SUBMITED BY-

18BIT0242 EESHAN PANDEY

18BIT0097 CHINMAY NRUSINGH CHOUDHURY

18BIT0162 PRAKASH KUMAR

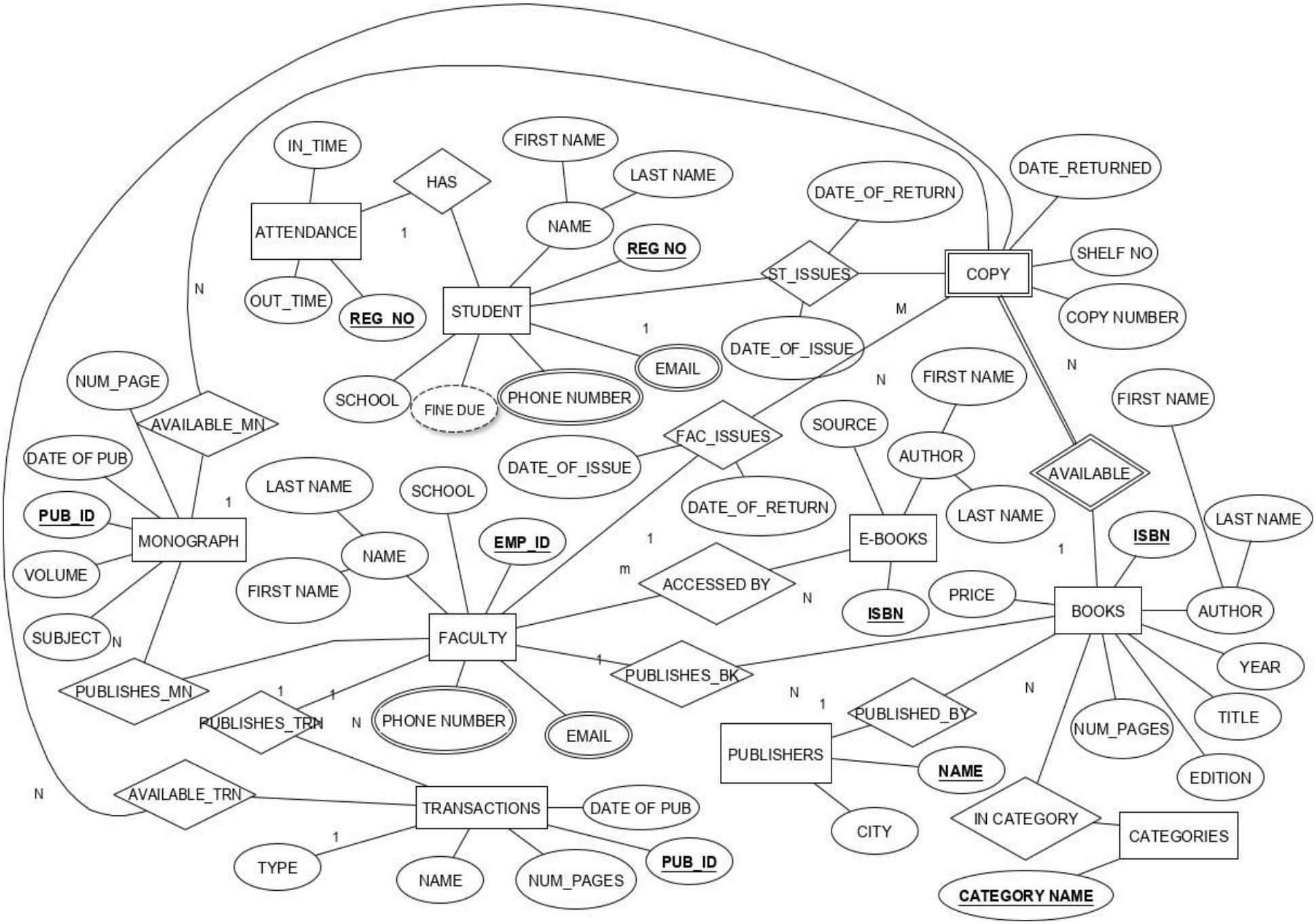
In fulfilment of the requirements for Project J Component-ITE1003

DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

VELLORE INSTITUE OF TECHNOLOGY

NOVEMBER 4, 2019


IN_TIME
HAS
FIRST NAME
LAST NAME
ATTENDANCE
NAME
REG NO
OUT_TIME
REG_NO
STUDENT
DATE_OF_RETURN
DATE_RETURNED
ST_ISSUES
COPY
SHELF NO
COPY NUMBER
DATE_OF_ISSUE
EMAIL
SCHOOL
FINE DUE
PHONE NUMBER
NUM_PAGE
AVAILABLE_MN
FIRST NAME
FIRST NAME
SOURCE
FAC_ISSUES
DATE OF PUB
DATE_OF_ISSUE
AUTHOR
AVAILABLE
LAST NAME
SCHOOL
DATE_OF_RETURN
LAST NAME
PUB_ID
MONOGRAPH
E-BOOKS
ISBN
LAST NAME
VOLUME
NAME
EMP_ID
ACCESSED BY
PRICE
BOOKS
AUTHOR
FIRST NAME
SUBJECT
FACULTY
ISBN
YEAR
PUBLISHES_MN
PUBLISHES_BK
PUBLISHED_BY
TITLE
PUBLISHES_TRN
PHONE NUMBER
EMAIL
NUM_PAGES
PUBLISHERS
NAME
EDITION
AVAILABLE_TRN
TRANSACTIONS
DATE OF PUB
IN CATEGORY
CATEGORIES
CITY
TYPE
NAME
NUM_PAGES
PUB_ID
CATEGORY NAME

INTRODUCTION:

This University Library is for both Faculty and Student. The Library accommodates a collection of Books, Journals, Monographs and Transactions. The Library Database system keeps account of all these books and the people issuing and publishing them. The library has a huge specialization collection of technical books advantageous to the engineering Students. It also has a fairly good set of Novels and Social Science resources. An Online Library portal is also maintained so that users can view information about their issues (like Return date, late fee, etc.). The Library also keeps E-books which can be accessed only by the faculty through an online portal. Monographs and Journals published by the faculty is also kept in the Library which can be issued by students and faculty. Plenty of studying space is available in the building and a very peaceful atmosphere is maintained. Book issuing process is also fast as it happens using automatic book sensors. Every book has number of copies hence every copy can be identified uniquely. Administrators have access to the Library Database which has details about every Faculty and Student and all their transactions with the Library. The Library only provides to the students of the University and not from outside.

