College Library Database System Implementation and Security

ECS740P – Database Systems

Full name

Nahid Topalovic

Introduction

The aim of the following report is to implement the database designed in the first part of this coursework using SQL. This will be fulfilled in the following ways:

- implementing the relational database schema by using ORACLE SQL;
- populating the database with a set of typical data;
- defining specialised views which are appropriate to various sub-groups of users;
- defining SQL queries which could be used as canned queries for naive users;
- creating triggers; and
- discussing security considerations and compliance with the Data Protection Act 2018.

Relational Database Schema

The conceptual relation schema from coursework one can be found below. Underlined attributes represent the primary key whereas attributes in italics represent a foreign key.

Library (<u>LibraryID</u>, LibraryName, Address)
Floor (<u>FloorNo</u>, *LibraryID*)

Shelf (<u>ShelfNo</u>, *FloorNo*)
Resource(<u>ResourceID</u>, *CourseNo, ShelfNo*, ResourceType, ResourceName, NumOfCopies, LoanPeriod, TotalNumOfLoans, Author)

Course (<u>CourseNo</u>, CourseName)
Loan (<u>*CardID, ResourceID, LoanDate*</u>, DueDate, IsOverdue)

Fine (<u>*CardID, ResourceID, LoanDate*</u>, FineAmount, IsPaid, PayDate)

LibraryCard (<u>CardID</u>, *LibraryID*, Name, Email, IssueDate, IsSuspended, NumOfResourcesLoaned)

StudentCard (<u>*CardID*</u>, StudentID)

StaffCard (<u>*CardID*</u>, StaffID)

Several changes were made in our relational database schema from the coursework one solution. We adapted the relations by considering the feedback on the coursework one solution and new ideas that came up during the implementation of the database.

The Library entity was removed since the specification for coursework did not request that our database must provide functionality for multiple libraries, therefore it was deemed unnecessary. Along with the Library entity, the Floor and Shelf entities were removed because they did not add any meaningful functionality, and hence they were replaced with the Location attribute of the Resource relation that uses a code to indicate the location of a resource. For example, a resource with a code F3S8 is located on Shelf 8 (S8) of the third floor (F3).

The Resource relation was renamed to LResource because such relation is already used by Oracle. Since the NumOfCopies and TotalNumOfLoans are derived attributes, they have been removed from the Resource table and stored in corresponding views instead. Another attribute called ResourceTypeID was added to this table, which is a foreign key from the ResourceType table that lists types of resources.

Next, the previous composite primary key consisting of CardID, ResourceID and LoanDate was replaced by the LoanID primary key in a bid to simplify implementation. The derived attributes DueDate and IsOverdue were removed from the schema and added to the views, while the Returndate attribute was added to the Loan relation for calculation of fines. Compared to the previous iteration, the Fine relation now uses the foreign key LoanID from the Loan table. The StudentCard and StaffCard specializations of the LibraryCard entity were replaced by the MemberTypeID attribute, which is a foreign key to the MemberType table that differentiates between the types of members such as students and staff. Also, the attribute NumOfResourcesLoaned was removed from the relation since it is a derivable attribute, which was calculated in the views.

After tidying up all the changes mentioned above, we have come up with the finalized relational schema as presented below:

LResource (ResourceID, *CourseNo, ResourceTypeID*, Location*,* ResourceName, NumOfCopies, LoanPeriod, Author)

ResourceType (ResourceTypeID, ResourceType)

Course (CourseNo, CourseName)
Loan (LoanID*, CardID, ResourceID,* LoanDate, ReturnDate)

Fine (*LoanID*, FineAmount, IsPaid, PayDate)

LibraryCard (CardID, *MemberTypeID,* Firstname, Lastname, Email, IssueDate, IsSuspended)

MemberType (MemberTypeID, MemberType)

Creating Tables

In this section, the SQL statements adopted to create the tables within the database will be outlined, and the description of the constraints imposed will also be presented.

Course

Within the Course table, CourseNo is given the primary key constraint. It implies that the value CourseNo is considered both unique and not null, while the attribute CourseName has a unique constraint.

The string datatypes noted for the attributes have been set with a character limit, such as thirty for CourseNo and eighty for CourseName. Therefore, it indicates that any details within the table must not exceed the characters limit.

```
CREATE TABLE COURSE (
   COURSENO VARCHAR2(30),
   COURSENAME VARCHAR2(80) NOT NULL,
   CONSTRAINT PK_COURSE PRIMARY KEY (COURSENO),
   CONSTRAINT UK_COURSE_COURSENAME UNIQUE(COURSENAME)
   );
```

ResourceType

Within the ResourceType table, ResourceTypeID is given the primary key constraint, while ResourceType attribute must not be a null value.

```
CREATE TABLE RESOURCETYPE (
   RESOURCETYPEID NUMBER(2),
   RESOURCETYPE VARCHAR2(30) NOT NULL,
   CONSTRAINT PK_RESOURCETYPE PRIMARY KEY (RESOURCETYPEID)
   );
```

LResource

Within the LResource table, ResourceID is given the primary key constraint. Two foreign key constraints have been defined for both the CourseNo and ResourceTypeID attributes. The ResourceName, LoanPeriod and Author attributes have NOT NULL constraint, because that information must be included upon insertion of new resources.

```
CREATE TABLE LRESOURCE (
   RESOURCEID NUMBER(10),
   COURSENO VARCHAR2(30),
   LOCATION VARCHAR2(100),
   RESOURCETYPEID NUMBER(2),
   RESOURCENAME VARCHAR2(100) NOT NULL,
   LOANPERIOD NUMBER(2) NOT NULL,
   AUTHOR VARCHAR2(100) NOT NULL,
   CONSTRAINT PK_RESOURCE PRIMARY KEY(RESOURCEID),
   CONSTRAINT FK_RESOURCE_COURSENO FOREIGN KEY(COURSENO)
      REFERENCES COURSE,
   CONSTRAINT FK_RESOURCE_RESOURCETYPE FOREIGN KEY(RESOURCETYPEID)
      REFERENCES RESOURCETYPE
   );
```

MemberType

Within the MemberType table, MemberTypeID is given the primary key constraint and the attribute Membertype has a NOT NULL constraint.

```
CREATE TABLE MEMBERTYPE (
   MEMBERTYPEID NUMBER(1),
   MEMBERTYPE VARCHAR2(10) NOT NULL,
   CONSTRAINT PK_MEMBERTYPE PRIMARY KEY(MEMBERTYPEID)
   );
```

LibraryCard

Within the LibraryCard table, CardID is given the primary key constraint. The attribute MembertypeID has a foreign key constraint with reference to the MemberType relation. Firstname, Lastname, IssueDate and Email have been assigned a NOT NULL constraint, since this information is necessary for the registration of a new library card user. The attribute IsSuspended has a default value of 0 and it has a check constraint that ensures that the value of that attribute is either 0 or 1—where 0 means that the user is not suspended, while 1 indicates the suspended status. The Email attribute is assigned two check constraints that ensure a value contains both the “@” and “dot” characters.

```
CREATE TABLE LIBRARYCARD (
   CARDID NUMBER(15),
   FIRSTNAME VARCHAR2(50) NOT NULL,
   LASTNAME VARCHAR2(50) NOT NULL,
   ISSUEDATE DATE NOT NULL,
   EMAIL VARCHAR2(100) NOT NULL,
   ISSUSPENDED NUMBER(1) DEFAULT 0,
   MEMBERTYPEID NUMBER(1),
   CONSTRAINT PK_LIBRARYCARD PRIMARY KEY(CARDID),
   CONSTRAINT FK_LIBRARYCARD_MEMBERTYPEID FOREIGN KEY(MEMBERTYPEID)
      REFERENCES MEMBERTYPE,
   CHECK (EMAIL LIKE '%@%'),
   CHECK (EMAIL LIKE '%.%'),
   CHECK (ISSUSPENDED IN (0,1))
   );
```

Loan

Within the Loan table, LoanID is given the primary key constraint, while CardID and ResourceID are given the foreign key constraint. The LoanDate attribute has a NOT NULL constraint, whereas ReturnDate can have a null value, which means that a resource has not yet been returned.

```
CREATE TABLE LOAN (
  LOANID NUMBER(10),
  CARDID NUMBER(15),
  RESOURCEID NUMBER(10),
  LOANDATE DATE NOT NULL,
  RETURNDATE DATE,
  CONSTRAINT PK_LOAN PRIMARY KEY (LOANID),
  CONSTRAINT FK_LOAN_CARDID FOREIGN KEY (CARDID)
    REFERENCES LIBRARYCARD,
  CONSTRAINT FK_LOAN_RESOURCEID FOREIGN KEY (RESOURCEID)
    REFERENCES LRESOURCE
  );
```

