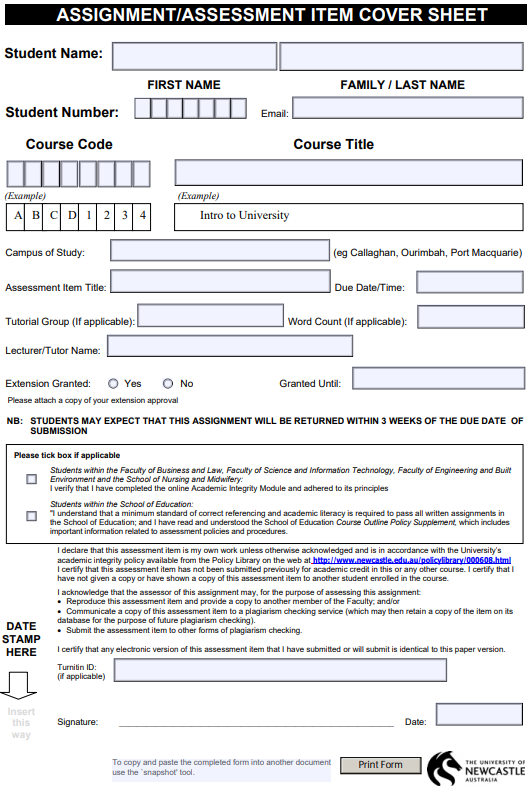
**COMP1140 Assignment 2**

6/11/20

Suhuai Luo

Friday 10-12pm Lab

Assignment 3 6/11/20 11:59pm

Callaghan

C O M P 1 1 4 0 Database and Information management

3 3 4 9 8 2 8 c3349828@uon.edu.au

Harlan De Jong

Due: Wednesday 11:59pm, September 30, 2020

Harlan De Jong c3349828

**Reflection (for A1)**

During the process of creating the EER and report of assignment 1 I have learnt many techniques and common practices to rely on when interpreting the specifications for both assignment 1 and 2. As for assignment 1, I received a mark of 14.78/15. I lost marks only on the EER diagram for the multiplicity values assigned to a select number of major data items. This was due to a misinterpretation of the question. For assignment 2, I have rectified my mistake of the multiplicity issues and provided a change to the reservation and loan data items by removing the attributes memberID and resourceID from both. By removing them, it develops a more cohesive structure, especially for the task to map the EER into the relational model as these removed attributes can now be allocated the role of foreign keys for each. Thus, in conclusion, the changes made through the feedback provided allows the entire report to flow better and have a more linear structure from start to finish.

**Reflection (for A2)**

During the process of normalising my relational schema to Boyce-Codd Normal Form, I encountered many misunderstandings and was unsure about many aspects of the transformation. Especially with the identification of whether the relation was in BCNF or not. I have attempted to rectify my mistakes with the normalisation although I am aware it is still not to the highest quality as I feel I have not entirely grasped the concept yet. In terms of creating the database, I feel I made steps in the right direction and am more confident about this section of the assignment. The changes I have made do feel correct and seem to flow more cohesively through help via the feedback sheet.

**Data Requirements**

**Member**: A member is the superclass containing staff and students. These members have reservation and loan rights to resources that their course privilege allows them to have. Staff have privilege to all database interactions and in some cases priority over the students. For an individual to be considered a member, they must be enrolled in a at least one course offered by SCS or be a staff member. Members are what query the database system for resource acquisition, loans or reservations. The key data describing a member includes a unique member ID, name of the individual, address, phone number (up to a maximum of 3), email address, status (“disabled” or “active”) that’s initially “active” and a comments field.

**Student**: A student is the subclass of member and refers to the individuals enrolled in a course within SCS. The student has access to everything its privilege allows it to indicative of the course. The student can loan, reserve and request acquisitions of resources immediately. 12 points are allocated initially for the student and points are deducted according to various punishments, if the student has more than 0 points, they can still loan and reserve resources. Punishments:

1. Failing to return resources by due date (-3 points for each overdue day)
2. Non-cancellation of reservation if not picked up after a day of the required date (-1 point)

The data describing a student includes points (initially 12) and status (borrowing privilege, “enabled” or “disabled”) that is initially “enabled”.

**Acquisition**: An acquisition is a request performed by a member, either staff or student to acquire a resource. This process is prioritised for the staff. The data describing an acquisition includes a unique acquisition ID, the ID of the member requesting the resource, name of resource, make, manufacturer, model, year, description of resource and urgency indicator (“Urgent”, “Non-urgent”). The administrator of the system later assigns a status (“Acquired”, “Pending”, “Denied”), a fund code, a vendor code, a price and any other notes pertaining to the request.

**Course**: A course is an entity a student member can enrol in. All students must be enrolled in at least 1 course but no more than 4. The course/s a student is enrolled in determines the privileges they are allowed when it comes to resource loaning and reservation. The course information about course offerings and student enrolments are maintained. The data describing a course includes a unique course offering ID, a course ID, course name, semester offered, year offered, date the course begins and date the course ends.

