**MSc Computing and Information Systems**

**ECS740 Database Systems Coursework**

Date due: December 1st, 2023, 12.00

By

Group 21

Ritika Mishra - 230965889

Lorena Estrada - 230149445

Catherine Gunaseelan - 210469969

Lorraine Bonney - 231025876

# CONTENTS

[CONTENTS 1](#_Toc1522554650)

[Part 1 – Database Design for a College Library System 2](#_Toc1229788035)

[LIST OF ALL ASSUMPTIONS: 3](#_Toc1402971573)

[1A. The Conceptual Schema (UML diagram) 3](#_Toc1933360862)

[1B. The relational database schema 5](#_Toc820155988)

[1C. Normalisation 8](#_Toc142628093)

[Part2: Implementation of the College Library System 13](#_Toc1380055260)

[2A. Table creation 14](#_Toc1249217122)

[2B. Views Code 25](#_Toc2127473156)

[2C. Query Output 27](#_Toc784902621)

[Data Dictionary 39](#_Toc96332652)

[References: 41](#_Toc1744015278)

# Part 1 – Database Design for a College Library System

## LIST OF ALL ASSUMPTIONS:

* Members (such as Student/Staff) will use their lib\_cardNo which will be a unique number to access the resources.
* Students will have a maximum of 5 resources per loan. However, Staff will be entitled to a maximum of 10 resources per loan.
* Members get fines, which must be paid. However, if the amount exceeds £10, then their status will change from active to suspended.
* All resources will be tracked by a shelf, class, and floor number within the library.
* Each resource pertaining to a particular subject will have a class number.
* The default loan period is 2 weeks for books. However, there is a 2-day period for short loans for items such as CD’s and DVD’s. On the other hand, some reference-only resources can only be accessed within the library such as video. This is because a member will require their unique login and password.
* A reservation will be generated for unavailable resources requested by members.
* Any member who has been suspended cannot loan new items, unless they clear the debt and return the resources.
* The database will be accessible to all authorised members.
* The database will be secure and protected from unauthorised access.
* The database will allow members to perform the following tasks:
  + Search for resources (e.g., books, videos, CDs and DVDs) by title
  + View information about a specific resource
  + Check out a resource
  + Renew a resource
  + Return a resource
  + View a list of all resources borrowed by a student/staff
  + View a list of all resources due to be returned by a student/staff
  + Generate reports on resource circulation, overdue resources, and other metrics

## 1A. The Conceptual Schema (UML diagram)

To represent the conceptual schema for the college library system, we will use a UML class diagram. A class diagram is a visual representation of the classes in a system and the associations between them (Cepeda Porras and Guéhéneuc, 2010)

The Unified Modelling Language (UML) has been chosen as the foundational model to conceptualise the database design. The UML class diagram is an object-oriented conceptual data model used to describe the design of software systems using a set of rules and models (Pinheiro da Silva and Paton, 2000). The UML diagram is a technique used to represent the different components of a system, such as the classes, subclasses, associations, and the methods within a system (Dennis et al., 2015).

To represent the conceptual schema for the college library system, we will use a UML class diagram. A class diagram is a visual representation of the classes in a system and the associations between them (Cepeda Porras and Guéhéneuc, 2010)

The UML class diagram was derived from a systematic analysis of the application domain and the requirements for the data in the project specification by first identifying requirements as stated in the project specification for the Database application and understanding their functional and informational requirements. The requirements include creating a conceptual schema in the form of a UML class diagram, deriving a relational schema from the UML diagram, normalising the relations, implementing this schema by using SQL in Oracle, populating the database with a set of typical, significant and manageable data, defining specialised views which are appropriate to various sub-classes of users and defining SQL queries which could be used as canned queries for naive users.

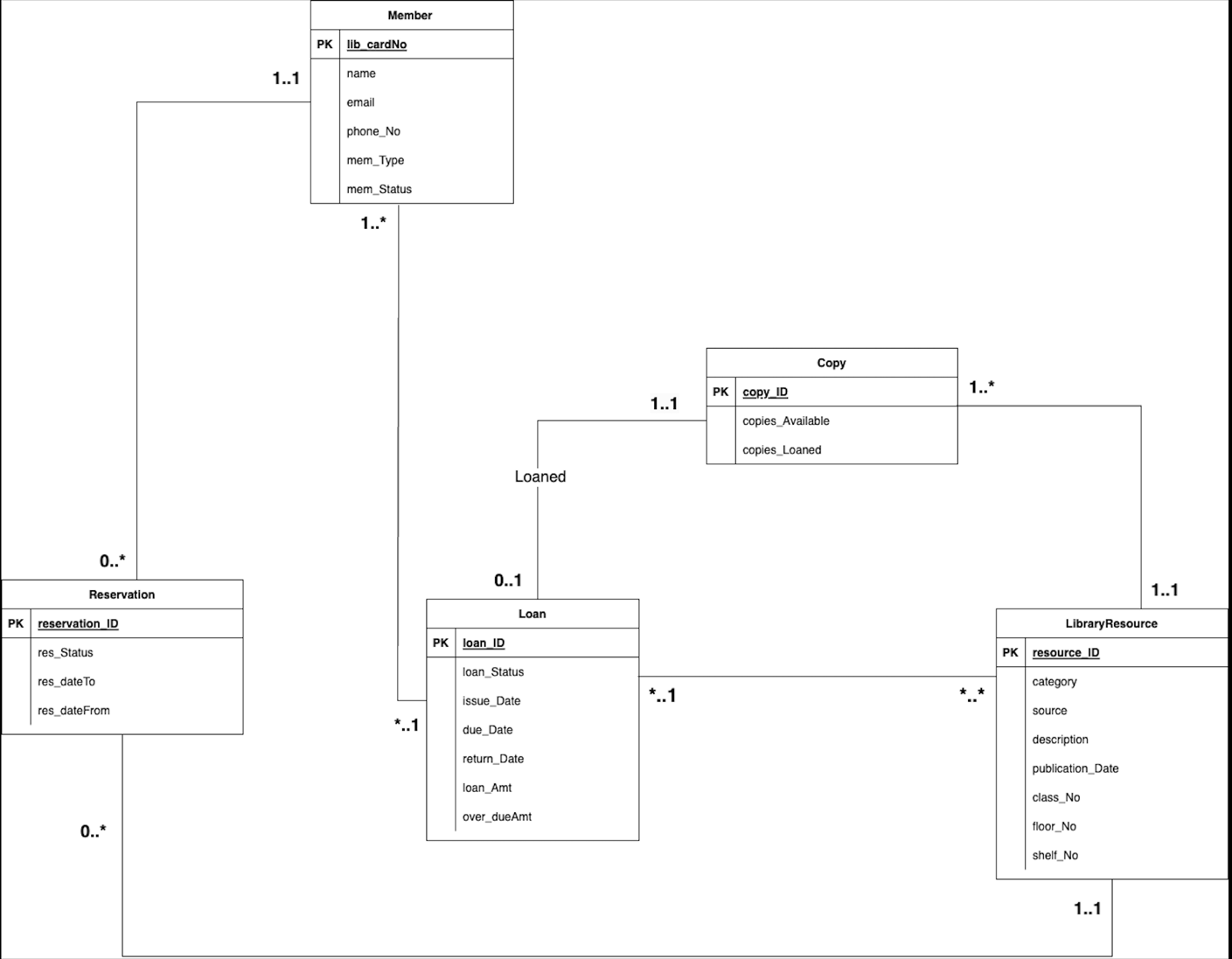
Consequently, we identified the leading classes and their variables based on information that has been provided. The associations between the classes and dependencies were also established. Before the conception and design phase, preliminary considerations were considered to create a premise conducive to a normalised database structure.

Based on these requirements, we identified the following key classes:

1. Member
2. Reservation
3. Resource
4. Copy
5. Loan

We identified that student and staff are subclasses of members.

Then we identified the entities and their attributes of the database. (See UML Diagram below



[(View data dictionary section for clarification on the terms.)](#_Data_Dictionary)

## 1B. The relational database schema

**Member** (lib\_cardNo{PK}, f\_Name, l\_Name, email, phone\_No, mem­\_Type, mem\_Status)

**Student** (lib\_cardNo{PK}, Max\_borrowableStu)

**Staff** (lib\_cardNo{PK}, Max\_borrowableStaff)

**Reservation**(reservation\_ID{PK}, res\_DateTo, res\_DateFrom, res\_Status, lib\_cardNo {FK}, resource\_ID {FK})

**Loan**(loan\_ID {PK}, loan\_Status, issue\_Date, due\_Date, return\_Date, loan\_Amt, over\_dueAmt)

**Copy**(copy\_ID {PK}, copies\_Available, copies\_Loaned, resource\_ID {FK})

**LibraryResource** (resource\_ID {PK}, category, source, description, publication­­­\_Date, floor\_No, class\_No, shelf\_No)

**Book**(resource\_ID {PK}, book\_Title)

**Video**(resource\_ID {PK}, video\_Title)

**DVD**(resource\_ID {PK}, dvd\_Title)

**CD**(resource\_ID {PK} , cd\_Title)

**Member table relations**

The member entity has relations with two other entities: Member: Reservation and Member: Loan

Member:Reservation

**Member** (lib\_cardNo {PK}, f\_name, l\_name email, phone\_No, mem­\_Type, mem\_Status)

**Reservation (**reservation\_ID {PK}, res\_DateFrom, res\_DateTo, res\_Status, lib\_cardNo{FK}, resource\_ID {FK})

Member to reservation relation is binary relationship. One-to-Many (1..\*). A member can have zero, one, or many reservations. This means that a member may not have any reservations, or they can have multiple reservations over time. However, for the reservation to member relationship is one-to-one (1..1) because each reservation is associated with exactly one member.

Member: loan

**Member** (lib\_cardNo {PK}, f\_name, l\_name email, phone\_No, mem­\_Type, mem\_Status)

**Loan** (loan\_ID {PK}, loan\_Status, issue\_Date, due\_Date, return\_Date, loan\_Amt, over\_dueAmt)

This is a One-to-Many (1..\*) relationship between the member and the loan because a member can have multiple loans, whereas the loan and member is one-to-one (1..1), because each loan is associated with only one member. This also assumes that a library card number in the "Member" table uniquely identifies a member, and a member can borrow multiple books (loans) over time.

**Member and its Subclasses (Student/Staff) Relation**

**Member** (lib\_cardNo {PK}, f\_name, l\_name email, phone\_No, mem­\_Type, mem\_Status)

**Student** (lib\_cardNo {PK}, f\_name, l\_name email, phone\_No, mem­\_Type, mem\_Status, Max\_borrowable)

**Staff** (lib\_cardNo {PK}, f\_name, l\_name email, phone\_No, mem­\_Type, mem\_Status, Max\_borrowable)

A generalisation/specialisation relationship exists between the member table and student/staff tables. The member table is a generalisation (superclass), while the student and staff tables are specialisations (subclasses). In other words, student and staff tables inherit all the attributes and relationships of member table, but add unique attributes and relationships related to their individual roles. For example, the unique attribute of a student allows them to borrow a maximum of 5 items. Notably, staffs are entitled to a maximum of 10 items per loan.

