Principle of Databases Course

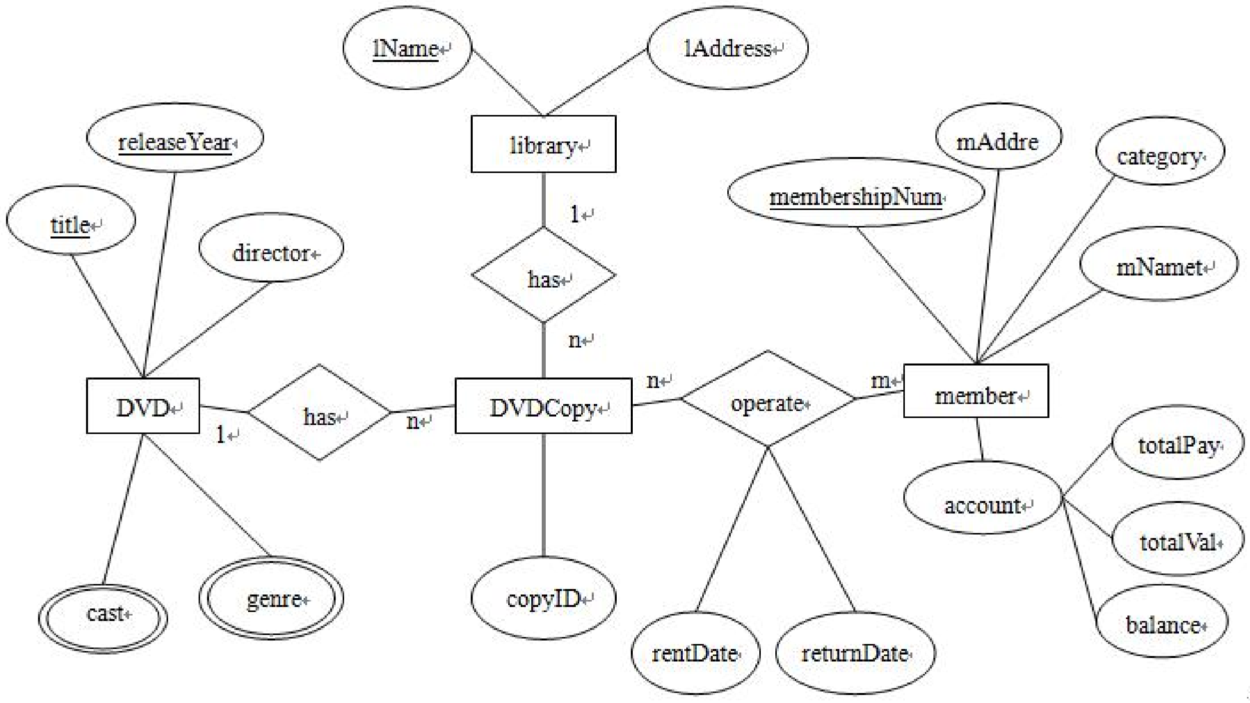
Assignment Report

Student Name	Student Number
xxxx	xxxxx

XXXX	**XXXXX**

1 EntityRelationship Diagram

1.1 Diagra m



1.2 Translating Entity-Relationship Data Models to Relations

library (<u>lName</u>, lAddress)

DVD (<u>title</u>, <u>releaseYear</u>, director)

DVDCast (<u>**title**</u>, <u>**releaseYear**</u>, <u>cast</u>)

DVDGenre (<u>**title**</u>, <u>**releaseYear**</u>, <u>genre</u>)

DVDCopy (<u>copyID</u>, title, releaseYear, lName) operate (**copyID**, **membershipNum**, returnDate, rentDate)

member (membershipNum, mName, mAddress, category, balance, totalVal, totalPay)

2 Normalization

2.1 Analysis

- **Attributes**

Company (lName, lAddress, title, releaseYear, director, cast, genre, copyID, membershipNum, returnDate, rentDate, mName, mAddress, category, balance, totalVal, totalPay)

- **Requirements and Dependencies**

Requirements Analysis	Functional Dependency
Given the library's name, we can get its address.	1. lName → lAddres
Given the DVD's title and its release year, we can get its director.	2. title, releaseYear → director
Given the DVD's title and its release year, we can get its cast, which can be multi-valued.	3. title, releaseYear →→ cast
Given the DVD's title and its release year, we can get its genre, which can be multi-valued.	4. title, releaseYear →→ genre
Given a copy's copyID, we can get its title, release year and the current library it is in.	5. copyID → title, releaseYear, lName

