University Library System

AMAANULLAH KHAN
JIRAPAT BOONMEE
ZUL ABSAR ALI

Table of Contents

Part 1 DATABASE DESIGN

List of All Assumptions

- All copies of one resource will be kept in only one place (on one shelf and floor), separated from other resources. The combination of shelf, floor and a ‘code’, which represents the first three letters of the creator’s (author for books) name, will be needed to locate the physical resource in the library.
- Members' name consists of two fields: first name and second name.
- The attribute ‘creator’ will be used synonymously with the ‘author’ of a book and ‘director’ of a DVD.
- The library is open 24 hours and seven days a week to allow people to return items on the whole day of the due date.
- The system will handle fines. Therefore, no actual code is needed at this stage to calculate fines.

The Conceptual Schema (ER diagram)

Our conceptual schema contains the following entities, attributes, and relations.

1. LIBRARY MEMBER whose attributes are:
   a. Member_id, a simple attribute and a primary key to uniquely identify each member.
   b. Fullname, a composite attribute containing a member’s first and last name.
   c. Email, a simple attribute
   d. Member_status, a derived attribute which can be active, suspended, or maximum items reached, to work out if a member still can request any more resources.
2. MEMBER TYPE whose attributes are:
   a. Quantity, a simple attribute to indicate how many resources each member type can hold.
   b. member_type, simple attribute and a primary key representing two different member types.

LIBRARY MEMBER and MEMBER TYPE are connected with the relationship “Library member is Member type”. This relationship is a One-to-Many relationship.

3. RESERVATION whose attributes are:
   a. Reserve_date, a simple attribute indicating a date when a loan offer is made.
   b. Reserve_duration, a simple attribute dictating how long an offer will last before being cancelled.
   c. Number_of_offer, a derived attribute to keep track of how many times an offer has been made. Once it reaches three, the offer of a resource to a member is cancelled.
   d. status, a derived attribute indicating if an offer has been taken, rejected, or cancelled.
4. LOAN whose attributes are:
   a. Loan_date, a simple attribute indicating a date when a loan is made.
   b. Duedate, a derived attribute indicating when a resource should be returned, depending on how many days are allowed for a particular resource.

    c. Return_date, a simple attribute indicating a date when a resource is returned.

The relationship of Library member to Reservation and Loan is “Make”. These two relationships are One-to-Many relationships

5. LOAN ARCHIVE, which is an entity of loan with the status “Returned”. It serves to keep track of popular resources.

LOAN and LOAN ARCHIVE are connected with the relationship “Loan Become Loan archive”. This relationship is a One-to-One relationship.

6. ITEM whose attributes are:
    a. Resource_id, a simple attribute and a primary key to identify each resource.
    b. Resource_name, a simple attribute defining a name of a resource.
    c. Days_allow, a simple attribute defining different loan duration of a resource.
    d. Creator, a simple attribute defining the creator of a resource.
    e. Year, a simple attribute defining the year a resource is published or created.
    f. Type, a simple attribute defining the type of a resource (e.g. CD, DVD, and Books).
    g. Class, a simple attribute defining the types of a resource, i.e., subject.
    h. Location, a composite attribute of shelf, floor, and code (In this case, code is the first three letters of its creator (used synonymously with author and director).

LOAN and RESERVATION relate to ITEM with a relationship of “Loan Include Item” and “Reservation Include Item”. These two relationships are One-to-Many.

7. COPY whose attribute is Copy_number, a simple attribute to identify different copies of a resource.

ITEM and COPY are connected with the relationship “Item Has Copy”. This relationship is a One-to-Many relationship.

8. FINE whose attributes are:
    a. paid, a simple attribute indicating if a fine has been paid or not (Boolean: true or false).

LOAN and FINE are connected with the relationship “Loan Incur Fine”. This relationship is a One-to-Many relationship

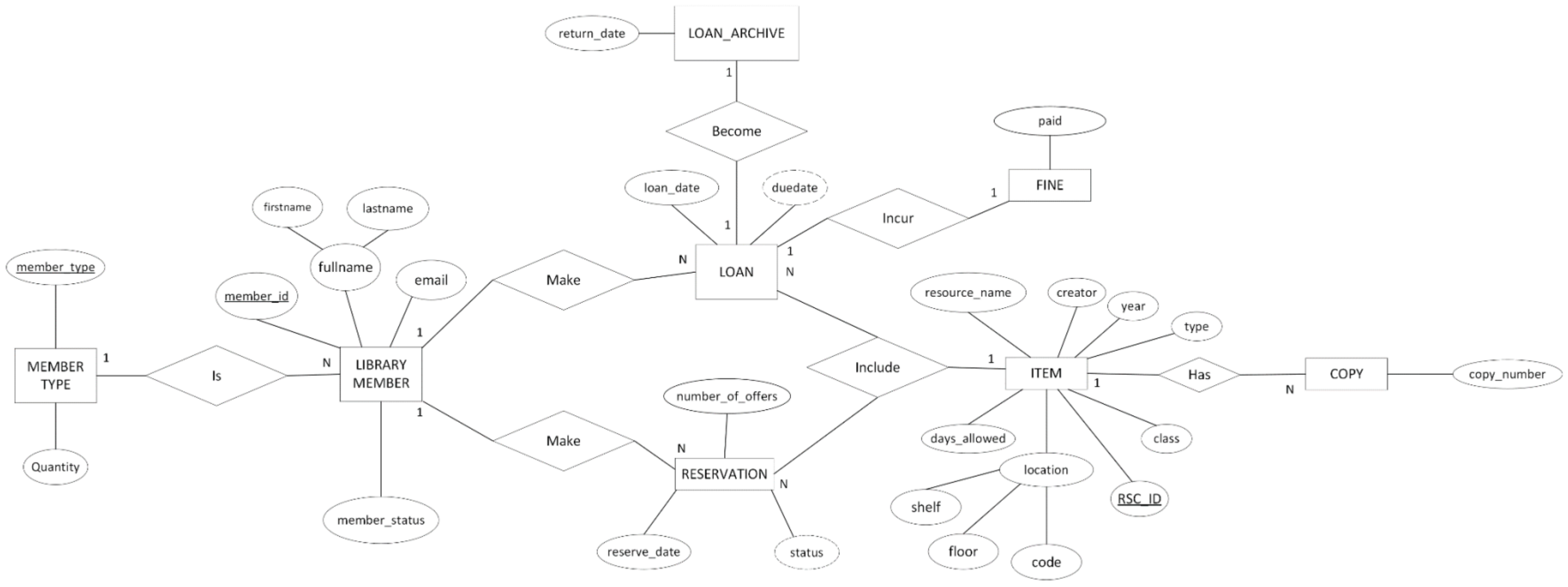


Figure 1 The Library ER Diagram

Mapping conceptual ER Model to a Relational Model

Using the ER diagram, where there is a 1 to N relationship between entity types, these have been mapped by taking a column from the 1-side of the relationship and added as a foreign key to the N-side. For example, the MEMBER TYPE entity type has a primary key, 'member_type', and this is placed as a foreign key in the LIBRARY MEMBER relation. This is the same for the COPY entity type, which is on the many side of the relationship between the ITEM entity type, so it has a foreign key, 'RSC_ID', in its relation. This foreign key in COPY becomes the primary key of the relation too.

As seen in the ER diagram, a single library member can have many loans and reservations. This has been mapped by placing the 'member_ID' primary key from the LIBRARY MEMBER entity type as a foreign key in the LOAN and RESERVATION relation. A similar situation is found with the relationships between items and loans, and items and reservations. A single item can be included in many loans and reservations, as each item has several copies. Therefore, the resource ID 'RSC_ID' from the ITEM entity type is added as a foreign key in the LOAN and RESERVATION relations. The combination of the 'RSC_ID' and 'member_id' foreign keys make up the primary key in the LOAN and RESERVATION relations.

Where there is a one-to-one relationship, such as between LOAN and LOAN ARCHIVE, we have inserted the primary key from the LOAN entity type to the LOAN ARCHIVE. The primary key consists of a combination of the resource ID and the members ID number. We have not assigned the LOAN ARCHIVE to have its own primary key as we decided it would have been redundant, and the table is an almost copy of the LOAN entity type with an extra attribute type in 'return_date'.

In the case of the one-to-one relationship between LOAN and FINE, the primary key from the LOAN entity type has been placed into the FINE relation, and again it was decided that a FINE would not have its own primary key.

The following is the mapping of the conceptual ER model to the relational model:

Notes:

- Primary key of table is underlined using a single line
- Foreign key is underlined using a dashed line
- If the key is both a primary and foreign key, it is underlined in italics

LIBRARY MEMBER (member_ID, firstname, lastname, email, member_type, member_status)

MEMBER TYPE (member_type, quantity)

LOAN (*member_id*, *RSC_ID*, loan_date)

LOAN ARCHIVE (*member_id*, *RSC_ID*, return_date)

RESERVATION (*member_id*, *RSC_ID*, number_of_offers, reserve_date)

ITEM (RSC_ID, floor, shelf, code, resource_name, year, creator, days_allowed, type, class)

COPY (*RSC_ID*, copy_number)

FINE (*member_id*, *RSC_ID*, paid)

Data Normalisation

We have resources with a unique ID known as 'RSC_ID', members with a unique ID known as 'member_ID', and member type as a unique value known as 'member_type'.