**Reservation table relations**

**Reservation:Member**

**Reservation** (reservation\_ID {PK}, res\_DateFrom, res\_DateTo, res\_Status, lib\_cardNo {FK}), resource\_ID {FK})

**Member** (lib\_cardNo {PK}, fname, email, phone\_No, mem­\_Type, mem\_Status)

This is a one-to-One (1:1): Records in the reservation table can only be associated with one and only one record in the member table. In contrast, one member, can be associated with multiple reservation, which means it is a zero to many relationship (0..\*).

**Reservation:Resource**

**Reservation** (reservation\_ID {PK}, res\_Date, res\_Status, lib\_cardNo, resource\_ID {FK})

**Resource** (resource\_ID {PK}, type, author, description, publication\_Date)

There is a one-to-one (1:1) relationship between a reservation and a resource because a reservation can be associated with only 1 resource. However, there is zero to many (0..\*) relationship between a resource and reservation as one resource can have multiple reservations.

**Resource Table relations**

**Resource and Subclass (Book, Video, DVD, CD) Relation**

**Book** (resource\_ID {PK}, type, author, description, publication\_Date, book\_Title)

**Video** (resource\_ID {PK}, type, author, description, publication\_Date, video\_Title)

**DVD** (resource\_ID {PK}, type, author, description, publication\_Date, dvd\_Title)

**CD** (resource\_ID {PK}, type, author, description, publication\_Date, cd\_Title)

There is a generalisation/specialisation relationship between the resource table and the subclasses (Book, Video, DVD, CD). Thus, resource represents the general concept of a library resource, while Books, Videos, DVDs, and CDs represent specific types of library resource. Generalisations represent the common attributes and behaviours of a set of subclasses. For example, the resource tables would show attributes and behaviours that are common to all library resources, such as title, type, and class number. In essence, a specialisation is a subclass that carries the attributes and behaviours of the superclass but represents a more specific type of attribute.

**Loan table relations**

**Loan: Member:**

There is a many-to-one (M:1) relationship between the loan and member tables.Many loans can be associated with one member. In other words, a single member can have multiple Loans, but a single Loan can be associated with only one member. For example, the library may have loaned the student or staff several books throughout the semester. A unique lib\_cardNo is associated with each loan record, indicating which loan belongs to which member.

**Loan:Resource**

A many-to-one (\*..1) relationship exists between loans and resources because a loan can be associated with more than one resource. However, each loan is associated with a unique resource. For example, there might be numerous loan records for a popular book, but each record would be associated with a single copy.

**Copy table relations**

**Copies to Loan Relation**

There is a zero-to-many (0..\*) relationship between copies and loans in a library system and a one-to-one (1..1) relationship between loans and copies. In other words, one copy of a resource can have multiple loans associated with it, however, one loan can only be associated with a copy of the resource.

**Copies to Resource**

There is a one-to-many (I..\*) relationship between copy and resource. Unlike a copy, which can be associated with multiple resources, a copy can only be associated with one resource at a time, which also indicates one-to-one (1..1) relationship between resource and copy.

## 1C. **Normalisation**

The process of normalisation is employed in database design to minimise the number of redundancies and dependency issues and to ensure that the data is organised in an efficient manner as well as maintaining its integrity (Eessaar, 2016). During the normalisation process, tables are gradually decomposed into higher normal forms, each building on the principles established by the previous state until the final normal form is achieved (Harrington, 2016).

**Initial Tables:**

* We identified that student and staff are subclasses of members.
* We also identified that Book, Video, DVD, and CD are the subclasses of LibraryResource

**Member**(lib\_cardNo, name, email, phoneNo, mem­\_Type, mem\_Status)

Student(lib\_cardNo {PK}, max\_borrowableStu)

Staff(lib\_cardNo {PK}, max\_borrowableStaff)

**Reservation**(reservationID, res\_DateTo, res\_DateFrom, res\_Status, lib\_cardNo, resource\_ID)

**Loan**(loan\_ID, loan\_Status, issue\_Date, due\_Date, return\_Date, loan\_Amt, over\_dueAmt)

**Copy**(copy\_ID, copies\_Available, copies\_Loaned, resource\_ID)

**LibraryResource**(resource\_ID, category, source, description, publication­­­\_Date, floor\_No, class\_No, shelf\_No)

Book(resource\_ID {PK}, book\_Title)

Video(resource\_ID {PK}, video\_Title)

DVD (resource\_ID {PK}, dvd\_Title)

CD (resource\_ID{PK}, cd\_Title)

**Universal Relation**

U(lib\_cardNo, name, email, phone\_No, mem­\_Type, mem\_Status, max\_borrowableStu, max\_borrowableStaff, reservation\_ID, res\_DateTo, res\_DateFrom, res\_Status, lib\_cardNo, loan\_ID, loan\_Status, issue\_Date, due\_Date, return\_Date, loan\_Amt, over\_dueAmt, copy\_ID, copies\_Available, copies\_Loaned, category, source, description, publication­­­\_Date, floor\_No, class\_No, shelf\_No, resource\_ID, book\_Title, resource\_ID, video\_Title, resource\_ID , dvd\_Title, resource\_ID , cd\_Title)

**Functional Dependencies:**

Given a lib\_CardNo we can uniquely determine the member’s personal attributes including, their borrowing status (i.e., active or cancelled) and the maximum number of books a member can borrow at a given time.

lib\_cardNo—> {f\_Name, l\_name, email, mem\_Type, mem\_Status, max\_borrowableStu, max\_borrowableStaff}

From the reservation\_ID we can uniquely determine what date a reservation was made and the reservation status.

reservation\_ID --> {res\_Status, res\_dateFrom, res\_dateTo}

Given a loan\_ID we can uniquely determine information about the loan including the loan amount and any overdue fine a member has incurred.

loan\_ID -> {loan\_Status, issue\_Date, due\_Date, return\_Date, loan\_Amt, over\_dueAmt}

From a copy\_ID we can determine the availability of the resource as well as the copy status (i.e., availability, on-loan).

copy\_ID -> {copies\_Available, copies\_Loaned}

Given a resource\_ID we can uniquely determine the resource type (i.e., CD, Books, etc.), its location including floor and shelf number as well as class number.

resource\_ID -> {type, author, description, publication\_Date, book\_Title, video\_Title, dvd\_Title, cd\_Title, floor\_No, class\_No, shelf\_No}

By analysing the functional dependencies, we can readily identify Primary Keys and assign them to corresponding classes containing non-primary variables.

Student: lib\_cardNo max\_borrowableStu,

Staff: lib\_cardNo max\_borrowableStaff

Resources (Book, Video, DVD, and CD) classes:

For the classes representing specific resource types (book, video, DVD, CD), the primary key remains the resource\_ID. This ensures that each resource is uniquely identified, while the type of variable further categorises it within the library's collection.

**First normal form**

In the F**irst Normal Form (1NF),** only single values are permitted at the intersection of each row and column; hence, there are no repeating groups (multi-valued attributes). To normalize the relation that contains a repeating group, the repeating group must be decomposed to form two new relations. Each attribute must contain atomic values. A column should contain value from same domain. Each domain should have unique name. No ordering of rows and columns and no duplicating rows (Watt and Eng, 2014

Based on the rules of the 1NF (First Normal Form), the universal relational table below meets all the requirements of the 1NF (First Normal Form) rules.

**Member Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Member** | | | | | |
| lib\_cardNo | name | email | phone\_No | mem\_Type | mem\_Status |

Here we notice that **Name** is being grouped hence we separate them into different columns.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Member** | | | | | | |
| lib\_cardNo | f\_Name | l\_Name | email | phone\_No | mem\_Type | mem\_Status |

**Second Normal Form (2NF)**

The **second normal form** requires that the table must first be in 1NF. The table is automatically in 2NF if, and only if, the Primary Key (PK) comprises a single attribute. If the table has a composite PK, then each non-key attribute must be fully dependent on the entire PK and not on a subset of the PK (i.e., there must be no partial dependency or augmentation) (Watt and Eng, 2014)

Hence, we create separate tables for Student and Staff information as they depend on lib\_cardNo:

**Member Table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Member** | | | | | | |
| lib\_cardNo | f\_Name | l\_Name | email | phone\_No | mem\_Type | mem\_Status |

|  |  |
| --- | --- |
| **Student** | |
| lib\_cardNo | max\_borrowableStu |

|  |  |
| --- | --- |
| **Staff** | |
| lib\_cardNo | max\_borrowableStaff |

Similarly, we create separate tables for LibraryResource as they all depend on resource\_ID.

**LibraryResource Tables:**

**Book Table:**

|  |  |
| --- | --- |
| **Book** | |
| resource\_ID | book\_Title |

**Video Table:**

|  |  |
| --- | --- |
| **Video** | |
| resource\_ID | video\_Title |

**DVD Table:**

|  |  |
| --- | --- |
| **DVD** | |
| resource\_ID | DVD\_Title |

**CD Table:**

|  |  |
| --- | --- |
| **CD** | |
| resource\_ID | CD\_Title |

**Third Normal Form (3NF)**

To satisfy the **third normal form rules,** the table must be in second normal form. There should not be any transitive dependencies for non-prime attributes, a non-key attribute may not be functionally dependent on another non-key attribute. A table must be in 3NF if all the non-primary-key attributes are mutually independent. (Watt and Eng, 2014).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Member** | | | | | | |
| lib\_cardNo | f\_Name | l\_Name | email | phone\_No | mem\_Type | mem\_Status |

|  |  |
| --- | --- |
| **Student** | |
| lib\_cardNo | max\_borrowableStu |

|  |  |
| --- | --- |
| **Staff** | |
| lib\_cardNo | max\_borrowableStaff |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Reservation** | | | | | |
| reservation\_ID | lib\_cardNo | resource\_ID | res\_DateFrom | res\_DateTo | res\_Status |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **LibraryResource** | | | | | | | |
| resource\_ID | category | source | description | publication\_Date | class\_No | floor\_No | shelf\_No |