Given a membership number, we can get the member’s name, address, category, and the account information.	6.	membershipNum → mName, mAddres, category, balance, totalVal, totalPay
Given a copy ID, membership number and a rent date, we can uniquely identify a rental record, including the return date.	7.	copyID, membershipNum, rentDate → returnDate

2.2 Decomposition

Table
S0

Attributes	lName, lAddress, title, releaseYear, director, cast, genre, copyID, membershipNum, returnDate, rentDate, mName, mAddress, category, balance, totalVal, totalPay	
FD	lName → lAddres title, releaseYear → director title, releaseYear →→ cast	✗

	title, releaseYear →→ genre copyID → title, releaseYear, lName membershipNum → mName, mAddres, category, balance, totalVal, totalPay copyID, membershipNum, rentDate → returnDate	

Decompose Table S0 on **lName → lAddress**

S1	(lName, lAddres)	✓
S2	(lName, title, releaseYear, director, cast, genre, copyID, membershipNum, returnDate, rentDate, mName, mAddres, category, balance, totalVal, totalPay)	✗

Decompose S2 on **title, releaseYear → director**

S3	(title, releaseYear, director)	✓
S4	(lName, title, releaseYear, cast, genre, copyID, membershipNum, returnDate, rentDate, mName, mAddress, category, balance, totalVal, totalPay)	✗

Decompose S4 on **title, releaseYear →→ cast**

S5	(title, releaseYear, cast)	✓
S6	(lName, title, releaseYear, genre, copyID, membershipNum, returnDate, rentDate, mName, mAddress, category, balance, totalVal, totalPay)	✗

Decompose S6 on **title, releaseYear →→ genre**

S7	(title, releaseYear, genre)	✓
S8	(lName, title, releaseYear, copyID, membershipNum, returnDate, rentDate, mName, mAddress, category, balance, totalVal, totalPay)	✗

Decompose S8 on **copyID → title, releaseYear, lName**

S9	(copyID, title, releaseYear, lName)	✓
S10	(copyID, membershipNum, returnDate, rentDate, mName, mAddress, category, balance, totalVal, totalPay)	✗

Decompose S10 on **membershipNum → mName, mAddress, category, balance, totalVal, totalPay**

S11	(membershipNum, mName, mAddress, category, balance, totalVal, totalPay)	✓

S12	(copyID, membershipNum, returnDate, rentDate)	✗

The tables are:

S1	(lName, lAddres)	✓
S3	(title, releaseYear, director)	✓
S5	(title, releaseYear, cast)	✓
S7	(title, releaseYear, genre)	✓
S9	(copyID, title, releaseYear, lName)	✓
S11	(membershipNum, mName, mAddress, category, balance, totalVal, totalPay)	✓
S12	(copyID, membershipNum, returnDate, rentDate)	✓

3 Comparison & Result

We use ER diagram and decomposition and get two sets of tables. As a result, the comparison as follows shows that they are completely the same.

Tables from **Entity Relationship Diagram**	**Tables from** **Normalization**

library (lName, lAddress)	S1 (lName, lAddres)
DVD (title, releaseYear, director)	S3 (title, releaseYear, director)
DVDCast (**title**, **releaseYear**, cast)	S5(title, releaseYear, cast)
DVDGenre (**title**, **releaseYear**, genre)	S7(title, releaseYear, genre)
DVDCopy (copyID, title, releaseYear, lName)	S9(copyID, title, releaseYear, lName)
member (membershipNum, mName, mAddress, category, balance, totalVal, totalPay)	S11 (membershipNum, mName, mAddress, category, balance, totalVal, totalPay)
operate (**copyID**, **membershipNum**, returnDate, rentDate)	S12(copyID, membershipNum, returnDate, rentDate)

Comparing with the two model, here is the logical data model being used.

Logical Data Model

TABLE NAME	ATTRIBUTES
library	(lName, lAddress)
DVD	(title, releaseYear, director)
DVDCast	(**title**, **releaseYear**, cast)
DVDGenre	(**title**, **releaseYear**, Director)
DVDCopy	(copyID, title, releaseYear, lName)
member	(membershipNum, mName, mAddress, category, balance, totalVal, totalPay)
operate	(**copyID**, **membershipNum**, returnDate, **rentDate**)