**Privilege:** Privilege refers to the accessibility a student member has to a category of certain resources. It directly follows course and acts as a gateway until student course requirements are met. The data describing privilege includes a name, description, a category to which a privilege is granted for and a maximum number of resources that can be borrowed or booked at any given time from the category.

**Resource:** Resource refers to the commodity primarily interacted with among the SCS database via members. The resource is split down into two different entities known as movable and immovable. The resource belongs to a category of types indictive of the course, and the location of the resource is always monitored. A resource can be reserved by members of appropriate privilege. The data describing a resource includes a unique resource ID, a description and a status of the resource (“In Use”, “Maintenance”, “Available”, “Borrowed”, “Lost”, “Damaged”) that is initially always “available” until otherwise changed.

**Movable:** Movable is a subclass of resource and it refers to the physically movable resources that can have an updated location at any time. The movable resource is the only resource that can be loaned via a member, whereas all resources can be reserved. The data describing a movable resource includes the name, make, manufacturer, model, year and asset value. Zero to many movable resources can be loaned at any given time.

**Immovable:** Immovable is a subclass of resource and it refers to the physically immovable resource that exists only in one location for the entire time. The immovable resource cannot be loaned, only reserved via a member. The data describing an immovable resource includes the capacity of members it can withhold.

**Category:** Category refers to the group of resources various courses can use. The resources are categorised to allow the privilege to either allow or deny student access to a whole category of resources rather than individual resources indicative of the course. The data that describes the category includes a unique category ID, name, description and a max borrow time (in days and hours). Zero to many resources belong in a category and a privilege allows a category.

**Location:** Location refers to the physical location a resource is situated. Both movable and immovable resources have a location but due to the immovable resource’s immovability, only the movable resource can change location. The data that describes location include a unique location ID, room, building and campus. Zero to many resources have a location.

**Loan:** Loan refers to intermediate between member and movable, its role is to allow the member to loan a resource for a period of time requiring the member has valid privileges. There are no limits on the number of resources staff can loan. The data describing loan includes the date and time borrowed, date and time returned and the date and time due. Zero to many members can loan zero to many movable resources.

**Reservation:** Reservation refers to the intermediate between member and resource, its role is to allow the member to reserve a resource for a period of time requiring the member has valid privileges. The are no limits on the number of resources a staff can reserve. No two reservations can occur on the same date and time. The data that describes reservation includes the date and time reserved and date and time due. Zero to many members can reserve zero to many resources.

**Business Rules**

**Rules relating to Student Member**

1. A student’s borrowing privileges are taken away when the current date is later than the end date of all his/her enrolled course offerings. The status of student member is set to “disabled”.
2. A student member must be enrolled in more than 1 course but no more than 4.

**Rules relating to Privilege/Borrowing**

1. A member cannot borrow, or reserve more than the maximum number of items specified in his/her privileges at any given time.
2. Each student member has a default set of points earned, initially 12.
3. A penalty of 3 points is incurred for each overdue day.
4. If the points are reduced to 0, member status is “Disabled”, disallowing borrowing/reservation privileges.
5. The administrator has the rights to reset/amend points.

**Rules relating to Reservation**

1. A reserved resource is cancelled if it is not picked up after a day of the required date or due date (whichever is earlier).
2. Non-cancellation of a reservation by student member incurs a 1-point penalty.
3. The administrator holds the right to cancel any reservation.
4. No two reservations can occur on the same time and date.

**Rules relating to Category**

1. The duration of borrowing/reservation periods (either number of days or hours) are determined by the category to which the resource belongs

**Rules relating to Loaning**

1. Immovable resources cannot be loaned.

**Transaction Requirements**

**Data Entry**

* Enter the details for a student member
* Enter the details for a staff member
* Enter the request details for the acquisition via member input
* Enter the details of a course when a student enrols
* Enter the reservation details when a member reserves a resource
* Enter the loan details when a member loans a resource
* Enter the location of the resource
* Enter resource in the category list when a new resource is acquired
* Enter privileges of a student enrolled in courses
* Enter privileges for student whose course date ended
* Enter student points after reduction
* Enter student member status if points reach 0
* Enter new student points via admin reset/amend
* Enter resource reserved cancellation

**Data insert/deletion**

* Insert/delete the details for a student member
* Insert/delete the details for a staff member
* Insert/delete the request details for the acquisition via member input
* Insert/delete the details of a course when a student enrols
* Insert/delete the reservation details when a member reserves a resource
* Insert/delete the loan details when a member loans a resource
* Insert/delete the location of the resource
* Insert/delete resource in the category list when a new resource is acquired
* Insert/delete privileges of a student enrolled in courses
* Insert/delete privileges for student whose course date ended
* Insert/delete student points after reduction
* Insert/delete student member status if points reach 0
* Insert/delete new student points via admin reset/amend
* Insert/delete resource reserved cancellation

