Assignment 3

SCS Resource Access Database Design Project Logical Database Design.

|  |  |
| --- | --- |
| Student number: C3238805 | Name: NI ZENG |
| COMP1140: Database and Information Management. | Due: 11:59 pm, Thursday, October 05, 2020 WORTH 15% of the final assessment mark. |

### 1.1 Do concept database design, including Requirement Specification.

For the last Assignment 2: SCS Resource Management Database Design Project- Requirement Analysis and Conceptual Design. The comments and suggestions are being provided by the markers both on the Business rules and EER model/diagram section.

Base on the marker’s feedback In the Data Dictionary part of last assignment 2, I have also corrected the error in this assignment 3. This Assignment 3 mainly focus on the rightness and completion of the last Assignment 2.

### 1.11. Requirement Specification (including data requirements, transaction requirements and business rules).

#### \* Data Requirements (Rightness, Clarity and completeness)

In the Data Requirements, I have included Resources, Location, Category, Members, Staff, Student, Course offering, Privileges, Acquisition Request, reservations and loan section of the last Assignment. The clarity and completeness for those sections are well described.

I have also included some business rules inside the Data Requirements section which it should not have. Those relative business rules should only be included in Business Rules section.

In the Category section “Each category can contain no limited resources at any one category.” In the staff section “a staff member can unlimited resources at any one time.” In the Privileges section “SCS Resource Management has designed each privilege can allow to hold unlimited categories.”.

#### \* Transaction Requirements

For the transaction Requirements, I have 3 main section which are Data entry, Data update/deletion and Data queries. Most of the details are completed and well described but for the Data update/deletion.

Insert/Update/delete the details of a resource.

Insert/Update/delete the details of a member of student’s offering and points.

Insert/Update/delete the details of the privilege related to the

course.

Insert/Update/delete the details of a category.

Insert/Update/delete the details of a member of staff.

Insert/Update/delete the details of an acquisition request.

**Data queries**

Search a loaned item based on a member’s id number, on a date.

Find the student with a late returned item number.

Find member with its specific member id.

List all the student’s privilege status.

List all the reservations for a specific item.

Report of points earned for specific student during a time period.

List the resource stored locations.

List of the resource’s category provided by the school.

Identify the details of student’s privileges.

List of the course offering information.

Search course offering by its offered semester, year offered, date begins and date ends.

Identify the member that has reach their maximum borrowing items capacity.

List the name, address, phone, email, status, item borrowed of student.

List the name, address, phone, email, status and item reserved of staff.

Find the acquisition request by member id.

#### \* Business Rules

The business rules are to Organising resources to provide efficient access to them. So that the member can have a far opportunity to access the resources they need for their study.

Student has an initial of 12 points, a penalty of 3 points is incurred for each overdue day. Once the point is reduced to 0, the student member’s status is disabled, disallowing any borrowing and reservation privileges.

The student can have minimum 1 course offering and no maximum limits on how many courses offering a student can have.

The administrator has rights to reset or amend points.

If the student has non cancellation of reservation, the system will automatically reward 1 point back to the student member.

When a student’s enrolled course end date is later than the current date, the borrowing privileges are automatically taken away and the status of the student member is set to disabled.

Student must have at least one course offering. the student can have minimum 1 course offering and no maximum limits on how many courses offering a student can have.

Staff members have privileges access to all the resources in SCS. A member can make unlimited acquisition requests.

A staff can borrow and reserve resources unlimited. A reserved item will be cancelled if it is not pick up a day after the required date.

The administrator holds the right to cancel any reservation.

If an item is reported damage or lost by the administrator, the member lending the resource must pay the fees of item’s original asset value.

A resource can either store as movable or immovable type of resource. A resource can only be stored in one location, but one location may contain multiple resources.

### 3. EER Diagram and Data Dictionary

#### \* EER Diagram (Rightness)

Included in the assignment files.