|  |  |
| --- | --- |
| **Book** | |
| resource\_ID | book\_Title |

|  |  |
| --- | --- |
| **Video** | |
| resource\_ID | video\_Title |

|  |  |
| --- | --- |
| **DVD** | |
| resource\_ID | DVD\_Title |

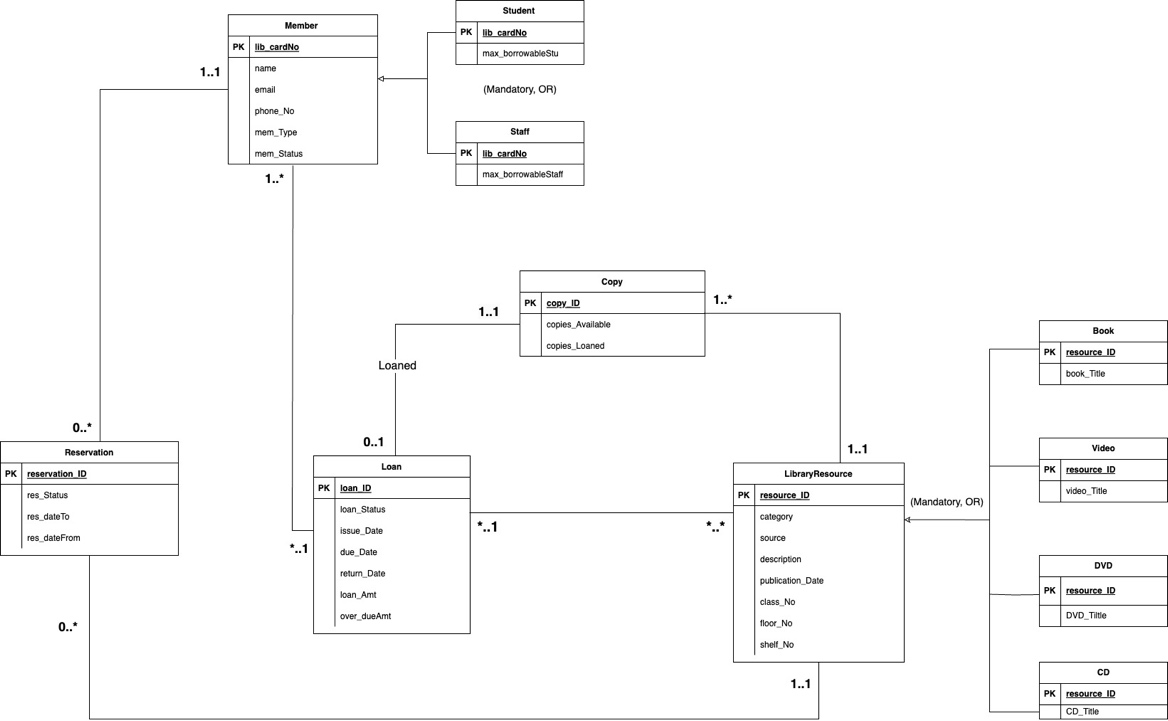
|  |  |
| --- | --- |
| **CD** | |
| resource\_ID | CD\_Title |

|  |  |  |  |
| --- | --- | --- | --- |
| **Copy** | | | |
| copy\_ID | copies\_Available | Resource\_ID | copies\_Loaned |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Loan** | | | | | | | |
| loan\_ID | loan\_Status | lib\_cardNo | resource\_ID | Issue\_Date | due\_Date | return\_Date | loan\_Amt |

This architecture minimizes redundancy and improves data integrity by classifying resources into discrete tables based on their categories. Foreign keys are needed to maintain the connections across tables, and each type of resource has its own set of attributes.

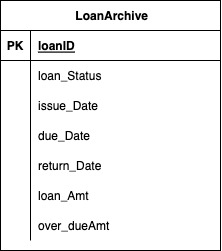
**Updated UML Diagram after Normalisation**



# Part2: Implementation of the College Library System

We have chosen to include a Loan Archive table to maintain track of prior loans, reservations, and resource demand separate from the loan table. This is due to the phyisical restraints if the database system in terms of memory.

The historical loan records are available in the Loan Archive table.



## 2A. Table creation

In this section, you will find the code that was used to create the tables

CREATE TABLE MEMBER

(lib\_cardNo NUMBER(4) NOT NULL,

f\_name VARCHAR(10),l\_name VARCHAR (10),email

VARCHAR(20),phone\_No NUMBER(13),mem\_Type VARCHAR(7),

mem\_Status VARCHAR(10))

INSERT INTO MEMBER VALUES

([7890,'HARRY','STYLES','HS@QMUL.AC.UK',441482664408](mailto:7890,'HARRY','STYLES','HS@QMUL.AC.UK',441482664408),

'STUDENT','ACTIVE')

INSERT INTO MEMBER VALUES

(2345,'LOUIS', 'TOMLINSON',

'[LT@QMUL.AC.UK',447922722477](mailto:LT@QMUL.AC.UK',447922722477), 'STUDENT','SUSPENDED')

INSERT INTO MEMBER VALUES

(3489,'NIALL', 'HORAN',

'[NH@QMUL.AC.UK',447570937590](mailto:NH@QMUL.AC.UK',447570937590), 'STAFF','OVERDUE')

INSERT INTO MEMBER VALUES

(5678,'JENNIFER', 'ANISTON',

'[JA@QMUL.AC.UK',443350157822](mailto:JA@QMUL.AC.UK',443350157822), 'STAFF','ACTIVE')

INSERT INTO MEMBER VALUES

(5072,'LIAM', 'PAYNE',

'[LP@QMUL.AC.UK',447525371530](mailto:LP@QMUL.AC.UK',447525371530), 'STUDENT','INACTIVE')

INSERT INTO MEMBER VALUES

(2039,'EMMA', 'WATSON',

'[EW@QMUL.AC.UK',441277937297](mailto:EW@QMUL.AC.UK',441277937297), 'STAFF','OVERDUE')

INSERT INTO MEMBER VALUES

(2601,'ROWAN', 'ATIKNSON',

'[RA@QMUL.AC.UK',447324292457](mailto:RA@QMUL.AC.UK',447324292457), 'STUDENT','ACTIVE')

INSERT INTO MEMBER VALUES

(5100,'ZAYN', 'MALIK',

'[ZM@QMUL.AC.UK',441362464511](mailto:ZM@QMUL.AC.UK',441362464511), 'STAFF','ACTIVE')

INSERT INTO MEMBER VALUES

(2789,'JACK', 'BLACK',

'[JB@QMUL.AC.UK',442364899223](mailto:JB@QMUL.AC.UK',442364899223), 'STUDENT','OVERDUE')

INSERT INTO MEMBER VALUES

(2789,'EREN', 'YEAGAR',

'[EY@QMUL.AC.UK',446769342623](mailto:EY@QMUL.AC.UK',446769342623), 'STUDENT','ACTIVE')

CREATE TABLE STUDENT

(lib\_cardNo NUMBER (4) NOT NULL,

max\_borrowableStu NUMBER (1))

INSERT INTO STUDENT VALUES

(7890,5)

INSERT INTO STUDENT VALUES

(5237,5)

INSERT INTO STUDENT VALUES

(5072,5)

INSERT INTO STUDENT VALUES

(2601,5)

INSERT INTO STUDENT VALUES

(2789,5)

INSERT INTO STUDENT VALUES

(2345,5)

CREATE TABLE STAFF

(lib\_cardNo NUMBER (4) NOT NULL,

max\_borrowableStaff NUMBER (2))

INSERT INTO STAFF VALUES

(3489,10)

INSERT INTO STAFF VALUES

(5678,10)

INSERT INTO STAFF VALUES

(2039,10)

INSERT INTO STAFF VALUES

(5110,10)

CREATE TABLE RESERVATION

(reservation\_ID NUMBER (5),

lib\_cardNo NUMBER (4),

resource\_ID VARCHAR (5),

res\_dateFrom DATE,

res\_dateTo DATE,

res\_Status VARCHAR(10))

INSERT INTO RESERVATION VALUES

(57382,7890,'740B',

TO\_DATE('01-9-2023', 'DD-MM-YYYY'),

TO\_DATE('02-9-2023', 'DD-MM-YYYY'),'notified')

INSERT INTO RESERVATION VALUES

(57383,7890,'7010B',

TO\_DATE('05-9-2023', 'DD-MM-YYYY'),

TO\_DATE('06-9-2023', 'DD-MM-YYYY'),'accepted')

INSERT INTO RESERVATION VALUES

(57384,3489,'740B',

TO\_DATE('07-9-2023', 'DD-MM-YYYY'),

TO\_DATE('08-9-2023', 'DD-MM-YYYY'),'cancelled')

INSERT INTO RESERVATION VALUES

(57385,5678,'780C',

TO\_DATE('05-11-2023', 'DD-MM-YYYY'),

TO\_DATE('06-11-2023', 'DD-MM-YYYY'),'notified')

INSERT INTO RESERVATION VALUES

(57386, 5072,'740B',

TO\_DATE('18-10-2023', 'DD-MM-YYYY'),

TO\_DATE('20-10-2023', 'DD-MM-YYYY'),'accepted')

INSERT INTO RESERVATION VALUES

(57387,2039,'7009B',

TO\_DATE('20-10-2023', 'DD-MM-YYYY'),

TO\_DATE('21-10-2023', 'DD-MM-YYYY'),'declined')

INSERT INTO RESERVATION VALUES

(57388,2601,'7009B',

TO\_DATE('22-10-2023', 'DD-MM-YYYY'),

TO\_DATE('23-10-2023', 'DD-MM-YYYY'),'expired')

INSERT INTO RESERVATION VALUES

(57389,5237,'7010B',

TO\_DATE('24-10-2023', 'DD-MM-YYYY'),

TO\_DATE('24-10-2023', 'DD-MM-YYYY'),'notified')

CREATE TABLE LIBRARYRESOURCE

(resource\_ID VARCHAR (5),category VARCHAR (5), source VARCHAR (20),

description VARCHAR (90), publication\_Date NUMBER (4),

class\_No NUMBER (1),floor\_No NUMBER (1),shelf\_No VARCHAR (2))

INSERT INTO LIBRARYRESOURCE VALUES

('740B','book','Allen G. Taylor',

'Guide for Database Development and Implementation 6th Edition',

2011,1,1,'1a')

INSERT INTO LIBRARYRESOURCE VALUES

('7010B','book',

'Tanenbaum et al',' Explain how networks work from the inside out.6th Edition',

2021,2,2,'2c')

INSERT INTO LIBRARYRESOURCE VALUES

('7009B','book',

'Ronald J. Leach','Practical Guidance on the Efficient Development of High-Quality Software second Edition',

2020,3,2,'3f')

INSERT INTO LIBRARYRESOURCE VALUES

('780V','video',

'Luna Lovegood', 'National Geographic' ,

2008,4,3, NULL)

INSERT INTO LIBRARYRESOURCE VALUES

('7010D','dvd',

'Brendan Fraser', 'A introduction to the culture of Egypt',

2016,4,3,'3a')

INSERT INTO LIBRARYRESOURCE VALUES

('7011D','dvd',

'J.J. Abrams', 'Star Wars: The Skywalker Saga',

2018,5,3, '3b')

INSERT INTO LIBRARYRESOURCE VALUES

('779C','cd',

'Juan Gonzalez', 'Spanish 101 for beginners',

2012,6,3,'3c')

INSERT INTO LIBRARYRESOURCE VALUES

('780C','cd',

'Pratap Mehta', 'Shaping the emerging world: India and the multilateral order',

2022,4,3, '3d')

CREATE TABLE BOOK

(resource\_ID VARCHAR (5), book\_Title VARCHAR (50))

INSERT INTO BOOK VALUES

('740B','Database for dummies')

INSERT INTO BOOK VALUES

('7010B', 'Computer Networks')

INSERT INTO BOOK VALUES

('7009B', 'Introduction to Software Engineering')

CREATE TABLE VIDEO

(resource\_ID VARCHAR (5), video\_Title VARCHAR (40))

INSERT INTO VIDEO VALUES

('780V', 'National Geographic: the Arctic Cirlce')

CREATE TABLE DVD

(resource\_ID VARCHAR (5), DVD\_Title VARCHAR (40))

INSERT INTO DVD VALUES

('7010D' , 'Egyptology:A documentary')