4 SQL code

4.1 Build

```
create schema library;
use library;

create table library (
  lName varchar(50) not null,
  lAddress varchar(50) NULL,
  primary key (lName)
);

create table member(
 membershipNum INT not null,
 mName varchar(50) null,
 mAddress varchar(50) null,
 category varchar(50),
 balance double(7,2) default 0.00,
 totalVal double(7,2) default 0.00,
 totalPay double(7,2) default 0.00,
 primary key(membershipNum),
 CHECK (balance>=0),
 CHECK (totalVal>=0),
 CHECK (totalPay>=0),
 CHECK(totalVal>=totalPay),
 check(category in ('normal','premium'))
);

create table DVD(
 title varchar(50) not null,
 releaseYear year(4) not null,
 director varchar(50) null,
 primary key(title,releaseYear)
);
```

```
create table DVDCast(
 title varchar(50) not null,
 releaseYear year(4) not null,
 castName varchar(50) not null,
 primary key(title,releaseYear,castName),
 foreign key(title, releaseYear) references DVD(title, releaseYear)
);

create table DVDGenre(
 title varchar(50) not null,
 releaseYear year(4) not null,
 genre varchar(50) not null,
 primary key(title,releaseYear,genre),
 foreign key(title, releaseYear) references DVD(title, releaseYear)
);

create table DVDCopy(
 copyID varchar(50) not null,
 title varchar(50)null,
 releaseYear year(4) null,
 lName varchar(50) null,
 primary key(copyID),
 foreign key(title, releaseYear) references DVD(title, releaseYear),
 foreign key(lName) references library(lName)
);

create table operate(
 rentDate date not null,
 returnDate date null,
 membershipNum int not null,
 copyID varchar(50) not null,
 primary key(rentDate, membershipNum, copyID),
 foreign key(membershipNum) references member(membershipNum),
 foreign key(copyID) references DVDCopy(copyID)
);
```

4.2 Populate

```
insert into library(lAddress, lName) values ('Shenying Road', 'NEUlibrary');
 insert into library(lAddress, lName) values ('Sanhao Street', 'Nanhulibray');
 insert into library(lAddress, lName) values ('6th Avenue', 'SYlibrary');
 insert into library(lAddress, lName) values ('7th Road', 'LNlibrary');
 insert into library(lAddress, lName) values ('Wenguan Road', 'Somelibrary');

 -- 会员信息
 insert into member(membershipNum, mName, mAddress, category, balance, totalVal, totalPay)
values (20161111,'Abigail', 'A1', 'normal', 5.5, 0, 0);
……
 insert into member(membershipNum, mName, mAddress, category, balance, totalVal, totalPay)
values (20165555,'Emily','A5','normal',112,10,10);

 -- DVD
 insert into DVD(title, releaseYear, director) values ('Handmaid''s Tale', '2017', 'Reed Morano');
……
 insert into DVD(title, releaseYear, director) values ('La La Land','2016','Damien Chazelle');

 -- DVD cast
 insert into DVDCast(title, releaseYear, castName) values ('Handmaid''s Tale', '2017','Elisabeth Moss');
……
 insert into DVDCast(title, releaseYear, castName) values ('La La Land','2016','Emma Stone');

 -- DVD Genre
 insert into DVDGenre(title, releaseYear, genre) values ('Handmaid''s Tale', '2017', 'Horror');
……
 insert into DVDGenre(title, releaseYear, genre) values ('La La Land','2016', 'Romance');

 -- DVD Copy
```

insert into DVDCopy(copyID, title, releaseYear, lName) values('HMT2017_1', 'Handmaid''s Tale', '2017', 'NEUlibrary');

......

insert into DVDCopy(copyID, title, releaseYear, lName) values('LLD2016_4', 'La La Land','2016', 'Somelibrary'

4.3 Manipulate

```
-- History
insert into operate(rentDate, returnDate, membershipNum, copyID) values ('2017-1-1', '2017-1-15', 20161111,'HMT2017_1');
......
membershipNum, copyID) values ('2018-4-11', '2018-4-29', 20165555,'LLD2016_4');
 insert into operate(rentDate, returnDate, membershipNum, copyID) values ('2018-3-11', '2018-4-7', 20165555,'HMT2017_1');

insert into operate(rentDate, membershipNum, copyID) values ('2018-5-10',20165555,'LLD2016_4');
 update DVDCopy SET lName = null where copyID = 'LLD2016_4';
 insert into operate(rentDate, membershipNum, copyID) values ('2018-5-11',20165555,'SM1948_2');
 update DVDCopy SET lName = null where copyID = 'SM1948_2';
 insert into operate(rentDate, membershipNum, copyID) values ('2018-5-11',20164444,'WW2016_4');
 update DVDCopy SET lName = null where copyID = 'WW2016_4';
```