### \* Data Dictionary (Rightness and completeness)

The Data Dictionary section for Assignment 1 needs to be slightly adjusted due to the changes that I made in this submission. For the relationship types in Data Dictionary.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Multiplicity** | **Relationship** | **Multiplicity** | **Entity Name** |
| resources | 1..1 | Has | 0..\* | movable |
| 1..1 | has | 0..\* | immovable |
| 0.\* | Is stored in | 1..1 | location |
| 1..\* | Is Belongs to | 0..\* | category |
| category | 1..1 | Is assigned to | 1..\* | privileges |
| privileges | 0..\* | Base on | 0..\* | Course offering |
| students | 0..\* | Is a | 1..1 | member |
| 1..1 | is offered to | 1..\* | Course offering |
| staff | 0..\* | Is a | 1..1 | member |
| 0..\* | Can borrow | 0..\* | resources |
| members | 1..1 | Can make | 0..\* | Acquisition request |
| reservations | 0..\* | to | 1..1 | member |
| 1..\* | Is Of | 1..1 | resources |
| loan | 0..\* | Is of | 1..1 | movable |
| 0..\* | To | 1..1 | member |

1.2 Map the EER model to the relational model. Document the relational schema in DBDL.

Rule 1 is: for super/subclass, Mandatory & Disjoint: create multi- relations, with 1 relation for each combined superclass/subclass.

**movable resource** (resource\_id, description, status, name, make, manufacturer, model, year, asset\_value, max\_borrowing\_time)

**Primary Key** resource\_id

**immovable resource** (resource\_id, description, status, capacity, max\_borrowing\_time)

**Primary Key** resource\_id

**staff members** (member\_id, name, address, phone, email, status, comment\_field, number\_items)

**Primary Key** member\_id

**student members** (member\_id, name, address, phone, email, status, comment\_field, points)

**Primary Key** member\_id

Rule 2 is: 1 to \*: for resources is stored in locations, resources is belong to category, category is assigned to privileges, students is offered to course\_offering, members if request acquisitionRequest ----add Foreign Key at \* side.

**resources** (resource\_id, description, status, location\_id, category\_code)

**Primary Key** resource\_id

more than one resources can be in a single location **Foreign Key** location\_id **references** locations (location\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Foreign Key** category\_code **references** category (category\_code) **ON UPDATE CASCADE ON DELETE CASCADE**

**Locations** (location\_id, room, building, campus)

**Primary Key** location\_id

**Category** (category\_code, name, description, maxTime\_borrow/book)

**Primary Key** category\_code

**privileges** (course\_id, name, description, category\_code, max\_resources)

**Primary Key** course\_id

**Foreign Key** category\_code **references** category (category\_code) **ON UPDATE CASCADE ON DELETE CASCADE**

**Student** (member\_id, points)

**Foreign Key** member\_ **references** student members(member\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Course\_offering** (offering\_id, course\_id, semester\_offered, year\_offered, date\_begins, date\_ends)

**Primary Key** offering\_id

**Foreign Key** course\_id **references** privileges(course\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Foreign Key** member\_id **references** students member (member\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Alternate Key** course\_id

**acquisitionRequest** (acquisition\_id, member\_id, itemName, make, manufacturer, model, year, description, urgency, admin\_note)

**Primary Key** acquisition\_id

**Foreign Key** member\_id **references** student and Staff (member\_id) **ON UPDATE CASCADE ON de CASCADE**

**Alternate Key** member\_id

Rule 3 is: \* to \*: for staff & resources, members reservations resources, members loan movable, course\_offering& privileges ----create a “Relationship” relation, add 2 foreign keys.

**Relation staff resources** (member\_id, resource\_id, number\_items)

**Primary Key** member\_id, resource\_id

**Foreign Key** member\_id **references** staff members(member\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Alternate Key** resource\_id **references** resources(resource\_id)

**Reservations** (member\_id, resource\_id, dateTimeRequested, dataTime\_Due)

**Primary Key** member\_id, resource\_id, dateTimeRequested

**Foreign Key** member\_id **references** members(member\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Foreign Key** resource\_id **references** resources(resource\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Loan** (member\_id, resource\_id, dataTimeLoaned, dateTimeDue, dateReturned)