Using these, we can create the Universal relation:

Universal Relation (member_ID, RSC_ID, member_type, name, email, member_status, quantity, loan_date, return_date, number_of_offers, reserve_date, location, resource_name, year, creator, days_allowed, type, class, copy_number, paid)

First Normal Form (1 NF)

The composite attribute 'location' is broken down into single attributes of 'shelf', 'floor' and a 'code' which represents the first three letters of the creator's name, as well as the composite attribute 'name', which is split into 'firstname' and 'lastname'.

Location -> (shelf, floor, code)

Name -> (firstname, lastname)

All of the attribute types are now atomic, bringing this relation into 1 NF:

Universal Relation (member_ID, RSC_ID, member_type, firstname, lastname, email, member_status, quantity, loan_date, return_date, number_of_offers, reserve_date, floor, shelf, code, resource_name, year, creator, days_allowed, class, type, copy_number, paid)

Second Normal Form (2 NF) and Third Normal Form (3 NF)

For 2 NF we create relations where attribute types in a relation are fully functionally dependent on the primary key of the relation. For 3 NF we remove any transitive dependencies in the relations if they are present.

From the primary key 'member_ID' alone, we can determine firstname, lastname, email, member_type (foreign key), and member_status. This gives us the relation:

LIBRARY MEMBER (member_ID, firstname, lastname, email, member_type, member_status)

The attribute type 'quantity' is dependent fully on the primary key 'member_type.' This gives us the relation:

MEMBER TYPE (member_type, quantity)

From the final primary key 'RSC_ID' we can fully determine floor, shelf, code, resource_name, creator, days_allowed, type, class and copy_number, giving us the relation:

ITEM (RSC_ID, floor, shelf, code, resource_name, year, creator, days_allowed, type, class, copy_number)

The ITEM relation is broken down further, and a COPY relation is created to avoid repeating information in the ITEM relation. As each resource has multiple copies, the COPY relation is formed using the primary key ‘RSC_ID’ on which the ‘copy_number’ is fully dependent:

COPY (*RSC_ID*, copy_number)

For the remaining attribute types ‘loan_date, return_date, number_of_offers, reserve_date and paid’, these can only be derived by the combination of two foreign keys, ‘member_id’ and ‘RSC_ID,’ to become a primary key in the relation:

LOAN (*member_id*, *RSC_ID*, loan_date, return_date, number_of_offers, reserve_date, paid)

However, now the LOAN relation contains information on active loans, expired loans, reservations, and fines. We can split these off into their different entity types giving us the following sets of relations:

LOAN (*member_id*, *RSC_ID*, loan_date)

LOAN ARCHIVE (*member_id*, *RSC_ID*, return_date)

RESERVATION (*member_id*, *RSC_ID*, number_of_offers, reserve_date)

FINE (*member_id*, *RSC_ID*, paid)

Our final relations are now normalised up to 3 NF, as each attribute type in each relation is fully functionally dependent on the primary key of the relation and there are no transitive dependencies present.

Final set of normalised relations:

LIBRARY MEMBER (member_ID, firstname, lastname, email, member_type, member_status)

MEMBER TYPE (member_type, quantity)

LOAN (*member_id*, *RSC_ID*, loan_date)

LOAN ARCHIVE (*member_id*, *RSC_ID*, return_date)

RESERVATION (*member_id*, *RSC_ID*, number_of_offers, reserve_date)

ITEM (RSC_ID, floor, shelf, code, resource_name, year, creator, days_allowed, type, class)

COPY (*RSC_ID*, copy_number)

FINE (*member_id*, *RSC_ID*, paid)