**Data queries**

1. List the amount of points a student member has
2. List the location of a given resource
3. List the capacity of a given immovable resource
4. List the privileges a given course provides
5. Identify the urgency of an acquisition request
6. List the due date and time for a given loaned resource
7. List the date and time of a given reserved resource
8. Identify comments stored in a given member
9. Identify the description of a given category of resources
10. Identify the semester a course is offered
11. List a given members phone number/s
12. Identify a given student members status
13. Identify the price of a given acquisition request
14. Identify the room a given resource is located in
15. Identify the model of a given movable resource

**Data Dictionary**

**Entity Types**

|  |  |  |  |
| --- | --- | --- | --- |
| Entity Name | Description | Aliases | Occurrence |
| Member | A member is the term used for individuals at SCS (Staff or Students). | Individual | A **member** is either a student or staff. Staff can access all the entities while students must have specific requirements to do so, with reference to **course** and **privilege**. |
| Student | A subclass of Member, Student represents the individuals enrolled in a course within SCS |  | A student cannot access all the entities without proper **privilege,** and they must be enrolled in at least one **course** of the maximum four. |
| Staff | A subclass of Member, Staff represents the individuals teaching within SCS |  | Staff can access all the entities and has priority when requesting **acquisitions** of new/external **resources**. |
| Acquisition | Acquisition is the request for new/external resources at SCS. | Request | **Members** can request an acquisition of a new/external **resource**, staff have priority over this, but students can also provide input. **Acquisition** requests can be provided by any number of **members** zero to many times. |
| Course | The course refers to the degree the student is studying currently within SCS. | Degree | Any number of students can enrol in a **course**, but a student must be enrolled in one to four **courses** to be considered a student. The **course** a student is enrolled in is indicative to the **privileges** they get for **resource** use within a **category**. |
| Privilege | The privilege refers to the student’s ability to access resources if they are enrolled in a course that allows them to access it. |  | All students have a **privilege** for their course to access a **category** of **resources**. The privilege also stores the maximum **resources** a student can borrow. |
| Resource | General term describing the physical items being either loaned, reserved or acquired. Can be movable or immovable. |  | A **resource** is broken down into subcategories, being movable and immovable. All **resources** can be reserved, while only the movable resources can be loaned - by **members** of appropriate **privilege**. Zero to many **resources** can be utilised at any occurrence. |
| Movable | A subclass of resource that can be moved from one location to another |  | A **movable** **resource** can be borrowed unlike an **Immovable resource** and can be reserved. |
| Immovable | A subclass of resource that cannot be moved from one location to another |  | An **Immovable resource** cannot be borrowed, only reserved. |
| Category | Category holds the information of the various types of resources and stores them in different categories according to course privilege. |  | The **category** entity stores **course** relative materials in different sections. The **privilege** allows students to gain access to the required **category** for their degree while also denying access for those without. |
| Location | Location stores the information of where all resources are located within SCS |  | The **location** allows **resources** to be identified via a location ID. Any **resource** of a certain type will always have a single **location**, although many of the same **resources** can be scattered all over the university. |
| Loan | Stores the date and time of resources both borrowed and returned. Along with the date and time the resources are due back |  | The **loan** is the intermediate between **member** and **movable** **resource** and allows the **members** to borrow the **resource**. A **member** cannot borrow more than the maximum number of **resources** specified by their **privilege.** |
| Reservation | Stores the date and time a resource is reserved and the date and time a resource is due back |  | The **reservation** entity is the intermediate between **member** and **resource**. A **member** with the correct **privileges** can **reserve** any **resource** within their **course’s** respective **category** |

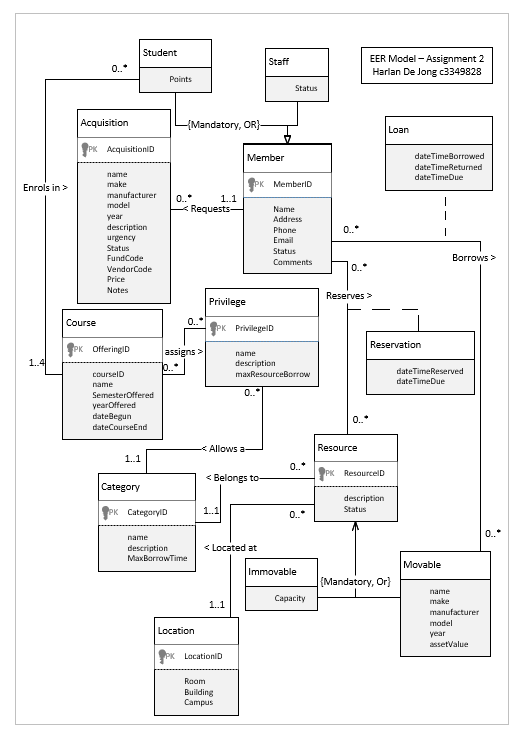
**Relationship Types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity name | Multiplicity | Relationship | Multiplicity | Entity name |
| Student | 0..\* | {Mandatory, OR}  Enrols in | 1..4 | Member  Course |
| Staff |  | {Mandatory, OR} |  | Member |
| Member | 1..1  0..\*  0..\* | Requests  Borrows  Reserves | 0..\*  0..\*  0..\* | Acquisition  Movable  Resource |
| Course | 0..\* | Assigns | 0..\* | Privilege |
| Privilege | 0..\* | Allows a | 1..1 | Category |
| Immovable |  | {Mandatory, OR} |  | Resource |
| Movable |  | {Mandatory, OR} |  | Resource |
| Resource | 0..\*  0..\* | Belongs to  Located at | 1..1  1..1 | Category  Location |