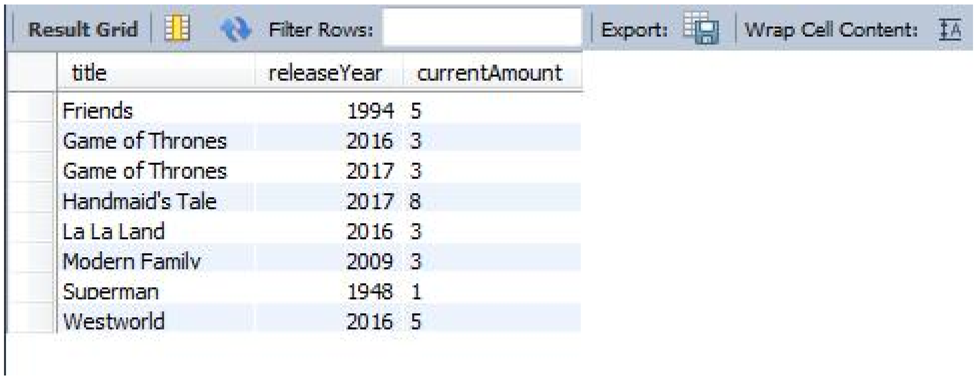
4.4 Query

Query 1 generate a list of all available DVDs

Code

```
select title, releaseYear, count(title) as currentAmount
from DVDCopy
where lName is not null
group by title, releaseYear;
```

Result set



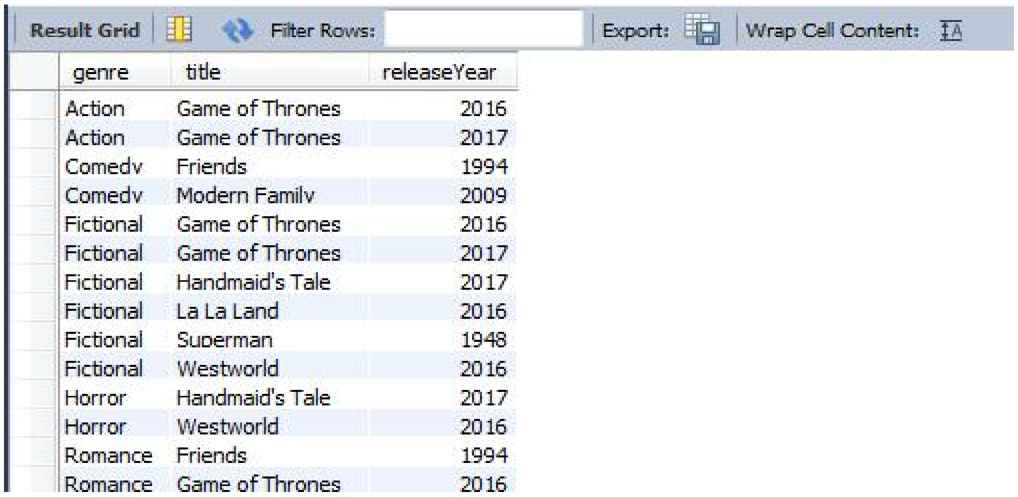
title	releaseYear	currentAmount
Friends	1994	5
Game of Thrones	2016	3
Game of Thrones	2017	3
Handmaid's Tale	2017	8
La La Land	2016	3
Modern Family	2009	3
Superman	1948	1
Westworld	2016	5

Query 2 generate a list of DVDs available in each genre

Code

```
select genre, dvdgenre.title, dvdgenre.releaseYear
from dvdgenre,dvdcopy
where dvdgenre.title = dvdcopy.title and dvdgenre.releaseYear = dvdcopy.releaseYear and
dvdcopy.lName is not null
group by genre,dvdgenre.title, dvdgenre.releaseYear;
```