Part 2 DATABASE IMPLEMENTATION

Create Table Command

```
/******** Creat Table Code ********/
DROP TABLE COPY;
DROP TABLE LOAN;
DROP TABLE LOAN_ARCHIVE;
DROP TABLE RESERVATION;
DROP TABLE FINE;
DROP TABLE LIBRARY_MEMBER;
DROP TABLE MEMBER_TYPE;
DROP TABLE ITEM;

DROP VIEW CATALOGUE;
DROP VIEW LOAN_INFORMATION;
DROP VIEW RESERVATION_INFORMATION;

CREATE TABLE MEMBER_TYPE
    (MEMBER_TYPE VARCHAR2(10) CONSTRAINT PK_MEMBER_TYPE PRIMARY
KEY,
    QUANTITY NUMBER(2));

INSERT INTO MEMBER_TYPE VALUES
    ('Student',5);
INSERT INTO MEMBER_TYPE VALUES
    ('Staff',10);

CREATE TABLE LIBRARY_MEMBER
    (MEMBER_ID NUMBER(4) CONSTRAINT PK_MEMBER_ID PRIMARY KEY,
    FIRSTNAME VARCHAR2(20),
    LASTNAME VARCHAR2(20),
    EMAIL VARCHAR2(40),
    MEMBER_TYPE VARCHAR2(7)/*Student or Staff*/,
    CONSTRAINT FK_MEMBER_TYPE FOREIGN KEY(MEMBER_TYPE)
REFERENCES MEMBER_TYPE,
    MEMBER_STATUS VARCHAR2(10)/*ACTIVE,SUSPEND,MAX BORROW*/);

INSERT INTO LIBRARY_MEMBER VALUES
(1001,'Lisa','Jordan','LisaKJordan@queenmary.ac.uk','Student',NU
LL);
INSERT INTO LIBRARY_MEMBER VALUES
(1002,'Allen','Hodges','AllenHHodges@queenmary.ac.uk','Student',
NULL);
INSERT INTO LIBRARY_MEMBER VALUES
(1003,'Raymond','Cable','RaymondSCable@queenmary.ac.uk','Student
',NULL);
```

```
INSERT INTO LIBRARY_MEMBER VALUES
(1004,'Maria','Jenkins','MariaRJenkins@queenmary.ac.uk','Staff',
NULL);
INSERT INTO LIBRARY_MEMBER VALUES
(1005,'Sarah','Dell','SarahCDell@queenmary.ac.uk','Staff',NULL);
INSERT INTO LIBRARY_MEMBER VALUES
(1006,'Ila','Pernell','IlaPPernell@queenmary.ac.uk','Staff',NULL
);

CREATE TABLE ITEM
    (RSC_ID NUMBER(6) CONSTRAINT PK_RSC_ID PRIMARY KEY/*resource
id*/,
    FLOOR VARCHAR2(3),
    SHELF VARCHAR2(3),
    CODE VARCHAR2(3),
    RESOURCE_NAME VARCHAR2(100),
    YEAR NUMBER(4),
    CREATOR VARCHAR2(50),
    DAYS_ALLOWED NUMBER(2),
    TYPE VARCHAR2(5),
    CLASS VARCHAR2(20));

INSERT INTO ITEM VALUES (100001,'F02','A02','COL','How to Draw
Cat and Dog', 2016,'Collins',14,'Book','Kids');
INSERT INTO ITEM VALUES (100002,'F01','A02','ROB','Natural
systems & human responses',2010,'Robert
Prosser',14,'Book','Biology');
INSERT INTO ITEM VALUES (100003,'F01','A02','CAL','Restless
Earth',2012,'Nigel Calder',14,'Book','Biology');
INSERT INTO ITEM VALUES (100004,'F02','A01','ADD','The organism
and the environment',2007,'John Adds and
others',14,'Book','Biology');
INSERT INTO ITEM VALUES (100005,'F02','A03','BIS','Home! Sweet
home!',2013,'Bishop H R',3,'DVD','Music');
INSERT INTO ITEM VALUES (100006,'F02','A03','HAY','my mother
bids me bind my hair',2000,'Haydn F J',3,'DVD','Music');
INSERT INTO ITEM VALUES (100007,'F03','A03','JIR','How to DIY A
Whole Library Resource Table',2022,'Jirapat
Boonmee',2,'Book','Computer Science');
INSERT INTO ITEM VALUES (100008,'F01','A02','GRO','How to Design
the Most Advance Database for A Library',2020,'Group
40',2,'Book','Computer Science');

CREATE TABLE COPY
    (RSC_ID NUMBER(6),
    CONSTRAINT FK_RSC_ID FOREIGN KEY(RSC_ID) REFERENCES ITEM,
    COPY_NUMBER NUMBER(2)/*how many copy is available*/);
```

```
INSERT INTO COPY VALUES (100001,01);
INSERT INTO COPY VALUES (100001,02);
INSERT INTO COPY VALUES (100001,03);
INSERT INTO COPY VALUES (100001,04);
INSERT INTO COPY VALUES (100001,05);
INSERT INTO COPY VALUES (100002,01);
INSERT INTO COPY VALUES (100002,02);
INSERT INTO COPY VALUES (100002,03);
INSERT INTO COPY VALUES (100002,04);
INSERT INTO COPY VALUES (100002,05);
INSERT INTO COPY VALUES (100003,01);
INSERT INTO COPY VALUES (100003,02);
INSERT INTO COPY VALUES (100003,03);
INSERT INTO COPY VALUES (100003,04);
INSERT INTO COPY VALUES (100003,05);
INSERT INTO COPY VALUES (100004,01);
INSERT INTO COPY VALUES (100004,02);
INSERT INTO COPY VALUES (100004,03);
INSERT INTO COPY VALUES (100004,04);
INSERT INTO COPY VALUES (100004,05);
INSERT INTO COPY VALUES (100005,01);
INSERT INTO COPY VALUES (100005,02);
INSERT INTO COPY VALUES (100005,03);
INSERT INTO COPY VALUES (100005,04);
INSERT INTO COPY VALUES (100005,05);
INSERT INTO COPY VALUES (100006,01);
INSERT INTO COPY VALUES (100006,02);
INSERT INTO COPY VALUES (100006,03);
INSERT INTO COPY VALUES (100006,04);
INSERT INTO COPY VALUES (100006,05);
INSERT INTO COPY VALUES (100007,01);
INSERT INTO COPY VALUES (100007,02);
INSERT INTO COPY VALUES (100007,03);
INSERT INTO COPY VALUES (100007,04);
INSERT INTO COPY VALUES (100007,05);
INSERT INTO COPY VALUES (100008,01);
INSERT INTO COPY VALUES (100008,02);
INSERT INTO COPY VALUES (100008,03);
INSERT INTO COPY VALUES (100008,04);
INSERT INTO COPY VALUES (100008,05);

CREATE TABLE LOAN
     (MEMBER_ID NUMBER(4),
     CONSTRAINT FK_LOAN_MEMBER_ID FOREIGN KEY(MEMBER_ID)
REFERENCES LIBRARY_MEMBER,
     RSC_ID NUMBER(6),
    CONSTRAINT FK_LOAN_RSC_ID FOREIGN KEY(RSC_ID) REFERENCES
ITEM,
    LOAN_DATE DATE);

INSERT INTO LOAN VALUES (1001,100003,TO_DATE('27-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1002,100006,TO_DATE('15-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1002,100007,TO_DATE('30-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1003,100003,TO_DATE('16-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1004,100002,TO_DATE('30-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1005,100003,TO_DATE('10-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1001,100008,TO_DATE('20-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1002,100004,TO_DATE('09-NOV-2022','DD-
MON-YYYY'));

CREATE TABLE LOAN_ARCHIVE
    (MEMBER_ID NUMBER(4),
    CONSTRAINT FK_ARCHIVE_MEMBER_ID FOREIGN KEY(MEMBER_ID)
REFERENCES LIBRARY_MEMBER,
    RSC_ID NUMBER(6),
    CONSTRAINT FK_ARCHIVE_RSC_ID FOREIGN KEY(RSC_ID) REFERENCES
ITEM,
    RETURN_DATE DATE);

INSERT INTO LOAN_ARCHIVE VALUES (1005,100004,TO_DATE('30-NOV-
2022','DD-MON-YYYY'));
INSERT INTO LOAN_ARCHIVE VALUES (1001,100004,TO_DATE('22-NOV-
2022','DD-MON-YYYY'));
INSERT INTO LOAN_ARCHIVE VALUES (1001,100005,TO_DATE('16-NOV-
2022','DD-MON-YYYY'));
INSERT INTO LOAN_ARCHIVE VALUES (1006,100007,TO_DATE('19-NOV-
2022','DD-MON-YYYY'));
INSERT INTO LOAN_ARCHIVE VALUES (1003,100004,TO_DATE('30-NOV-
2022','DD-MON-YYYY'));
INSERT INTO LOAN_ARCHIVE VALUES (1002,100008,TO_DATE('30-NOV-
2022','DD-MON-YYYY'));

CREATE TABLE RESERVATION
    (MEMBER_ID NUMBER(4) NOT NULL,
    CONSTRAINT FK_RESERVATION_MEMBER_ID FOREIGN KEY(MEMBER_ID)
REFERENCES LIBRARY_MEMBER,
    RSC_ID NUMBER(6) NOT NULL,
    CONSTRAINT FK_RESERVATION_RSC_ID FOREIGN KEY(RSC_ID)
REFERENCES ITEM,
    NUMBER_OF_OFFERS NUMBER(2)/*how many times the member have
been notified*/,
    RESERVE_DATE DATE);

INSERT INTO RESERVATION VALUES (1002,100005, 0, TO_DATE('20-NOV-
2022','DD-MON-YYYY'));
INSERT INTO RESERVATION VALUES (1005,100007, 0, TO_DATE('22-NOV-
2022','DD-MON-YYYY'));
INSERT INTO RESERVATION VALUES (1005,100003, 0, TO_DATE('24-NOV-
2022','DD-MON-YYYY'));
INSERT INTO RESERVATION VALUES (1006,100007, 0, TO_DATE('25-NOV-
2022','DD-MON-YYYY'));
INSERT INTO RESERVATION VALUES (1002,100008, 0, TO_DATE('27-NOV-
2022','DD-MON-YYYY'));
INSERT INTO RESERVATION VALUES (1004,100008, 0, TO_DATE('22-NOV-
2022','DD-MON-YYYY'));

CREATE TABLE FINE
    (MEMBER_ID NUMBER(4) NOT NULL,
    CONSTRAINT FK_FINE_MEMBER_ID FOREIGN KEY(MEMBER_ID)
REFERENCES LIBRARY_MEMBER,
    RSC_ID NUMBER(6),
    CONSTRAINT FK_FINE_RSC_ID FOREIGN KEY(RSC_ID) REFERENCES
ITEM,
    PAID NUMBER(1)/*check whether the fine is paid or not*/);

INSERT INTO FINE VALUES (1002,100006,0);
INSERT INTO FINE VALUES (1002,100007,0);
INSERT INTO FINE VALUES (1003,100003,0);
INSERT INTO FINE VALUES (1005,100003,0);
INSERT INTO FINE VALUES (1001,100008,0);
INSERT INTO FINE VALUES (1002,100004,0);
INSERT INTO FINE VALUES (1004,100005,1);
INSERT INTO FINE VALUES (1004,100006,1);

COMMIT;
```

Sample Test Data

LIBRARY MEMBER

member_id	First name	Last name	email	member_type	member_status
1001	Lisa	Jordan	LisaKJordan@queenmary.ac.uk	Student	
1002	Allen	Hodges	AllenHHodges@queenmary.ac.uk	Student	
1003	Raymond	Cable	RaymondSCable@queenmary.ac.uk	Student	
1004	Maria	Jenkins	MariaRJenkins@queenmary.ac.uk	Staff	
1005	Sarah	Dell	SarahCDell@queenmary.ac.uk	Staff	
1006	Ila	Pernell	IlaPPernell@queenmary.ac.uk	Staff	

MEMBER TYPE

Member type	Quantity
Student	5
Staff	10

LOAN

MEMBER_ID	RSC_ID	LOAN_DATE
1001	100003	27-Nov-22
1002	100006	15-Nov-22
1002	100007	30-Nov-22
1003	100003	16-Nov-22
1004	100002	30-Nov-22
1005	100003	10-Nov-22
1001	100008	20-Nov-22
1002	100004	09-Nov-22

LOAN ARCHIVE

MEMBER_ID	RSC_ID	RETURN_DATE
1005	100004	30-Nov-22
1001	100004	22-Nov-22
1001	100005	16-Nov-22
1006	100007	19-Nov-22
1003	100004	30-Nov-22
1002	100008	30-Nov-22

RESERVATION

MEMBER_ID	RSC_ID	NUMBER_OF_OFFERS	RESERVE_DATE
1002	100005	0	20-Nov-22
1005	100007	0	22-Nov-22
1005	100003	0	24-Nov-22
1006	100007	0	25-Nov-22
1002	100008	0	27-Nov-22
1004	100008	0	22-Nov-22

ITEM

Resource_ID	Floor	Shelf	Type	Code	resource_name	Class	Creator	Year	days_allowed
100001	F02	A01	Book	THO	Melodies of Scotland Vol.1	Music	Thomson G (ed)	2018	0
100002	F01	A02	Book	ROB	Natural systems & human responses	Biology	Robert Prosser	2010	14
100003	F01	A02	Book	CAL	Restless Earth	Biology	Nigel Calder	2012	14
100004	F02	A01	Book	ADD	The organism and the environment	Biology	John Adds and others	2007	14
100005	F02	A03	DVD	BIS	Home! Sweet home!	Music	Bishop H R	2013	3
100006	F02	A03	DVD	HAY	my mother bids me bind my hair	Music	Haydn F J	2000	3
100007	F03	A03	Book	JIR	How to DIY A Whole Library Resource Table	Computer Science	Jirapat Boonmee	2022	2
100008	F01	A02	Book	GRO	How to Design the Most Advance Database for A Library	Computer Science	Group 40	2020	2

COPY

Resource_ID	Copy_number
100001	01
100001	02
100001	03
100001	04
100001	05
100002	01
100002	02
100002	03
100002	04
100002	05
100003	01
100003	02
100003	03
100003	04
100003	05
100004	01
100004	02
100004	03
100004	04
100004	05
100005	01
100005	02
100005	03
100005	04
100005	05
100006	01
100006	02
100006	03
100006	04
100006	05
100007	01
100007	02
100007	03
100007	04
100007	05
100008	01
100008	02
100008	03
100008	04
100008	05