INSERT INTO DVD VALUES

('7011D' , 'Star Wars: The Skywalker Saga')

CREATE TABLE CD

(resource\_ID VARCHAR (5), CD\_Title VARCHAR (70))

INSERT INTO CD VALUES

('779C','Spanish 101 for beginners')

INSERT INTO CD VALUES

('780C' , 'Shaping the emerging world: India and the multilateral order')

CREATE TABLE COPY

( copy\_ID NUMBER (3) , copies\_Available NUMBER (1), resource\_ID VARCHAR (5) , copies\_Loaned NUMBER (1) )

INSERT INTO COPY VALUES

( 101 ,1 , '7040B' ,1)

INSERT INTO COPY VALUES

(102 , 2 , '7010B',3)

INSERT INTO COPY VALUES

(103, 3 , '7009B',1)

INSERT INTO COPY VALUES

(104,NULL,'780V',NULL)

INSERT INTO COPY VALUES

(105,1 , '7010D', 0)

INSERT INTO COPY VALUES

(106,2,'7011D',2)

INSERT INTO COPY VALUES

(118,1,'780C' , 1)

INSERT INTO COPY VALUES

(119,1, '779C',0)

CREATE TABLE LOAN

(loan\_ID VARCHAR (7), loan\_Status VARCHAR (8) , lib\_cardNo NUMBER (4) , resource\_ID VARCHAR (5) , issue\_Date DATE , due\_Date DATE , return\_Date DATE , loan\_Amt CHAR (7) , over\_dueAmt CHAR (7))

INSERT INTO LOAN VALUES

('a28633','overdue', 7890,'7010B' , TO\_DATE ('02-09-2023', 'DD-MM-YYYY'), TO\_DATE ('16-09-2023', 'DD-MM-YYYY'),

TO\_DATE ('18-09-2023', 'DD-MM-YYYY'), '£2.00' , '£2.00')

INSERT INTO LOAN VALUES

('b34940','borrowed', 5678,'740B' , TO\_DATE ('06-09-2023', 'DD-MM-YYYY'), TO\_DATE ('20-09-2023', 'DD-MM-YYYY'),

TO\_DATE ('19-09-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

INSERT INTO LOAN VALUES

('c74894','overdue', 3489,'7011D' , TO\_DATE ('21-09-2023', 'DD-MM-YYYY'), TO\_DATE ('23-09-2023', 'DD-MM-YYYY'),

TO\_DATE ('28-09-2023', 'DD-MM-YYYY'), '£5.00' , '£5.00')

INSERT INTO LOAN VALUES

('g18730','borrowed', 7890,'7009B' , TO\_DATE ('28-09-2023', 'DD-MM-YYYY'), TO\_DATE ('12-10-2023', 'DD-MM-YYYY'),

TO\_DATE ('11-10-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

INSERT INTO LOAN VALUES

('e09876','overdue', 2789,'7011D' , TO\_DATE ('12-10-2023', 'DD-MM-YYYY'), TO\_DATE ('14-10-2023', 'DD-MM-YYYY'),

TO\_DATE ('15-10-2023', 'DD-MM-YYYY'), '£1.00' , '£1.00')

INSERT INTO LOAN VALUES

('h45678','overdue', 2345,'7010B' , TO\_DATE ('13-10-2023', 'DD-MM-YYYY'), TO\_DATE ('27-10-2023', 'DD-MM-YYYY'),

TO\_DATE ('07-11-2023', 'DD-MM-YYYY'), '£11.00' , '£11.00')

INSERT INTO LOAN VALUES

('d72628','borrowed', 5072,'740B' , TO\_DATE ('20-10-2023', 'DD-MM-YYYY'), TO\_DATE ('03-11-2023', 'DD-MM-YYYY'),

TO\_DATE ('30-10-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

INSERT INTO LOAN VALUES

('i23930','borrowed', 2601,'7010B' , TO\_DATE ('24-10-2023', 'DD-MM-YYYY'), TO\_DATE ('07-11-2023', 'DD-MM-YYYY'),

TO\_DATE ('01-11-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

INSERT INTO LOAN VALUES

('f81693','borrowed', 5287,'780C' , TO\_DATE ('24-10-2023', 'DD-MM-YYYY'), TO\_DATE ('7-11-2023', 'DD-MM-YYYY'),

TO\_DATE ('6-11-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

CREATE TABLE LOANARCHIVE

( loan\_ID VARCHAR (7),loan\_Status VARCHAR (8) , resource\_ID VARCHAR (5) , issue\_Date DATE , due\_Date DATE , return\_Date DATE , loan\_Amt CHAR (7) , over\_dueAmt CHAR (7))

INSERT INTO LOANARCHIVE VALUES

('j28633','overdue', '7010B' , TO\_DATE ('02-06-2023', 'DD-MM-YYYY'), TO\_DATE ('16-06-2023', 'DD-MM-YYYY'),

TO\_DATE ('15-06-2023', 'DD-MM-YYYY'), '£2.00' , '£2.00')

INSERT INTO LOANARCHIVE VALUES

('k34940','borrowed', '740B' , TO\_DATE ('06-06-2023', 'DD-MM-YYYY'), TO\_DATE ('20-06-2023', 'DD-MM-YYYY'),

TO\_DATE ('19-06-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

INSERT INTO LOANARCHIVE VALUES

('l74894','overdue', '7011D' , TO\_DATE ('21-06-2023', 'DD-MM-YYYY'), TO\_DATE ('22-06-2023', 'DD-MM-YYYY'),

TO\_DATE ('27-06-2023', 'DD-MM-YYYY'), '£5.00' , '£5.00')

INSERT INTO LOANARCHIVE VALUES

('m18730','borrowed', '7009B' , TO\_DATE ('28-06-2023', 'DD-MM-YYYY'), TO\_DATE ('12-07-2023', 'DD-MM-YYYY'),

TO\_DATE ('12-7-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

INSERT INTO LOANARCHIVE VALUES

('n09876','overdue', '7011D' , TO\_DATE ('12-07-2023', 'DD-MM-YYYY'), TO\_DATE ('13-07-2023', 'DD-MM-YYYY'),

TO\_DATE ('14-07-2023', 'DD-MM-YYYY'), '£1.00' , '£1.00')

INSERT INTO LOANARCHIVE VALUES

('o45678','overdue', '7010B' , TO\_DATE ('13-07-2023', 'DD-MM-YYYY'), TO\_DATE ('27-07-2023', 'DD-MM-YYYY'),

TO\_DATE ('07-08-2023', 'DD-MM-YYYY'), '£11.00' , '£11.00')

INSERT INTO LOANARCHIVE VALUES

('p72628','borrowed', '740B' , TO\_DATE ('20-07-2023', 'DD-MM-YYYY'), TO\_DATE ('03-07-2023', 'DD-MM-YYYY'),

TO\_DATE ('30-07-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

INSERT INTO LOANARCHIVE VALUES

('q23930','borrowed', '7010B' , TO\_DATE ('24-07-2023', 'DD-MM-YYYY'), TO\_DATE ('07-08-2023', 'DD-MM-YYYY'),

TO\_DATE ('01-08-2023', 'DD-MM-YYYY'), 0.00, 0.00)

INSERT INTO LOANARCHIVE VALUES

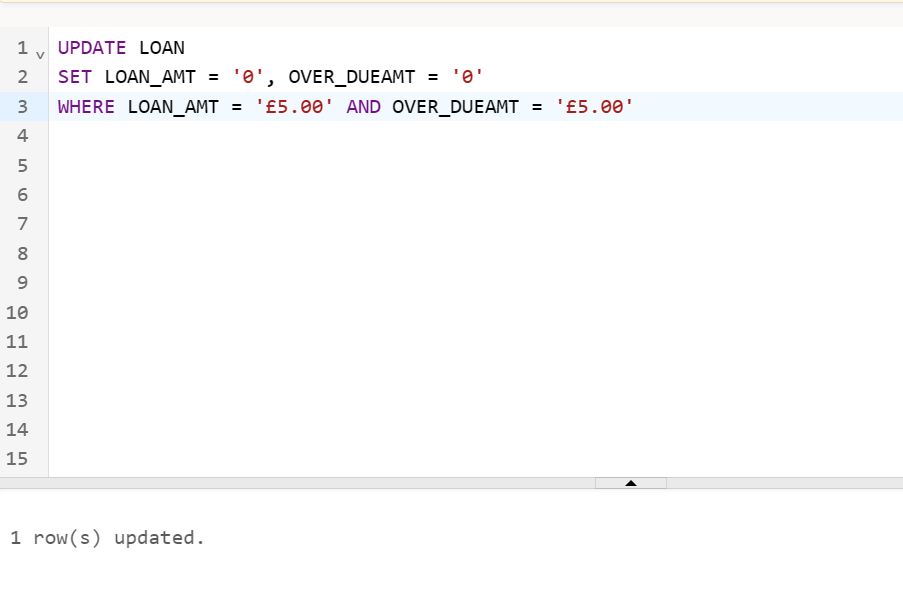
('r81693','borrowed', '780C' , TO\_DATE ('24-07-2023', 'DD-MM-YYYY'), TO\_DATE ('25-07-2023', 'DD-MM-YYYY'),