Result set



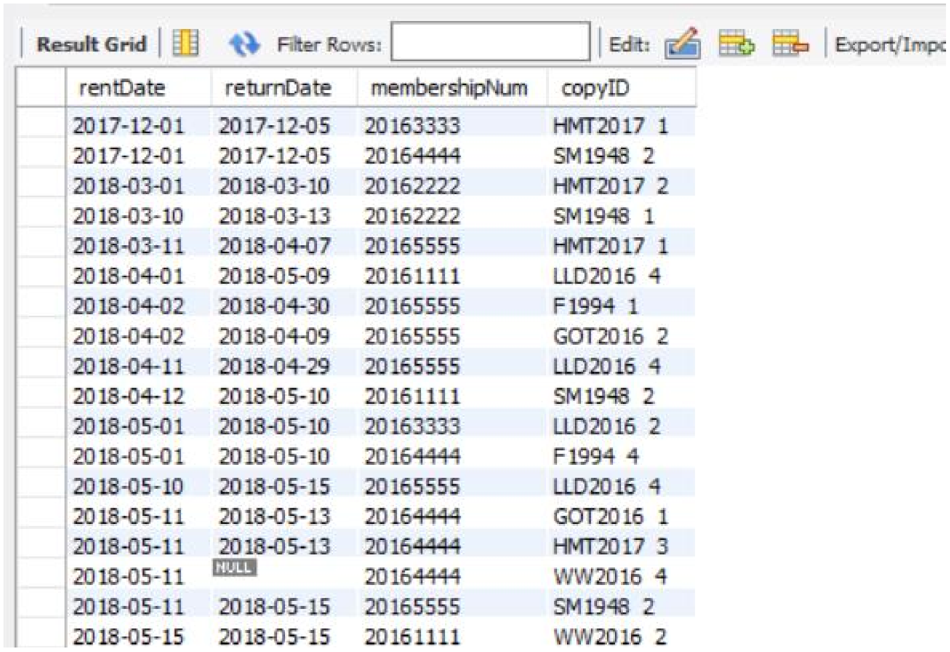
genre	title	releaseYear
Action	Game of Thrones	2016
Action	Game of Thrones	2017
Comedv	Friends	1994
Comedv	Modern Familv	2009
Fictional	Game of Thrones	2016
Fictional	Game of Thrones	2017
Fictional	Handmaid's Tale	2017
Fictional	La La Land	2016
Fictional	Suoerman	1948
Fictional	Westworld	2016
Horror	Handmaid's Tale	2017
Horror	Westworld	2016
Romance	Friends	1994
Romance	Game of Thrones	2016

Query 3 return the records of DVDs being rented and returned

Code

```
select *
from operate;
```

Result set



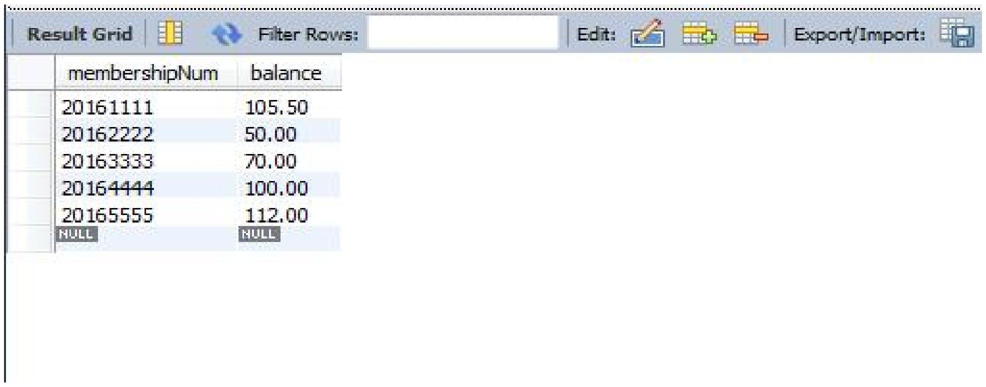
rentDate	returnDate	membershipNum	copyID
2017-12-01	2017-12-05	20163333	HMT2017 1
2017-12-01	2017-12-05	20164444	SM1948 2
2018-03-01	2018-03-10	20162222	HMT2017 2
2018-03-10	2018-03-13	20162222	SM1948 1
2018-03-11	2018-04-07	20165555	HMT2017 1
2018-04-01	2018-05-09	20161111	LLD2016 4
2018-04-02	2018-04-30	20165555	F1994 1
2018-04-02	2018-04-09	20165555	GOT2016 2
2018-04-11	2018-04-29	20165555	LLD2016 4
2018-04-12	2018-05-10	20161111	SM1948 2
2018-05-01	2018-05-10	20163333	LLD2016 2
2018-05-01	2018-05-10	20164444	F1994 4
2018-05-10	2018-05-15	20165555	LLD2016 4
2018-05-11	2018-05-13	20164444	GOT2016 1
2018-05-11	2018-05-13	20164444	HMT2017 3
2018-05-11	NULL	20164444	WW2016 4
2018-05-11	2018-05-15	20165555	SM1948 2
2018-05-15	2018-05-15	20161111	WW2016 2