FINE

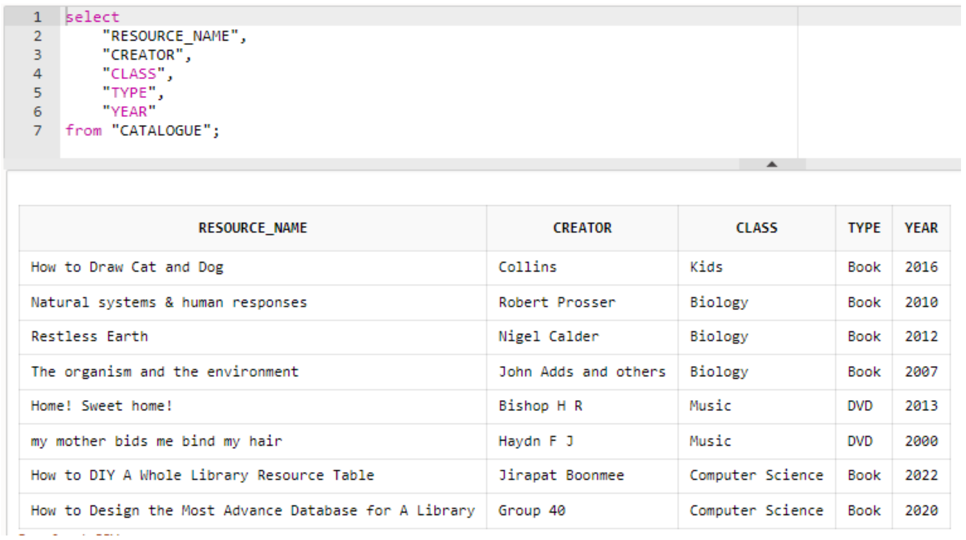
MEMBER_ID	RSC_ID	PAID
1002	100006	0
1002	100007	0
1003	100003	0
1005	100003	0
1001	100008	0
1002	100004	0
1004	100005	1
1004	100006	1

View Definitions

CATALOGUE View

CATALOGUE view will show all the item information available in the item table.

```
CREATE VIEW CATALOGUE AS
    SELECT RESOURCE_NAME, CREATOR, ITEM.CLASS, ITEM.TYPE,
ITEM.YEAR
    FROM ITEM;
```



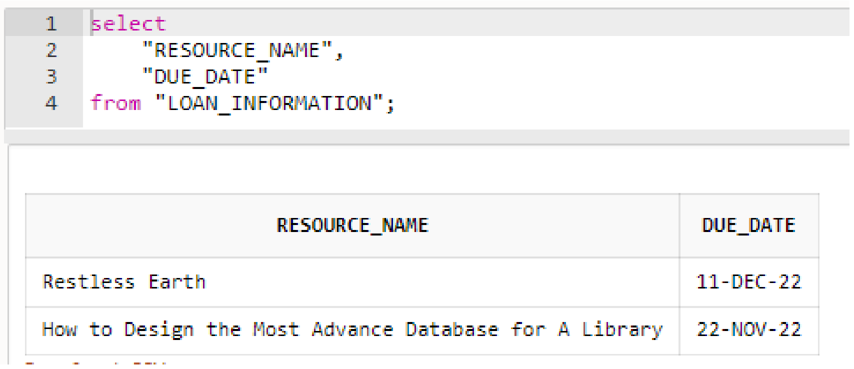
```
select
    "RESOURCE_NAME",
    "CREATOR",
    "CLASS",
    "TYPE",
    "YEAR"
from "CATALOGUE";
```

RESOURCE_NAME	CREATOR	CLASS	TYPE	YEAR
How to Draw Cat and Dog	Collins	Kids	Book	2016
Natural systems & human responses	Robert Prosser	Biology	Book	2010
Restless Earth	Nigel Calder	Biology	Book	2012
The organism and the environment	John Adds and others	Biology	Book	2007
Home! Sweet home!	Bishop H R	Music	DVD	2013
my mother bids me bind my hair	Haydn F J	Music	DVD	2000
How to DIY A Whole Library Resource Table	Jirapat Boonmee	Computer Science	Book	2022
How to Design the Most Advance Database for A Library	Group 40	Computer Science	Book	2020

LOAN_INFORMATION View

LOAN_INFORMATION view will show the items that a member loans. In this scenario, the member is 1001.

```
CREATE VIEW LOAN_INFORMATION (RESOURCE_NAME, DUE_DATE) AS
    SELECT RESOURCE_NAME,LOAN_DATE+DAYS_ALLOWED
    FROM ITEM, LOAN
    WHERE LOAN.MEMBER_ID = 1001 AND ITEM.RSC_ID = LOAN.RSC_ID;
```



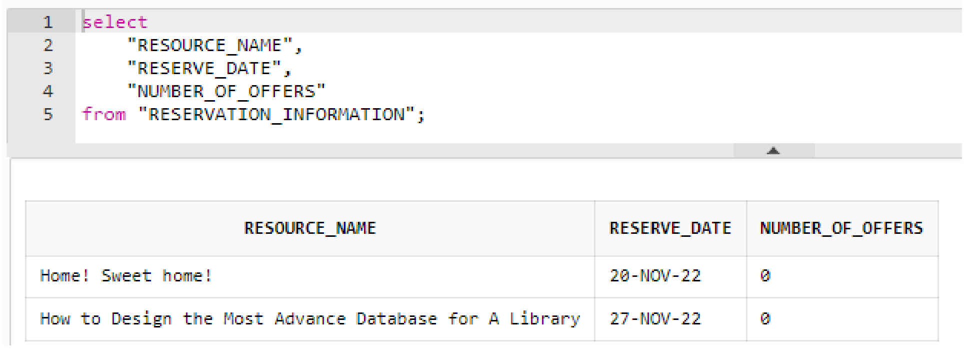
```
select
    "RESOURCE_NAME",
    "DUE_DATE"
from "LOAN_INFORMATION";
```

RESOURCE_NAME	DUE_DATE
Restless Earth	11-DEC-22
How to Design the Most Advance Database for A Library	22-NOV-22

RESERVATION_INFORMATION View

RESERVATION_INFORMATION view will show the items that are on reservation by a member and number of times they have been notified to pick up the item. In this scenario, the member is 1002.

```
CREATE VIEW RESERVATION_INFORMATION AS
    SELECT RESOURCE_NAME, RESERVE_DATE, NUMBER_OF_OFFERS
    FROM ITEM, RESERVATION
    WHERE RESERVATION.MEMBER_ID = 1002 AND RESERVATION.RSC_ID =
ITEM.RSC_ID;
```



```
select
    "RESOURCE_NAME",
    "RESERVE_DATE",
    "NUMBER_OF_OFFERS"
from "RESERVATION_INFORMATION";
```

RESOURCE_NAME	RESERVE_DATE	NUMBER_OF_OFFERS
Home! Sweet home!	20-NOV-22	0
How to Design the Most Advance Database for A Library	27-NOV-22	0

Triggers

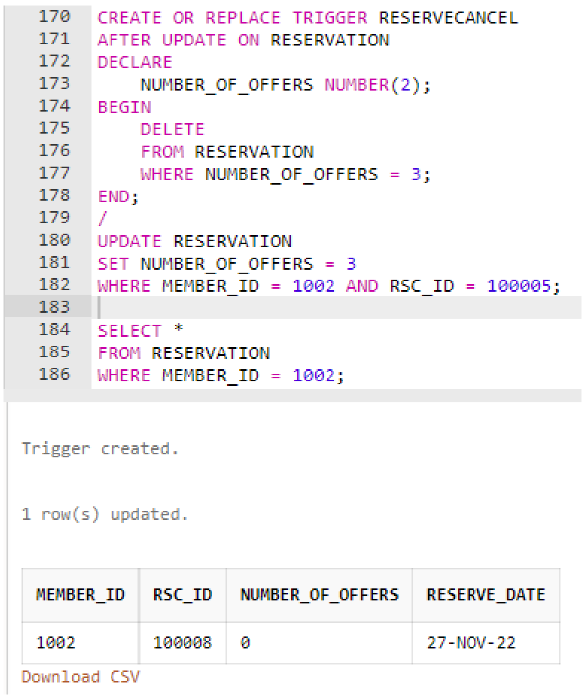
RESERVECANCEL Trigger

Reservecancel trigger is used to cancel the reservation of a member when the number of times the member is notified is equal to three. When this happens, the trigger will delete that reservation row. In this testing scenario, member 1002 have reserve resource 100008 and 100005. The number_of_offers on resource 100005 is updated to three.

```
CREATE OR REPLACE TRIGGER RESERVECANCEL
AFTER UPDATE ON RESERVATION
DECLARE
    NUMBER_OF_OFFERS NUMBER(2);
BEGIN
    DELETE
    FROM RESERVATION
    WHERE NUMBER_OF_OFFERS = 3;
END;

UPDATE RESERVATION
SET NUMBER_OF_OFFERS = 3
WHERE MEMBER_ID = 1002 AND RSC_ID = 100005;

SELECT *
FROM RESERVATION
WHERE MEMBER_ID = 1002;
```



```
CREATE OR REPLACE TRIGGER RESERVECANCEL
AFTER UPDATE ON RESERVATION
DECLARE
    NUMBER_OF_OFFERS NUMBER(2);
BEGIN
    DELETE
    FROM RESERVATION
    WHERE NUMBER_OF_OFFERS = 3;
END;
/
UPDATE RESERVATION
SET NUMBER_OF_OFFERS = 3
WHERE MEMBER_ID = 1002 AND RSC_ID = 100005;

SELECT *
FROM RESERVATION
WHERE MEMBER_ID = 1002;
```

```
Trigger created.

1 row(s) updated.
```