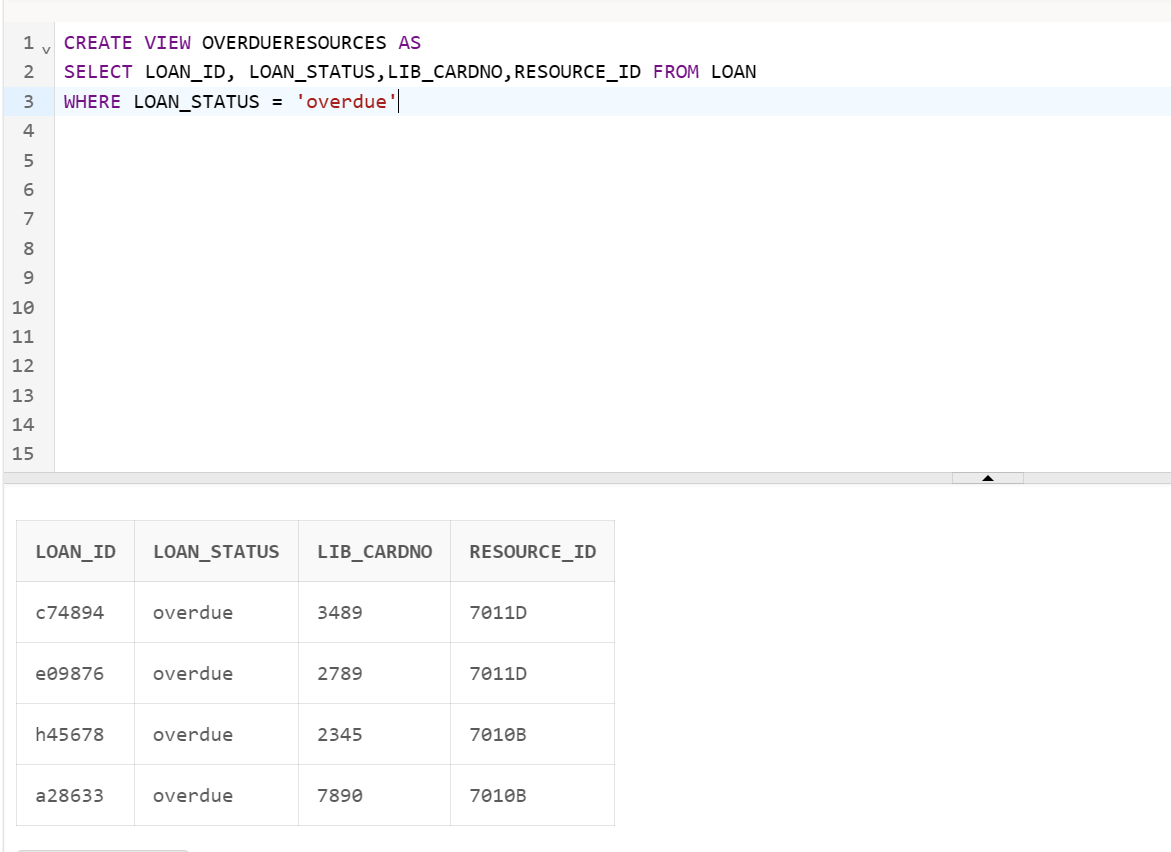
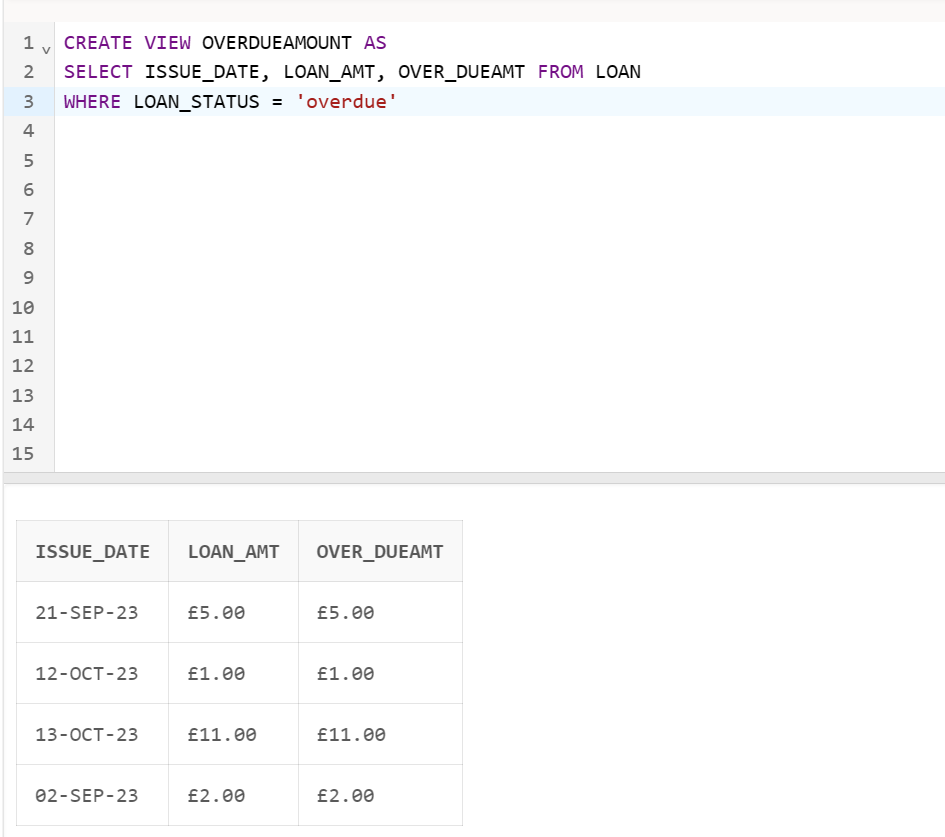
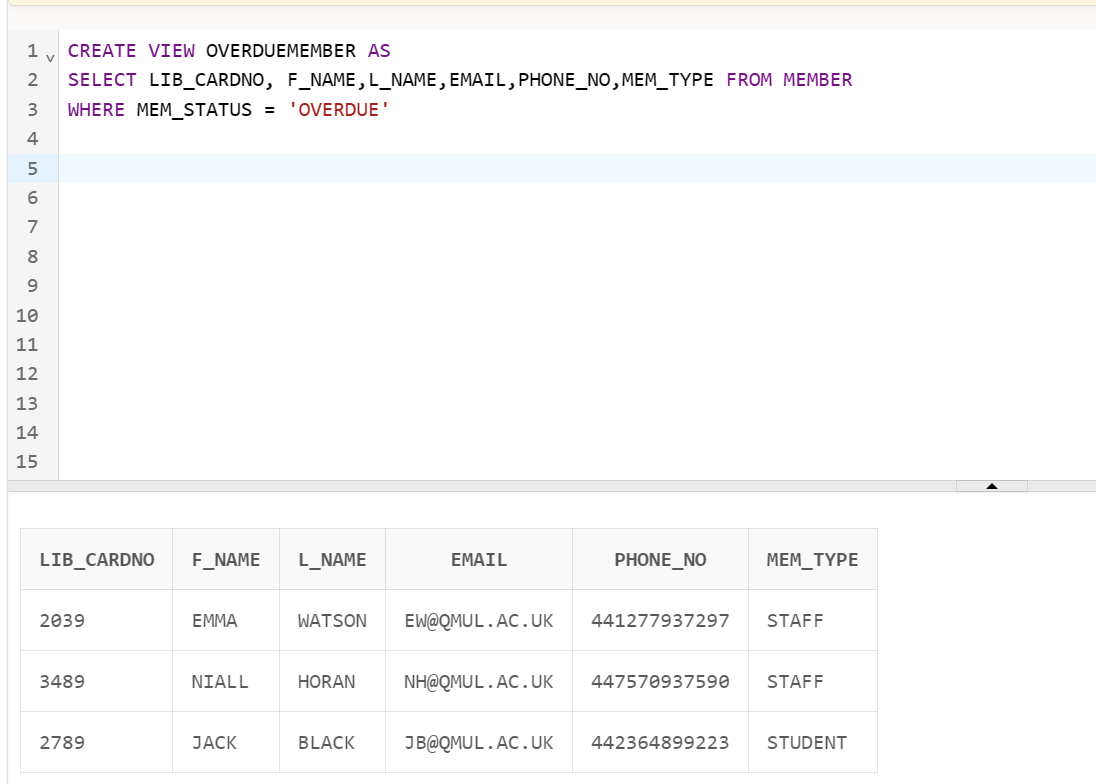
TO\_DATE ('25-07-2023', 'DD-MM-YYYY'), 0.00 , 0.00)

## 2B. Views Code

This section Insert code used to create **views** plus screenshots

1. Create a view to see who has a loan status of borrowed. Show loan ID, loan status, library card number, and resource ID.



1. Create a view to see who has a loan status of overdue. Show loan ID, loan status, library card number, and resource ID
2. Create a view to see the amount of overdue loan status. Show issue date, loan amount, and overdue amount.
3. Create a view to see the overdue members. Show library card number, first name, last name , email , phone , and member type. 

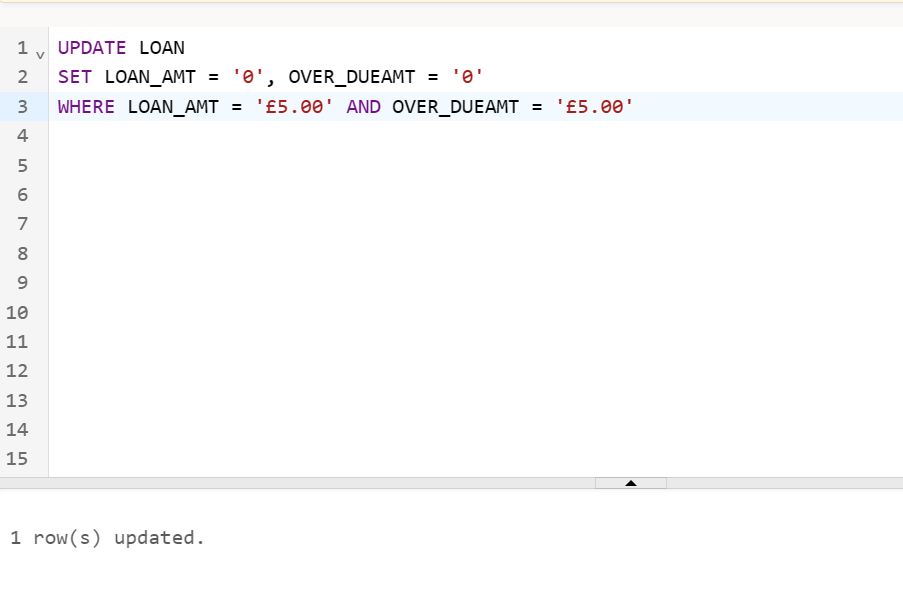
## **2C. Query Output**

**Update loan from 5 pounds to show 0 pounds**

UPDATE LOAN

SET LOAN\_AMT = '0', OVER\_DUEAMT = '0'

WHERE LOAN\_AMT = '£5.00' AND OVER\_DUEAMT = '£5.00'



**Show Description, Source, Category, Floor Number, Shelf Number From Library Resource where description beginning with S and equals Floor Number 3. Order by source in descending order.**

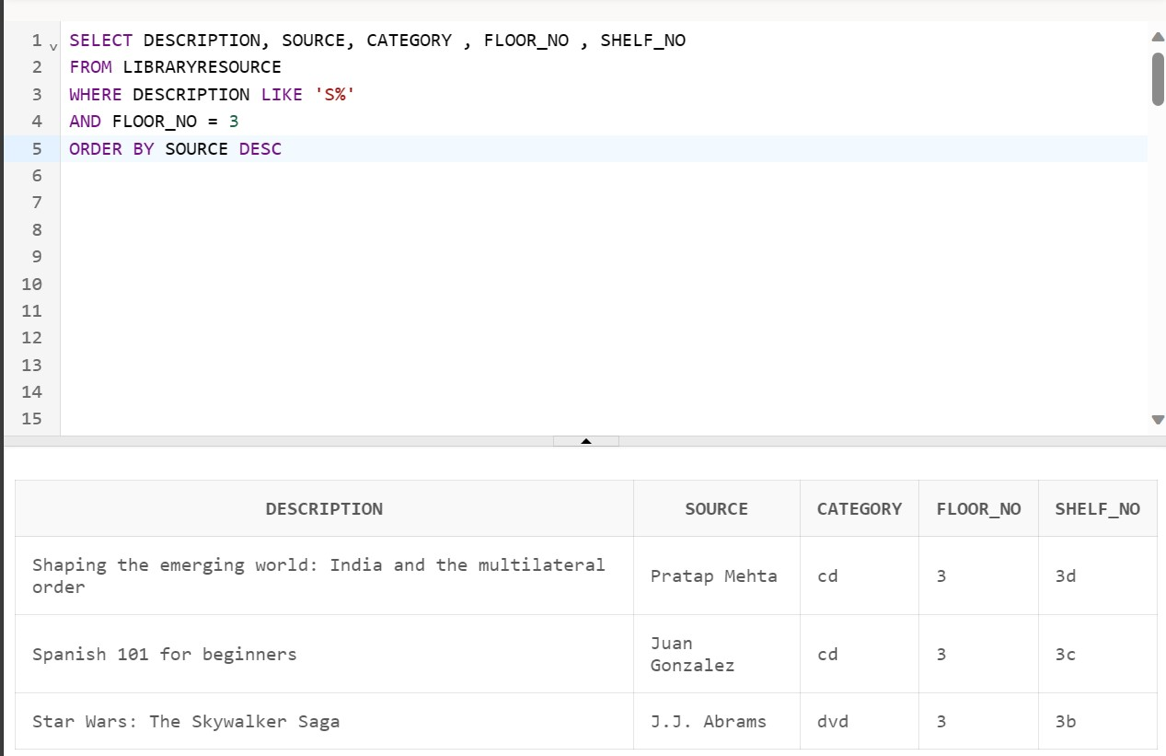
SELECT DESCRIPTION, SOURCE, CATEGORY , FLOOR\_NO , SHELF\_NO

FROM LIBRARYRESOURCE

WHERE DESCRIPTION LIKE 'S%'

