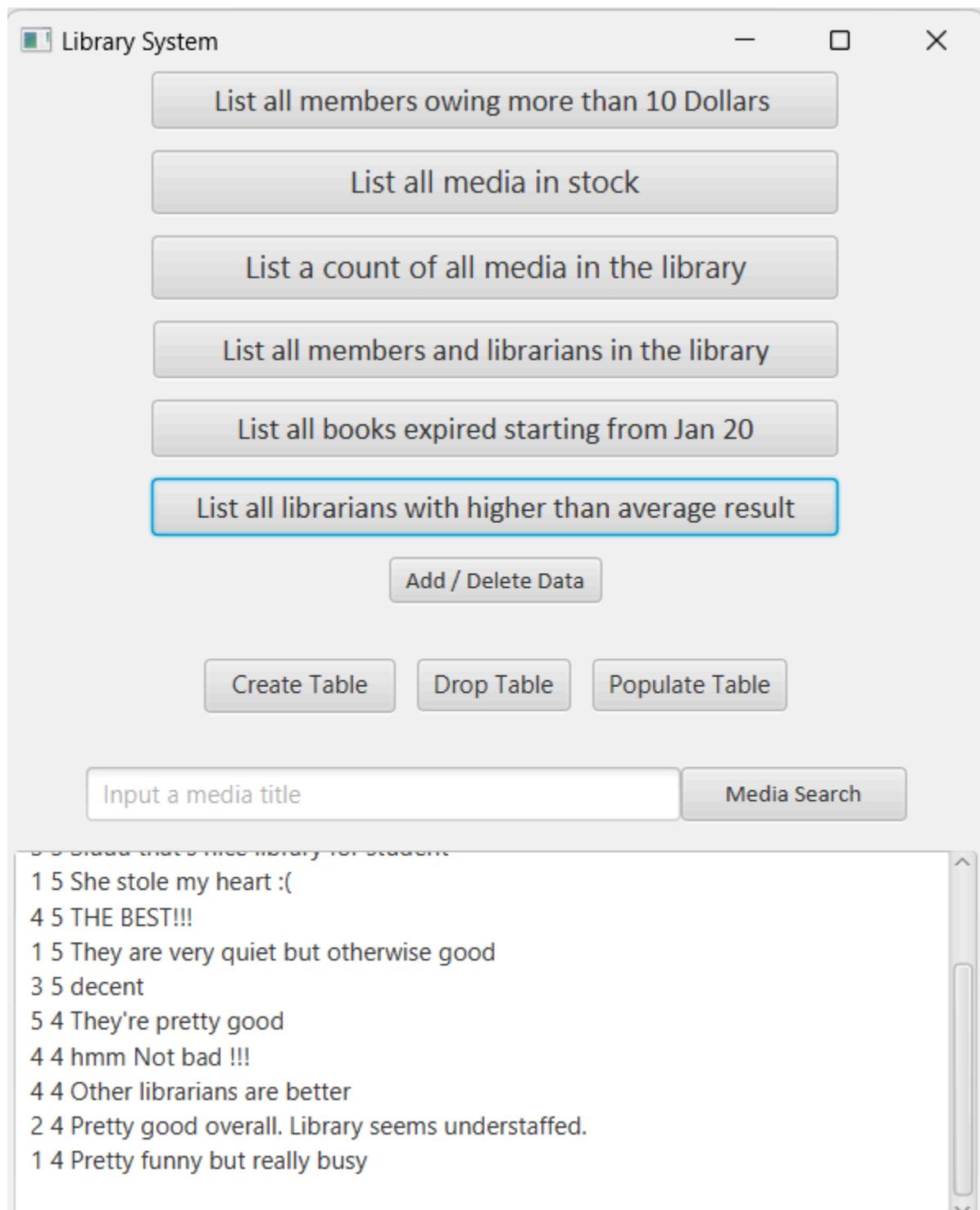
Media Search

Allan allanturing@gmail.com Member
Allan ucorn100@gmail.com Member
Andy andytherandyy@gmail.com Member
Bart imbaroque@gmail.com Member
Brekke brekke.robb@gmail.com Member
Ellen ehu0098@gmail.com Member
Emerson Emerson.watsica@gmail.com Member
Ho Kim Lee null Librarian
Jacob liajacobsone@gmail.com Member
Jenkins jenkins.fritz@gmail.com Member