MEMBER_ID	RSC_ID	NUMBER_OF_OFFERS	RESERVE_DATE
1002	100008	0	27-NOV-22

Download CSV

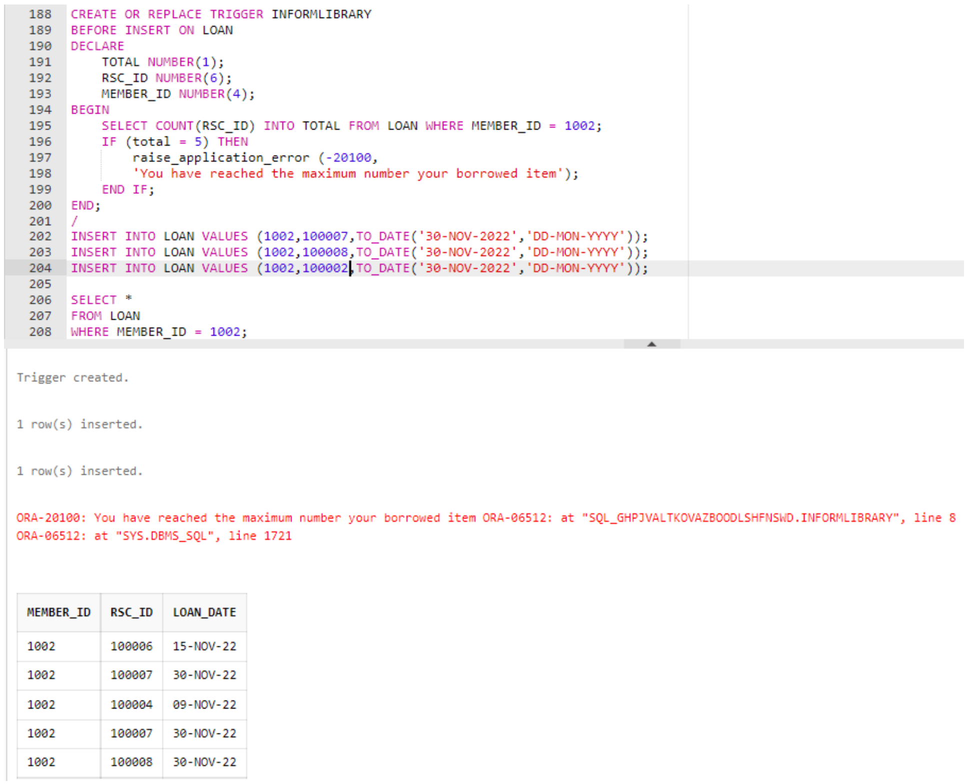
INFORMLIBRARY Trigger

Informlibrary trigger is used to notify members that they have reached the maximum number of items they can loan. In this testing scenario, member 1002 has already borrowed three items and decided to borrow three more. Before the six rows of data are inserted, the SQL code will raise an error and prevent the data from being inserted into the table.

```
CREATE OR REPLACE TRIGGER INFORMLIBRARY
BEFORE UPDATE ON LOAN
DECLARE
    TOTAL NUMBER(1);
    RSC_ID NUMBER(6);
    MEMBER_ID NUMBER(4);
BEGIN
    SELECT COUNT(RSC_ID) INTO TOTAL FROM LOAN WHERE MEMBER_ID =
100001;
    IF (total = 5) THEN
        raise_application_error (-20100,
        'You have reached the maximum number your borrowed
item');
    END IF;
END;

INSERT INTO LOAN VALUES (1002,100007,TO_DATE('30-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1002,100008,TO_DATE('30-NOV-2022','DD-
MON-YYYY'));
INSERT INTO LOAN VALUES (1002,100002,TO_DATE('30-NOV-2022','DD-
MON-YYYY'));

SELECT *
FROM LOAN
WHERE MEMBER_ID = 1002;
```



```sql
CREATE OR REPLACE TRIGGER INFORMLIBRARY
BEFORE INSERT ON LOAN
DECLARE
    TOTAL NUMBER(1);
    RSC_ID NUMBER(6);
    MEMBER_ID NUMBER(4);
BEGIN
    SELECT COUNT(RSC_ID) INTO TOTAL FROM LOAN WHERE MEMBER_ID = 1002;
    IF (total = 5) THEN
        raise_application_error (-20100,
        'You have reached the maximum number your borrowed item');
    END IF;
END;
/
INSERT INTO LOAN VALUES (1002,100007,TO_DATE('30-NOV-2022','DD-MON-YYYY'));
INSERT INTO LOAN VALUES (1002,100008,TO_DATE('30-NOV-2022','DD-MON-YYYY'));
INSERT INTO LOAN VALUES (1002,100002,TO_DATE('30-NOV-2022','DD-MON-YYYY'));

SELECT *
FROM LOAN
WHERE MEMBER_ID = 1002;
```

```
Trigger created.


1 row(s) inserted.


1 row(s) inserted.


ORA-20100: You have reached the maximum number your borrowed item ORA-06512: at "SQL_GHPJVALTKOVAZBOODLSHFNSWD.INFORMLIBRARY", line 8
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

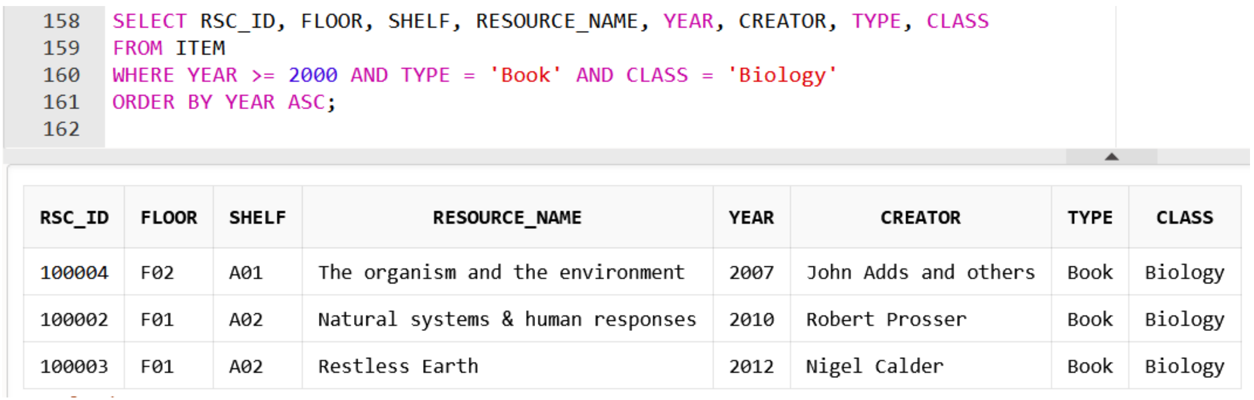
MEMBER_ID	RSC_ID	LOAN_DATE
1002	100006	15-NOV-22
1002	100007	30-NOV-22
1002	100004	09-NOV-22
1002	100007	30-NOV-22
1002	100008	30-NOV-22

SQL Queries

Query 1

Display resources which are biology books from the second floor which was created before the year 2000, sorted in ascending order

```sql
SELECT RSC_ID, FLOOR, SHELF, RESOURCE_NAME, YEAR, CREATOR, TYPE,
CLASS
FROM ITEM
WHERE YEAR >= 2000 AND TYPE = 'Book' AND CLASS = 'Biology'
ORDER BY YEAR ASC;
```



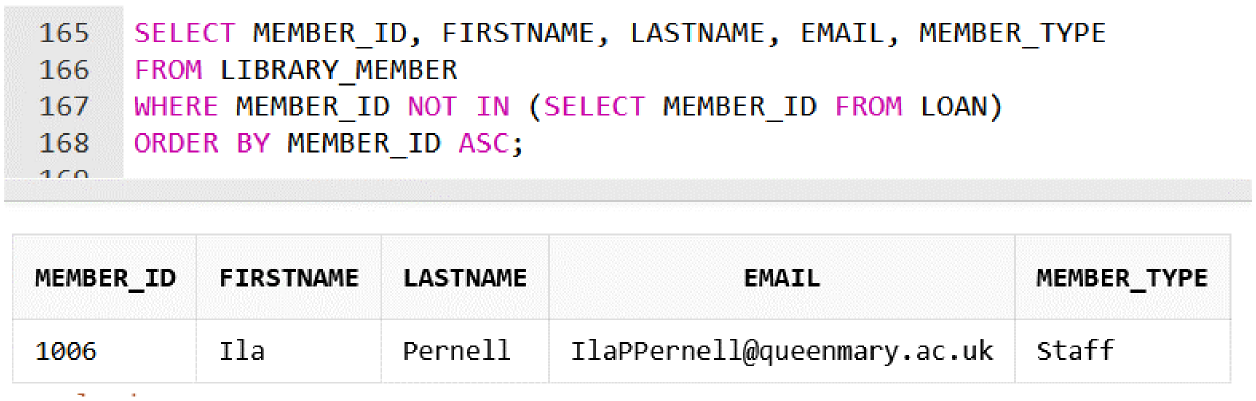
```sql
SELECT RSC_ID, FLOOR, SHELF, RESOURCE_NAME, YEAR, CREATOR, TYPE, CLASS
FROM ITEM
WHERE YEAR >= 2000 AND TYPE = 'Book' AND CLASS = 'Biology'
ORDER BY YEAR ASC;
```

RSC_ID	FLOOR	SHELF	RESOURCE_NAME	YEAR	CREATOR	TYPE	CLASS
100004	F02	A01	The organism and the environment	2007	John Adds and others	Book	Biology
100002	F01	A02	Natural systems & human responses	2010	Robert Prosser	Book	Biology
100003	F01	A02	Restless Earth	2012	Nigel Calder	Book	Biology

Query 2

Display library members who do not currently hold any resource, sorted in ascending order.