DESCRIPTION:

The project represents the library system. The library contains several copies of a single book. Each **COPY** has its **DATE_OF_RETURN** (when the copy should be returned), shelf no. (where the book is kept)and **DATE_OF_ISSUE** (when the copy was issued).
Each book is uniquely identified by its **ISBN** number. **BOOK**'s have **AUTHOR**, **YEAR**, **TITLE**, **EDITION**, **NUM_PAGES** (number of pages), **PRICE** and **TITLE**. A Publisher publishes multiple books but a book only relates to a specific publisher. **PUBLISHER** is identified by its **NAME**. Publisher also has city of publication (**CITY**).

All books have a category (genre).

Each **COPY** is related to **STUDENT** with an attribute **DATE_RETURNED** (which signifies the date when the copy was returned by the student), since a student may borrow many books at a time but a copy is issued to a particular student at a specific point of time.

Each student has a unique **REG NO**, **SCHOOL**, **PHONE NO** and **EMAIL**.
FACULTY has been assigned a unique **EMP_ID** along with other information like **SCHOOL**, **PHONE NUMBER** and **EMAIL. FACULTY** can also access **E_BOOK**s which are available on the Library's website.

EBOOKS are classified uniquely using **ISBN**. Other information about the E-Book consists of **AUTHER NAME** and **SOURCE**.

ATTENDANCE of students in the library is kept track of. It has details such as **IN_TIME**, **OUT_TIME** and **REG_NO**.

Faculties also provide **MONOGRAPHS** and **TRANSACTIONS** each identified using a publication id, date of publication, name and number of pages**.**
Each MONOGRAPH has a **VOLUME** and a corresponding **SUBJECT.** Whereas **TRANSACTIONS** have **TYPE** (the type of transcripts such as reports, journals etc).

FUNCTIONAL REQUIREMENTS and CONCEPTUAL VIEW:

STUDENT:
Represents the STUDENTS entity type.

Reg_No	F_Name	L_Name	School	**Phone_No**	**Email**

1NF:

Normalization 1: no multi-valued attribute, allow only atomic values.

1. STUDENT

Reg_No	F_Name	L_Name	School

Attribute	**Data Type**	**Constraints**
REG_NO	Varchar(9)	Primary key
F_NAME	Varchar(15)	Not Null
L_NAME	Varchar(15)	Not Null
SCHOOL	Varchar(6)	Not Null

Functional dependency:
REG_NO -> {F_NAME,L_NAME,SCHOOL}

2. STUDENT_PH

Reg_No	Phone_No

Attribute	**Data Type**	**Constraints**
REG_NO	Varchar(9)	Primary Key
Phone_No	Number(10)	Primary Key

PHONE_NO-> { REG_NO }

3. STUDENT_EMAIL

Reg_No	Email

Attribute	**Data Type**	**Constraints**
REG_NO	Varchar(9)	Primary Key
EMAIL	Varchar(25)	Primary Key

EMAIL -> {REG_NO}

4. ATTENDANCE:
   To represent the last in time and out time of the student in the library.
   ST_REG refers to the to the reg_no of student relation.

Attribute	**Data Type**	**Constraints**
REG_NO	Varchar(9)	Primary key
IN_TIME	Timestamp	Not Null
OUT_TIME	Timestamp	Not Null
ST_REG	Varchar(9)	Foreign_Key(ST_REG) references STUDENT (REG_NO)

REG_NO	IN_TIME	OUT_TIME	ST_REG

Functional dependencies:
REG_NO->{IN_TIME, OUT_TIME,ST_REG}
ST_REG -> { IN_TIME, OUT_TIME, REG_NO}

REG_NO	IN_TIME	OUT_TIME
18BIT0097	19-09-2019 14:15:23	19-09-2019 16:45:54
18BIT0242	18-09-2019 10:15:15	18-09-2019 13:25:08
18BEC0635	18-09-2019 12:30:14	18-09-2019 15:30:15
18BEC0756	19-09-2019 11:12:29	19-09-2019 13:52:30

FACULTY

Stores the faculty information.
Each faculty is uniquely identified by their employee id. They have first name, Last name and school to which they belong to.

Emp_id	School	F_Name	L_Name	Phone_No	Email

Normalization 1: no multi-valued attribute, allow only atomic values.

5. FACULTY

Emp_id	School	F_Name	L_Name

Attribute	**Data Type**	**Constraints**
EMP_ID	Number(4)	Primary Key, Check(length(EMP_ID)==5)
FNAME	Varchar(15)	Not Null
LNAME	Varchar(15)	Not Null
SCHOOL	Varchar(7)	Not Null

Functional dependencies:
EMP_ID->{FNAME,LNAME,SCHOOL}

EMP_ID	FNAME	LNAME	SCHOOL
5412	KARTHIKA	K	SAS
7854	SENTHIL	KUMAR	SMEC
8613	PRAVEEN	T	SAS
7943	DELHI	BABU	SCOPE
1271	JOHN	DOE	SCOPE
1843	THOMAS	PANDEY	SMEC
1300	DAVID	IGOR	SITE

6. FACULTY_PH

Emp_id	**Phone_No**

Attributes	**Data Type**	**Constraints**
EMP_ID	Number(4)	Primary Key, Check(length(EMP_ID==5))

PHONE_NUMBER	Number(10)	Primary Key, Check(length(EMP_ID==10))