Fine

Within the Fine table, LoanID has a unique, foreign key constraint with reference to the Loan relation. The IsPaid attribute can be either 0 or 1 where 1 means that a fine is paid. This behaviour is ensured by a check constraint on the IsPaid attribute. FineAmount is also stored in the Fine table even though it is a derivable attribute. It is essential to store this information in the Fine table and its validation is performed by a trigger that will be explained in the Triggers section.

```
CREATE TABLE FINE (
  LOANID NUMBER(10),
  ISPAID NUMBER(1),
  PAYDATE DATE,
  FINEAMOUNT NUMBER(10),
  CONSTRAINT FK_FINE_LOANID FOREIGN KEY (LOANID)
    REFERENCES LOAN,
  CONSTRAINT UK_FINE_LOANID UNIQUE (LOANID),
  CHECK (ISPAID IN (0,1))
  );
```

Triggers

Triggers listed below will allow us to implement business rules and perform validation in the system.

The first trigger inserts the primary key value generated in a sequence (resource_sequence) for the tuple in LResource table. This allows users to insert resources in the table without keeping track of resource IDs.

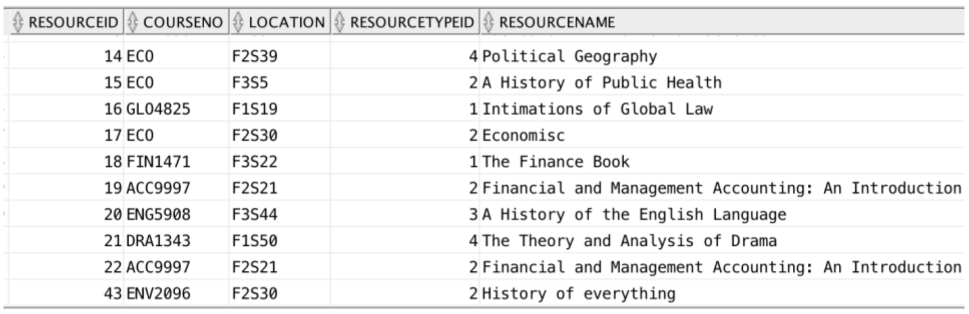
```
CREATE SEQUENCE resource_sequence
START WITH 1;

CREATE OR REPLACE TRIGGER resource_on_insert
BEFORE INSERT ON LRESOURCE
FOR EACH ROW
BEGIN
:NEW.RESOURCEID := resource_sequence.nextval;
END;
/
```

Demonstration:
A new resource item was inserted with the following SQL command:

```
INSERT INTO LResource (COURSENO, LOCATION, RESOURCETYPEID, RESOURCENAME,
LOANPERIOD, AUTHOR) VALUES
('ENV2096', 'F2S30', 2, 'History of everything', 5, 'Prof Pauline Weetman');
```

After querying for all results in the LResource table, it can be observed that a new resource item has been added with the ResourceID value of 43. The reason for this value (rather than 23) is that, during the implementation, the system was tested with different inputs; consequently, the values between 23 and 42 have already been used.



RESOURCEID	COURSENO	LOCATION	RESOURCETYPEID	RESOURCENAME
14	ECO	F2S39	4	Political Geography
15	ECO	F3S5	2	A History of Public Health
16	GL04825	F1S19	1	Intimations of Global Law
17	ECO	F2S30	2	Economisc
18	FIN1471	F3S22	1	The Finance Book
19	ACC9997	F2S21	2	Financial and Management Accounting: An Introduction
20	ENG5908	F3S44	3	A History of the English Language
21	DRA1343	F1S50	4	The Theory and Analysis of Drama
22	ACC9997	F2S21	2	Financial and Management Accounting: An Introduction
43	ENV2096	F2S30	2	History of everything

The next trigger performs validation of the fine amount value before an insertion in the Fine table. This attribute is derived but is important to have it as a stored value in the system. In the trigger, a variable VALIDAMOUNT is declared, where the correct fine amount is stored. The Fine amount is calculated as a difference between the return date and the loan date plus the loan period (the detailed explanation of how the fine amount is calculated will be given under the Views section). By using the IF statement, the inserted value is compared to the calculated value stored in the variable VALIDAMOUNT. If there is a disparity between the two values, the application will flag an error.

```
CREATE OR REPLACE TRIGGER fineamount_check_on_insert
BEFORE INSERT ON FINE
FOR EACH ROW
DECLARE
VALIDAMOUNT NUMBER(10);
BEGIN
SELECT ROUND((L.RETURNDATE - L.LOANDATE - R.LOANPERIOD), 3) INTO VALIDAMOUNT
FROM LOAN L
INNER JOIN LRESOURCE R ON L.RESOURCEID = R.RESOURCEID
WHERE L.LOANID = :NEW.LOANID;
IF :NEW.FINEAMOUNT != VALIDAMOUNT
THEN
raise_application_error(-20100, 'Fine amount does not match the dates');
END IF;
END;
/
```

To perform a demonstration of this trigger, we will try to insert an incorrect fine amount with the following insert statement.

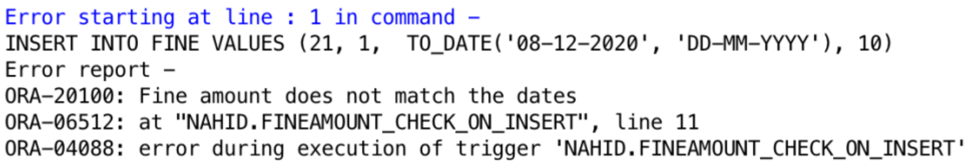
First, we insert a loan that is overdue:

```
INSERT INTO LOAN VALUES
(21, 935012719, 7, TO_DATE('10-11-2020', 'DD-MM-YYYY'), TO_DATE('08-12-2020', 'DD-MM-YYYY'));
```

Then, we try to insert a fine amount of $10 for this loan:

```
INSERT INTO FINE VALUES (21, 1, TO_DATE('08-12-2020', 'DD-MM-YYYY'), 10);
```

This results in an error:



```
Error starting at line : 1 in command -
INSERT INTO FINE VALUES (21, 1,  TO_DATE('08-12-2020', 'DD-MM-YYYY'), 10)
Error report -
ORA-20100: Fine amount does not match the dates
ORA-06512: at "NAHID.FINEAMOUNT_CHECK_ON_INSERT", line 11
ORA-04088: error during execution of trigger 'NAHID.FINEAMOUNT_CHECK_ON_INSERT'
```

But when we insert a correct value, the trigger allows it to run through.

```
INSERT INTO FINE VALUES (21, 1,  TO_DATE('08-12-2020', 'DD-MM-YYYY'), 21);
```

The third trigger prevents an insertion of new loans if the number of current loans for a user exceeds the allowed quota. Student users can loan up to five resources, whereas staff users can borrow a maximum number of ten resources.

```
CREATE OR REPLACE TRIGGER limit_loans
BEFORE INSERT ON LOAN
FOR EACH ROW
DECLARE
NUMOFLOANS NUMBER(5);
LOANSLIMIT NUMBER(5);
BEGIN
SELECT (CASE WHEN LC.MEMBERTYPEID = 0 THEN 5 ELSE 10 END) INTO LOANSLIMIT
FROM LIBRARYCARD LC
WHERE LC.CARDID=:NEW.CARDID;
SELECT COUNT(*) INTO NUMOFLOANS
FROM LOAN
WHERE LOAN.RETURNDATE IS NULL AND LOAN.CARDID = :NEW.CARDID;

IF NUMOFLOANS >= LOANSLIMIT
THEN
raise_application_error(-20100, 'Number of loans exceeded');
END IF;
END;
/
```