AND FLOOR\_NO = 3

ORDER BY SOURCE DESC

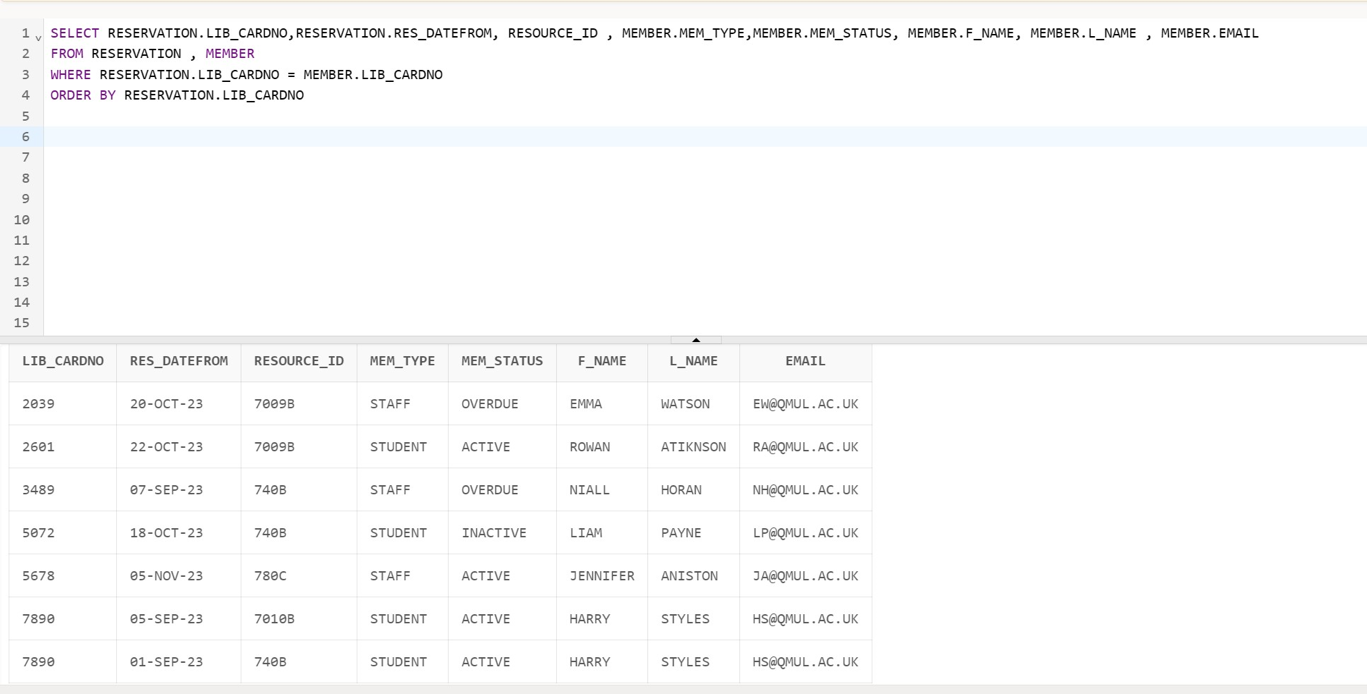


**Join tables Reservation and Member**

SELECT RESERVATION.LIB\_CARDNO,RESERVATION.RES\_DATEFROM, RESOURCE\_ID , MEMBER.MEM\_TYPE,MEMBER.MEM\_STATUS, MEMBER.F\_NAME, MEMBER.L\_NAME , MEMBER.EMAIL

FROM RESERVATION , MEMBER

WHERE RESERVATION.LIB\_CARDNO = MEMBER.LIB\_CARDNO

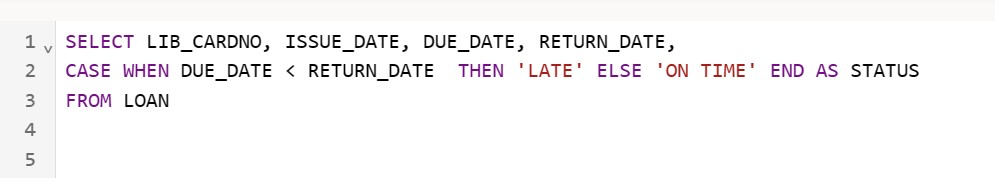
ORDER BY RESERVATION.LIB\_CARDNO

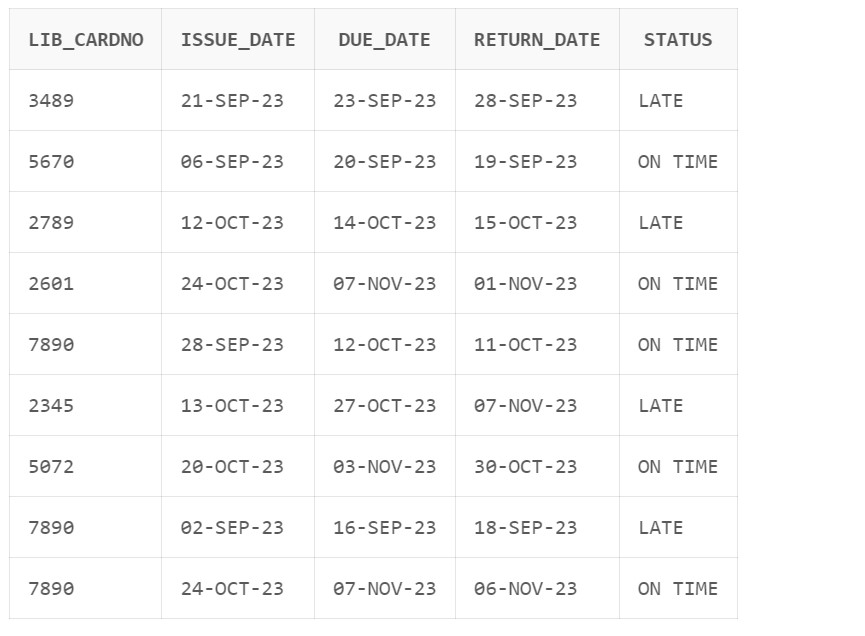
**Show how we can indicate if a resource is late or on time to return**

SELECT LIB\_CARDNO, ISSUE\_DATE, DUE\_DATE, RETURN\_DATE,

CASE WHEN DUE\_DATE < RETURN\_DATE THEN 'LATE' ELSE 'ON TIME' END AS STATUS

FROM LOAN



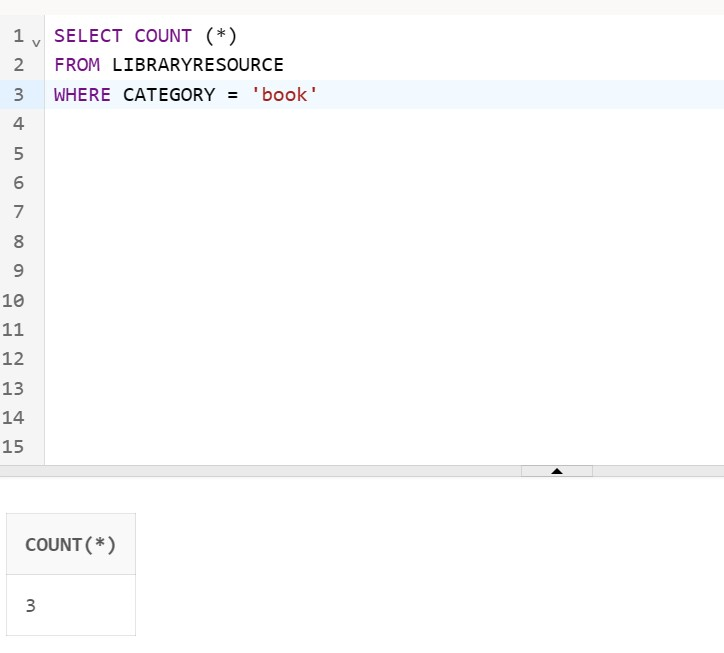


**How many books are in the Library Resource.**

SELECT COUNT (\*)

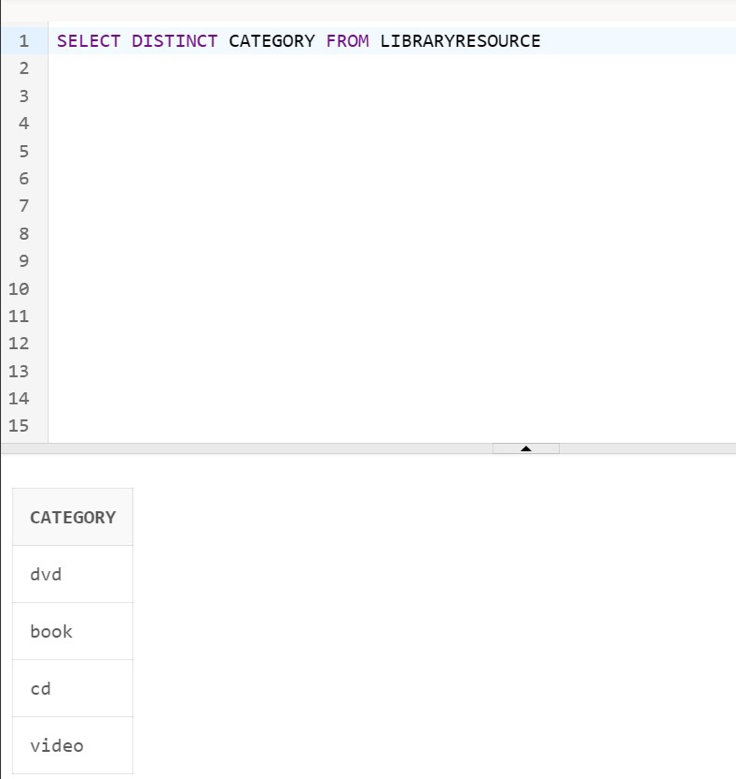
FROM LIBRARYRESOURCE

WHERE CATEGROY = 'book'