Query 4 return the account balance for each member

Code

```
select membershipNum,balance
from member;
```

Result set



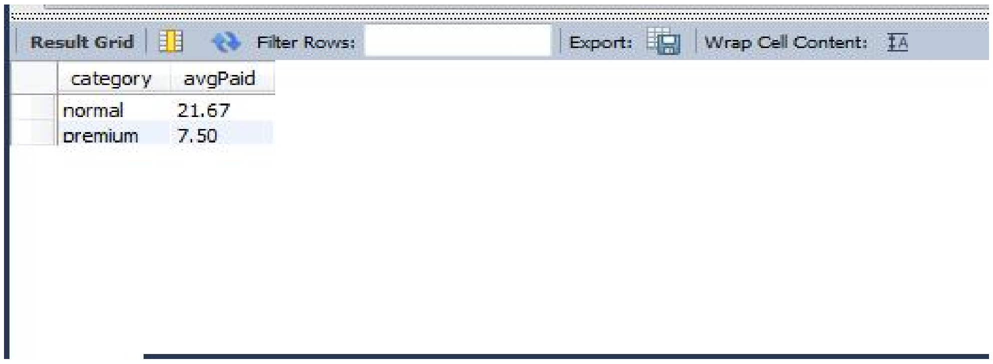
membershipNum	balance
20161111	105.50
20162222	50.00
20163333	70.00
20164444	100.00
20165555	112.00
NULL	NULL

Query 5 return the average money that normal members and premium members have spent

Code

```
select category, round(avg(totalPay),2) as avgPaid
from member
group by category;
```

Result set



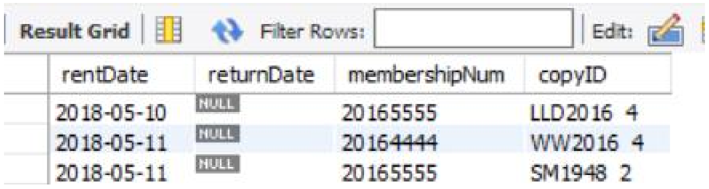
```
category   avgPaid
normal     21.67
premium    7.50
```

Query 6 return the list of DVDs being rented by members

Code

```
select *
from operate
where returnDate is null;
```

Result set



```
rentDate     returnDate   membershipNum   copyID
2018-05-10   NULL         20165555        LLD2016  4
2018-05-11   NULL         20164444        WW2016  4
2018-05-11   NULL         20165555        SM1948  2
```

5 JDBC

README

A simple java program to manipulate the library. When you enter the program, first, enter your membershipID (showed in the table member). Then you will get the menu of operations. You can top up your account, rent copies of DVDs from and return them to a library.