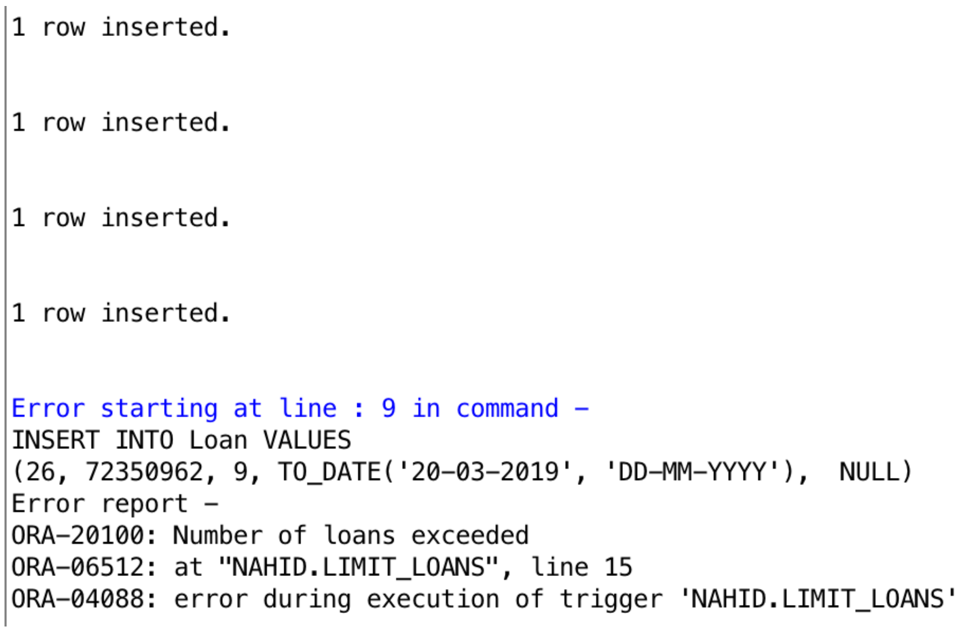
In the first part of the trigger, we declared two variables: NUMOFLOANS to store the current number of loans for a user; and LOANSLIMIT to store the loan limit of a user.

The first query stores the limit of a user using a case statement to check whether a user is a student or a staff member. Then the number of current loans (where the return date is set to null) is stored to the second variable. These two values are subsequently compared. If NUMOFLOANS is bigger or equal to the LOANSLIMIT value, the trigger will raise an application error.

To test this trigger, we will add five more loan items for a student user with the CardID 72350962. This user has already had a loan item in his/her record, so the trigger should expectedly forbid the insertion of the last loan.

```
INSERT INTO Loan VALUES
(22, 72350962, 5, TO_DATE('20-03-2019', 'DD-MM-YYYY'),  NULL);
INSERT INTO Loan VALUES
(23, 72350962, 6, TO_DATE('20-03-2019', 'DD-MM-YYYY'),  NULL);
INSERT INTO Loan VALUES
(24, 72350962, 7, TO_DATE('20-03-2019', 'DD-MM-YYYY'),  NULL);
INSERT INTO Loan VALUES
(25, 72350962, 8, TO_DATE('20-03-2019', 'DD-MM-YYYY'),  NULL);
INSERT INTO Loan VALUES
(26, 72350962, 9, TO_DATE('20-03-2019', 'DD-MM-YYYY'),  NULL);
```

The result is shown in the following figure, where the trigger works as intended.



```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


Error starting at line : 9 in command -
INSERT INTO Loan VALUES
(26, 72350962, 9, TO_DATE('20-03-2019', 'DD-MM-YYYY'),  NULL)
Error report -
ORA-20100: Number of loans exceeded
ORA-06512: at "NAHID.LIMIT_LOANS", line 15
ORA-04088: error during execution of trigger 'NAHID.LIMIT_LOANS'
```

The fourth trigger suspends a user after an update of the return date in a Loan table. The trigger checks if the user has an outstanding fine amount that exceeds $10; if they do, they will automatically be suspended.

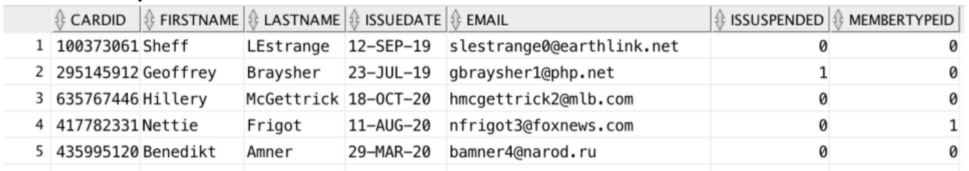
```
CREATE OR REPLACE TRIGGER suspend_check
AFTER UPDATE OF RETURNDATE ON LOAN
FOR EACH ROW
DECLARE
FINEAMOUNT NUMBER(10);
LIBCARDID NUMBER(20);
BEGIN
SELECT ROUND((:NEW.RETURNDATE - :OLD.LOANDATE - R.LOANPERIOD), 3) INTO
FINEAMOUNT
FROM LRESOURCE R
WHERE :OLD.RESOURCEID = R.RESOURCEID;
LIBCARDID := :OLD.CARDID;

IF FINEAMOUNT > 10
THEN
UPDATE LIBRARYCARD SET ISSUSPENDED = 1
WHERE CARDID = LIBCARDID;
END IF;
END;
/
```

In the trigger, we declared two variables—FINEAMOUNT and LIBCARDID. The calculated fine amount for a particular loan is stored in the first variable, while the second variable accommodates the CardID of the user possessing that loan.

When the fine amount is larger than $10, the update statement is executed where we set the IsSuspended attribute value to 1 in a LibraryCard table.

The following example is used to test this trigger.
The LibraryCard table before insertion:



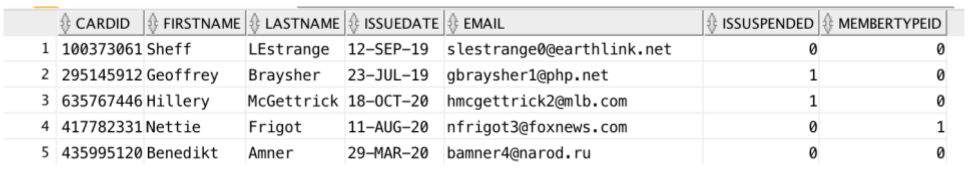
	CARDID	FIRSTNAME	LASTNAME	ISSUEDATE	EMAIL	ISSUSPENDED	MEMBERTYPEID
1	100373061	Sheff	LEstrange	12-SEP-19	slestrange0@earthlink.net	0	0
2	295145912	Geoffrey	Braysher	23-JUL-19	gbraysher1@php.net	1	0
3	635767446	Hillery	McGettrick	18-OCT-20	hmcgettrick2@mlb.com	0	0
4	417782331	Nettie	Frigot	11-AUG-20	nfrigot3@foxnews.com	0	1
5	435995120	Benedikt	Amner	29-MAR-20	bamner4@narod.ru	0	0

A new loan is inserted into the record of a user with the CardID 635767446 who does not have a “suspended” status:

```
INSERT INTO Loan VALUES
(28, 635767446, 4, TO_DATE('20-04-2019', 'DD-MM-YYYY'),  NULL);
```

Then, to find out whether the user will be effectively suspended, a return date is updated to deliberately create a late return condition, which entails a fine amount that exceeds $10 for this loan item.

```
UPDATE Loan set RETURNDATE = TO_DATE('20-05-2019', 'DD-MM-YYYY')
WHERE LOANID = 28;
```



	CARDID	FIRSTNAME	LASTNAME	ISSUEDATE	EMAIL	ISSUSPENDED	MEMBERTYPEID
1	100373061	Sheff	LEstrange	12-SEP-19	slestrange0@earthlink.net	0	0
2	295145912	Geoffrey	Braysher	23-JUL-19	gbraysher1@php.net	1	0
3	635767446	Hillery	McGettrick	18-OCT-20	hmcgettrick2@mlb.com	1	0
4	417782331	Nettie	Frigot	11-AUG-20	nfrigot3@foxnews.com	0	1
5	435995120	Benedikt	Amner	29-MAR-20	bamner4@narod.ru	0	0

As a result, we can see that the user has been suspended, as reflected by the IsSuspended attribute value “1”.

Data

This section comprises a set of sample data that we used to test the stability of our database.

Course

The following data will be the sample data used for the Course table.

CourseNo	CourseName
BIO2476	Biomedical Science
CLI2782	Clinical Psychology
MEC5903	Mechanical Engineering
COM4408	Computer Science
GLO4825	Global Law

ECO7910	Economics
FIN1471	Finance
ACC9997	Accounting and Management
ENG5908	English Language
DRA1343	Drama
ROB7138	Robotics Engineering
SOF7257	Software Engineering
ENV2096	Environmental Science

ResourceType

The following data will be the sample data used for the ResourceType table.