**Attribute Types**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Entity Name | Attributes | Description | Data Type  Length | Null | Multi-  valued | Derived | Default |
| Student | Points | Three points are deducted from students when possessing overdue resource and one point for non-cancellation of reserved resource not picked up after due date. When points reach zero, member status is disabled. The admin can reset/amend points | Int(2) | N | N | N | 12 |
|  | Status | Represents borrowing privilege for student (enabled/disabled). | Varchar(8) | N | N | N | “Enabled” |
| Member | MemberID | Unique identifier for each member | Char(8) | N | N | N |  |
|  | Name | The members name | Varchar(50) | N | N | N |  |
|  | Address | The members address | Varchar(50) | N | N | N |  |
|  | Phone | The members phone number | Int(8) | Y | Y | N |  |
|  | Email | The members email | Varchar(50) | N | Y | N |  |
|  | Status | The members status (disabled/active) | Varchar(8) | N | N | N | “Active” |
|  | Comments | Comments regarding the member | Varchar(100) | Y | Y | N |  |
| Acquisition | AcquisitionID | Unique identifier for an active acquisition | Char(5) | N | N | N |  |
|  | MemberID | The ID of the member requesting the resource | Char(8) | N | N | N |  |
|  | Name | The name of the requested resource | Varchar(50) | N | N | N |  |
|  | Make | The make of the requested resource | Varchar(50) | Y | Y | N |  |
|  | Manufacturer | The manufacturer of the requested resource | Varchar(50) | Y | N | N |  |
|  | Model | The model of the requested resource | Varchar(50) | Y | N | N |  |
|  | Year | The year model of the requested resource | Year | Y | N | N |  |
|  | Description | A description of the requested resource | Varchar(100) | Y | Y | N |  |
|  | Urgency | An indictor to the need of the required resource | Varchar(10) | Y | N | N | “Non-urgent” |
|  | Status | An indicator of the status (“Acquired”, “Pending”, “Denied”) | Varchar(8) | N | N | N |  |
|  | FundCode | A code referencing the fund | Char(8) | N | N | N |  |
|  | VendorCode | A code referencing the vendor | Char(8) | N | N | N |  |
|  | Price | The price of the resource | Float() | N | N | N |  |
|  | Notes | Any further notes pertaining to the request | Varchar(100) | Y | Y | N |  |
| Course | OfferingID | Unique identifier for the course offering | Char(8) | N | N | N |  |
|  | CourseID | Identifier for the course | Char(8) | N | N | N |  |
|  | Name | The name of the course | Varchar(50) | N | N | N |  |
|  | Semester-Offered | The semester the course is offered | Int(1) | N | Y | N |  |
|  | YearOffered | The year the course is offered | Year | N | Y | N |  |
|  | dateBegun | The date the course begun | Date | N | N | N |  |
|  | dateCourse-End | The date the course ends | Date | N | N | N |  |
| Privilege | PrivilegeID | Unique identifier for student privilege | Char(8) | N | N | N |  |
|  | Name | The name of the privilege | Varchar(50) | N | N | N |  |
|  | Description | A description of the privilege | Varchar(100) | Y | Y | N |  |
|  | MaxResource-Borrow | The maximum resources a student can borrow from the category | Int() | N | N | N |  |
| Resource | ResourceID | Unique identifier for the resource | Char(8) | N | N | N |  |
|  | Description | A description of the resource | Varchar(100) | Y | Y | N |  |
|  | Status | The status of the resource (“In Use”, “Maintenance”, “Available”, “Borrowed”, “Lost”, “Damaged”) | Varchar(11) | Y | N | N | “Available” |
| Movable | Name | The name of the movable resource | Varchar(50) | N | N | N |  |
|  | Make | The make of the movable resource | Varchar(50) | Y | Y | N |  |
|  | Manufacturer | The manufacturer of the movable resource | Varchar(50) | Y | N | N |  |
|  | Model | The model of the movable resource | Varchar(50) | Y | N | N |  |
|  | Year | The year model of the movable resource | Year | Y | N | N |  |
|  | AssetValue | The asset value of the movable resource | Float() | N | N | N |  |
| Immovable | Capacity | The capacity of members the immovable object can hold | Int() | N | N | N |  |
| Category | CategoryID | Unique identifier for the categories | Char(8) | N | N | N |  |
|  | Name | The name of the category | Varchar(50) | N | N | N |  |
|  | Description | A description of the category | Varchar(100) | Y | Y | N |  |
|  | MaxBorrow-Time | The maximum time allowed to borrow a resource | DateTime | N | N | N |  |
| Location | LocationID | Unique identifier for the location | Char(8) | N | N | N |  |
|  | Room | The room name of the stored resource | Varchar(50) | N | N | N |  |
|  | Building | The building name of the store resource | Varchar(50) | N | N | N |  |
|  | Campus | The campus name of the stored resource | Varchar(50) | N | N | N |  |
| Loan | DateTime-Borrowed | The date and time a resource was borrowed by a member (Unique) | DateTime | N | N | N |  |
|  | DateTime-Returned | The date and time a resource was returned by a member | DateTime | N | N | N |  |
|  | DateTimeDue | The date and time a resource is due back | DateTime | N | N | N |  |
| Reservation | DateTime-Reserved | The date and time a resource was reserved (Unique) | DateTime | N | N | N |  |
|  | DateTimeDue | The date and time a resource is due | DateTime | N | N | N |  |