Functional dependencies:
PHONE -> EMP_ID

EMP_ID	PHONE
5412	9642158787
7854	7845123654
8613	9845757112
7943	9785664213
5412	7799647851
7854	7561201498
8613	8972104336
7943	9942035556
1271	9969264332
1843	5674224312
1843	9898765409
1271	9999870019
1300	7007989071

7. FACULTY_EMAIL

Emp_id	Email

Attributes	**Data Type**	**Constraints**
EMP_ID	Number(4)	Primary Key, Check(length(EMP_ID==5))
EMAIL	Number(10)	Primary Key, Check(length(EMP_ID==10))

Functional dependencies:
EMAIL -> EMP_ID

EMP_ID	EMAIL
5412	karthikak23@gmail.com
7854	senthilsenthil@gmail.com
8613	praveenalgebra@rediffmail.com
8613	praveenalg123@gmail.com

1271	john123@gmail.com
1271	Johnhello1@rediffmail.com
1843	tommy88@gmail.com
1300	davidpop121@hotmail.com
1300	davidpop121@hotmail.com

8. EBOOKS:
   Stores the details of all the Ebooks available. Each ebook has an unique ISBN, a title, first_name and last_name of the author and the source of the ebook.

Attributes	**Data Type**	**Constraints**
ISBN	Varchar(5)	Primary Key, Check(length(ISBN==5))
TITLE	Varchar(20)	Not Null
FNAME	Varchar(15)	Not Null
LNAME	Varchar(15)	Not Null
SOURCE	Varchar(50)	Not Null

ISBN	TITLE	FNAME	LNAME	SOURCE

Functional dependencies:
ISBN -> {TITLE, SOURCE}

ISBN	TITLE	F_NAME	L_NAME	SOURCE
E1333	Trigonometry Concepts	Jonathan	Wick	www.open bookworld.net
E2422	Basic Steganography	Ethan	Hunt	www.openbookworld.net
E2133	Advanced Java and applications	Chuck	Noland	www.mitcoursebooks.com
E1783	Modern Physics	Morgan	Freeman	www.vitebooks.com
E3003	Relational Database for Dummies	Tom	McDonald	www.mitcoursebooks.com

9. ACCESSED_BY:
   Stores the information about which faculty accessed which ebook.

Attributes	Data Type	Constraints
EMP_ID	Number(4)	Primary Key, Check(length(EMP_ID==5))
ISBN	Varchar(5)	Primary Key, Check(length(ISBN==5))

EMP_ID	ISBN

Functional dependencies:
ISBN->EMP_ID

EMP_ID	ISBN
1271	E1333
1843	E2422
1300	E2133
1300	E3003
1271	E2133
5412	E1783
7943	E3003

10. MONOGRAPHS:

Stores the details about the monographs in the library. Each monograph has a publication id, date_of_publication, Volume, subject, number of pages and emp_id of the faculty who published the monograph.

Attributes	Data Type	Constraints
PUB_ID	Varchar(6)	Primary Key, Check(length(PUB_ID==6))
DATE_OF_PUB	Date	Not Null
VOLUME	Number(2)	Not Null
SUBJECT	Varchar(10)	Not Null
NUM_OF_PAGE	Number(3)	Not Null
EMP_ID	Number(4)	Foreign_Key(EMP_ID) references FACULTY (EMP_ID)

PUB_ID	DATE_OF_PUB	VOLUME	SUBJECT	NUM_PAGE	EMP_ID

Functional dependencies:
PUB_ID -> {DATE_OF_PUB, VOLUME, SUBJECT, NUM_PAGE}
EMP_ID -> {SUBJECT, VOLUME}

PUB_ID	DATE_OF_PUB	VOLUME	SUBJECT	NUM_PAGE	EMP_ID
G10021	11-09-2010	2	Operating Systems	420	1271
G20325	14-09-2000	1	Advanced Cryptography	300	1843
H30113	05-10-2010	1	Semiconductors	630	1271
A18334	04-12-2017	3	Applied Linear Algebra	550	1843
J20041	20-01-2009	8	Communication Systems	430	1300

11. TRANSACTIONS:
Stores the details about the transactions published. Each transaction has a publication id, date_of_publication, type of transaction, number of pages, name of the transaction and emp_id of the faculty who published the transaction.

Attributes	**Data Type**	**Constraints**
PUB_ID	Varchar(6)	Primary Key, Check(length(PUB_ID==6))
DATE_OF_PUB	Date	Not Null
TYPE	Varchar(15)	Not Null
NAME	Varchar(30)	Not Null
NUM_OF_PAGE	Number(3)	Not Null
EMP_ID	Number(4)	Foreign_Key(EMP_ID) references FACULTY (EMP_ID)

PUB_ID	NUM_PAGES	DATE_OF_PUB	NAME	TYPE	EMP_ID

Functional dependencies:
PUB_ID -> {NUM_PAGES, DATE_OF_PUB, NAME, TYPE, EMP_ID}
EMP_ID -> {NAME, TYPE}

PUB_ID	NUM_PAGES	DATE_OF_PUB	NAME	TYPE	EMP_ID
TA4002	120	10-01-2014	Border Gateway Protocol Basics	IT: Network	1271
TA3200	200	15-01-2019	Fluid Dynamics in Space	Physics: Mechanics	1843
TG2933	130	13-11-2009	Computer Graphics Optimizations	IT: Graphics	5412
TS1672	135	31-12-2015	Cloud Architecture	IT: COA	1300
TA1003	190	10-09-2017	A Dragon and his friend	Novel	8613

COPY

COPY_ NUMBER	ISBN	DATE_ RETURNED	SHELF_ NO	ST_ DATE_OF _ISSUE_	ST_ DATE_ OF_ RETURN	REG_NO	EMP_ID	M_PUB_ID	T_PUB_ID
FC_ DATE_ OF_ ISSUE	FC_ DATE_ OF_ RETURN								