ResourceTypeID	ResourceType
1	Book
2	Video
3	DVD
4	CD

LResource

The following data will be the sample data used for the LResource table.

CourseNo	Location	ResourceTypeID	ResourceName	LoanPeriod	Author
BIO2476	F3S20	1	Cell Structure & Function	7	Guy Orchard & Brian Nation
CLI2782	F2S14	2	What is Clinical Psychology?	5	Susan P. Llewelyn & David J. Murphy
MEC5903	F3S17	3	Basic Mechanical Engineering	2	Mohan Sen
COM4408	F1S7	1	Learn Python 3 the Hard Way	7	Zed A. Shaw
GLO4825	F1S19	1	Intimations of Global Law	7	Neil Walker
ECO7910	F2S30	2	Economics	5	Alain Anderton
FIN1471	F3S22	1	The Finance Book	7	Stuart Warner & Si Hussain
ACC9997	F2S21	2	Financial and Management Accounting: An Introduction	5	Prof Pauline Weetman
ENG5908	F3S44	3	A History of the English Language	2	Elly van Gelderen
DRA1343	F1S50	4	The Theory and Analysis of Drama	3	Manfred Pfister
ROB7138	F1S33	1	Fundamentals of Robotics Engineering	7	Harry H. Poole
SOF7257	F2S47	1	Software Engineering and Quality Assurance	7	A.A. Puntambekar
ENV2096	F1S9	2	Basics of Environmental Science	5	Michael Allaby
ECO	F2S39	4	Political Geography	3	Ramesh Dutta Dikshit
ECO	F3S5	2	A History of Public Health	5	George Rosen, Gwen Rosen & Edward T. Morman
GLO4825	F1S19	1	Intimations of Global Law	7	Neil Walker

ECO	F2S30	2	Economics	5	Alain Anderton
FIN1471	F3S22	1	The Finance Book	7	Stuart Warner & Si Hussain
ACC9997	F2S21	2	Financial and Management Accounting: An Introduction	5	Prof Pauline Weetman
ENG5908	F3S44	3	A History of the English Language	2	Elly van Gelderen
DRA1343	F1S50	4	The Theory and Analysis of Drama	3	Manfred Pfister
ACC9997	F2S21	2	Financial and Management Accounting: An Introduction	5	Prof Pauline Weetman

MemberType

The following data will be the sample data used for the MemberType table.

MemberTypeID	**MemberType**
0	Student
1	Staff

LibraryCard

The following data will be the sample data used for the Library Card table.

CardID	Firstname	Lastname	Email	IssueDate	IsSuspended	MemberTypeID
100373061	Sheff	LEstrange	slestrange0@earthlink.net	12-09-2019	0	0
295145912	Geoffrey	Braysher	gbraysher1@php.net	23-07-2019	0	0
635767446	Hillery	McGettrick	Hmcgettrick2@mlb.com	18-10-2020	0	0
417782331	Nettie	Frigot	Nfrigot3@foxnews.com	11-08-2020	0	1
435995120	Benedikt	Amner	Bamner4@narod.ru	29-03-2020	0	0
297382959	Hogan	Darleston	Hdarleston5@discovery.com	25-05-2020	0	1
718693208	Flory	OCurran	Focurran6@netscape.com	22-11-2019	1	0
520284834	Mickey	De Vuyst	Mdevuyst7@umn.edu	28-04-2019	1	0
935012719	Dougy	Ruuffle	Druffle8@umn.edu	05-08-2020	0	1
293155114	Gwenneth	Arrowsmith	Garrowsmith9@dailymail.co.uk	07-07-2020	0	1
492010041	Merissa	Sloyan	Lmsloyana@behance.net	01-09-2019	0	0
90496836	Sharlene	Oxlade	Soxladeb@gnu.org	13-02-2020	0	0
172352102	Salvatore	Weatherell	Sweatherellc@privacy.gov.au	05-08-2020	0	1
510427213	Jamal	Munning	Jmuuningd@eepurl.com	16-10-2019	0	0
72350962	Maribelle	Shoobridge	Mshoobridgee@china.com.cn	20-03-2020	0	0

Loan

The following data will be the sample data used for the Loan table.

LoanID	**CardID**	**ResourceID**	**LoanDate**	**ReturnDate**
1	100373061	1	12-09-2019	15-09-2019
2	295145912	2	23-07-2019	30-08-2019
3	635767446	3	18-07-2019	19-07-2019
4	417782331	4	11-08-2020	14-08-2020
5	435995120	5	29-03-2020	30-03-2020
6	297382959	6	25-05-2020	27-05-2020

7	718693208	7	22-11-2019	23-11-2019
8	520284834	8	28-04-2019	28-05-2019
9	935012719	9	05-08-2020	05-08-2020
10	293155114	10	07-07-2020	10-07-2020
11	492010041	11	01-09-2019	01-11-2019
12	90496836	12	13-02-2020	18-02-2020
13	172352102	13	05-08-2020	06-08-2020
14	510427213	14	16-10-2019	19-10-2019
15	72350962	15	20-03-2020	05-04-2020
16	72350962	5	20-03-2019	21-03-2019
17	435995120	7	29-03-2020	30-03-2020
18	435995120	5	30-03-2020	04-04-2020
19	935012719	10	06-12-2020	08-12-2020
20	72350962	12	15-11-2020	

Fine

The following data will be the sample data used for the Fine table.

LoanID	isPaid	PayDate	FineAmount
2	1	12-09-2020	33
8	1	10-06-2020	25
11	1	01-10-2020	24
15	1	12-05-2020	11
20	0		

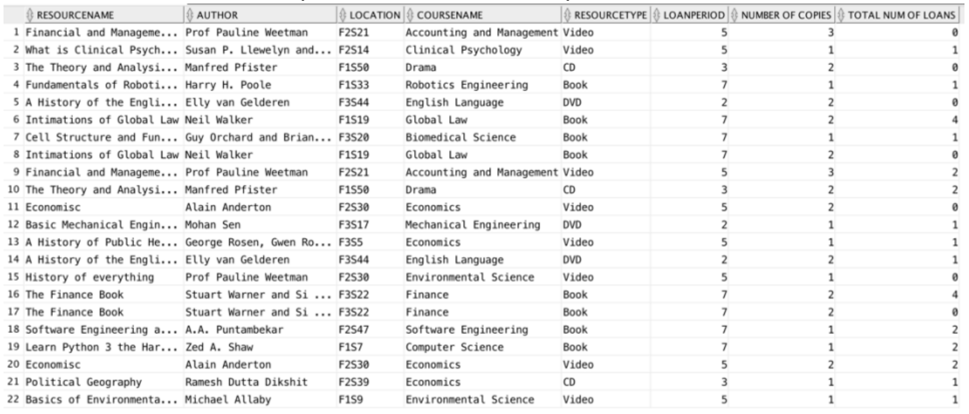
Views

Views allow us to represent complex or common queries for specific groups of users. This section will outline the various views created for our database system and their intended functions and outputs.