Scoure Code

```
package com.company;

import  java.sql.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class Main {



    public static String getDVDAvailable() {
        //a.search for the DVDs that can be rented by the member.
        //you can see the DVD available at each library
        String sql = "SELECT  * FROM  dvdcopy where lName is not null";
        return sql;
    }
    public static void executeDVDAvailable( ResultSet res) throws SQLException {
        System.out.println("copyID   title    releaseYear   lName ");
        while (res.next()) {
            String copyID = res.getString("copyID");
            String title = res.getString("title");
            String releaseYear = res.getString("releaseYear").substring(0, 4);
            String lName = res.getString("lName");
            System.out.println(copyID + "   " + title + "   " + releaseYear + "   " + lName);
        }
    }
    public static String getDVDAvailableInEachGenre() {
        //2. generate the list of DVDs        available in each genre
        String sql = "select genre, dvdgenre.title, dvdgenre.releaseYear " +
                "from dvdgenre,dvdcopy\r\n" +
                "where dvdgenre.title = dvdcopy.title and dvdgenre.releaseYear =
dvdcopy.releaseYear and dvdcopy.lName is not null " +
                "group by genre,dvdgenre.title, dvdgenre.releaseYear";
        return sql;
    }
    public static void executeDVDAvailableInEachGenre( ResultSet res) throws SQLException {
        System.out.println("genre         title         releaseYear");
        while (res.next()) {
            String genre = res.getString("genre");
            String title = res.getString("title");
            String releaseYear = res.getString("releaseYear").substring(0, 4);
            System.out.println(genre + "      " + title + "      " + releaseYear);
        }
    }
    public static String getBalance() {
        //3. select the account balance for each member
        String sql = "select membershipNum,balance " +
                "from member";
        return sql;
    }
    public static void executeGetBalance( ResultSet res) throws SQLException {
        System.out.println("membershipNum         balance");
        while (res.next()) {
            String membershipNum = res.getString("membershipNum");
            String balance = res.getString("balance");
            System.out.println(membershipNum + "      " + balance);
        }
    }
    public static String getRecordsOf2018() {
        //4. generate the rental records of all members in 2018
        String sql = "select membershipNum,rentDate,returnDate,copyID " +
                "from operate " +
                "where rentDate between '2018-1-1' and '2018-12-31' " +
                "order by membershipNum";
        return sql;
    }
    public static void executeRecordsOf2018( ResultSet res) throws SQLException {
        System.out.println("membershipNum        rentDate        returnDate        copyID");
        while (res.next()) {
            String membershipNum = res.getString("membershipNum");
            String rentDate = res.getString("rentDate");
            String returnDate = res.getString("returnDate");
            String copyID = res.getString("copyID");
            System.out.println(membershipNum + "      " + rentDate + "      " + returnDate + "      " +
copyID);
        }
    }

    public static String getDVDNotReturned() {
        //5. generate the records of DVDs that haven't been returned
        String sql = "select * " +
                "from operate " +
                "where returnDate is null";
        return sql;
    }
    public static void executeDVDNotReturned( ResultSet res) throws SQLException {
        System.out.println("rentDate        returnDate        membershipNum        copyID");
        while (res.next()) {
            String rentDate = res.getString("rentDate");
            String returnDate = res.getString("returnDate");
            String membershipNum = res.getString("membershipNum");
            String copyID = res.getString("copyID");
            System.out.println(rentDate + "      " + returnDate + "      " + membershipNum + "      " +
copyID);
        }
    }
    public static String getAverageMoney() {
        //6. return the average money that normal members and premium members have spent
        String sql = "select category, round(avg(totalPay),2) as avgPaid " +
                "from member " +
                "group by category";
        return sql;
    }
    public static void executeAverageMoney( ResultSet res) throws SQLException {
        System.out.println("category        avgPaid");
        while (res.next()) {
            String category = res.getString("category");
            String avgPaid = res.getString("avgPaid");
            System.out.println(category + "      " + avgPaid + "      ");
        }
    }

    public static String getPotentialPremium() {
        //7. generate a list of normal members whose balance>=50rmb and have 3 or more
rental
        String sql = "select * from (select member.membershipNum, balance,
count(member.membershipNum) as rentAmount from member, operate where
member.membershipNum = operate.membershipNum group by member.membershipNum,
balance ) as rentAmountTable where rentAmountTable.balance>=50 and
rentAmountTable.rentAmount>=3";
        return sql;
    }
    public static void executePotentialPremium( ResultSet res) throws SQLException {
```

```
    System.out.println("membershipNum        balance        rentAmount");
    while (res.next()) {
        String membershipNum = res.getString("membershipNum");
        String balance = res.getString("balance");
        String rentAmount = res.getString("rentAmount");
        System.out.println(membershipNum + "      " + balance + "      " + rentAmount);
    }
}

public static void main (String[]args){
        Scanner input = new Scanner(System.in);
        Scanner DVDid = new Scanner(System.in);
        Scanner libraryName=new Scanner(System.in);
        Scanner membershipNumber=new Scanner(System.in);
        Date rent_Date = null;
        String category = null;
        Double totalValue = null;
        Double totalPaid=null;
        Double yourBalance=null;
        int days=0;
        String sql;
        int update;
        String id;
        String library;
        ResultSet res = null;
        try {
            try {
                Class.forName("com.mysql.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                System.out.println("Driver could not be loaded");