**Show how many different resources are in Library resource.**

SELECT DISTINCT CATEGORY FROM LIBRARYRESOURCE

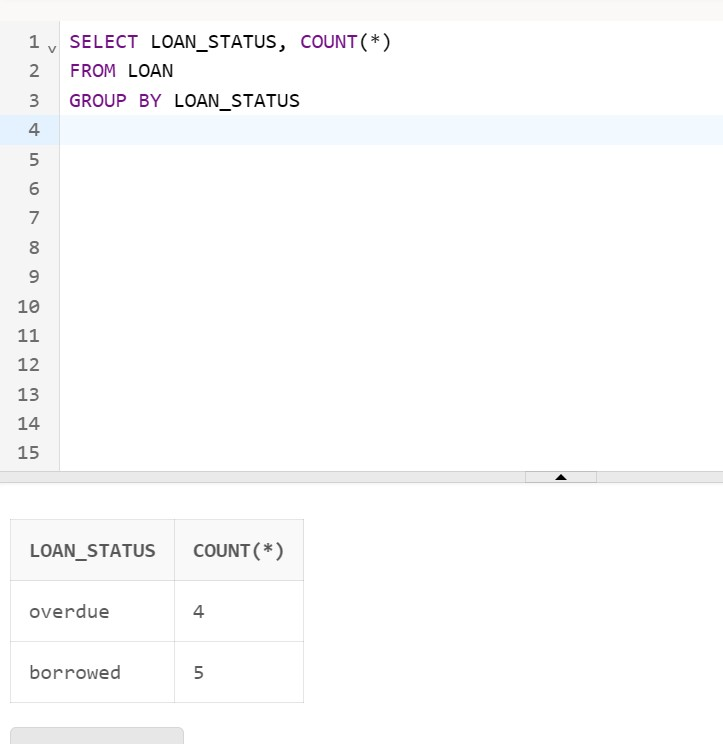


**Show how many status are overdue and borrowed.**

SELECT LOAN\_STATUS, COUNT(\*)

FROM LOAN

GROUP BY LOAN\_STATUS



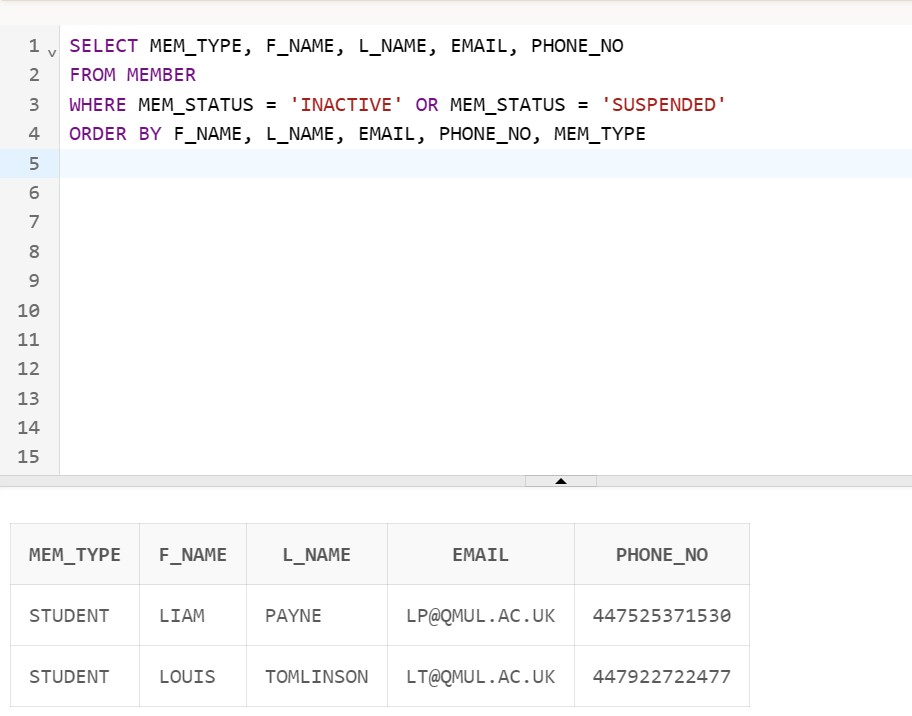
**Show many members from the member table are Inactive or Suspended.**

SELECT MEM\_TYPE, F\_NAME, L\_NAME, EMAIL, PHONE\_NO

FROM MEMBER

WHERE MEM\_STATUS = 'INACTIVE' OR MEM\_STATUS = 'SUSPENDED'

ORDER BY F\_NAME, L\_NAME, EMAIL, PHONE\_NO, MEM\_TYPE



**Show all resources joined together with resource\_ID, Category, and Titles**

SELECT LIBRARYRESOURCE.RESOURCE\_ID, LIBRARYRESOURCE.CATEGORY, BOOK.BOOK\_TITLE, VIDEO.VIDEO\_TITLE, DVD.DVD\_TITLE, CD.CD\_TITLE

FROM LIBRARYRESOURCE

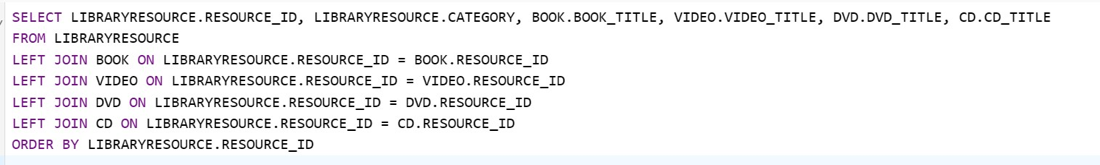
LEFT JOIN BOOK ON LIBRARYRESOURCE.RESOURCE\_ID = BOOK.RESOURCE\_ID

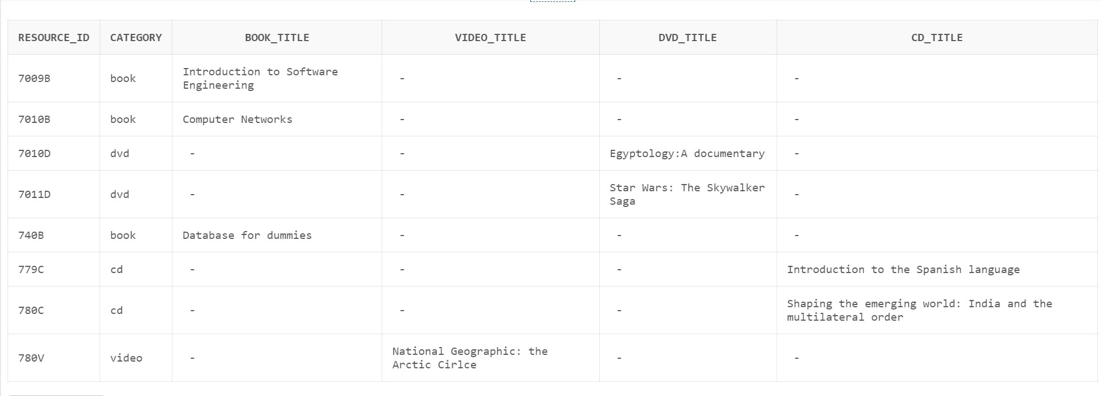
LEFT JOIN VIDEO ON LIBRARYRESOURCE.RESOURCE\_ID = VIDEO.RESOURCE\_ID

LEFT JOIN DVD ON LIBRARYRESOURCE.RESOURCE\_ID = DVD.RESOURCE\_ID

LEFT JOIN CD ON LIBRARYRESOURCE.RESOURCE\_ID = CD.RESOURCE\_ID

ORDER BY LIBRARYRESOURCE.RESOURCE\_ID

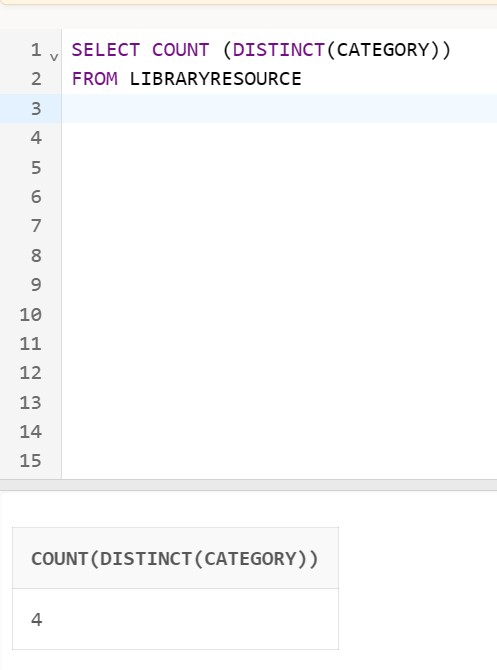




**Show how many different categories there are.**

SELECT COUNT (DISTINCT(CATEGORY))

FROM LIBRARYRESOURCE



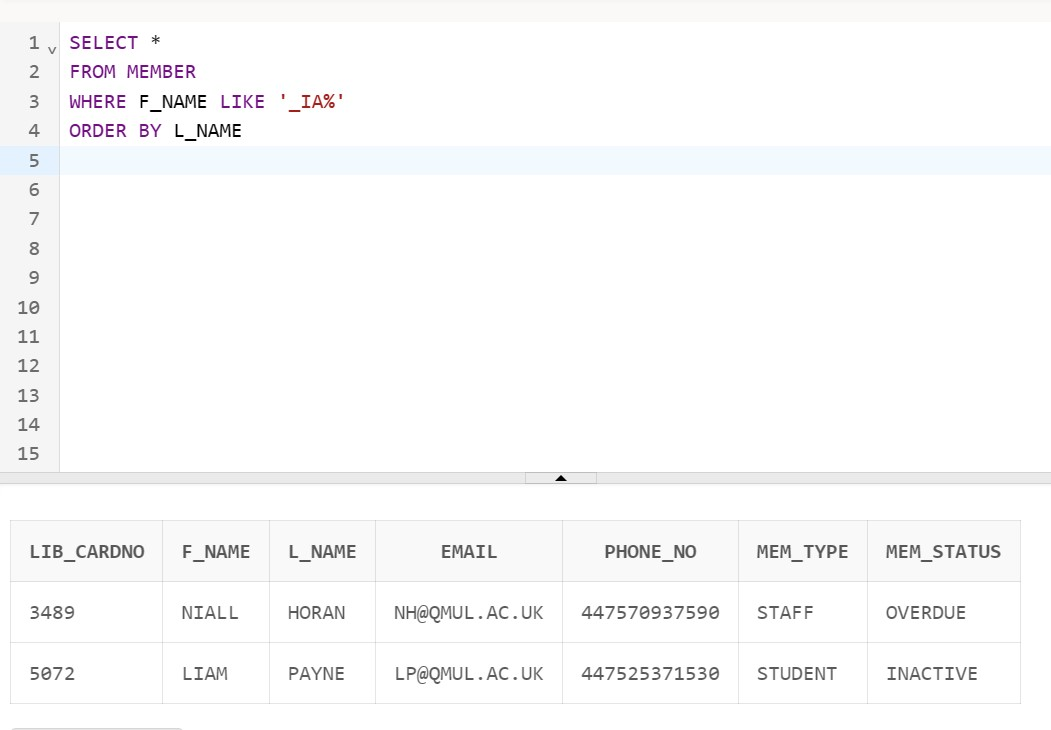
**Select member with first name that includes IA.**