List of resources

```
CREATE VIEW LISTOFRESOURCES AS (
SELECT DISTINCT R.RESOURCENAME, R.AUTHOR, R.LOCATION, C.COURSENAME ,
RT.RESOURCETYPE, R.LOANPERIOD,
(SELECT COUNT(*) FROM LRESOURCE R2 WHERE R.RESOURCENAME=R2.RESOURCENAME)
"NUMBER OF COPIES",
(SELECT COUNT(*) FROM LOAN WHERE R.RESOURCEID = LOAN.RESOURCEID) "TOTAL NUM
OF LOANS"
FROM LRESOURCE R
INNER JOIN COURSE C ON C.COURSENO = R.COURSENO
INNER JOIN RESOURCETYPE RT ON RT.RESOURCETYPEID = R.RESOURCETYPEID
);
```

The view here lists resources from the Resource table, which comprises the resource name, author, location, course name, resource type and loan period columns, along with calculated attributes including the number of copies and the total amount of loans for each resource item. The output of this view is captured as follows:

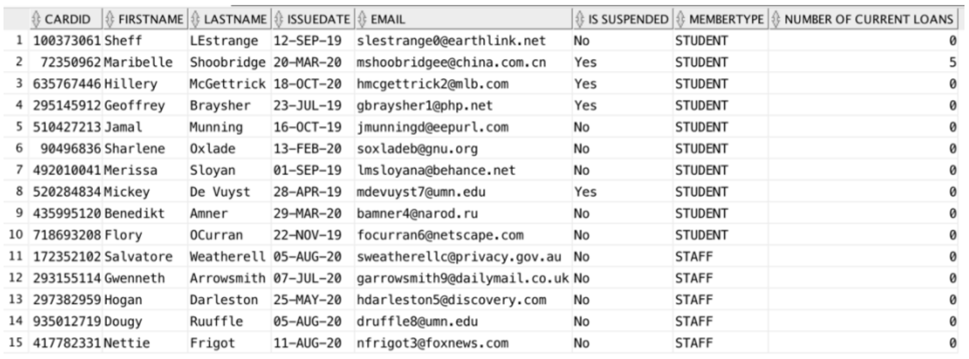


	RESOURCENAME	AUTHOR	LOCATION	COURSENAME	RESOURCETYPE	LOANPERIOD	NUMBER OF COPIES	TOTAL NUM OF LOANS
1	Financial and Manageme...	Prof Pauline Weetman	F2S21	Accounting and Management	Video	5	3	0
2	What is Clinical Psych...	Susan P. Llewelyn and...	F2S14	Clinical Psychology	Video	5	1	1
3	The Theory and Analysi...	Manfred Pfister	F1S50	Drama	CD	3	2	0
4	Fundamentals of Roboti...	Harry H. Poole	F1S33	Robotics Engineering	Book	7	1	1
5	A History of the Engli...	Elly van Gelderen	F3S44	English Language	DVD	2	2	0
6	Intimations of Global Law	Neil Walker	F1S19	Global Law	Book	7	2	4
7	Cell Structure and Fun...	Guy Orchard and Brian...	F3S20	Biomedical Science	Book	7	1	1
8	Intimations of Global Law	Neil Walker	F1S19	Global Law	Book	7	2	0
9	Financial and Manageme...	Prof Pauline Weetman	F2S21	Accounting and Management	Video	5	3	2
10	The Theory and Analysi...	Manfred Pfister	F1S50	Drama	CD	3	2	2
11	Economisc	Alain Anderton	F2S30	Economics	Video	5	2	0
12	Basic Mechanical Engin...	Mohan Sen	F3S17	Mechanical Engineering	DVD	2	1	1
13	A History of Public He...	George Rosen, Gwen Ro...	F3S5	Economics	Video	5	1	1
14	A History of the Engli...	Elly van Gelderen	F3S44	English Language	DVD	2	2	1
15	History of everything	Prof Pauline Weetman	F2S30	Environmental Science	Video	5	1	0
16	The Finance Book	Stuart Warner and Si ...	F3S22	Finance	Book	7	2	4
17	The Finance Book	Stuart Warner and Si ...	F3S22	Finance	Book	7	2	0
18	Software Engineering a...	A.A. Puntambekar	F2S47	Software Engineering	Book	7	1	2
19	Learn Python 3 the Har...	Zed A. Shaw	F1S7	Computer Science	Book	7	1	2
20	Economisc	Alain Anderton	F2S30	Economics	Video	5	2	2
21	Political Geography	Ramesh Dutta Dikshit	F2S39	Economics	CD	3	1	1
22	Basics of Environmenta...	Michael Allaby	F1S9	Environmental Science	Video	5	1	1

List of students and staff

```
CREATE VIEW LISTOF_STUDENTSANDSTAFF AS (SELECT CARDID, FIRSTNAME, LASTNAME,
ISSUEDATE, EMAIL,
(CASE WHEN ISSUSPENDED = 1 THEN 'Yes' ELSE 'No' END) "IS SUSPENDED",
MT.MEMBERTYPE,
(SELECT COUNT(*) FROM LOAN L WHERE L.CARDID = LIBRARYCARD.CARDID AND
L.RETURNDATE IS NULL) "NUMBER OF CURRENT LOANS"
FROM LIBRARYCARD, MEMBERTYPE MT
WHERE LIBRARYCARD.MEMBERTYPEID = MT.MEMBERTYPEID);
```

This view lists relevant information about students and staff: CardID, first and last names of the user, issue date of the library card, email, “suspended” status, member type and the calculated attribute – number of current loans. The output of this view is displayed below:

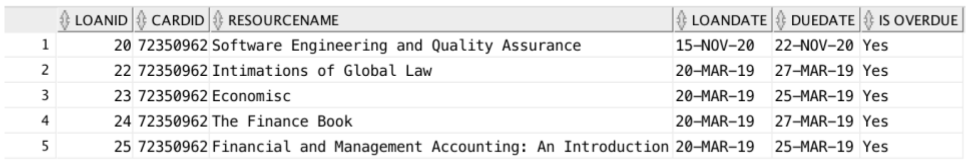


	CARDID	FIRSTNAME	LASTNAME	ISSUEDATE	EMAIL	IS SUSPENDED	MEMBERTYPE	NUMBER OF CURRENT LOANS
1	100373061	Sheff	LEstrange	12-SEP-19	slestrange0@earthlink.net	No	STUDENT	0
2	72350962	Maribelle	Shoobridge	20-MAR-20	mshoobridgee@china.com.cn	Yes	STUDENT	5
3	635767446	Hillery	McGettrick	18-OCT-20	hmcgettrick2@mlb.com	Yes	STUDENT	0
4	295145912	Geoffrey	Braysher	23-JUL-19	gbraysher1@php.net	Yes	STUDENT	0
5	510427213	Jamal	Munning	16-OCT-19	jmunningd@eepurl.com	No	STUDENT	0
6	90496836	Sharlene	Oxlade	13-FEB-20	soxladeb@gnu.org	No	STUDENT	0
7	492010041	Merissa	Sloyan	01-SEP-19	lmsloyana@behance.net	No	STUDENT	0
8	520284834	Mickey	De Vuyst	28-APR-19	mdevuyst7@umn.edu	Yes	STUDENT	0
9	435995120	Benedikt	Amner	29-MAR-20	bamner4@narod.ru	No	STUDENT	0
10	718693208	Flory	OCurran	22-NOV-19	focurran6@netscape.com	No	STUDENT	0
11	172352102	Salvatore	Weatherell	05-AUG-20	sweatherellc@privacy.gov.au	No	STAFF	0
12	293155114	Gwenneth	Arrowsmith	07-JUL-20	garrowsmith9@dailymail.co.uk	No	STAFF	0
13	297382959	Hogan	Darleston	25-MAY-20	hdarleston5@discovery.com	No	STAFF	0
14	935012719	Dougy	Ruuffle	05-AUG-20	druffle8@umn.edu	No	STAFF	0
15	417782331	Nettie	Frigot	11-AUG-20	nfrigot3@foxnews.com	No	STAFF	0

List of current loans

```
CREATE VIEW CURRENT_LOANS AS (SELECT LOANID, CARDID, R.RESOURCENAME, LOANDATE,
(LOANDATE + R.LOANPERIOD) "DUEDATE",
(CASE WHEN (LOANDATE + R.LOANPERIOD) < TO_DATE('01-DEC-2020', 'DD-MON-YYYY')
THEN 'Yes' ELSE 'No' END) "IS OVERDUE"
FROM LOAN, LRESOURCE R
WHERE LOAN.RESOURCEID = R.RESOURCEID AND LOAN.RETURNDATE IS NULL);
```

This view gives us information about current loans. By running this view, any loan records with the ReturnDate being a NULL value, which indicates that the resources have not been returned, will be grasped and presented with their corresponding LoanID, CardID, ResourceName, LoanDate, two calculated attributes DueDate and IsOverdue. We have set 01 December 2020 as SYSDATE, but in the real application it would be in sync with the current date. The output of this view is presented below:



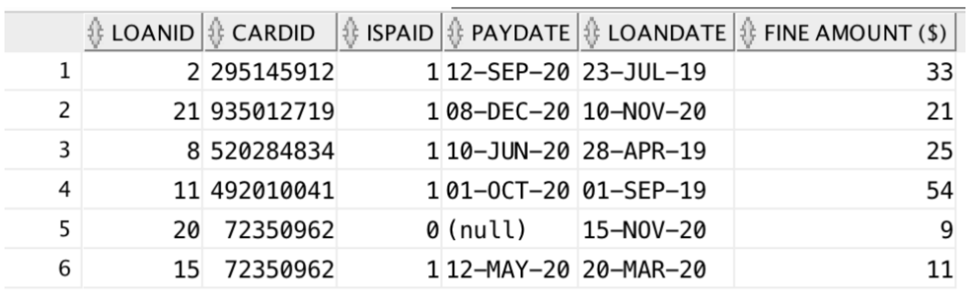
	LOANID	CARDID	RESOURCENAME	LOANDATE	DUEDATE	IS OVERDUE
1	20	72350962	Software Engineering and Quality Assurance	15-NOV-20	22-NOV-20	Yes
2	22	72350962	Intimations of Global Law	20-MAR-19	27-MAR-19	Yes
3	23	72350962	Economisc	20-MAR-19	25-MAR-19	Yes
4	24	72350962	The Finance Book	20-MAR-19	27-MAR-19	Yes
5	25	72350962	Financial and Management Accounting: An Introduction	20-MAR-19	25-MAR-19	Yes

Fines

```
CREATE VIEW FINES AS (SELECT F.LOANID, L.CARDID, F.ISPAID, F.PAYDATE, L.LOANDATE,
ROUND((CASE WHEN L.RETURNDATE IS NOT NULL THEN (L.RETURNDATE - L.LOANDATE -
R.LOANPERIOD)
ELSE (TO_DATE('01-DEC-2020', 'DD-MON-YYYY') - L.LOANDATE - R.LOANPERIOD) END),3)
"FINE AMOUNT ($)"
FROM FINE F
INNER JOIN LOAN L ON F.LOANID = L.LOANID
INNER JOIN LRESOURCE R ON L.RESOURCEID = R.RESOURCEID);
```

This view allows users to see a list of fines. This list includes attributes from the Fine table along with derived attributes such as the fine amount where its value is calculated by taking two scenarios into account. The first scenario involves a valid ReturnDate value (user has returned the resource), and the fine amount is calculated by considering how many days the ReturnDate has passed the required return date. The second scenario is where the ReturnDate has a null value; the fine amount is calculated from the difference between the required return date (as defined by the default loan period) and the SYSDATE (current date).

The output of this view is presented below:



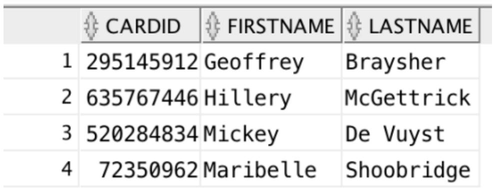
	LOANID	CARDID	ISPAID	PAYDATE	LOANDATE	FINE AMOUNT ($)
1	2	295145912	1	12-SEP-20	23-JUL-19	33
2	21	935012719	1	08-DEC-20	10-NOV-20	21
3	8	520284834	1	10-JUN-20	28-APR-19	25
4	11	492010041	1	01-OCT-20	01-SEP-19	54
5	20	72350962	0	(null)	15-NOV-20	9
6	15	72350962	1	12-MAY-20	20-MAR-20	11

<u>**Queries**</u>

The following section contains a list of queries relevant to the individuals or groups of users of the library system.

1. The following query returns a list of suspended students and staff:

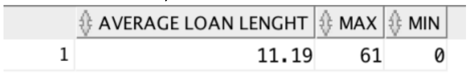
```
SELECT CARDID, FIRSTNAME, LASTNAME
FROM LIBRARYCARD
WHERE ISSUSPENDED = 1;
```



	CARDID	FIRSTNAME	LASTNAME
1	295145912	Geoffrey	Braysher
2	635767446	Hillery	McGettrick
3	520284834	Mickey	De Vuyst
4	72350962	Maribelle	Shoobridge

2. This query gives us information about the average, maximum and minimum loan lengths:

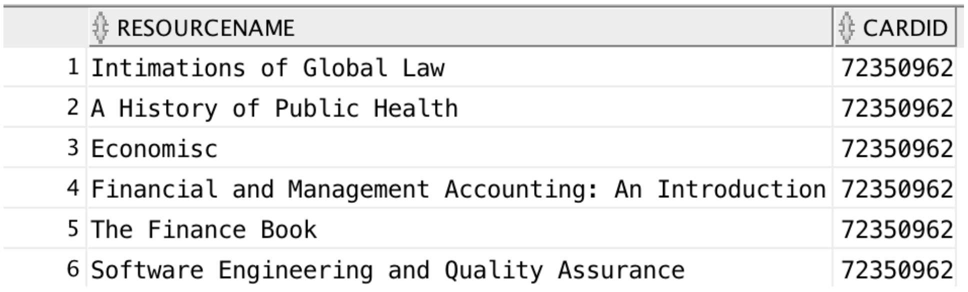
```
SELECT ROUND(AVG(RETURNDATE - LOANDATE),2) "AVERAGE LOAN LENGHT",
MAX(RETURNDATE - LOANDATE) MAX,
MIN(RETURNDATE - LOANDATE) MIN
FROM LOAN;
```



	AVERAGE LOAN LENGHT	MAX	MIN
1	11.19	61	0

3. This query lists resources that user A loaned and resources that both user A and user B loaned:

```
SELECT DISTINCT LR.RESOURCENAME, CARDID
FROM LOAN
INNER JOIN LRESOURCE LR ON LR.RESOURCEID = LOAN.RESOURCEID
LEFT OUTER JOIN
(SELECT LIBR.RESOURCENAME FROM LRESOURCE LIBR, LOAN
WHERE LIBR.RESOURCEID = LOAN.RESOURCEID
AND LOAN.CARDID = 435995120) LIBR ON LR.RESOURCENAME =
LIBR.RESOURCENAME
WHERE CARDID = 72350962;
```

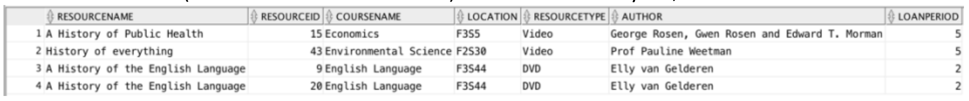


	RESOURCENAME	CARDID
1	Intimations of Global Law	72350962
2	A History of Public Health	72350962
3	Economisc	72350962
4	Financial and Management Accounting: An Introduction	72350962
5	The Finance Book	72350962
6	Software Engineering and Quality Assurance	72350962

4. With this query we can search for a resource containing certain title:

```
SELECT LR.RESOURCENAME, LR.RESOURCEID, C.COURSENAME, LOCATION,
RT.RESOURCETYPE, LR.AUTHOR, LR.LOANPERIOD
```

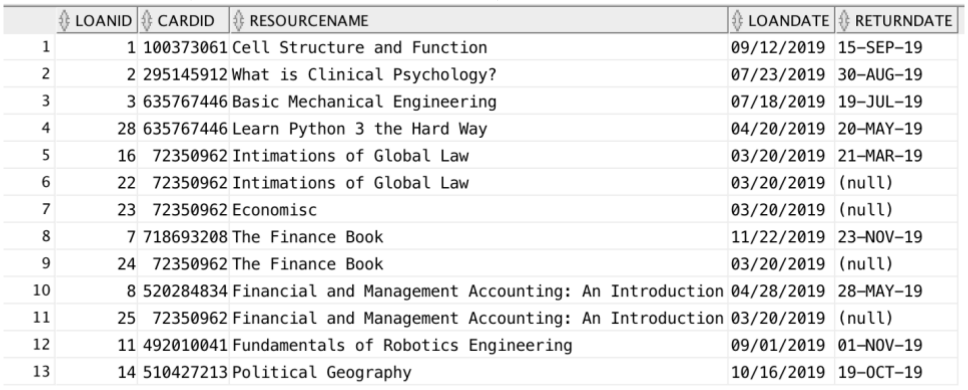
```
FROM LRESOURCE LR
INNER JOIN COURSE C ON LR.COURSENO = C.COURSENO
INNER JOIN RESOURCETYPE RT ON LR.RESOURCETYPEID = RT.RESOURCETYPEID
WHERE lower(LR.RESOURCENAME) LIKE '%history%';
```