{REG_NO, M_PUB_ID, COPY_NUMBER} -> {ST_DATE_OF_ISSUE, ST_DATE_OF_RETURN, DATE_RETURNED}

{REG_NO, T_PUB_ID, COPY_NUMBER} -> {ST_DATE_OF_ISSUE, ST_DATE_OF_RETURN, DATE_RETURNED }

{REG_NO, ISBN, COPY_NUMBER} -> {ST_DATE_OF_ISSUE, ST_DATE_OF_RETURN, DATE_RETURNED }

{EMP_ID, ISBN, COPY_NUMBER} -> {ST_DATE_OF_ISSUE, ST_DATE_OF_RETURN, DATE_RETURNED }

{EMP_ID, T_PUB_ID, COPY_NUMBER} -> { FC_DATE_OF_ISSUE, FC_DATE_OF_RETURN, DATE_RETURNED }

{EMP_ID, M_PUB_ID, COPY_NUMBER} -> {FC_DATE_OF_ISSUE, FC_DATE_OF_RETURN, DATE_RETURNED }

{COPY_NUMBER, ISBN}->SHELF_NO

{COPY_NUMBER, T_PUB_ID}->SHELF_NO
{COPY_NUMBER, M_PUB_ID} ->SHELF_NO

1NF: SATISFIED
2NF: Key = {COPY_NUMBER, ISBN, M_PUB_ID, T_PUB_ID}
Hence, 2NF condition is violated.

12. ST_BOOK_ISSUE

Attribute	**Data Type**	**Constraints**
ISBN	Varchar(6)	Primary key
REG_NO	Varchar(9)	Primary key
COPY_NUMBER	Number(2)	Primary key
DATE_OF_ISSUE	Date	Not NULL
DATE_OF_RETURN	Date	Not NULL
DATE_RETURNED	Date	Not NULL

{REG_NO, ISBN, COPY_NUMBER} -> {ST_DATE_OF_ISSUE, ST_DATE_OF_RETURN}

13. FAC_BOOK_ISSUE

Attribute	**Data Type**	**Constraints**
ISBN	Varchar(6)	Primary key
EMP_ID	Number(4)	Primary key
COPY_NUMBER	Number(2)	Primary key
DATE_OF_ISSUE	Date	Not NULL
DATE_OF_RETURN	Date	Not NULL
DATE_RETURNED	Date	Not NULL

ISBN	**EMP_ID**	**COPY _NUMBER**	DATE_ OF ISSUE	DATE_OF_ RETURN	DATE_ RETURNED
TA8185	5412	1	12-JUN-19	21-JUN-19	21-JUN-19
CG1212	7854	2	10-MAR-19	17-MAR-19	25-MAR-19
TB1988	1271	1	12-OCT-18	21-OCT-18	20-OCT-18

14. ST_MON_ISSUE

Attribute	**Data Type**	**Constraints**
M_PUB_ID	Varchar(6)	Primary key
REG_NO	Varchar(9)	Primary key
COPY_NUMBER	Number(2)	Primary key
DATE_OF_ISSUE	Date	Not NULL
DATE_OF_RETURN	Date	Not NULL
DATE_RETURNED	Date	Not NULL

{REG_NO, M_PUB_ID, COPY_NUMBER } -> { ST_DATE_OF_ISSUE, ST_DATE_OF_RETURN}

15. ST_TR_ISSUE

Attribute	**Data Type**	**Constraints**
T_PUB_ID	Varchar(6)	Primary key
REG_NO	Varchar(9)	Primary key
COPY_NUMBER	Number(2)	Primary key
DATE_OF_ISSUE	Date	Default NULL
DATE_OF_RETURN	Date	Default NULL
DATE_RETURNED	Date	Not NULL

{REG_NO, T_PUB_ID, COPY_NUMBER} -> {ST_DATE_OF_ISSUE, ST_DATE_OF_RETURN}

16. FAC_MON_ISSUE

Attribute	**Data Type**	**Constraints**
M_PUB_ID	Varchar(6)	Primary key
EMP_ID	Number(4)	Primary key
COPY_NUMBER	Number(2)	Primary key
DATE_OF_ISSUE	Date	Default NULL
DATE_OF_RETURN	Date	Default NULL
DATE_RETURNED	Date	Not NULL

{EMP_ID, M_PUB_ID, COPY_NUMBER} -> {FC_DATE_OF_ISSUE, FC_DATE_OF_RETURN}

PUB_ID	**EMP_ID**	**COPY _NUMBER**	DATE_ OF ISSUE	DATE_OF_ RETURN	DATE_ RETURNED
TG2933	5412	1	21-OCT-19	28-OCT-19	28-OCT-19

PUB_ID	**EMP_ID**	**COPY _NUMBER**	DATE_ OF ISSUE	DATE_OF_ RETURN	DATE_ RETURNED
G10021	1843	1	01-APR-19	08-JUN-19	09-APR-19
J20041	5412	1	12-JUN-19	19-JUN-19	19-JUN-19

17. FAC_TR_ISSUE

Attribute	**Data Type**	**Constraints**
T_PUB_ID	Varchar(6)	Primary key
EMP_ID	Number(4)	Primary key
COPY_NUMBER	Number(2)	Primary key
DATE_OF_ISSUE	Date	Default NULL

DATE_OF_RETURN	Date	Default NULL
DATE_RETURNED	Date	Not NULL

{EMP_ID, T_PUB_ID, COPY_NUMBER,} -> {FC_DATE_OF_ISSUE, FC_DATE_OF_RETURN}

PUB_ID	**EMP_ID**	**COPY _NUMBER**	DATE_ OF ISSUE	DATE_OF_ RETURN	DATE_ RETURNED
TG2933	5412	1	21-OCT-19	28-OCT-19	28-OCT-19

18. COPY_BOOK

Stores the information about the copies of book available.

Attribute	**Data Type**	**Constraints**
COPY_NUMBER	Number(2)	Primary key
ISBN	Varchar(6)	Primary key
SHELF_NO	Number(2)	Not Null

COPY_ NUMBER	ISBN	SHELF_NO

Functional dependencies:
{COPY_NUMBER, ISBN}->{SHELF_NO }

COPY NUMBER	**ISBN**	SHELF_NO
1	TA4002	02
1	TA3200	02
1	TG2933	09
2	TG2933	10

19. COPY_MON

Represents the copies available of Monographs.