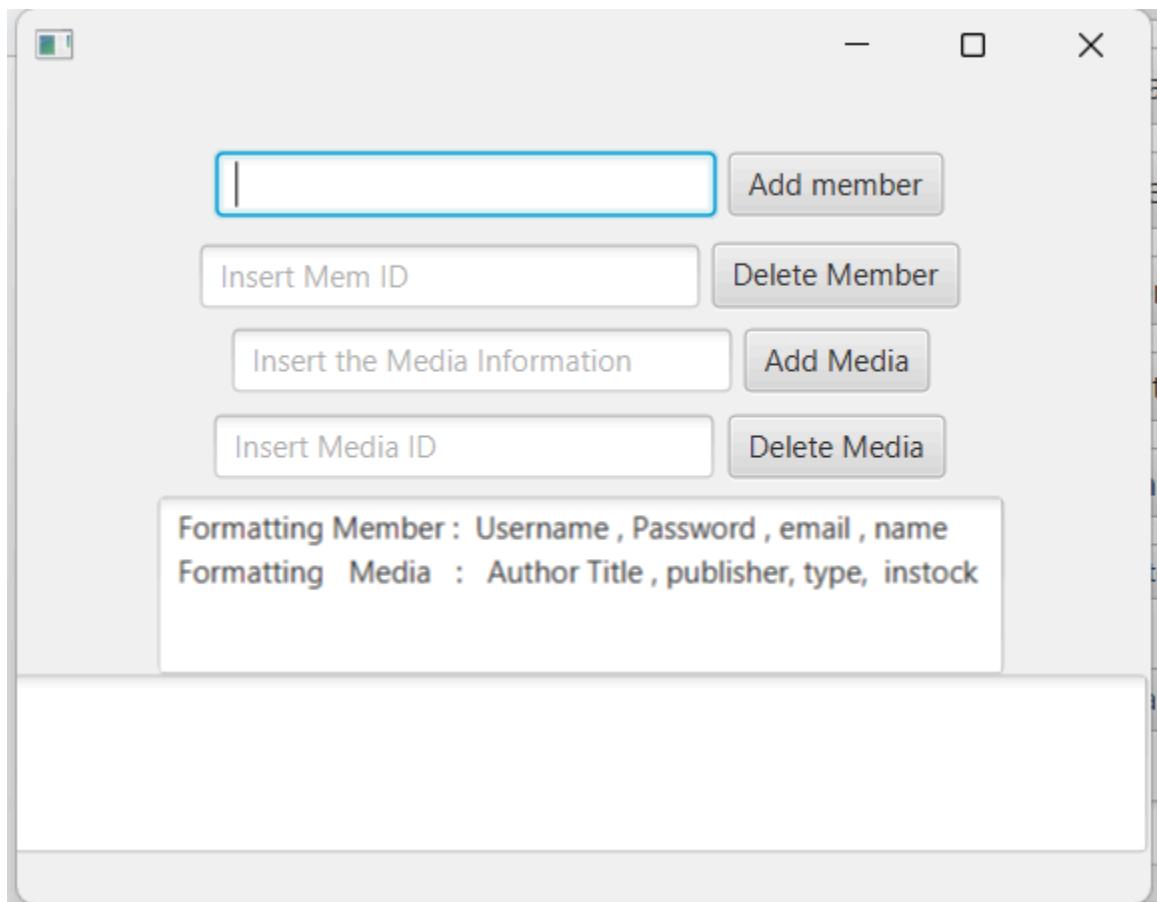
Query 5 : List all books expired starting from Jan 20 on javaFX graphical user interface