	RESOURCENAME	RESOURCEID	COURSENAME	LOCATION	RESOURCETYPE	AUTHOR	LOANPERIOD
1	A History of Public Health	15	Economics	F3S5	Video	George Rosen, Gwen Rosen and Edward T. Morman	5
2	History of everything	43	Environmental Science	F2S30	Video	Prof Pauline Weetman	5
3	A History of the English Language	9	English Language	F3S44	DVD	Elly van Gelderen	2
4	A History of the English Language	20	English Language	F3S44	DVD	Elly van Gelderen	2

5. This query allows us to list all loans from the year of 2019:

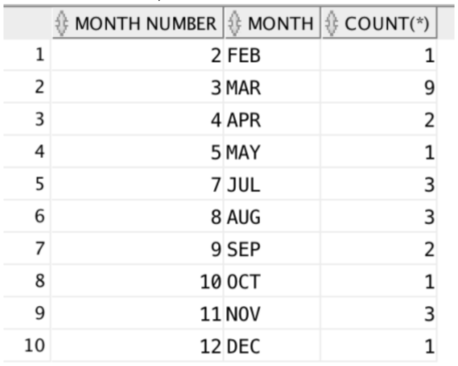
```
SELECT LOANID, CARDID, LR.RESOURCENAME, TO_CHAR(LOANDATE, 'MM/DD/YYYY')
LOANDATE, RETURNDATE
FROM LOAN, LRESOURCE LR
WHERE LOAN.RESOURCEID = LR.RESOURCEID
AND EXTRACT(YEAR FROM LOANDATE) = 2019;
```



	LOANID	CARDID	RESOURCENAME	LOANDATE	RETURNDATE
1	1	100373061	Cell Structure and Function	09/12/2019	15-SEP-19
2	2	295145912	What is Clinical Psychology?	07/23/2019	30-AUG-19
3	3	635767446	Basic Mechanical Engineering	07/18/2019	19-JUL-19
4	28	635767446	Learn Python 3 the Hard Way	04/20/2019	20-MAY-19
5	16	72350962	Intimations of Global Law	03/20/2019	21-MAR-19
6	22	72350962	Intimations of Global Law	03/20/2019	(null)
7	23	72350962	Economisc	03/20/2019	(null)
8	7	718693208	The Finance Book	11/22/2019	23-NOV-19
9	24	72350962	The Finance Book	03/20/2019	(null)
10	8	520284834	Financial and Management Accounting: An Introduction	04/28/2019	28-MAY-19
11	25	72350962	Financial and Management Accounting: An Introduction	03/20/2019	(null)
12	11	492010041	Fundamentals of Robotics Engineering	09/01/2019	01-NOV-19
13	14	510427213	Political Geography	10/16/2019	19-OCT-19

6. This query gives us the number of loans per each month:

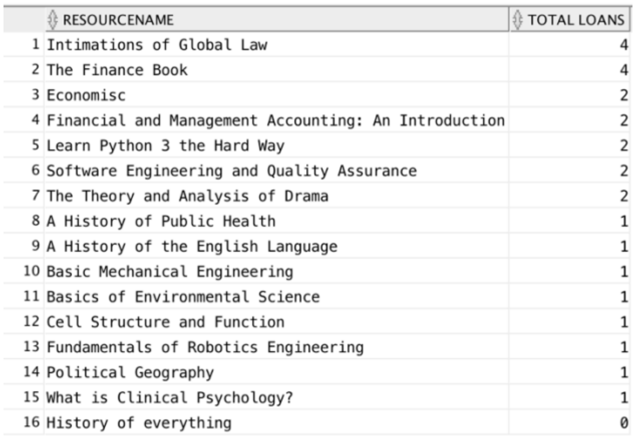
```
SELECT EXTRACT(MONTH FROM LOANDATE) "MONTH NUMBER",
TO_CHAR(LOANDATE,'MON') "MONTH", COUNT(*)
FROM LOAN
GROUP BY EXTRACT(MONTH FROM LOANDATE), TO_CHAR(LOANDATE,'MON')
ORDER BY 1;
```



	MONTH NUMBER	MONTH	COUNT(*)
1	2	FEB	1
2	3	MAR	9
3	4	APR	2
4	5	MAY	1
5	7	JUL	3
6	8	AUG	3
7	9	SEP	2
8	10	OCT	1
9	11	NOV	3
10	12	DEC	1

7. This query lists most popular resources in the descending order:

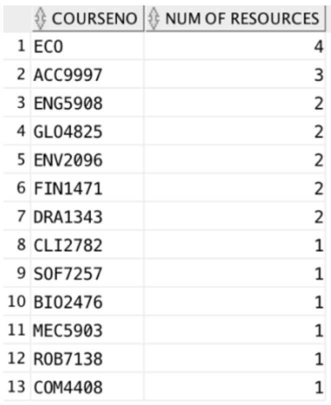
```
SELECT DISTINCT R.RESOURCENAME,
(SELECT COUNT(*) FROM LOAN WHERE LOAN.RESOURCEID
IN (SELECT RESOURCEID FROM LRESOURCE WHERE RESOURCENAME =
R.RESOURCENAME)
) "TOTAL LOANS"
FROM LRESOURCE R
ORDER BY 2 DESC;
```



	RESOURCENAME	TOTAL LOANS
1	Intimations of Global Law	4
2	The Finance Book	4
3	Economisc	2
4	Financial and Management Accounting: An Introduction	2
5	Learn Python 3 the Hard Way	2
6	Software Engineering and Quality Assurance	2
7	The Theory and Analysis of Drama	2
8	A History of Public Health	1
9	A History of the English Language	1
10	Basic Mechanical Engineering	1
11	Basics of Environmental Science	1
12	Cell Structure and Function	1
13	Fundamentals of Robotics Engineering	1
14	Political Geography	1
15	What is Clinical Psychology?	1
16	History of everything	0

8. This query lists the number of resources available for each course:

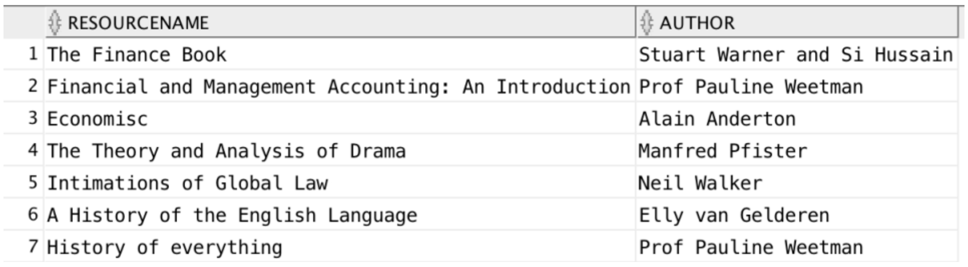
```
SELECT LR.COURSENO, COUNT(*) "NUM OF RESOURCES"
FROM LRESOURCE LR, COURSE C
WHERE LR.COURSENO = C.COURSENO
GROUP BY LR.COURSENO
ORDER BY 2 DESC;
```



	COURSENO	NUM OF RESOURCES
1	ECO	4
2	ACC9997	3
3	ENG5908	2
4	GLO4825	2
5	ENV2096	2
6	FIN1471	2
7	DRA1343	2
8	CLI2782	1
9	SOF7257	1
10	BIO2476	1
11	MEC5903	1
12	ROB7138	1
13	COM4408	1

9. This query shows a list of resources that have never been loaned:

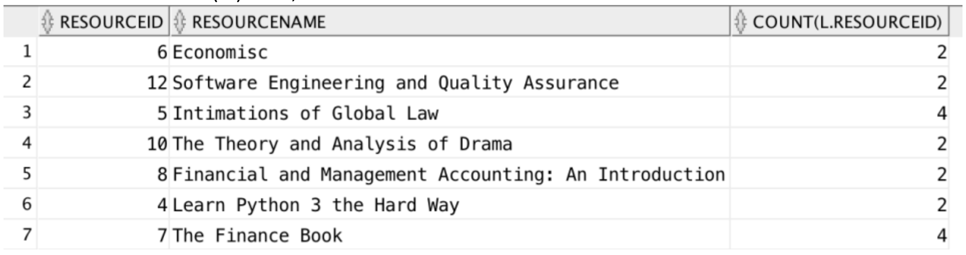
```
SELECT DISTINCT RESOURCENAME, AUTHOR
FROM LRESOURCE
WHERE RESOURCEID NOT IN (SELECT RESOURCEID FROM LOAN);
```