**Primary Key** member\_id, resource\_id, dataTimeLoaned

**Foreign Key** member\_id **references** members(member\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Foreign Key** resource\_id **references** movable(resource\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Relation course\_offering privileges** (offering\_id, course\_id, member\_id)

**Primary Key** offering\_id

**Foreign Key** member\_id **reference** student members (member\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

**Foreign Key** course\_id **references** privileges (course\_id) **ON UPDATE CASCADE ON DELETE CASCADE**

### 1.3 Normalize the relational schema to Boyce-Codd Normal Form (Point out what norm form what norm form)

Normalise relationship up to BCNF:

**resources** (resource\_id, description, status, location\_id, category\_code) -> 3NF (Transitive dependency)

so normalised it as:

**Resources** (resource\_id, description, status, room, building, campus, category\_code)

**Category** (category\_code, name, description, maxTime\_borrow/book, resource\_id)

**privileges** (course\_id, name, description, category\_code, max\_resources) 🡪 3NF (Full functionally dependency)

so normalised it as:

**privileges** (course\_id, maxTime\_borrow/book, category\_code)

**Category** (category\_code, name, description)

**acquisitionRequest** (acquisition\_id, member\_id, itemName, make, manufacturer, model, year, description, urgency, admin\_note) 🡪 3NF to BCNF

so normalised it as:

**acquisitionRequest** (acquisition\_id, member\_id)

**member** **acquisitionRequest** (acquisition\_id, itemName, make, manufacturer, model, year, description, urgency, admin\_note)

**Course\_offering** (offering\_id, course\_id, semester\_offered, year\_offered, date\_begins, date\_ends) 🡪 2NF (Full functionally dependency) to BCNF

so normalised it as:

**Course\_offering** (offering\_id, semester\_offered, year\_offered, date\_begins, date\_ends)

**privileges** (course\_id)

**list of all the relations that are in BCNF. list of all the relations that are in BCNF:**

A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. for each functional dependency ( X → Y ), X should be a super Key, R must be in 3rd Normal Form.

**movable resource** (resource\_id, description, status, name, make, manufacturer, model, year, asset\_value, max\_borrowing\_time) 🡪 BCNF

Functional dependencies:

resource\_id -> description, status, name, make, manufacturer, model, year, asset\_value, max\_borrowing\_time

**immovable resource** (resource\_id, description, status, capacity, max\_borrowing\_time) 🡪 BCNF

Functional dependencies:

resource\_id -> description, status, capacity, max\_borrowing\_time

**staff members** (member\_id, name, address, phone, email, status, comment\_field, number\_items) 🡪 BCNF

Functional dependencies:

member\_id -> name, address, phone, email, status, comment\_field, number\_items

**student members** (member\_id, name, address, phone, email, status, comment\_field, points) 🡪 BCNF

Functional dependencies:

member\_id -> name, address, phone, email, status, comment\_field, points

**Locations** (location\_id, room, building, campus) 🡪 BCNF

Functional dependencies:

location\_id -> room, building, campus

**Category** (category\_code, name, description, maxTime\_borrow/book) 🡪 BCNF

Functional dependencies:

category\_code -> name, description, maxTime\_borrow/book

**Student** (member\_id, points) 🡪 BCNF

Functional dependencies:

member\_id -> points

**Relation staff resources** (member\_id, resource\_id, number\_items) 🡪 BCNF

Functional dependencies:

member\_id, resource\_id -> number\_items

**Reservations** (member\_id, resource\_id, dateTimeRequested, dataTime\_Due) 🡪 BCNF

Functional dependencies:

member\_id, resource\_id, dateTimeRequested -> dataTime\_Due

**Loan** (member\_id, resource\_id, dataTimeLoaned, dateTimeDue, dateReturned) 🡪 BCNF

Functional dependencies:

member\_id, resource\_id, dataTimeLoaned -> dataTime\_Due

**Relation course\_offering privileges** (offering\_id, course\_id, member\_id) 🡪 BCNF

Functional dependencies:

Offering\_id -> member\_id, course\_id

## SQL scripts

--to create some tables of assignment 3

Drop TABLE AcquisitionRequest

Drop TABLE Reservations

Drop TABLE Loan

Drop TABLE Students

Drop TABLE Staffs

Drop TABLE Members

Drop TABLE Course\_offering

Drop TABLE Privileges

Drop TABLE Moveable

Drop TABLE Immoveable

Drop TABLE Resources

Drop TABLE Category

Drop TABLE Locations

go

CREATE TABLE Category (

category\_code VARCHAR(10) NOT NULL,

name VARCHAR(20),

Descriptions varchar(50),

maxTime\_borrow\_book INT Not Null,

PRIMARY KEY (category\_code),

);

go

CREATE TABLE Locations (

location\_id VARCHAR(20) NOT NULL,

room Varchar(20),

building Varchar(20),

campus Varchar(20),

PRIMARY KEY (location\_id),

);

go

Create Table Resources(

resource\_id VARCHAR (10) NOT NULL,

description VARCHAR(50) NOT NULL,

status VARCHAR(20) DEFAULT 'available' CHECK (status IN ('available', 'occupied', 'damaged'))NOT NULL,

location\_id VARCHAR(20),

category\_id VARCHAR(10),

PRIMARY KEY (resource\_id),

FOREIGN KEY (category\_id) REFERENCES Category(Category\_code) ON UPDATE CASCADE ON DELETE NO ACTION,

FOREIGN KEY (location\_id) REFERENCES locations(Location\_id) ON UPDATE CASCADE ON DELETE NO ACTION

);

go

CREATE TABLE Moveable (

resource\_id VARCHAR (10) NOT NULL,

name VARCHAR (20),

make VARCHAR(10),

manufacturer VARCHAR(30),

model VARCHAR(30),

year DATETIME2,

asset\_value VARCHAR(15),

max\_borrowing\_time INT,

PRIMARY KEY (resource\_id),

FOREIGN KEY (resource\_id) REFERENCES resources(resource\_id) ON UPDATE CASCADE ON DELETE NO ACTION

);

go

CREATE TABLE Immoveable (

resource\_id VARCHAR (10) NOT NULL,

name VARCHAR (20),

capacity INT,

max\_borrowing\_time INT,

PRIMARY KEY (resource\_id),

FOREIGN KEY (resource\_id) REFERENCES resources(resource\_id) ON UPDATE CASCADE ON DELETE NO ACTION

);

go

CREATE TABLE Members (

member\_id VARCHAR(10) NOT NULL,

name VARCHAR(20) NOT NULL,

address VARCHAR(20),

phone INT,

email VARCHAR(20) NOT NULL,

status VARCHAR(8) DEFAULT 'active' CHECK (Status IN ('active', 'expire')) NOT NULL,

comments\_field VARCHAR(20),

PRIMARY KEY (member\_id),

);

go

CREATE TABLE Privileges (

course\_id VARCHAR(10) NOT NULL,

name VARCHAR(20) NOT NULL,

description VARCHAR(20),

category\_code VARCHAR(10),

max\_resources INT,

PRIMARY KEY (course\_id),

FOREIGN KEY (category\_code) REFERENCES Category(category\_code) ON UPDATE CASCADE ON DELETE NO ACTION

);

go

CREATE TABLE Course\_offering (

offering\_id VARCHAR(10) NOT NULL,

course\_id VARCHAR(10) NOT NULL,

name VARCHAR(20) NOT NULL,

semester\_offered INT,

year\_offered DATE,

date\_begins DATETIME2,

date\_ends DATETIME2,

PRIMARY KEY (offering\_id),

FOREIGN KEY (course\_id) REFERENCES Privileges(course\_id) ON UPDATE CASCADE ON DELETE NO ACTION

);

go

CREATE TABLE Students (

member\_id VARCHAR(10) NOT NULL,

offering\_id VARCHAR(10) NOT NULL,

points INT,

PRIMARY KEY (member\_id),

Foreign Key (member\_id) references Members(member\_id) On Update Cascade On Delete NO ACTION,

Foreign Key (offering\_id) references Course\_offering(offering\_id) On Update Cascade On Delete NO ACTION