**Revised EER Diagram**:



**Mapped EER Model to Relational Model**:

**Member(**MemberID, name, address, phone, email, status, comments**)**

**Primary Key**: MemberID

**Student(**StudentID, Points**)**

**Primary Key**: StudentID

**Foreign Key**: studentD **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE CASCADE**

**Staff(**StaffID,Status**)**

**Primary Key**: StaffID

**Foreign Key**: staffID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Acquisition(**AcquisitionID, MemberID, name, make, manufacturer, model, year, description, urgency, Status, FundCode, VendorCode, Price, Notes**)**

**Primary Key:** AcquisitionID

**Foreign Key**: MemberID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Course(**OfferingID, StudentID, courseID, name, semesterOffered, yearOffered, dateBegun, dateCourseEnd**)**

**Primary Key:** OfferingID

**Foreign Key:** PrivilegeID **references** Privilege(PrivilegeID)

**Privilege(**PrivilegeID, OfferingID, name, description, category, maxResourceBorrow**)**

**Primary Key:** PrivilegeID

**Foreign Key:** CategoryID **references** Category(CategoryID)

**Category(**CategoryID, PrivilegeID, ResourceID, name, description, MaxBorrowTime**)**

**Primary Key:** CatgeoryID

**Foreign Key**: PrivilegeID **references** Privilege(PrivilegeID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Foreign Key**: ResourceID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Resource(**ResourceID, description, status**)**

**Primary Key:** ResourceID

**Immovable(**ImmovableID, Capacity**)**

**Primary Key**: ImmovableID

**Foreign Key**: ImmovableID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE CASCADE**

**Movable(**MovableID, name, make, manufacturer, model, year, assetValue**)**

**Primary Key**: MovableID

**Foreign Key**: MovableID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE CASCADE**

**Location(**LocationID, ResourceID, Room, Building, Campus**)**

**Primary Key:** LocationID

**Loan(**LoanID, dateTimeBorrowed, MemberID, MovableID, dateTimeReturned, dateTimeDue**)**

**Primary Key:** LoanID

**Foreign Key:** MemberID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Foreign Key:** MovableID **references** Movable(MovableID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Reservation(**ReservationID,dateTimeReserved, MemberID, ResourceID, dateTimeDue**)**

**Primary Key:** ReservationID -

**Foreign Key:** MemberID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Foreign Key:** ResourceID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Course(**OfferingID, StudentID, courseID, name, semesterOffered, yearOffered, dateBegun, dateCourseEnd**)**

**Primary Key:** OfferingID

**Foreign Key:** StudentID **references** Student(StudentID)

**Privilege(**PrivilegeID, OfferingID, name, description, category, maxResourceBorrow**)**

**Primary Key:** PrivilegeID

**Foreign Key:** OfferingID **references** Course(OfferingID)

**Normalising to BCNF:**

**Member(**MemberID, name, address, phone, email, status, comments**)**

**Primary Key**: MemberID

2nd Normal Form Check:

* the key attribute is: MemberID and the candidate key is: MemberID.
* The Functional Dependency is: MemberID 🡪 name, address, phone, email, status, comments
* The left side is a not a subset of the candidate keys and the right side are non key attribute
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above it is not transitive
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above and there are no other FDs
* Thus, in BCNF

**Student(**StudentID, MemberID, Points**)**

**Primary Key**: StudentID

**Foreign Key**: MemberID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE CASCADE**

2nd Normal Form Check:

* the key attributes are: StudentID and MemberID and the candidate keys are: StudentID and MemberID.
* The Functional Dependency is: StudentID, MemberID 🡪 Points
* The left side is a subset of the candidate keys and the right side are not all key attributes thus, StudentId 🡪points (partial dependency)
* Thus, not in 2nf

**Staff(**StaffID,MemberID, Status**)**

**Primary Key**: StaffID

**Foreign Key**: MemberID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

2nd Normal Form Check:

* the key attributes are: StaffID and MemberID and the candidate keys are: StaffID and MemberID.
* The Functional Dependency is: StaffID, MemberID 🡪 Status
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Acquisition(**AcquisitionID, MemberID, name, make, manufacturer, model, year, description, urgency, Status, FundCode, VendorCode, Price, Notes**)**

**Primary Key:** AcquisitionID

**Foreign Key**: MemberID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

2nd Normal Form Check:

* the key attributes are: AcquisitionID and MemberID and the candidate keys are: AcquisitionID and MemberID.
* The Functional Dependency is: AcquisitionID, MemberID 🡪 name, make, manufacturer, model, year, description, urgency, Status, FundCode, VendorCode, Price, Notes
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Course(**OfferingID, StudentID, courseID, name, semesterOffered, yearOffered, dateBegun, dateCourseEnd**)**

**Primary Key:** OfferingID

**Foreign Key**: StudentID **references** Student(StudentID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

2nd Normal Form Check:

* the key attributes are: OfferingID and the candidate keys are: OfferingID. StudentID is a FK, It is not a part of a key in this relation.
* The Functional Dependency is: OfferingID 🡪 courseID, name, semesterOffered, yearOffered, dateBegun, dateCourseEnd
* The left side is a subset of the candidate keys and the right side are not all key attributes
* courseID 🡪 name (transitive)
* and other Candiate Keys in this relation.
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Privilege(**PrivilegeID, OfferingID, name, description, maxResourceBorrow**)**

**Primary Key:** PrivilegeID

**Foreign Key**: OfferingID **references** Course(OfferingID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

2nd Normal Form Check:

* the key attributes are: PrivilegeID and OfferingID and the candidate keys are: PrivilegeID and OfferingID.
* The Functional Dependency is: PrivilegeID, OfferingID 🡪 name, description, maxResourceBorrow
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Category(**CategoryID, PrivilegeID, ResourceID, name, description, MaxBorrowTime**)**

**Primary Key:** CatgeoryID

**Foreign Key**: PrivilegeID **references** Privilege(PrivilegeID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Foreign Key**: ResourceID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

2nd Normal Form Check:

* the key attributes are: CategoryID, PrivilegeID and ResourceID and the candidate keys are: CategoryID, PrivilegeID and ResourceID.
* The Functional Dependency is: CategoryID, PrivilegeID, ResourceID 🡪 name, description, MaxBorrowTime
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Resource(**ResourceID, description, status**)**

**Primary Key:** ResourceID

2nd Normal Form Check:

* the key attribute is: ResourceID and the candidate key is: ResourceID.
* The Functional Dependency is: ResourceID 🡪 description, status
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Immovable(**ImmovableID, ResourceID, Capacity**)**

**Primary Key**: ImmovableID

**Foreign Key**: ResourceID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE CASCADE**

2nd Normal Form Check:

* the key attributes are: ImmovableID and ResourceID and the candidate keys are: ImmovableID and ResourceID.
* The Functional Dependency is: ImmovableID, ResourceID 🡪 Capacity
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Movable(**MovableID, ResourceID, name, make, manufacturer, model, year, assetValue**)**

**Primary Key**: MovableID

**Foreign Key**: ResourceID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE CASCADE**

2nd Normal Form Check:

* the key attributes are: MovableID and ResourceID and the candidate keys are: MovableID and ResourceID.
* The Functional Dependency is: MovableID, ResourceID 🡪 name, make, manufacturer, model, year, assetValue
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Location(**LocationID, ResourceID, Room, Building, Campus**)**

**Primary Key:** LocationID

**Foreign Key**: ResourceID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

2nd Normal Form Check:

* the key attributes are: LocationID and ResourceID and the candidate keys are: LocationID and ResourceID.
* The Functional Dependency is: LocationID, ResourceID 🡪 Room, Building, Campus
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Loan(**LoanID, dateTimeBorrowed, MemberID, MovableID, dateTimeReturned, dateTimeDue**)**

**Primary Key:** LoanID **references** Loan(dateTimeBorrowed, dateTimeDue)

**Foreign Key:** MemberID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Foreign Key:** MovableID **references** Movable(MovableID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

2nd Normal Form Check:

* the key attributes are: LoanID, MemberID and MovableID and the candidate keys are: LoanID, MemberID and MovableID.
* The Functional Dependency is: LoanID, MemberID, MovableID 🡪 dateTimeBorrowed, dateTimeReturned, dateTimeDue
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Reservation(**ReservationID,dateTimeReserved, MemberID, ResourceID, dateTimeDue**)**

**Primary Key:** ReservationID **references** Reservation(dateTimeReserved, dateTimeDue)

**Foreign Key:** MemberID **references** Member(MemberID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

**Foreign Key:** ResourceID **references** Resource(ResourceID)

**ON UPDATE CASCADE, ON DELETE NO ACTION**

2nd Normal Form Check:

* the key attributes are: ReservationID, MemberID and ResourceID and the candidate keys are: ReservationID, MemberID and ResourceID.
* The Functional Dependency is: ReservationID, MemberID, ResourceID 🡪 dateTimeReserved, dateTimeDue
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Conclusion:**

|  |  |
| --- | --- |
| **Relation** | **Normalisation Form** |
| Member | Boyce-Codd Normal Form |
| Student | Boyce-Codd Normal Form |
| Staff | Boyce-Codd Normal Form |
| Acquisition | Boyce-Codd Normal Form |
| Course | Boyce-Codd Normal Form |
| Privilege | Boyce-Codd Normal Form |
| Resource | Boyce-Codd Normal Form |
| Movable | Boyce-Codd Normal Form |
| Immovable | Boyce-Codd Normal Form |
| Category | Boyce-Codd Normal Form |
| Location | Boyce-Codd Normal Form |
| Loan | Boyce-Codd Normal Form |
| Reservation | Boyce-Codd Normal Form |

**Examples of Relations not in BCNF converted to BCNF:**

* For the examples to come, they have all been manipulated for explanation purposes and only serves as a demonstration on how to normalise from lower forms to BCNF. **These are examples of the relations above if they weren’t already in BCNF.**

**Example 1:**

**This is the relation that will be normalised to BCNF**

Course(offeringID, courseID, name, semesterOffered, yearOffered, dateBegun, dateCourseEnd, privilegeID, name, description, maxResourceBorrow)

2nd Normal Form Check:

* the key attributes are: OfferingID and PrivilegeID and the candidate keys are: OfferingID and PrivilegeID.
* The Functional Dependency is: OfferingID 🡪 CourseID, Name, semesterOffered, yearOffered, dateBegun, dateCourseEnd and also

PrivilegeID 🡪 name, description, maxResourceBorrow

* The left side is not a proper subset of the candidate keys
* Thus, not in 2nd normal form.

Therefor this relation is in unnormalized form.

Converting to BCNF:

1. To convert we first need to understand that there are two different Functional Dependencies:
2. OfferingID 🡪 CourseID, Name, semesterOffered, yearOffered, dateBegun, dateCourseEnd
3. And
4. PrivilegeID, OfferingID 🡪 name, description, maxResourceBorrow
5. Having OfferingID referenced as a determinant of the second Functional Dependencies and the first indicates that it should be converted into a primary key for a new relation and thus referenced to privilege as a foreign key. For example,
6. We can now have Course(OfferingID, CourseID, Name, semesterOffered, yearOffered, dateBegun, dateCourseEnd) and Privilege(PrivilegeID, OfferingID, name, description, maxResourceBorrow)
7. Now for both relations we can run a normalisation test:

2nd Normal Form Check (For Course(…)):

* the key attribute is: OfferingID and the candidate key is: OfferingID.
* The Functional Dependency is: OfferingID 🡪 CourseID, Name, semsterOffered, yearOffered, dateBegun, dateCourseEnd
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

2nd Normal Form Check (For Privilege(…)):

* the key attributes are: PrivilegeID and OfferingID and the candidate keys are: PrivilegeID and OfferingID.
* The Functional Dependency is: PrivilegeID, OfferingID 🡪 name, description, maxResourceBorrow
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**Example 2:**

**This is the relation that will be normalised to BCNF**

Category(CategoryID, name, description, maxBorrowTime, ResourceID, description, status)

2nd Normal Form Check:

* the key attributes are: CategoryID and ResourceID and the candidate keys are: CategoryID and ResourceID.
* The Functional Dependency is: CategoryID, ResourceID 🡪 name, description, maxBorrowTime and also

ResourceID 🡪 description, status

* The left side is not a proper subset of the candidate keys
* Thus, not in 2nd normal form.

Therefor this relation is in unnormalized form.

Converting to BCNF:

1. To convert we first need to understand that there are two different Functional Dependencies:
2. ResourceID 🡪 description, status
3. And
4. CategoryID, ResourceID 🡪 name, description, maxBorrowTime
5. Having ResourceID referenced as a determinant of the second Functional Dependencies and the first indicates that it should be converted into a primary key for a new relation and thus referenced to privilege as a foreign key. For example,
6. We can now have Resource(ResourceID, description, status) and Category(CategoryID, ResourceID, name, description, maxBorrowTime)
7. Now for both relations we can run a normalisation test:

2nd Normal Form Check (For Resource(…)):

* the key attribute is: ResourceID and the candidate key is: ResourceID.
* The Functional Dependency is: ResourceID 🡪 description, status
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

2nd Normal Form Check (For Category (…)):

* the key attributes are: CategoryID and ResourceID and the candidate keys are: CategoryID and ResourceID.
* The Functional Dependency is: CategoryID, ResourceID 🡪 name, description, maxBorrowTime
* The left side is a subset of the candidate keys and the right side are not all key attributes
* Thus, in 2nd normal form at least

3rd Normal Form Check

* The left side is a super key in the Functional Dependency above
* Thus, in 3rd normal form at least

BCNF Check

* The left side is a super key in the Functional Dependency above
* Thus, in BCNF