	RESOURCENAME	AUTHOR
1	The Finance Book	Stuart Warner and Si Hussain
2	Financial and Management Accounting: An Introduction	Prof Pauline Weetman
3	Economisc	Alain Anderton
4	The Theory and Analysis of Drama	Manfred Pfister
5	Intimations of Global Law	Neil Walker
6	A History of the English Language	Elly van Gelderen
7	History of everything	Prof Pauline Weetman

10. This query returns a list of resources that have been loaned more than once:

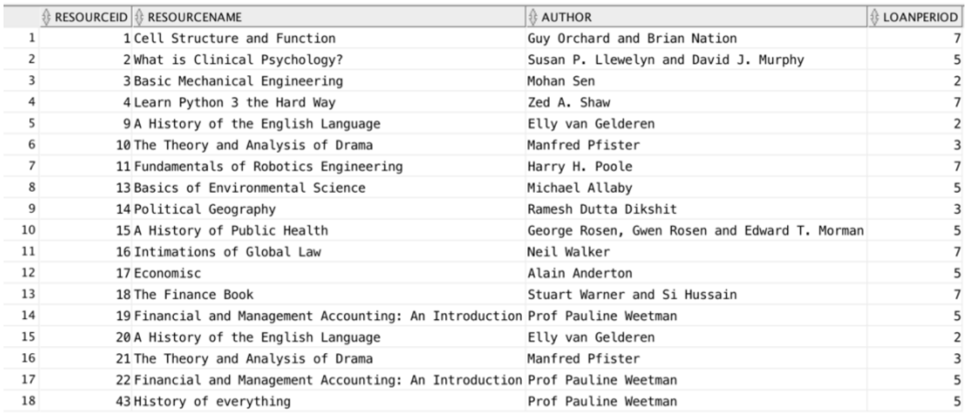
    SELECT L.RESOURCEID, LR.RESOURCENAME, COUNT(L.RESOURCEID)
    FROM LOAN L, LRESOURCE LR
    WHERE L.RESOURCEID = LR.RESOURCEID
    GROUP BY L.RESOURCEID, LR.RESOURCENAME
    HAVING COUNT(*) > 1;



	RESOURCEID	RESOURCENAME	COUNT(L.RESOURCEID)
1	6	Economisc	2
2	12	Software Engineering and Quality Assurance	2
3	5	Intimations of Global Law	4
4	10	The Theory and Analysis of Drama	2
5	8	Financial and Management Accounting: An Introduction	2
6	4	Learn Python 3 the Hard Way	2
7	7	The Finance Book	4

11. This query gives a list of resources currently available for loan:

    SELECT RESOURCEID, RESOURCENAME, AUTHOR, LOANPERIOD
    FROM LRESOURCE
    WHERE LOANPERIOD > 0
    AND RESOURCEID
    NOT IN (SELECT RESOURCEID FROM LOAN WHERE RETURNDATE IS NULL);

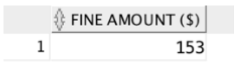


	RESOURCEID	RESOURCENAME	AUTHOR	LOANPERIOD
1	1	Cell Structure and Function	Guy Orchard and Brian Nation	7
2	2	What is Clinical Psychology?	Susan P. Llewelyn and David J. Murphy	5
3	3	Basic Mechanical Engineering	Mohan Sen	2
4	4	Learn Python 3 the Hard Way	Zed A. Shaw	7
5	9	A History of the English Language	Elly van Gelderen	2
6	10	The Theory and Analysis of Drama	Manfred Pfister	3
7	11	Fundamentals of Robotics Engineering	Harry H. Poole	7
8	13	Basics of Environmental Science	Michael Allaby	5
9	14	Political Geography	Ramesh Dutta Dikshit	3
10	15	A History of Public Health	George Rosen, Gwen Rosen and Edward T. Morman	5
11	16	Intimations of Global Law	Neil Walker	7
12	17	Economisc	Alain Anderton	5
13	18	The Finance Book	Stuart Warner and Si Hussain	7
14	19	Financial and Management Accounting: An Introduction	Prof Pauline Weetman	5
15	20	A History of the English Language	Elly van Gelderen	2
16	21	The Theory and Analysis of Drama	Manfred Pfister	3
17	22	Financial and Management Accounting: An Introduction	Prof Pauline Weetman	5
18	43	History of everything	Prof Pauline Weetman	5

12. This query shows the sum of all fines rounded to two decimal places:

    SELECT SUM(ROUND((CASE WHEN L.RETURNDATE IS NOT NULL THEN
    (L.RETURNDATE - L.LOANDATE - R.LOANPERIOD)
    ELSE (TO_DATE('01-DEC-2020', 'DD-MON-YYYY') - L.LOANDATE - R.LOANPERIOD)
    END),2)) "FINE AMOUNT ($)"
    FROM FINE F

```
INNER JOIN LOAN L ON F.LOANID = L.LOANID
INNER JOIN LRESOURCE R ON L.RESOURCEID = R.RESOURCEID;
```



```
    FINE AMOUNT ($)
1               153
```

Data Security Issues

This section is devoted to potential security issues pertaining to our library database system that should be addressed by the administrators and users, as well as effective measures to tackle these issues.

The library database system is subject to potential threats that may damage the integrity of the system. Firstly, there could be a security breach of the network through which library users access the database, resulting in the hacking into the database system by unscrupulous internet users who intend to use the personal data for illicit activities. Secondly, data corruption or data loss may ensue owing to an abrupt power or hardware failure that renders data storage unavailable for a period until such failure has been fixed. Furthermore, there could be an infringement of privacy or unwanted access of data by irrelevant parties of the system; for example, staff users with access privileges may browse or alter the database during non-working hours.

The Data Protection Act 2018 (Section 57) (referred to as "the Act" below) stipulates that the administrators of the library database system are liable to implement technical and organizational measures that upload data protection principles in an effective manner. To protect the database system against the threats described in the previous paragraph, the administrators will need to first secure the hardware and network that supports all activities associated with the database. Given that a disk is often the piece of hardware that is most likely to fail in the system, the Redundant Array of Independent Disks (RAID) Technology is adopted to guarantee that a duplicate, or backup, of data is being periodically stored in a redundant disk so that the entre set of data can be retained in the event of a disk failure (Stockman, 2020). Moreover, to prevent leaking of data, an encryption procedure, which involves the encoding of the data by a specific algorithm, should be carried out such that the data will no longer be readable by any programme without the decryption key.

It is also one of the administrators' duties to control who have access privileges by using the GRANT and REVOKE commands when user accounts are created (Stockman, 2020). In our system, staff users can access their own personal information, all users' current and past loan information, any fines incurred for late returns, and whether they have been suspended. Student users, on the other hand, will be able to view a record of these pieces of information that is only relevant to themselves. To achieve this, views, which are a simplified presentation of information, are created upon request by a staff or student user. These views should only contain data that are relevant to the user who files a request. There is a log file of all database activities or changes to keep track of each count of data access and retrieval, which lists the information about who accessed the data, what data were accessed and what time such access took place. This can expose any unauthorized or unwanted access to the database, ensuring that any data access should abide by sections 87 and 88 of the Act, which states that the data is only used for specified, explicit purposes, as well as in a way that is adequate, relevant and restricted to only what is necessary.

In addition to the abovementioned measures to be carried out by the administrators, users have a specific role to play in enhancing overall security of the system. Users will be required to secure their own account by setting a password combination that is difficult to be speculated. While the use of a combination including own birthday and ID card number is discouraged, the users will need to come up with a combination that consists of symbols, upper- and lower-case letters, and numbers. According to the Act (Section 45), users have the rights to be informed how their personal data are used and processed by the library, and to request access of their personal data. While they are unable to update the data of their account, they ought to report to the system administrators when they discover any errors concerning their personal data, or when there is a faulty display of other users' information on their account so that the mistakes can be swiftly rectified. Staff users who have extra access privileges should be informed of the potential consequences of violating the Act.

References

Stockman, A. (2020). 'Slides on Database Administration'. *ECS740P: Database Systems.* Available at https://qmplus.qmul.ac.uk/mod/resource/view.php?id=1451748 (Accessed: 9 December 2020).

Stockman, A. (2020). 'Slides for the Database Security Lecture'. *ECS740P: Database Systems.* Available at: https://qmplus.qmul.ac.uk/mod/resource/view.php?id=1476733 (Accessed: 9 December 2020).

Data Protection Act 2018. (c. 12). Available at: https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted (Accessed: 9 December 2020).