);

go

CREATE TABLE Staffs (

member\_id VARCHAR(10) NOT NULL,

number\_items INT NOT NULL,

PRIMARY KEY (member\_id),

Foreign Key (member\_id) references Members(member\_id) On Update Cascade On Delete NO ACTION

);

go

CREATE TABLE Loan (

member\_id VARCHAR(10) NOT NULL,

resource\_id VARCHAR(10) NOT NULL,

dateTimeLoaned DATE NOT NULL,

dateTimeDue DATE NOT NULL,

dateReturned DATE,

PRIMARY KEY (dateTimeLoaned),

FOREIGN KEY (member\_id) REFERENCES Members ON UPDATE CASCADE ON DELETE NO ACTION,

FOREIGN KEY (resource\_id) REFERENCES Moveable ON UPDATE CASCADE ON DELETE NO ACTION,

);

go

CREATE TABLE Reservations (

member\_id VARCHAR(10) NOT NULL,

resource\_id VARCHAR(10) NOT NULL,

dateTimeRequested DATETIME NOT NULL,

dateTimeDue DATETIME NOT NULL,

PRIMARY KEY (dateTimeRequested),

FOREIGN KEY (member\_id) REFERENCES Members ON UPDATE CASCADE ON DELETE NO ACTION,

FOREIGN KEY (resource\_id) REFERENCES Resources ON UPDATE CASCADE ON DELETE NO ACTION,

);

go

CREATE TABLE AcquisitionRequest (

acquisitiion\_id VARCHAR(10) NOT NULL,

member\_id VARCHAR(10) NOT NULL,

itemName VARCHAR(20) NOT NULL,

make VARCHAR(15) NOT NULL,

manufacturer VARCHAR(20) NOT NULL,

model VARCHAR(20),

year DATETIME2,

description VARCHAR(20),

urgency VARCHAR(15),

admin\_note VARCHAR(20),

PRIMARY KEY (acquisitiion\_id),

FOREIGN KEY (member\_id) REFERENCES Members ON UPDATE CASCADE ON DELETE NO ACTION

);

go

--Data creation

INSERT INTO locations VALUES

('Robotics Lab','101','ES','Callaghan'),

('AC Storage' ,'102','CT','Callaghan'),

('Optical Store','103','ICT','Callaghan')

go

INSERT INTO Category VALUES

('C1','MATH2001','math','7'),

('C2' ,'COMP2002','computer','14'),

('C3','CIVL2003','civil','5'),

('C4','camera','computer','14')

go

INSERT INTO Resources VALUES

('MR00001', 'bluetooth 50W speaker ','available','Robotics Lab','C2'),

('MR00002', 'wireless speaker','available','AC Storage','C1'),

('MR00003', 'high-end headphone','available','Optical Store','C3'),

('MR00004', 'high-end camera','available','Optical Store','C4'),

('R001', 'Lab','available','Robotics Lab','C3'),

('R002', 'Lecture','available','Robotics Lab','C2'),

('R003', 'Libary','available','Robotics Lab','C1')

go

INSERT INTO Moveable VALUES

('MR00001','50W speaker','A','SONY','SX11','2020','$97','7'),

('MR00002','speaker','B','Philips','WE69','2009','$34','5'),

('MR00003','headphone','A','Philips','HD380','2019','$189','4'),

('MR00004','camera','B','cannon','HD1080','2020','$300','2')

go

INSERT INTO Immoveable VALUES

('R001','Lab','30','4'),

('R002','Lecture','200','5'),

('R003','Libary','120','2')

go

INSERT INTO Privileges VALUES

('SENG1050','Data Structure','IT','C2','5'),

('MATH2050','Math Advance','civil enginnering','C3','10'),

('COMP1140','Data Management','Software enginnering','C4','20')

go

INSERT INTO Course\_offering VALUES

('S2018','SENG1050','IT','1','2018','2018-09-01','2022-06-01'),

('S2019','MATH2050','civil enginnering','2','2019','2019-09-01','2023-06-01'),

('S2020','COMP1140','Software enginnering','2','2020','2020-09-01','2024-06-01')

go

INSERT INTO Members VALUES

('M009', 'Sophia S','Waterside Dr',0408888331,'11111@gmail.com','active','fees paid'),