Attribute	**Data Type**	**Constraints**
COPY_NUMBER	Number(2)	Primary key

PUB_ID	Varchar(6)	Primary key
SHELF_NO	Number(2)	Not Null

COPY_ NUMBER	PUB_ID	SHELF_NO

Functional dependencies:
{COPY_NUMBER, PUB_ID}->{SHELF_NO}

COPY NUMBER	PUB_ID	SHELF_NO
1	TA4002	02
1	TA3200	02
1	TG2933	09
2	TG2933	10

20. COPY_TRANS

Represents the copies available of Transactions.

Attribute	**Data Type**	**Constraints**
COPY_NUMBER	Number(2)	Primary key
PUB_ID	Varchar(6)	Primary key
SHELF_NO	Number(2)	Not Null

COPY_ NUMBER	PUB_ID	SHELF_NO

Functional dependencies:
{COPY_NUMBER, PUB_ID}->{ SHELF_NO }

COPY NUMBER	PUB_ID	SHELF_NO
1	TG2933	32
2	TG2933	32
3	TG2933	32
1	TS1672	31

BOOKS

Contains the details of books.

ISBN	F_NAME	L_NAME	TITLE	YEAR	EDITION	NUM_PAGES	PRICE	CATEGORIES	PUBLISHER_NAME	CITY

Functional dependencies:

TITLE-> {YEAR, EDITION}
ISBN -> {TITLE, YEAR, EDITION,PUBLISHER_NAME}
PUBLISHER_NAME->CITY

Normalization 1: no multi-valued attribute, allow only atomic values. Satisfied.

Normalization 2: Does **not** allow Partial Functional Dependency. Satisfied.

Normalization 3: transitive functional dependency exists

Hence we decompose into two relations:

21. BOOKS

Attribute	Data Type	Constraints
ISBN	Varchar(5)	Primary Key, Check(length(ISBN==5))
F_NAME	Varchar(15)	Not Null
L_NAME	Varchar(15)	Not Null
TITLE	Varchar(20)	Not Null
YEAR	Number(4)	Not Null
EDITION	Number(2)	Not Null
NUM_PAGES	Number(4)	Not Null
PRICE	Number(4)	Not Null
CATEGORIES	Varchar(15)	Foreign_Key(CATEGORIES) references CATEGORIES (CATEGORIES)

ISBN	F_NAME	L_NAME	TITLE	YEAR	EDITION	NUM_PAGES	PRICE	CATEGORIES

Funtional dependencies:

ISBN ->{F_NAME,L_NAME,TITLE,YEAR,EDITION,NUM_PAGES,PRICE,CATEGORIES}

ISBN	F_NAME	L_NAME	TITLE	YEAR	EDITION	NUM_PAGES	PRICE	CATEGORIES

A8185	John	Mathew	Fundamental Algebra	2015	2	455	300	Mathematics
B2213	Harish	Verma	Concepts of Physics	1999	8	6	480	Physics
B1988	David	Jones	Network and Information Security	2012	1	500	350	Information Technology
C3412	Stanley	Michaels	Business Analytics	2010	3	650	500	Business
G3341	Kunal	Nag	Circuit Analysis for beginners	2004	5	395	410	Physics
G1212	Paramecia	Ramone	Operation Systems	2013	2	555	500	Information Technology

22. PUBLISHERS:

Represents the publishers whose published books are present in the library. Each publisher is uniquely identified by his name and has a city where it is located.

Attributes	**Data Type**	**Constraints**
NAME	Varchar(20)	Primary Key
CITY	Varchar(20)	Not Null

NAME	CITY

Functional dependencies:
NAME -> CITY

NAME	CITY
Blue Stone Publications	Washington
Penguin	Delhi
Anuradha	Chennai
Willey	London
Hamill and Hamill	Albuquerque

23. CATEGORIES:

Represents the categories of books present in the library.

Attributes	**Data Type**	**Constraints**
Categories	Varchar(15)	Primary Key

CATEGORIES
Physics
Mathematics
Information Technology
Business
Novel

24. FAC_BOOKS

Represents the books published by faculties.

Attribute	**Data Type**	**Constraints**
ISBN	Varchar(5)	Primary Key
EMP_ID	Number(4)	Not Null

ISBN	EMP_ID

Functional dependencies:

ISBN -> EMP_ID

ISBN	EMP_ID
C3412	1271
G3341	1300
G1212	8613

DDL, DML Statements

1. Student:

SQL> create table student (reg_no varchar(9) primary key, f_name varchar(15) not null, l_name varchar(15) not null, school varchar(6) not null);

2. Student ph:

SQL> create table student_ph (reg_no varchar(9), phone_no number(10), primary key(reg_no,phone_no));

3. Student email:

SQL> create table student_email (reg_no varchar(9), email varchar(25), primary key(reg_no,email));

4. Attendance:

SQL> create table attendance(reg_no varchar(9),in_time timestamp, out_time timestamp, st_reg varchar(9), foreign key(st_reg) references student(reg_no), primary key(reg_no));

5. Faculty:

SQL> create table faculty (emp_id number(4) primary key, f_name varchar(15) not null, l_name varchar(15) not null, school varchar(6) not null);

6. Faculty ph

SQL> create table faculty_ph (emp_id number(4), phone_no number(10), primary key(emp_id,phone_no));

7. Faculty email:

SQL> create table faculty_email (emp_id number(4), email varchar(25), primary key(emp_id,email));

8. Ebooks:

SQL> create table ebooks(isbn varchar(5) primary key, title varchar(50), f_name varchar(15), l_name varchar(15), source varchar(50));

9. Accessed by:

SQL> create table accessed_by(emp_id number(4), isbn varchar(5), primary key(emp_id, isbn));

10. Monographs:
    SQL> create table monographs(pub_id varchar(6) primary key, date_of_pub date, volume number(2), subject varchar(10), num_of_page number(3), emp_id number(4), foreign key (emp_id) references faculty(emp_id));

11. Transactions:

SQL> create table transactions(pub_id varchar(6) primary key, date_of_pub date, type varchar(15) , name varchar(30), num_of_page number(3), emp_id number(4), foreign key(emp_id) references faculty(emp_id));

12. Copy book:

```
SQL> create table copy_book (
  copy_number number(2),
  isbn varchar(5),
  shelf_no number(2),
 primary key (copy_number, isbn));
```

13. Copy mon:

```
SQL> create table copy_mon(
 copy_number number(2),
 pub_id varchar(5),
 shelf_no number(2),
primary key(copy_number, pub_id));
```

Table created.