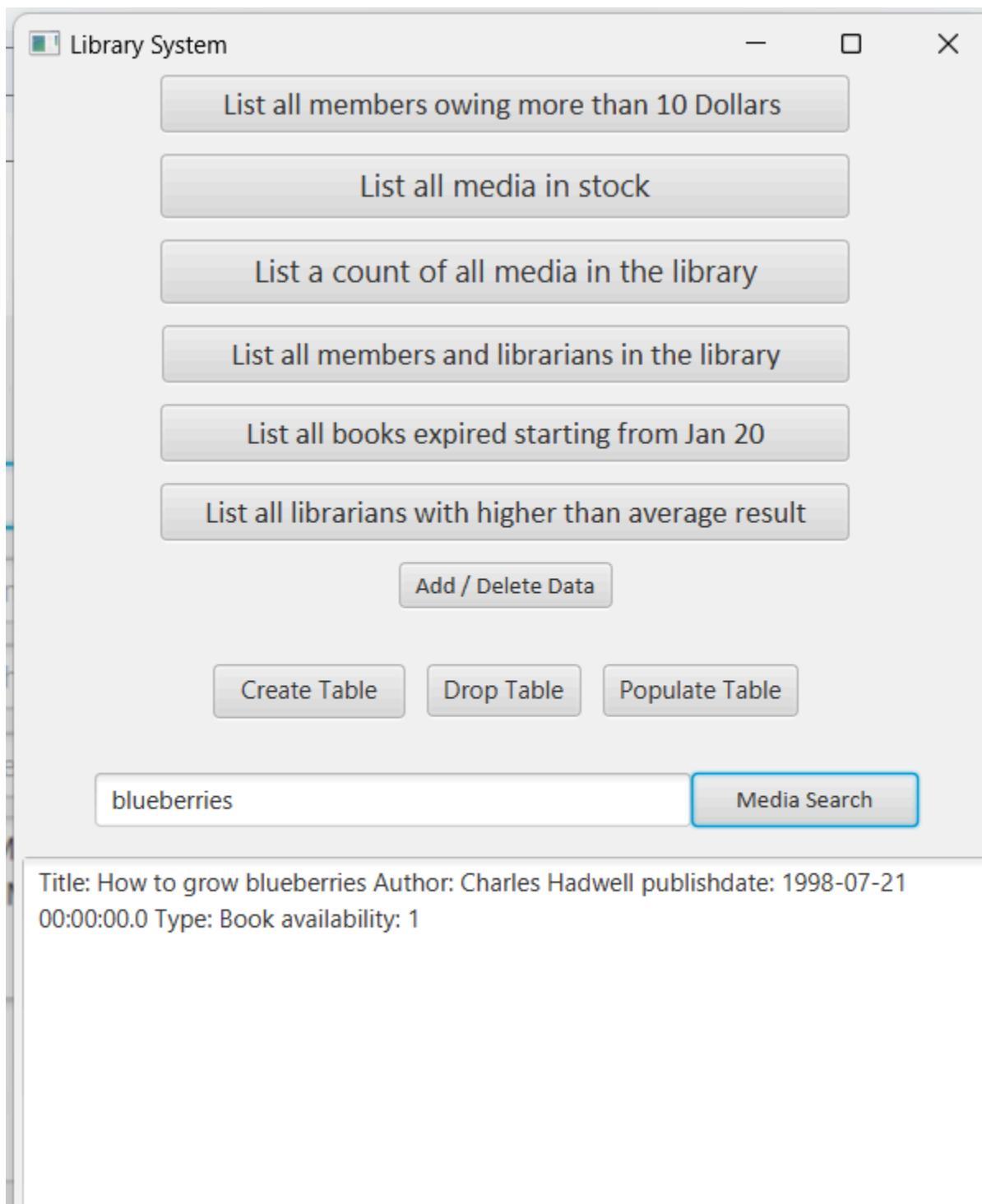
Query 6 : List all librarians with higher than average result on javaFX graphical user interface