('M010', 'Aiyana H','City Dr',0406666331,'2222222@gmail.com', 'active','fees paid'),

('M011', 'Oliver Z','Wallsend Dr',0405555331,'33333@gmail.com','active','fees paid'),

('M012', 'John A','Wallsend Dr',0405555331,'33333@gmail.com','active','fees paid'),

('M013', 'George B','Wallsend Dr',0405555331,'33333@gmail.com','active','fees paid'),

('M014', 'Shandra C','Wallsend Dr',0405555331,'33333@gmail.com','active','fees paid')

go

INSERT INTO Staffs VALUES

('M009', 12),

('M010', 0),

('M011', 8)

go

INSERT INTO Students VALUES

('M012', 'S2018','8'),

('M013', 'S2019','6'),

('M014', 'S2020','10')

go

INSERT INTO Loan VALUES

('M012', 'MR00001','2019-05-25','2019-06-02', '2019-06-01'),

('M012', 'MR00001','2020-01-01','2020-01-03', '2020-01-03'),

('M013', 'MR00003','2020-02-20','2020-02-24', '2020-02-22'),

('M013', 'MR00003','2020-11-01','2020-11-05', '2020-11-04'),

('M014', 'MR00004','2019-03-01','2019-03-05', '2019-03-05'),

('M014', 'MR00004','2020-11-02','2020-11-04', '2020-11-03'),

('M014', 'MR00004','2020-11-03','2020-1-05','2020-11-04')

go

INSERT INTO Reservations VALUES ('M011','R003','2019-07-02 08:00:00','2019-07-06 10:00:00');

INSERT INTO Reservations VALUES ('M009','R001','2020-05-01 08:00:00','2019-05-01 12:00:00');

INSERT INTO Reservations VALUES ('M009','R002','2020-06-05 08:00:00','2019-06-10 13:00:00');

INSERT INTO Reservations VALUES ('M010','R002','2020-09-19 08:00:00','2020-09-19 13:00:00');

INSERT INTO Reservations VALUES ('M011','R003','2020-09-19 08:00:00','2020-09-19 10:00:00');

INSERT INTO Reservations VALUES ('M011','R003','2020-09-19 10:00:00','2020-09-19 12:00:00');

INSERT INTO AcquisitionRequest VALUES

('a001','M009','Lab Top','2018','Sony','VS888','2018-05-14 14:00:00','color: black','very','approved'),

('a002','M010','keyboard','2019','Sony','SE888','2019-05-14 15:30:00','color: red','very','approved'),

('a003','M010','monitor','2020','Sony','FE888','2020-05-14 10:10:00','color: blue','very','approved'),

('a004','M011','PHONE','2020','iphone','12 pro','2020-05-14 11:10:00','color: blue','very','approved'),

('a005','M011','PHONE2','2020','iphone','11 pro','2020-05-14 12:10:00','color: yellow','very','approved')

go

DELETE FROM Locations

DELETE FROM Category

DELETE FROM Resources

DELETE FROM Moveable

DELETE FROM Immoveable

DELETE FROM Privileges

DELETE FROM Course\_offering

DELETE FROM Members

DELETE FROM Staffs

DELETE FROM Students

DELETE FROM Loan

DELETE FROM Reservations

DELETE FROM AcquisitionRequest

SELECT\* FROM Locations

SELECT\* FROM Category

SELECT\* FROM Resources

SELECT\* FROM Moveable

SELECT\* FROM Immoveable

SELECT\* FROM Privileges

SELECT\* FROM Course\_offering

SELECT\* FROM Members

SELECT\* FROM Staffs

SELECT\* FROM Students

SELECT\* FROM Loan

SELECT\* FROM Reservations

SELECT\* FROM AcquisitionRequest

---Q1: Print the name of student(s) who has/have enrolled in the course with course id xxx.

SELECT s.member\_id, m.name, c.course\_id

FROM Members m,Students s, Course\_offering c

WHERE m.member\_id = s.member\_id AND s.offering\_id = c.offering\_id

--Q2: Print the maximal number of speakers that the student with name xxx can borrow.