14. Copy trans:

```
SQL> create table copy_trans(
 copy_number number(2),
 pub_id varchar(5),
 shelf_no number(2),
 primary key(copy_number, pub_id));
```

15. Books:

```
SQL> create table books(
```

```
 isbn varchar(6) primary key,
 f_name varchar(15),
 l_name varchar(15),
 title varchar(50),
 year number(4),
 edition number(2),
 num_pages number(4),
 price number(4),
categories varchar(50),
foreign key(categories) references categories(categories));
```

16. Publisher:

```
SQL> create table publishers( name varchar(20) primary key, city varchar(20));
```

17. Categories:

```
SQL> create table categories(categories varchar(15) primary key);
```

18. Fac_books:

```
SQL> create table fac_books(isbn varchar(5) primary key, emp_id number(4));
```

19. St_book_issue:

```
create table st_book_issue (
  isbn varchar2(6) ,
   reg_no varchar2(9),
   copy_number number(2),
   date_of_issue date,
   date_of_return date,
   date_returned date,
primary key(isbn,reg_no,copy_number)
   );
```

20. St_mon_issue:

```
    create table st_mon_issue (
          m_pub_id varchar2(6) ,
           reg_no varchar2(9),
```

```
            copy_number number(2),
            date_of_issue date,
            date_of_return date,
          primary key(m_pub_id,reg_no,copy_number)
            );
```

21. St_tr_issue:

```
create table st_tr_issue (
  t_pub_id varchar2(6) ,
   reg_no varchar2(9),
   copy_number number(2),
   date_of_issue date,
   date_of_return date,
primary key(t_pub_id,reg_no,copy_number)
   );
```

22. Fac_book_issue:

```
create table fac_book_issue (
  isbn varchar2(6) ,
   emp_id number(4),
   copy_number number(2),
   date_of_issue date,
   date_of_return date,
   date_returned date,
primary key(isbn,emp_id,copy_number)
   );
```

23. Fac_mon_issue:

```
create table fac_mon_issue (
  m_pub_id varchar2(6) ,
   emp_id number(4),
   copy_number number(2),
   date_of_issue date,
   date_of_return date,
primary key(m_pub_id, emp_id,copy_number)
   );
```

24. Fac_tr_issue:

```
create table fac_tr_issue (
```

```
  t_pub_id varchar2(6) ,
   emp_id number(4),
   copy_number number(2),
   date_of_issue date,
   date_of_return date,
primary key(t_pub_id, emp_id,copy_number)
   );
```

Stored Procedures

1. Procedure for student fine calculation

```
CREATE OR REPLACE PROCEDURE calc_fine_stud(
reg in varchar,
type in number,
search_id in varchar,
cpy_num in number
)
IS

date_ret date; -- date when returned
exp_date date; -- date to be returned
ret_id varchar(5);
extra_days number(3);
fine number(3);

begin

IF type = 1 then
        SELECT isbn,date_returned, date_of_return
        INTO ret_id, date_ret, exp_date
        FROM st_book_issue
        WHERE isbn = search_id AND reg = reg_no AND cpy_num =
copy_number;
        extra_days:=date_ret - exp_date;
        IF extra_days < 0 then
                dbms_output.put_line('no fines');
        ELSE
                fine := extra_days*2;
                dbms_output.put_line('book:'||ret_id);
```

```
                dbms_output.put_line('issued by:'||reg);
                dbms_output.put_line('fine due:'||fine);
        END IF;
ELSIF type = 2 THEN
        SELECT pub_id,date_returned, date_of_return
        INTO ret_id, date_ret, exp_date
        FROM st_mon_issue
        WHERE search_id = pub_id AND reg = reg_no AND cpy_num =
copy_number;
        extra_days:=date_ret - exp_date;
        IF extra_days < 0 then
                dbms_output.put_line('no fines');
        ELSE
                fine := extra_days*2;
                dbms_output.put_line('book:'||ret_id);
                dbms_output.put_line('issued by:'||reg);
                dbms_output.put_line('fine due:'||fine);
        END IF;
ELSIF type = 3 THEN
        SELECT pub_id,date_returned, date_of_return
        INTO ret_id, date_ret, exp_date
        FROM st_tr_issue
        WHERE pub_id =search_id AND reg = reg_no AND cpy_num =
copy_number;
        extra_days:=date_ret - exp_date;
        IF extra_days < 0 then
                dbms_output.put_line('no fines');
        ELSE
                fine := extra_days*2;
                dbms_output.put_line('book:'||ret_id);
                dbms_output.put_line('issued by:'||reg);
                dbms_output.put_line('fine due:'||fine);
        END IF;
ELSE
        dbms_output.put_line('bad type');
END IF;
END;
```