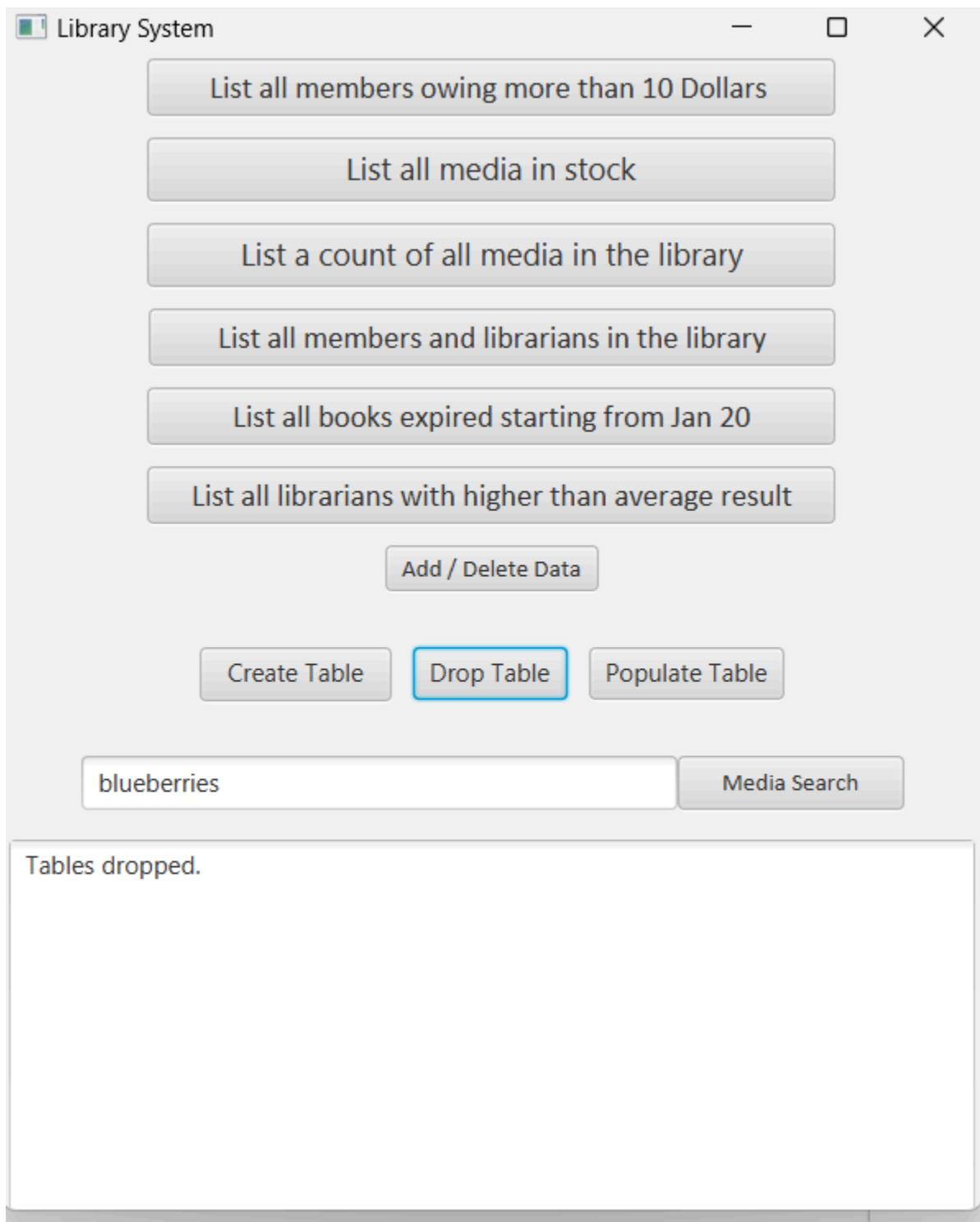
Adding and Deleting members and media using javaFX graphical user interface.