SELECT m.name, MAX(p.max\_resources) AS maximal\_number\_of\_speakers

FROM Members m,Students s, Course\_offering c , Privileges p,Category ca, Resources r

WHERE m.member\_id = s.member\_id AND s.offering\_id = c.offering\_id AND c.course\_id = p.course\_id AND p.category\_code = ca.category\_code AND ca.category\_code = r.category\_id

AND r.description LIKE '%speaker%'

GROUP BY m.name

--Q3: For a staff member with id number xxx, print his/her name and phone number, the total

--number of acquisition requests and the total number of reservations that the staff had

--made in 2019.

SELECT m.member\_id, m.name,m.phone,COUNT(DISTINCT a.acquisitiion\_id) AS total\_AcquisitionRequest, COUNT(DISTINCT r.dateTimeRequested) AS total\_AcquisitionRequest

FROM Members m, Staffs st, AcquisitionRequest a,Reservations r

WHERE m.member\_id = st.member\_id AND st.member\_id = a.member\_id AND a.member\_id = r.member\_id AND r.dateTimeRequested LIKE '%2019%'

GROUP BY m.member\_id, m.name,m.phone

--Q4: Print the name(s) of the student member(s) who has/have borrowed the category with

--the name of camera, of which the model is xxx, in this year. Note: camera is a category,

--and model attribute must be in movable table.

SELECT m.name, ca.name,mo.model,L.dateTimeLoaned

FROM Members m,Students s,Course\_offering c, Privileges p, Category ca, Resources r ,Moveable mo, Loan l

WHERE m.member\_id = s.member\_id AND s.offering\_id = c.offering\_id AND c.course\_id = p.course\_id AND p.category\_code = ca.category\_code

AND ca.category\_code = r.category\_id AND r.resource\_id = mo.resource\_id AND mo.resource\_id = l.resource\_id AND ca.name LIKE '%camera%'

AND l.dateTimeLoaned LIKE '%2020%'

--Q5: Find the moveable resource that is the mostly loaned in the current month. Print the

--resource id and resource name.

SELECT l.resource\_id,m.name, COUNT(\*) AS mostly\_loaned\_current\_month

FROM Loan l, Moveable m

WHERE l.resource\_id = m.resource\_id AND l.dateTimeLoaned BETWEEN '2020-11-01' AND '2020-11-30'

GROUP BY l.resource\_id,m.name

HAVING COUNT(m.resource\_id)>=

ALL (SELECT COUNT(\*)

FROM Loan l, Moveable m

WHERE l.resource\_id = m.resource\_id AND l.dateTimeLoaned BETWEEN '2020-11-01' AND '2020-11-30'

GROUP BY l.resource\_id,m.name)

--Q6: For each of the three days, including May 1, 2020, June 5, 2020 and September 19,

--2020, print the date, the name of the room with name xxx, and the total number of

--reservations made for the room on each day.

SELECT

r.dateTimeRequested,

i.name,

COUNT(CASE WHEN r.dateTimeRequested BETWEEN '2020-09-19 00:00:00' AND '2020-09-19 23:00:00' then 1 ELSE NULL END) as '2020-09-19',

COUNT(CASE WHEN r.dateTimeRequested BETWEEN '2020-06-05 00:00:00' AND '2020-06-05 23:00:00' then 1 ELSE NULL END) as '2020-06-05',

COUNT(CASE WHEN r.dateTimeRequested BETWEEN '2020-05-01 00:00:00' AND '2020-05-01 23:00:00' then 1 ELSE NULL END) as '2020-05-01'

FROM Reservations r, Immoveable i

WHERE r.resource\_id = i.resource\_id and ((r.dateTimeRequested BETWEEN '2020-09-19 00:00:00' AND '2020-09-19 23:00:00')

or ((r.dateTimeRequested BETWEEN '2020-06-05 00:00:00' AND '2020-06-05 23:00:00')or (r.dateTimeRequested BETWEEN '2020-05-01 00:00:00' AND '2020-05-01 23:00:00')))

GROUP BY r.dateTimeRequested,i.name