```sql
SELECT MEMBER_ID, FIRSTNAME, LASTNAME, EMAIL, MEMBER_TYPE
FROM LIBRARY_MEMBER
WHERE MEMBER_ID NOT IN (SELECT MEMBER_ID FROM LOAN)
ORDER BY MEMBER_ID ASC;
```



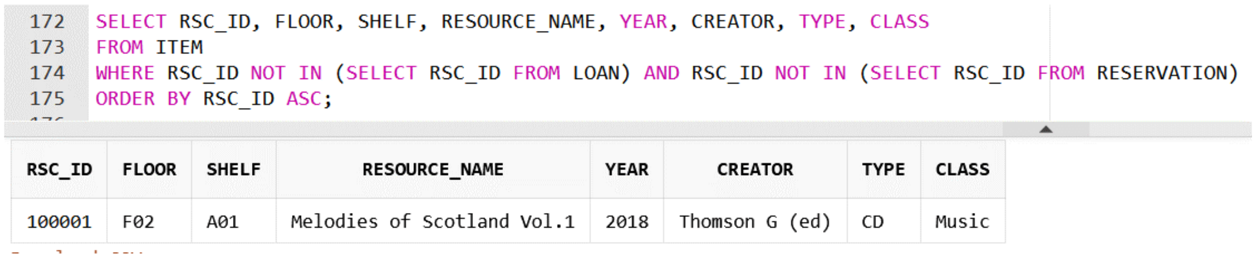
```sql
SELECT MEMBER_ID, FIRSTNAME, LASTNAME, EMAIL, MEMBER_TYPE
FROM LIBRARY_MEMBER
WHERE MEMBER_ID NOT IN (SELECT MEMBER_ID FROM LOAN)
ORDER BY MEMBER_ID ASC;
```

MEMBER_ID	FIRSTNAME	LASTNAME	EMAIL	MEMBER_TYPE
1006	Ila	Pernell	IlaPPernell@queenmary.ac.uk	Staff

Query 3

Display resources not currently on loan or reserved, sorted in ascending order.

```
SELECT RSC_ID, FLOOR, SHELF, RESOURCE_NAME, YEAR, CREATOR, TYPE,
CLASS
FROM ITEM
WHERE RSC_ID NOT IN (SELECT RSC_ID FROM LOAN) AND RSC_ID NOT IN
(SELECT RSC_ID FROM RESERVATION)
ORDER BY RSC_ID ASC;
```



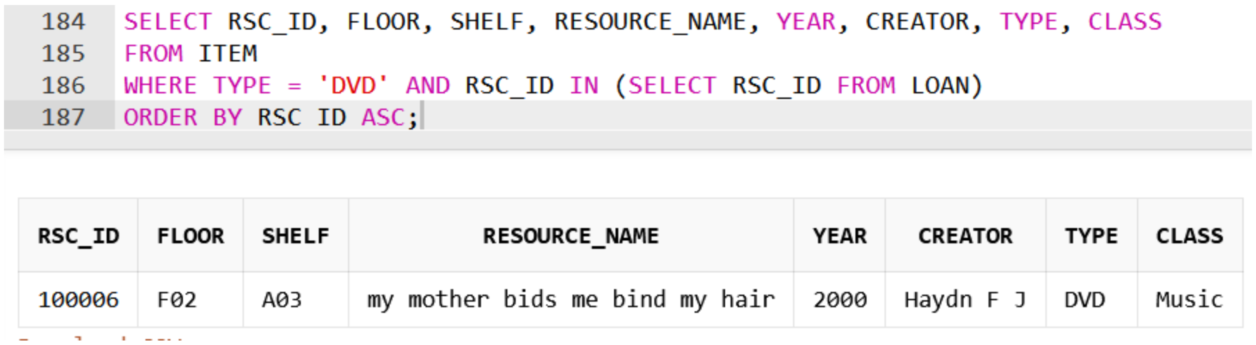
```
SELECT RSC_ID, FLOOR, SHELF, RESOURCE_NAME, YEAR, CREATOR, TYPE, CLASS
FROM ITEM
WHERE RSC_ID NOT IN (SELECT RSC_ID FROM LOAN) AND RSC_ID NOT IN (SELECT RSC_ID FROM RESERVATION)
ORDER BY RSC_ID ASC;
```

RSC_ID	FLOOR	SHELF	RESOURCE_NAME	YEAR	CREATOR	TYPE	CLASS
100001	F02	A01	Melodies of Scotland Vol.1	2018	Thomson G (ed)	CD	Music

Query 4

Display DVDs which are currently on loan.

```
SELECT RSC_ID, FLOOR, SHELF, RESOURCE_NAME, YEAR, CREATOR, TYPE,
CLASS
FROM ITEM
WHERE TYPE = 'DVD' AND RSC_ID IN (SELECT RSC_ID FROM LOAN)
ORDER BY RSC_ID ASC;
```



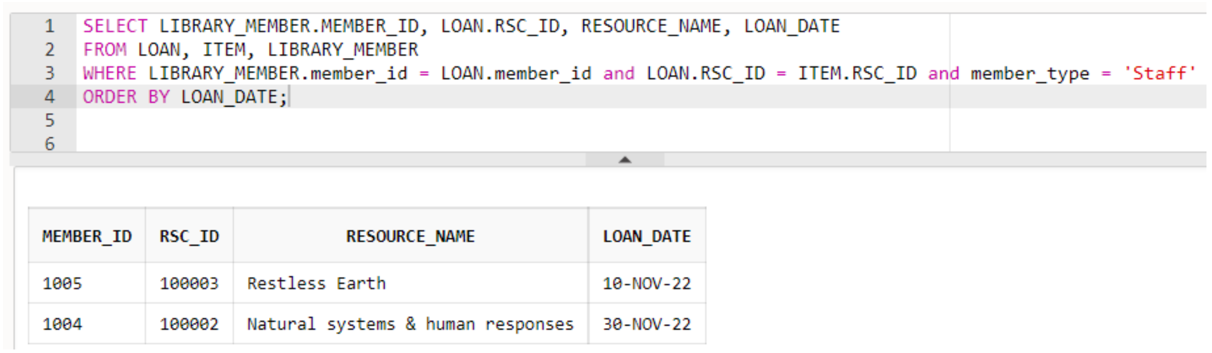
```
SELECT RSC_ID, FLOOR, SHELF, RESOURCE_NAME, YEAR, CREATOR, TYPE, CLASS
FROM ITEM
WHERE TYPE = 'DVD' AND RSC_ID IN (SELECT RSC_ID FROM LOAN)
ORDER BY RSC ID ASC;
```

RSC_ID	FLOOR	SHELF	RESOURCE_NAME	YEAR	CREATOR	TYPE	CLASS
100006	F02	A03	my mother bids me bind my hair	2000	Haydn F J	DVD	Music

Query 5

Display names and IDs of all resources on loan by staff members in order by earliest loan date.

```
SELECT LIBRARY_MEMBER.MEMBER_ID, LOAN.RSC_ID, RESOURCE_NAME,
LOAN_DATE
FROM LOAN, ITEM, LIBRARY_MEMBER
WHERE LIBRARY_MEMBER.member_id = LOAN.member_id and LOAN.RSC_ID
= ITEM.RSC_ID and member_type = 'Staff'
ORDER BY LOAN_DATE;
```



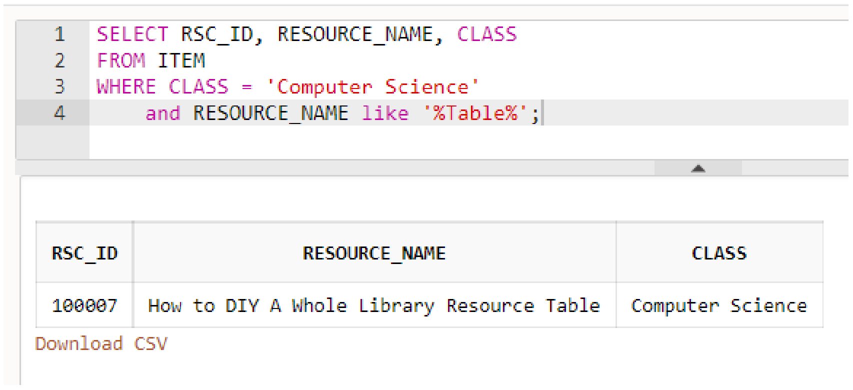
```
SELECT LIBRARY_MEMBER.MEMBER_ID, LOAN.RSC_ID, RESOURCE_NAME, LOAN_DATE
FROM LOAN, ITEM, LIBRARY_MEMBER
WHERE LIBRARY_MEMBER.member_id = LOAN.member_id and LOAN.RSC_ID = ITEM.RSC_ID and member_type = 'Staff'
ORDER BY LOAN_DATE;
```

MEMBER_ID	RSC_ID	RESOURCE_NAME	LOAN_DATE
1005	100003	Restless Earth	10-NOV-22
1004	100002	Natural systems & human responses	30-NOV-22

Query 6

Display all resources belonging to computer science classes where the word 'table' is in the item's name.