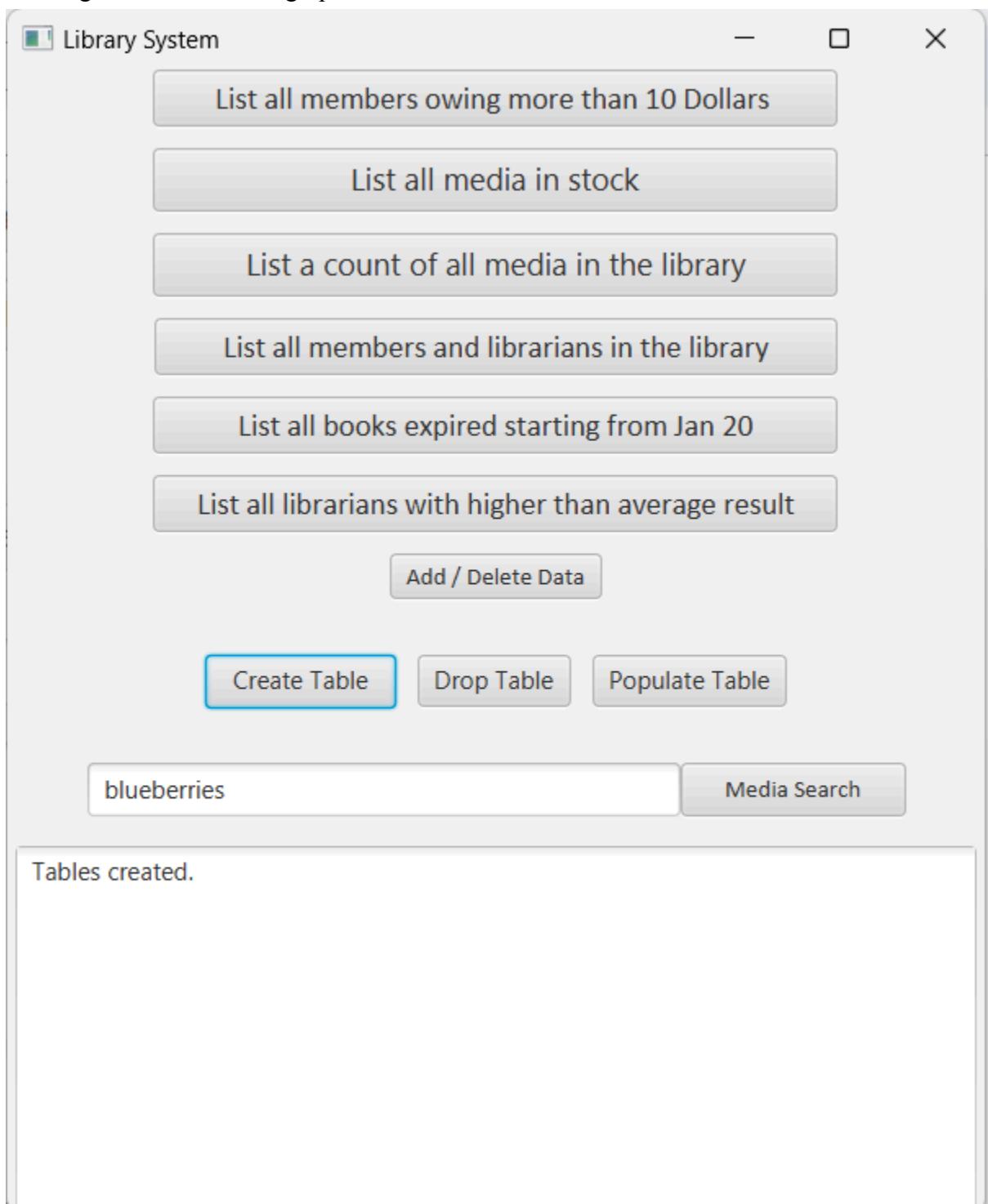
Query 7 : Entering Title keyword to search media on javaFX graphical user interface