2. Procedure for faculty fine calculation

```
CREATE OR REPLACE PROCEDURE calc_fine_stud(
e_id in number,
type in number,
search_id in varchar,
cpy_num in number
)
IS

date_ret date; -- date when returned
exp_date date; -- date to be returned
ret_id varchar(6);
extra_days number(3);
fine number(3);

begin

IF type = 1 then
        SELECT isbn,date_returned, date_of_return
        INTO ret_id, date_ret, exp_date
        FROM fac_book_issue
        WHERE isbn = search_id AND emp_id = e_id AND cpy_num =
copy_number;
        extra_days:=date_ret - exp_date;
        IF extra_days < 0 then
                dbms_output.put_line('no fines');
        ELSE
                fine := extra_days*2;
                dbms_output.put_line('book:'||ret_id);
                dbms_output.put_line('issued by:'||e_id);
                dbms_output.put_line('fine due:'||fine);
        END IF;
ELSIF type = 2 THEN
        SELECT m_pub_id, date_returned, date_of_return
        INTO ret_id, date_ret, exp_date
        FROM fac_mon_issue
        WHERE search_id = m_pub_id AND emp_id = e_id AND cpy_num =
copy_number;
        extra_days:=date_ret - exp_date;
        IF extra_days < 0 then
```

```
                dbms_output.put_line('no fines');
        ELSE
                fine := extra_days*2;
                dbms_output.put_line('book:'||ret_id);
                dbms_output.put_line('issued by:'||e_id);
                dbms_output.put_line('fine due:'||fine);
        END IF;
ELSIF type = 3 THEN
        SELECT t_pub_id,date_returned, date_of_return
        INTO ret_id, date_ret, exp_date
        FROM fac_tr_issue
        WHERE t_pub_id =search_id AND emp_id = e_id AND cpy_num =
copy_number;
        extra_days:=date_ret - exp_date;
        IF extra_days < 0 then
                dbms_output.put_line('no fines');
        ELSE
                fine := extra_days*2;
                dbms_output.put_line('book:'||ret_id);
                dbms_output.put_line('issued by:'||e_id);
                dbms_output.put_line('fine due:'||fine);
        END IF;
ELSE
        dbms_output.put_line('bad type');
END IF;
END;
```

3. Procedure to display attendance details of a student

```
CREATE OR REPLACE PROCEDURE stud_attendance(
reg_num in varchar)

IS
reg varchar(9);
in_t timestamp;
out_t timestamp;

BEGIN

SELECT reg_no, in_time, out_time
```

```
INTO reg, in_t, out_t
FROM attendance
WHERE reg_no=reg_num;
Dbms_output.put_line('student reg_no:'||reg);
Dbms_output.put_line('library in-time(last):'||in_t);
Dbms_output.put_line('library out-time(last):'||out_t);

END;
```

4. Procedure to find books of given category

```
CREATE OR REPLACE PROCEDURE category_check(
category_name in varchar)
declare
cursor curr is select p.* from
books b
where b.categories = category_name
r_book curr%rowtype;
BEGIN
open curr;
loop
        fetch curr into r_book;
        exit when curr%notfound;
        dbms_output.put_line('Category:'||r_book.categories);
        dbms_output.put_line('isbn:'||r_book.isbn);
        dbms_output.put('Author:'||r_book.f_name);
        dbms_output.put_line(r_book.l_name);
        dbms_output.put_line('Title:'||r_book.title);
        dbms_output.put_line('Edition:'||r_book.edition);
end loop;
close curr;
END;
```

5. Number of copies available for that book,monographs,transactions

```
 set serveroutput on
create or replace procedure copies_ava( b_name IN varchar2,
 authr_first IN varchar2,
authr_last IN varchar2,
no_of_pages In varchar2,
```

```
date_pub IN date,
type IN number,
cop_ava OUT number)
  IS  n number(4);
   begin
if type=1 then
   select count(isbn)
 into n from books
where title=b_name and f_name=authr_first and l_name=authr_last and
num_pages=no_of_pages;
elsif type=2 then
select count(pub_id)
into n from monographs
 where subject=b_name and num_of_page=no_of_pages and
date_of_pub=date_pub;
elsif type=3 then
select count(pub_id)
into  n from transactions
where name=b_name and num_of_page=no_of_pages and
date_of_pub=date_pub;
else
dbms_output.put_line('Unable to get the total no of copies for this book');
end if;
cop_ava:=n;
END copies_ava;
```

Triggers

1. Trigger to log transactions for faculty table

```
create or replace trigger log_faculty
  after update or delete on faculty
  for each row
  declare
  l_trans varchar(10);
  begin
  l_trans :=CASE
    WHEN updating THEN 'UPDATE'
    WHEN deleting THEN 'DELETE'
  end;
  insert into transaction_logs(table_name, transaction, by_user, trans_time)
```

```
values('FACULTY',l_trans,USER,SYSDATE);
end;
/
```

2. **Trigger to log transaction on student table**

```
create or replace trigger log_student
after update or delete on student
for each row
declare
l_trans varchar(10);
begin
l_trans :=CASE
  WHEN updating THEN 'UPDATE'
  WHEN deleting THEN 'DELETE'
end;
insert into transaction_logs(table_name, transaction, by_user, trans_time)
values('STUDENT',l_trans,USER,SYSDATE);
end;
/
```