```
SELECT RSC_ID, RESOURCE_NAME, CLASS
FROM ITEM
WHERE CLASS = 'Computer Science'
    and RESOURCE_NAME like '%Table%';
```



```
SELECT RSC_ID, RESOURCE_NAME, CLASS
FROM ITEM
WHERE CLASS = 'Computer Science'
    and RESOURCE_NAME like '%Table%';
```

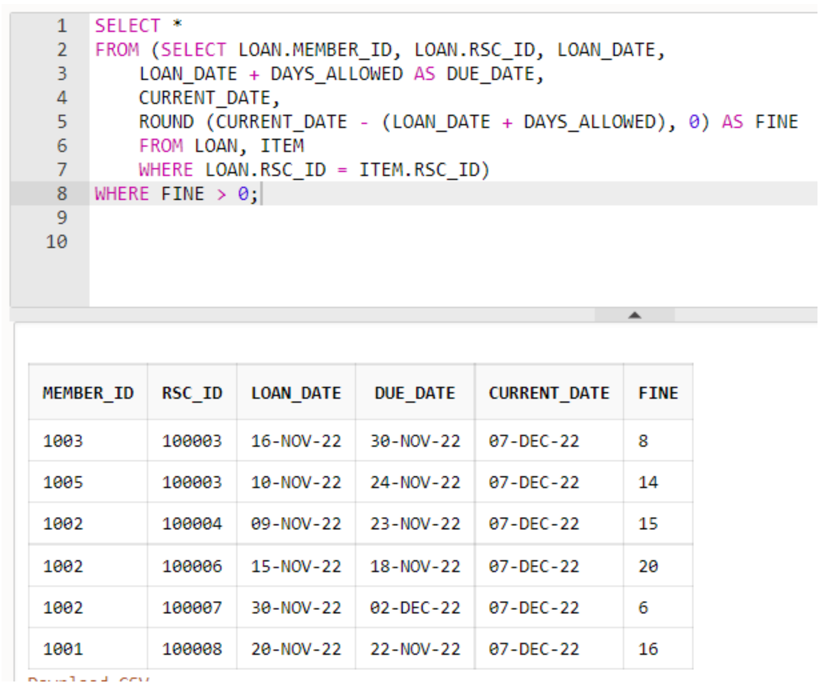
RSC_ID	RESOURCE_NAME	CLASS
100007	How to DIY A Whole Library Resource Table	Computer Science

Download CSV

Query 7

Display loan information of members with overdue loans and who have not yet paid their fines. Include the fine amount to the nearest £GBP for each loan up to the current date (7/12/2022).

```
SELECT *
FROM (SELECT LOAN.MEMBER_ID, LOAN.RSC_ID, LOAN_DATE,
    LOAN_DATE + DAYS_ALLOWED AS DUE_DATE,
    CURRENT_DATE,
    ROUND (CURRENT_DATE - (LOAN_DATE + DAYS_ALLOWED), 0) AS FINE
    FROM LOAN, ITEM
    WHERE LOAN.RSC_ID = ITEM.RSC_ID)
WHERE FINE > 0;
```



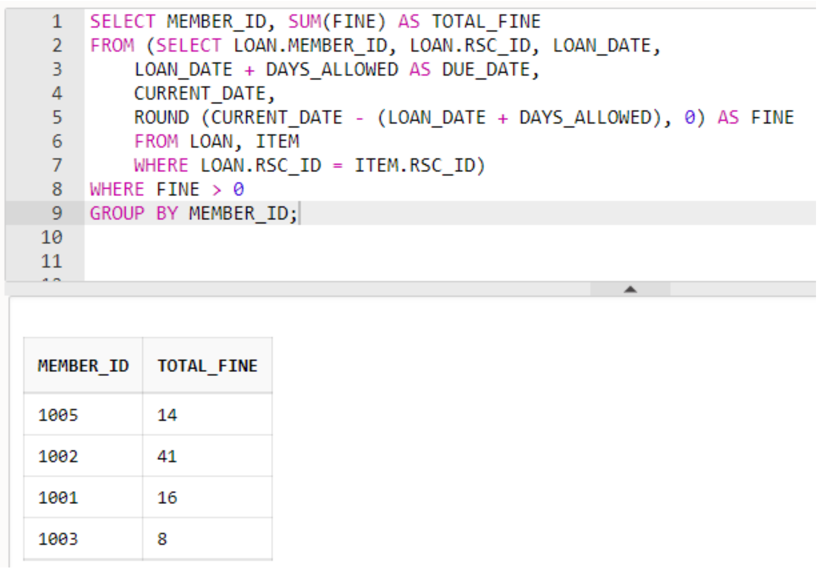
```
SELECT *
FROM (SELECT LOAN.MEMBER_ID, LOAN.RSC_ID, LOAN_DATE,
    LOAN_DATE + DAYS_ALLOWED AS DUE_DATE,
    CURRENT_DATE,
    ROUND (CURRENT_DATE - (LOAN_DATE + DAYS_ALLOWED), 0) AS FINE
    FROM LOAN, ITEM
    WHERE LOAN.RSC_ID = ITEM.RSC_ID)
WHERE FINE > 0;
```

MEMBER_ID	RSC_ID	LOAN_DATE	DUE_DATE	CURRENT_DATE	FINE
1003	100003	16-NOV-22	30-NOV-22	07-DEC-22	8
1005	100003	10-NOV-22	24-NOV-22	07-DEC-22	14
1002	100004	09-NOV-22	23-NOV-22	07-DEC-22	15
1002	100006	15-NOV-22	18-NOV-22	07-DEC-22	20
1002	100007	30-NOV-22	02-DEC-22	07-DEC-22	6
1001	100008	20-NOV-22	22-NOV-22	07-DEC-22	16

Query 8

Display a list of library members and the total fines they owe up until the current date (7/12/2022) and rounded to the nearest £GBP.

```
SELECT MEMBER_ID, SUM(FINE) AS TOTAL_FINE
FROM (SELECT LOAN.MEMBER_ID, LOAN.RSC_ID, LOAN_DATE,
    LOAN_DATE + DAYS_ALLOWED AS DUE_DATE,
    CURRENT_DATE,
    ROUND (CURRENT_DATE - (LOAN_DATE + DAYS_ALLOWED), 0) AS FINE
    FROM LOAN, ITEM
    WHERE LOAN.RSC_ID = ITEM.RSC_ID)
WHERE FINE > 0
GROUP BY MEMBER_ID;
```



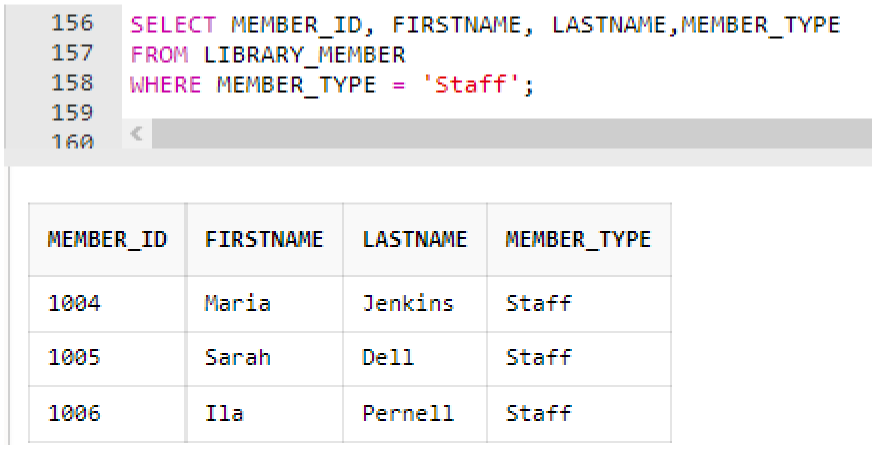
```
SELECT MEMBER_ID, SUM(FINE) AS TOTAL_FINE
FROM (SELECT LOAN.MEMBER_ID, LOAN.RSC_ID, LOAN_DATE,
    LOAN_DATE + DAYS_ALLOWED AS DUE_DATE,
    CURRENT_DATE,
    ROUND (CURRENT_DATE - (LOAN_DATE + DAYS_ALLOWED), 0) AS FINE
    FROM LOAN, ITEM
    WHERE LOAN.RSC_ID = ITEM.RSC_ID)
WHERE FINE > 0
GROUP BY MEMBER_ID;
```

MEMBER_ID	TOTAL_FINE
1005	14
1002	41
1001	16
1003	8

Query 9

Display all members who are staff