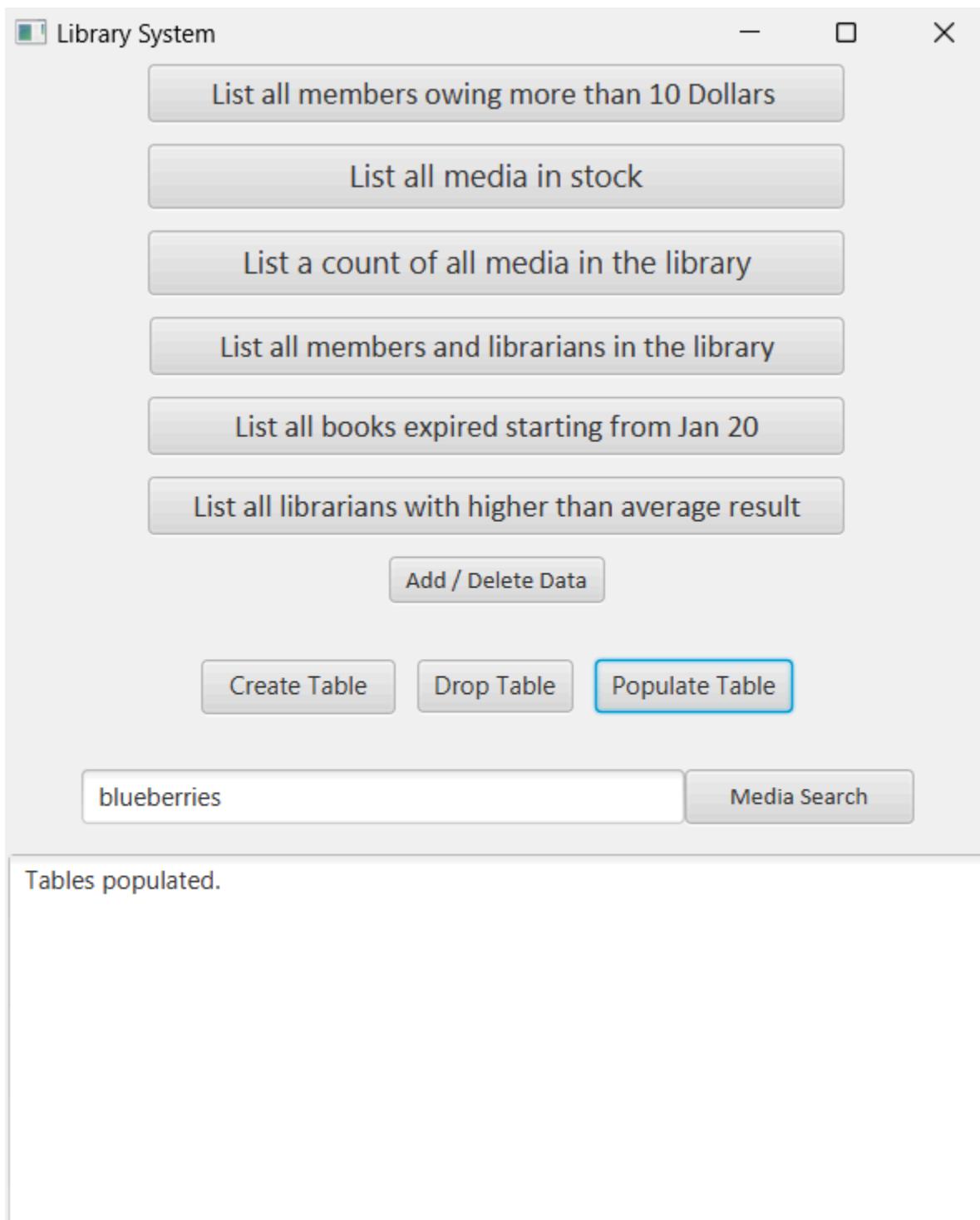
Dropping tables on JavaFX graphical user interface.



Creating tables on JavaFX graphical user interface.



Populate table on JavaFX graphical user interface.