                System.exit(0);
            }
            String url = "jdbc:mysql://localhost:3306/library?useSSL=false";
```

```
 String user = "root";
 String password = "cgp5226926+123";
 Connection conn = DriverManager.getConnection(url, user, password);
 Statement statement = conn.createStatement();

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");
 String currentDate=df.format(System.currentTimeMillis());
 System.out.println(currentDate);
 System.out.println("welcome to the library!");
 System.out.println("please input your membership number:");
 int mNum=membershipNumber.nextInt();

 System.out.println("menu\n" +
         "0. charge balance"+
         "1. rent DVD\n" +
         "2. return DVD\n" +
         "3. DVD available\n" +
         "4. DVDs available in each genre\n"+
 "5. balance for each member\n"+
 "6. rental records of members in 2018\n"+
 "7. NOT returned records of DVDs \n"+
 "8. average cost of member\n"+
 "9. potential premium\n"+
 "10. exit");
     while(true){
         System.out.println("your manipulation");
 int a= input.nextInt();
 switch (a){
     case 0:
         //charge your balance
         System.out.println("How much do you want to charge?");
         Scanner charge=new Scanner(System.in);
         Double money=charge.nextDouble();
```

```
                sql="UPDATE member SET balance=balance+"+"'"+money+"'"+"WHERE
membershipNum=" + "'" + mNum + "'";
                update=statement.executeUpdate(sql);
                break;
            case 1:
                //rent DVD from the library
                System.out.println("choose the book you want to rent");
                id="'"+DVDid.next()+"'";
                sql = "UPDATE dvdcopy SET lName = NULL WHERE copyID="+id;
                update =statement.executeUpdate(sql);

                sql="INSERT  INTO  operate (copyID, membershipNum, returnDate,
rentDate)values ("+id+","+"'"+mNum+"'"+", NULL ,"+"'"+currentDate+"'"+")";
                update=statement.executeUpdate(sql);
                break;
            case 2:
                //return your book at library
                System.out.println("please input the DVD's copyID that you want to return");
                id="'"+DVDid.next()+"'";
                System.out.println("please input the library you are in");
                library="'"+libraryName.next()+"'";
                sql="UPDATE dvdcopy SET lName="+library+" WHERE copyID="+id;
                update=statement.executeUpdate(sql);
                sql="SELECT rentDate FROM operate WHERE copyID="+id;
                res=statement.executeQuery(sql);
                while (res.next()) {
                    rent_Date=res.getDate("rentDate");
                }
                sql="UPDATE operate SET returnDate="+"'"+currentDate+"'"+"WHERE
copyID="+id+"AND membershipNum="+"'"+mNum+"'"+"AND rentDate="+"'"+rent_Date+"'";
                update=statement.executeUpdate(sql);
                sql="SELECT DATEDIFF(returnDate,rentDate) AS days FROM  operate
WHERE copyID="+id+"AND membershipNum="+"'"+mNum+"'"+"AND
```

```
rentDate="+"'"+rent_Date+"'";
                        res=statement.executeQuery(sql);
                        while (res.next()) {
                            days=res.getInt("days");
                        }
                        sql = "SELECT category,totalVal,totalPay,balance FROM member WHERE
membershipNum=" + "'" + mNum + "'";
                        res=statement.executeQuery(sql);
                        while (res.next()) {
                            category=res.getString("category");
                            totalValue=res.getDouble("totalVal");
                            totalPaid=res.getDouble("totalPay");
                            yourBalance=res.getDouble("balance");
                            System.out.println(category+totalValue);
                        }
                        if (category.equals("normal")){
                            totalValue=totalValue+5*days;
                            totalPaid=totalPaid+5*days;
                            yourBalance=yourBalance-5*days;
                        }else{
                            totalValue=totalValue+3*days;
                            totalPaid=totalPaid+3*days;
                            yourBalance=yourBalance-3*days;
                        }

                        sql="UPDATE member SET totalVal="+"'"+totalValue+"'"+",
totalPay="+"'"+totalPaid+"'"+", balance="+"'"+yourBalance+"'"+"WHERE
membershipNum="+"'"+mNum+"'";
                        update=statement.executeUpdate(sql);
                        break;
                    case 3:
                        //3.search for the DVDs that can be rented by the member.
                        //you can see the DVD available at each library
```

```
    sql = getDVDAvailable();
    res = statement.executeQuery(sql);
    executeDVDAvailable(res);
    break;
case 4:
    //search for the DVD that is available group by genre
    sql=getDVDAvailableInEachGenre();
    res=statement.executeQuery(sql);
    executeDVDAvailableInEachGenre(res);
    break;
case 5:
    //search for the balace of every member
    sql=getBalance();
    res=statement.executeQuery(sql);
    executeGetBalance(res);
    break;
case 6:
    //search for the record of every member this year
    sql=getRecordsOf2018();
    res=statement.executeQuery(sql);
    executeRecordsOf2018(res);
     break;
case 7:
    //search for the DVDs that are rented by members
    sql=getDVDNotReturned();
    res=statement.executeQuery(sql);
    executeDVDNotReturned(res);
    break;
case 8:
    //search for the average fee for the premium member
    sql=getAverageMoney();
    res=statement.executeQuery(sql);
    executeAverageMoney(res);
```

```
                break;
            case 9:
                //generate a list of normal members whose balance>=50rmb and have 3 or
more rental
                sql=getPotentialPremium();
                res=statement.executeQuery(sql);
                executePotentialPremium(res);
            case 10:
                //exit
                System.exit(1);
                break;
            default:
                System.out.println("input error, input again!");
                break;

        }}

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if (res != null) {
            try {
                res.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

}
```