```
SELECT MEMBER_ID, FIRSTNAME, LASTNAME,MEMBER_TYPE
FROM LIBRARY_MEMBER
WHERE MEMBER_TYPE = 'Staff';
```



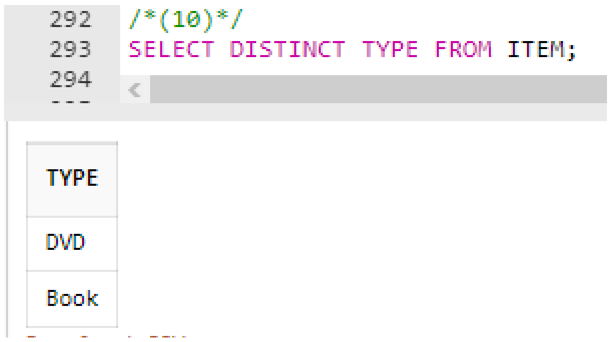
```
SELECT MEMBER_ID, FIRSTNAME, LASTNAME,MEMBER_TYPE
FROM LIBRARY_MEMBER
WHERE MEMBER_TYPE = 'Staff';
```

MEMBER_ID	FIRSTNAME	LASTNAME	MEMBER_TYPE
1004	Maria	Jenkins	Staff
1005	Sarah	Dell	Staff
1006	Ila	Pernell	Staff

Query 10

Display all unique types of resources in the library

```
SELECT DISTINCT TYPE FROM ITEM;
```



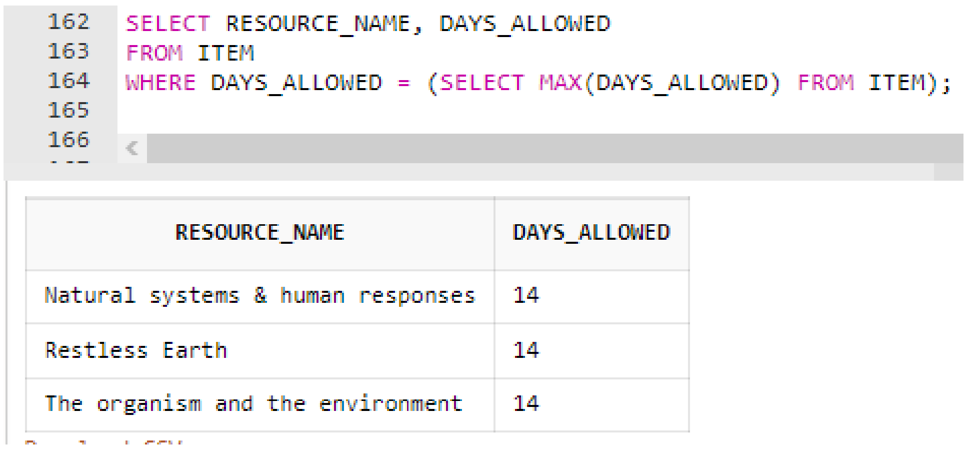
```
/*(10)*/
SELECT DISTINCT TYPE FROM ITEM;
```

TYPE
DVD
Book

Query 11

Display items that have the most extended duration allowed.

```
SELECT RESOURCE_NAME, DAYS_ALLOWED
FROM ITEM
WHERE DAYS_ALLOWED = (SELECT MAX(DAYS_ALLOWED) FROM ITEM);
```



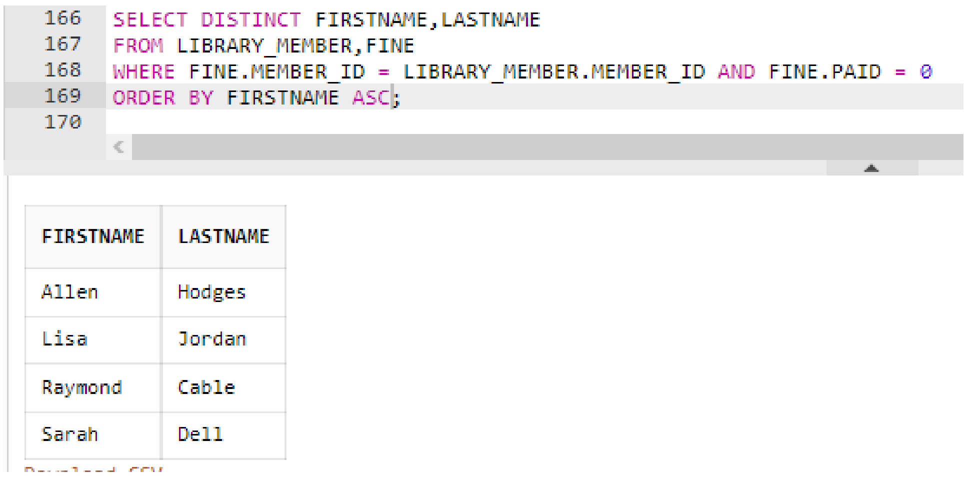
```
SELECT RESOURCE_NAME, DAYS_ALLOWED
FROM ITEM
WHERE DAYS_ALLOWED = (SELECT MAX(DAYS_ALLOWED) FROM ITEM);
```

RESOURCE_NAME	DAYS_ALLOWED
Natural systems & human responses	14
Restless Earth	14
The organism and the environment	14

Query 12

Display all library members who have not paid the fine.

```
SELECT DISTINCT FIRSTNAME,LASTNAME
FROM LIBRARY_MEMBER,FINE
WHERE FINE.MEMBER_ID = LIBRARY_MEMBER.MEMBER_ID AND FINE.PAID =
0
ORDER BY FIRSTNAME ASC;
```



```
SELECT DISTINCT FIRSTNAME,LASTNAME
FROM LIBRARY_MEMBER,FINE
WHERE FINE.MEMBER_ID = LIBRARY_MEMBER.MEMBER_ID AND FINE.PAID = 0
ORDER BY FIRSTNAME ASC;
```

FIRSTNAME	LASTNAME
Allen	Hodges
Lisa	Jordan
Raymond	Cable
Sarah	Dell

References

123ProjectLab. (n.d.). *ER Diagram for Library Management System*. Retrieved from 123ProjectLab: https://123projectlab.com/er-diagram-for-library-management-system/

Cacoo. (2022, February 9). *ER diagrams vs. EER diagrams: What's the difference?* Retrieved from Nulabs: https://nulab.com/learn/software-development/er-diagrams-vs-eer-diagrams-whats-the-difference/

Creatly. (n.d.). *E-R Diagram of Library Management System*. Retrieved from Creatly: https://creately.com/diagram/example/h7cw0wrb1/e-r-diagram-of-library-management-system

Database Star. (2021, June 22). *Database Design for a Library Management System*. Retrieved from Youtube: https://www.youtube.com/watch?v=yldrIdoXiYk

Edrawsoft. (n.d.). *What is an EER Diagram?* Retrieved from Edrawsoft: https://www.edrawsoft.com/article/what-is-eer-diagram.html

freeCodeCamp.org. (2018, July 2). *SQL Tutorial - Full Database Course for Beginners*. Retrieved from Youtube: https://www.youtube.com/watch?v=HXV3zeQKqGY

Geeksforgeeks. (2021, September 8). *Introduction of ER Model*. Retrieved from Geeksforgeeks: https://www.geeksforgeeks.org/introduction-of-er-model/

International Standard Serial Number, International Centre. (n.d.). *What is an ISSN?* Retrieved from International Standard Serial Number, International Centre: https://www.issn.org/understanding-the-issn/what-is-an-issn/

Jain, M. (2022, May 25). *Types of Attributes in DBMS*. Retrieved from Scaler Topics: https://www.scaler.com/topics/types-of-attributes-in-dbms/

Juniata College. (2019, September 2019). *The Entity-Relationship Model*. Retrieved from Juniata College: https://jcsites.juniata.edu/faculty/rhodes/dbms/ermodel.htm

Library of Congress. (n.d.). *Library of Congress Classification Outline*. Retrieved from Library of Congress: https://www.loc.gov/catdir/cpso/lcco/

Navathe, E. (2011). *Fundamental of Database Systems.* London: Pearson Education, Inc.

Park, J. (n.d.). The Enhanced EntityRelationship (EER) Model. Seoul, Korea: Depertment of Industrial Engineering, Seoul National University.

Soni, D. (2022, October 19). *Types of Relationship in DBMS*. Retrieved from Scaler Topics: https://www.scaler.com/topics/types-of-relationship-in-dbms/

Tutorialspoint. (n.d.). *ER Diagram Representation*. Retrieved from Tutorialspoint: https://www.tutorialspoint.com/dbms/er_diagram_representation.htm#:~:text=Total%20Participation%20%E2%88%92%20Each%20entity%20is,is%20represented%20by%20single%20lines.

University of Pennsylvenia. (n.d.). *Types of Materials*. Retrieved from Penn Libraries, University of Pennsylvenia: https://guides.library.upenn.edu/scholarlycommons/materials

Waltham Forest Library. (n.d.). London, United Kingdom. Retrieved November 19, 2022, from https://www.walthamforest.gov.uk/libraries

Wikipedia. (n.d.). *Library of Congress Classification*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Library_of_Congress_Classification#cite_note-:0-10