CONCLUSION

Conclusion: Minh Tran

Throughout our final project, we undertook ten assignments that guided us from conceptualization to developing a fully functional program capable of user interaction. These assignments deepened our comprehension of the SQL language and the Oracle database management system, particularly in working with data pertinent to our library system, including media, members, and transaction details.

Initiating assignments 1 and 2, we gained familiarity with entity concepts and ER diagrams, providing invaluable insights into translating our library system concept into a practical application. Assignment 3 marked the initial phase of constructing our SQL system by creating tables corresponding to our predefined entities—subsequently, assignment four involved formulating queries based on these entities crafting meaningful statements for our dataset.

While assignment 6 posed challenges in exporting queries via the terminal, teaching us to connect our SQL system with the school database and present results on the command line, assignments 7 and 8 delved into the intricacies of the third standard form and Bernstein algorithm. Assignment 9 proved the most complex, requiring us to grapple with formatting issues in integrating our work into the JavaFX interface.

Concluding with assignment 10, where we translated our SQL queries into relational algebra, the task demanded substantial time investment, given the approximately 40 queries developed in assignment 4. Despite the time consuming -, this process significantly augmented our understanding of relational model operations and the SQL standard query language for relational database management systems. In retrospect, this project has been a valuable learning experience, and we express sincere gratitude for the guidance and support provided by our professor and TA.

Conclusion: Woojin Lee

Throughout the whole lab, 10 labs were conducted and were able to understand and demonstrate the real process of database management and programming. Although all the assignments were handed in without extensions or late submissions, there were a lot of conflicts and resolutions throughout working with the assignments.

As for example, more tables were needed to be added and the modification of the system had to be done during the first few weeks of the semester, resulting in a brainstorming until late night to think of a new design and attribute for the database. This has taught us that even in the real work environment, these stressful situations occur on a daily basis, and it was a preparation, or a tutorial of how to come to a resolution.

Additionally, having a great schema, and preparation was a key aspect to finish this project successfully. By creating and maintaining a great basis and schematics has helped our group to create an optimized plan that will help everyone to manage their time. However, the last few assignments have not been the case, where having multiple different projects from different courses at the same time has made time management difficult. However, it was done and the graphic component was also finished on time to submit.

Overall, this lab has encouraged the students to get a grasp of how the data management system works and how the DBMS operates. Making a mock database system has enhanced the understanding of the database system, as well as the coding aspect of the project. A lot of independent studying had to be conducted as well, but has become a new skill for myself.

Conclusion: Talon Jiang

The purpose of this project was to create a complex Library System using Oracle 11g and the Java programming language. Throughout the duration of this project, we underwent a series of assignments to take the Library System from a conceptual level to an interactive user interface complete with graphics, user interactions and complex queries to retrieve and write data to a centralized Oracle database.

Through the first few assignments, we laid the groundwork for the conceptual level of the database system through the extensive use of SQL Developer. By determining the functions and entities in the system, we derived ER and relational diagrams for the system. Due to some oversights, we recognized that our system did not initially include the amount of functionality required, thus it was necessary to redesign the system before proceeding with further development.

In later assignments, our team took the time to develop a prototype of the system based on the centralized server provided by the institution using Bash, which proved useful in both finalizing the initial designs of our tables, and ensuring that the queries given to the system were meaningful and functioned as expected. We encountered several issues with database connections due to unfamiliarity with the server initially, however we were soon able to launch the system on time with the expected functionality as required.

When designing the tables for the System, we were initially unaware about whether our tables were optimized into normal forms. Further analysis revealed that most of the tables, fortunately, were in 3rd Normal form, although there were instances where some tables required further work to normalize them to an acceptable level.

By far, the most challenging part of this project has been the implementation phase, where copious amounts of research and testing were required to bring the project to an acceptable level of functionality. Due to time constraints leading into the project, our team was unable to fully implement all of the functions outlined in our planning phases, however we were able to bring out functionality for the majority of functions within the database system, and plan to further refine and grow the service of the system in the future.

Overall, this project has brought into light important lessons when building complex systems in a team, alongside the importance of planning and communication while implementing similar projects in the future.