3. **Trigger to limit no of issues to 4 for faculty and students**

```
create or replace trigger st_mon

before insert on st_mon_issue

for each row

declare

no_of_book int(3);

no_of_tr int(3);

no_of_mon int(3);

t_count int (3);

begin

select count(reg_no)  into no_of_mon from st_mon_issue where
reg_no=:new.reg_no;

select count(reg_no)  into no_of_book from st_book_issue where
reg_no=:new.reg_no;

select count(reg_no)  into no_of_tr from st_tr_issue where reg_no=:new.reg_no;

t_count:= no_of_mon + no_of_book + no_of_tr;
```

```
if inserting then
if t_count>4 then
raise_application_error(-20500,'More Than 4 Books is NOT ALLOWED');
end if;
end if;
end;
/
**********
Create or replace trigger st_tr
before insert on st_tr_issue
for each row
declare
no_of_book int(3);
no_of_tr int(3);
no_of_mon int(3);
t_count int (3);
begin
select count(reg_no)  into no_of_mon from st_mon_issue where
reg_no=:new.reg_no;
select count(reg_no)  into no_of_book from st_book_issue where
reg_no=:new.reg_no;
select count(reg_no)  into no_of_tr from st_tr_issue where reg_no=:new.reg_no;
t_count:= no_of_mon + no_of_book + no_of_tr;
if inserting then
if t_count>4 then
raise_application_error(-20500,'More Than 4 Books is NOT ALLOWED');
end if;
end if;
```

```
end;
/
**************************************************************
create or replace trigger st_book
before insert on st_book_issue
for each row
declare
no_of_book int(3);
no_of_tr int(3);
no_of_mon int(3);
t_count int (3);
begin
select count(reg_no)  into no_of_mon from st_mon_issue where
reg_no=:new.reg_no;
select count(reg_no)  into no_of_book from st_book_issue where
reg_no=:new.reg_no;
select count(reg_no)  into no_of_tr from st_tr_issue where reg_no=:new.reg_no;
t_count:= no_of_mon + no_of_book + no_of_tr;
if inserting then
if t_count>4 then
raise_application_error(-20500,'More Than 4 Books is NOT ALLOWED');
end if;
end if;
end;
/
**************************************************************
create or replace trigger fac_mon
before insert on fac_mon_issue
```

```
for each row
declare
no_of_book int(3);
no_of_tr int(3);
no_of_mon int(3);
t_count int (3);
begin
select count(emp_id)  into no_of_mon from fac_mon_issue where emp_id =:new.emp_id;
select count(emp_id)  into no_of_book from fac_book_issue where emp_id=:new.emp_id;
select count(emp_id)  into no_of_tr from fac_tr_issue where emp_id =: new.emp_id;
t_count:= no_of_mon + no_of_book + no_of_tr;
if inserting then
if t_count>4 then
raise_application_error(-20500,'More Than 4 Books is NOT ALLOWED');
end if;
end if;
end;
/
**************************************************************

Create or replace trigger fac_tr
before insert on fac_tr_issue
for each row
declare
no_of_book int(3);
no_of_tr int(3);
```

```
no_of_mon int(3);
t_count int (3);
begin
select count(emp_id)  into no_of_mon from fac_mon_issue where emp_id =:new.emp_id;
select count(emp_id)  into no_of_book from fac_book_issue where emp_id =:new.emp_id;
select count(emp_id)  into no_of_tr from fac_tr_issue where emp_id =:new.emp_id;
t_count:= no_of_mon + no_of_book + no_of_tr;
if inserting then
if t_count>4 then
raise_application_error(-20500,'More Than 4 Books is NOT ALLOWED');
end if;
end if;
end;
/
**********

create or replace trigger fac_tr
before insert on fac_tr_issue
for each row
declare
no_of_book int(3);
no_of_tr int(3);
no_of_mon int(3);
t_count int (3);
begin
select count(emp_id)  into no_of_mon from fac_mon_issue where emp_id =:new.emp_id;
```

```
select count(emp_id)  into no_of_book from fac_book_issue where emp_id
=:new.emp_id;

select count(emp_id)  into no_of_tr from fac_tr_issue where emp_id =:new.emp_id;

t_count:= no_of_mon + no_of_book + no_of_tr;

if inserting then

if t_count>4 then

raise_application_error(-20500,'More Than 4 Books is NOT ALLOWED');

end if;

end if;

end;

/
**********************************************************************


set serveroutput on;

create or replace trigger fac_book

before insert or delete on fac_book_issue

for each row

declare

no_of_book int(3);

no_of_tr int(3);

no_of_mon int(3);

t_count int (3);

begin

select count(emp_id)  into no_of_mon from fac_mon_issue where emp_id
=:new.emp_id;

select count(emp_id)  into no_of_book from fac_book_issue where emp_id
=:new.emp_id;

select count(emp_id)  into no_of_tr from fac_tr_issue where emp_id =:new.emp_id;

t_count:= no_of_mon + no_of_book + no_of_tr;
```

```
if inserting then

if (t_count>4) then

raise_application_error(-20500,'More Than 4 Books is NOT ALLOWED');

end if;

end if;

end;

/
```

****

Procedure to calculate whether students wear their ID card or not , they are with in the opening hours of the library and they are not allowed to carry more than 2 items.

```
set serveroutput on;
create or replace procedure last(time in timestamp,id in boolean,
items in number)
is count number(4);
begin
if (to_char(time,'hh24:mi') not between '07:00' and '23:30') then
raise_application_error(-20500,'Sorry library is closed');
end if;
if items < 3 then
if id=true then
dbms_output.put_line('Welcome to Library');
else
dbms_output.put_line('without id card no entrance');
end if;
else
dbms_output.put_line('Sorry more than 2 items not allowed');
end if;
end last;
/
```

Output:

```
SQL> set serveroutput on;
SQL> create or replace procedure last(time in timestamp,id in boolean, items in number)
  2  is count number(4);
  3  begin
  4  if (to_char(time,'hh24:mi') not between '07:00' and '23:30') then
  5  raise_application_error(-20500,'Sorry library is closed');
  6  end if;
  7  if items < 3 then
  8  if id=true then
  9  dbms_output.put_line('Welcome to Library');
 10  else
 11  dbms_output.put_line('without id card no entrance');
 12  end if;
 13  else
 14  dbms_output.put_line('Sorry more than 2 items not allowed');
 15  end if;
 16  end last;
 17  /

Procedure created.
```

```
SQL> exec last(to_timestamp('23:15','hh24:mi'),true,2);
Welcome to Library

PL/SQL procedure successfully completed.

SQL> exec last(to_timestamp('23:51','hh24:mi'),true,2);
BEGIN last(to_timestamp('23:51','hh24:mi'),true,2); END;

*
ERROR at line 1:
ORA-20500: Sorry library is closed
ORA-06512: at "HR.LAST", line 5
ORA-06512: at line 1


SQL> exec last(to_timestamp('23:15','hh24:mi'),true,5);
Sorry more than 2 items not allowed

PL/SQL procedure successfully completed.

SQL> exec last(to_timestamp('23:15','hh24:mi'),false,2);
without id card no entrance

PL/SQL procedure successfully completed.
```