SELECT \*

FROM MEMBER

WHERE F\_NAME LIKE '\_IA%'

ORDER BY L\_NAME



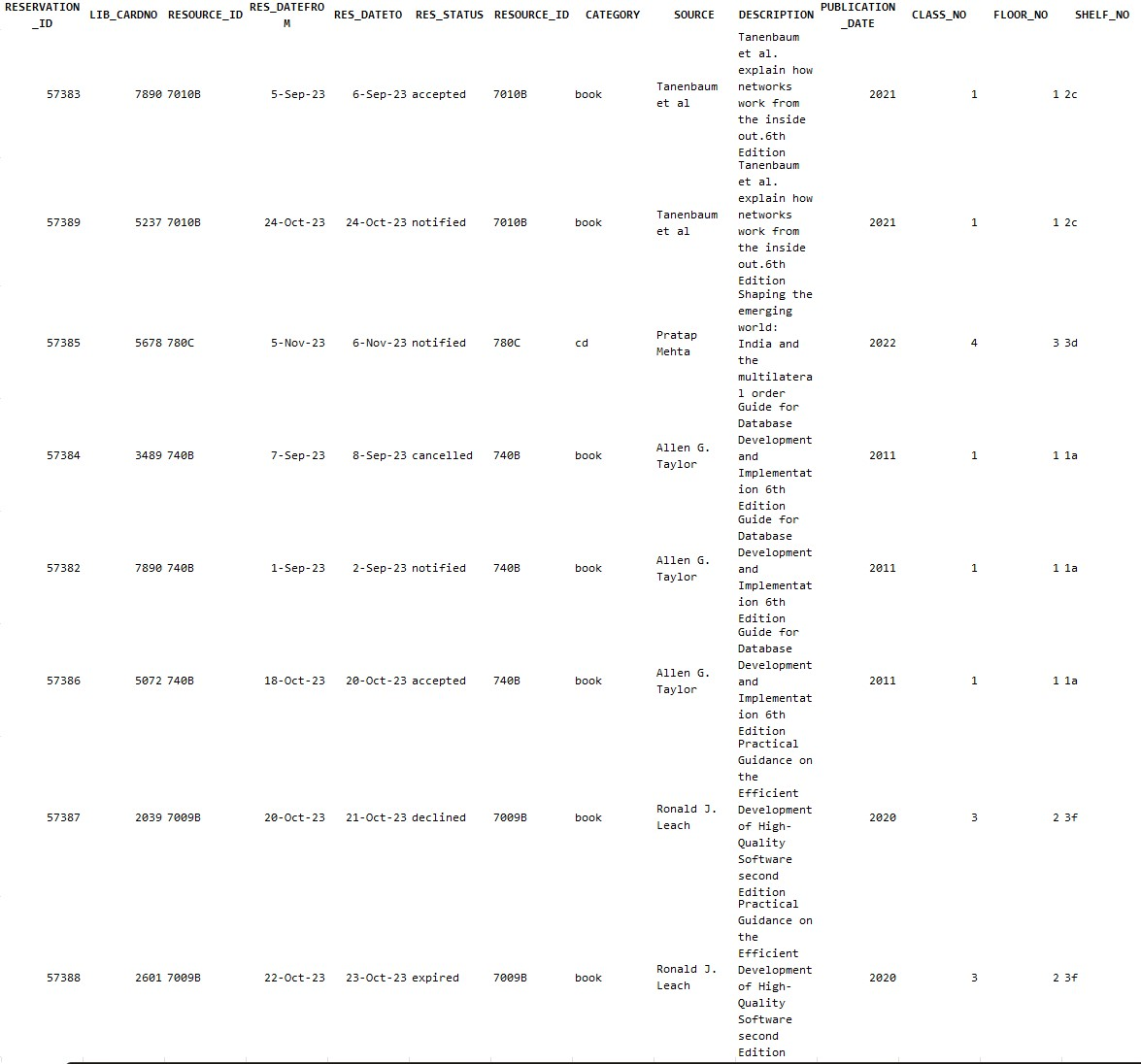
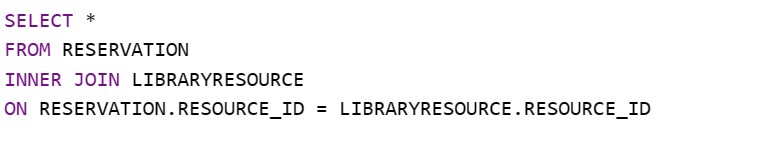
**Show joined tables Reservation and Library Resource.**

SELECT \*

FROM RESERVATION

INNER JOIN LIBRARYRESOURCE

ON RESERVATION.RESOURCE\_ID = LIBRARYRESOURCE.RESOURCE\_ID



# Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes Names** | **Contents (Description)** | **Format** | **PK OR FK** |
| **Member Table** | | | |
| lib\_cardNo | Unique number assigned to each member | NUMBER (4) | PK |
| f\_Name | Member first name | VARCHAR (10) | N/A |
| l\_Name | Member last name | VARCHAR (10) | N/A |
| email | Member email | VARCHAR (20) | N/A |
| phone\_No | Member telephone number | NUMBER (20) | N/A |
| mem\_Type | The type of member E.g. Student or Staff | VARCHAR (7) | N/A |
| mem\_Status | Status of member account E.g. Active, suspended | VARCHAR (10) | N/A |
| **Student Table** | | | |
| lib\_cardNo | Unique number assigned to each member | NUMBER (4) | PK |
| max\_BorrowableStu | The maximum amount of resources that can be borrowed by a student member | NUMBER (1) | N/A |
| **Staff Table** | | | |
| lib\_cardNo | Unique number assigned to each member | NUMBER (4) | PK |
| max\_BorrowableStaff | The maximum amount of resources that can be borrowed by a staff member | NUMBER (2) | N/A |
| **Reservation Table** | | | |
| reservation\_ID | ID that is created once a reservation request is placed for a resource that is current on loan | NUMBER (5) | PK |
| lib\_cardNo | Unique number assigned o each member | NUMBER (4) | FK |
| resource\_ID | a unique ID that assigned to each resource | VARCHAR (5) | FK |
| res\_dateFrom | the date that a reservation is available from | DATE | N/A |
| res\_dateTo | The date that a reservation is available up to | DATE | N/A |
| res\_Status | Status of a reservation E.g. Notified, Accepted, Cancelled, Declined | VARCHAR (10) | N/A |
| **LibraryResource Table** | | | |
| resource\_ID | a unique ID that assigned to each resource | NUMBER (5) | PK |
| category | Category or genre of resource | VARCHAR (5) | N/A |
| source | Author, director, etc | VARCHAR (20) | N/A |
| description | Description of the resource i.e. blurb | VARCHAR (90) | N/A |
| publication\_Date | publication date of resource | NUMBER (4) | N/A |
| class\_No | identification for the different classes | NUMBER (1) | N/A |
| floor\_No | Floor number | NUMBER (1) | N/A |
| shelf\_No | Shelf number | VARCHAR (2) | N/A |
| **Book** | | | |
| resource\_ID | a unique ID that assigned to each resource | VARCHAR (5) | PK |
| book\_Title | Book Title | VARCHAR (50) | N/A |
| **Video** | | | |
| resource\_ID | a unique ID that assigned to each resource | NUMBER (5) | PK |
| video\_Titile | Video Title | VARCHAR (40) | N/A |
| **DVD** | | | |
| resource\_ID | a unique ID that assigned to each resource | NUMBER (5) | PK |
| DVD\_Titile | DVD Title | VARCHAR (40) | N/A |
| **CD** | | | |
| resource\_ID | a unique ID that assigned to each resource | NUMBER (5) | PK |
| CD\_Title | CD Title | VARCHAR (70) | N/A |
| **Copy Table** | | | |
| copy\_ID | a unique ID that assigned to each copy | NUMBER (3) | PK |
| copies\_Available | Number of copies currently available that can be borrowed | NUMBER (1) | N/A |
| resource\_ID | a unique ID that assigned to each resource | VARCHAR (5) | FK |
| copies\_Loaned | Number of copies currently on loan | NUMBER (1) | N/A |
| **Loan Table** | | | |
| loan\_ID | a unique ID that assigned to each loan | VARCHAR (7) | PK |
| loan\_Status | Status of a loan E.g. overdue, Borrowed | VARCHAR (8) | N/A |
| lib\_cardNo | Unique number assigned to each member | NUMBER (4) | FK |
| resource\_ID | a unique ID that assigned to each resource | VARCHAR (5) | FK |
| Issue\_Date | Date the resource is loaned out by member | DATE | N/A |
| due\_Date | Date the resource is due | DATE | N/A |
| return\_Date | Date the resource is actually returned by member | DATE | N/A |
| loan\_Amt | fine amount due for a specific loan | CHAR (7) | N/A |
| over\_dueAmt | combined amount of fines due | CHAR (7) | N/A |
| **LoanArchive** | | | |
| loan\_ID | a unique ID that assigned to each loan | VARCHAR (7) | PK |
| loan\_Status | Status of the loan E.g. borrowed, available | VARCHAR (8) | N/A |
| resource\_ID | a unique ID that assigned to each resource | VARCHAR (5) | FK |
| issue\_Date | Date the resource is loaned out by member | DATE | N/A |
| due\_Date | Date the resource is due | DATE | N/A |
| return\_Date | Date the resource is actually returned by member | DATE | N/A |
| loan\_Amt | fine amount due for a specific loan | CHAR (7) | N/A |
| over\_dueAmt | combined amount of fines due | CHAR (7) | N/A |

Data Table

[DBS - Overall Table with Data.xlsx](https://qmulprod-my.sharepoint.com/:x:/r/personal/ec23991_qmul_ac_uk/Documents/DBS%20-%20Overall%20Table%20with%20Data.xlsx?d=w10a99f286c08429b9e1b23b87a57ded0&csf=1&web=1&e=b6GHbs)

# References:

Cepeda Porras, G. and Guéhéneuc, Y.G., 2010. An empirical study on the efficiency of different design pattern representations in UML class diagrams. Empirical Software Engineering, 15(5), pp.493-522. <https://doi.org/10.1007/s10664-009-9125-9>

Dennis, A., Wixom, B. and Tegarden, D., 2015. Systems analysis and design: An object-oriented approach with UML. John Wiley & sons.

Eessaar, E. (2016). The database normalization theory and the theory of normalized systems: Finding a common ground. *Baltic Journal of Modern Computing, 4*(1), 5-33. Available at: <https://www.proquest.com/scholarly-journals/database-normalization-theory-normalized-systems/docview/1785391026/se-2> Assessed on: 12th November 2023

Harrington, J.L., 2016. *Relational database design and implementation*. Morgan Kaufmann.

Pinheiro da Silva, P. and Paton, N.W., 2000, October. UML i: The Unified Modelling Language for Interactive Applications. In International Conference on the Unified Modelling Language (pp. 117-132). Berlin, Heidelberg: Springer Berlin Heidelberg.

<https://doi.org/10.1007/3-540-40011-7_9>

Watt, A. and Eng, N., 2014. *Database design*. BCcampus.