**SQL Database:**

drop table reservation

drop table loan

drop table location

drop table movable

drop table immovable

drop table category

drop table privilege

drop table course

drop table staff

drop table student

drop table acquisition

drop table resource

drop table member

create table member(

MemberID Char(8),

name varchar(50) not null,

address varchar(50) not null,

phone bigint,

email varchar(50) not null,

status varchar(8) default 'Active',

comments varchar(100),

primary key(MemberID),

check(phone > 0))

create table resource(

ResourceID char(8),

description varchar(50),

status varchar(11) default 'Available',

primary key(ResourceID))

create table acquisition(

AcquisitionID char(8),

MemberID char(8) not null,

name varchar(50) not null,

make varchar(50),

manufacturer varchar(50),

model varchar(50),

year int,

description varchar(100),

urgency varchar(10) default 'non-urgent',

Status varchar(8) not null,

FundCode char(8) not null,

VendorCode char(8) not null,

Price float not null,

Notes varchar(100),

primary key(AcquisitionID),

foreign key(MemberID) references member on update cascade on delete no action,

check(Price > 0),

check(year between 1000 and 2020))

create table student(

StudentID char(8),

MemberID char(8) not null,

Points tinyint default 12,

Status varchar(8) default 'Enabled',

primary key(StudentID),

foreign key(MemberID) references member on update cascade on delete no action,

check(Points between 0 and 12))

create table staff(

StaffID char(8),

MemberID char(8) not null,

Status varchar(8) default 'Enabled',

primary key(StaffID),

foreign key(MemberID) references member on update cascade on delete no action)

create table course(

OfferingID char(8),

StudentID char(8) not null,

courseID char(8) not null,

name varchar(50) not null,

semesterOffered tinyint not null,

yearOffered int not null,

dateBegun date not null,

dateCourseEnd date not null,

primary key(OfferingID),

foreign key(StudentID) references student on update cascade on delete no action,

check(semesterOffered between 1 and 2),

check(yearOffered between 1000 and 2020))

create table privilege(

PrivilegeID char(8),

OfferingID char(8) not null,

name varchar(50) not null,

description varchar(100),

category varchar(50) not null,

maxResourceBorrow tinyint not null,

primary key(PrivilegeID),

foreign key(OfferingID) references course,

check(maxResourceBorrow > 0))

create table category(

CategoryID char(8),

PrivilegeID char(8) not null,

ResourceID char(8) not null,

name varchar(50) not null,

description varchar(100),

MaxBorrowTime datetime not null,

primary key(CategoryID),

foreign key(PrivilegeID) references privilege on update cascade on delete no action,

foreign key(ResourceID) references resource on update cascade on delete no action)

create table immovable(

ImmovableID char(8),

ResourceID char(8) not null,

Capacity smallint not null,

primary key(ImmovableID),

foreign key(ResourceID) references resource on update cascade on delete cascade,

check(Capacity > 0))

create table movable(

MovableID char(8),

ResourceID char(8) not null,

name varchar(50) not null,

make varchar(50),

manufacturer varchar(50),

model varchar(50),

year int,

assetValue float not null,

primary key(MovableID),

foreign key(ResourceID) references resource on update cascade on delete cascade,

check(assetValue > 0),

check(year between 1000 and 2020))

create table location(

LocationID char(8),

ResourceID char(8) not null,

Room varchar(50) not null,

Building varchar(50) not null,

Campus varchar(50) not null,

primary key(LocationID),

foreign key(ResourceID) references resource on update cascade on delete no action)

create table loan(

LoanID char(8),

MemberID char(8) not null,

MovableID char(8) not null,

dateTimeBorrowed datetime not null,

dateTimeReturned datetime not null,

dateTimeDue datetime not null,

primary key(LoanID),

foreign key(MemberID) references member on update cascade on delete no action,

foreign key(MovableID) references movable on update cascade on delete no action)

create table reservation(

ReservationID char(8),

MemberID char(8) not null,

ResourceID char(8) not null,

dateTimeReserved datetime not null,

dateTimeDue datetime not null,

primary key(ReservationID),

foreign key(MemberID) references member on update cascade on delete no action,

foreign key(ResourceID) references resource on update cascade on delete no action)

insert into member(MemberID, name, address, phone, email, comments) values('M1234567', 'Cory', '34 East street', '1438765567', 'Cory@hotmail.com', 'I am Cory')

insert into member(MemberID, name, address, phone, email) values('M7654321', 'Matt', '12 Hello street', '9876543214', 'Matt@hotmail.com')

insert into member(MemberID, name, address, phone, email, comments) values('M1357911', 'Lukas', '76 Bright avenue', '65714631', 'Lukas@gmail.com', 'I am a member of SCS')

insert into member(MemberID, name, address, phone, email) values('M9999999', 'Mark', '14 West street', '5754678367', 'Mark@hotmail.com')

insert into member(MemberID, name, address, phone, email) values('M9888888', 'John', '10 GoodBye steet', '1800101010', 'John@hotmail.com')

insert into member(MemberID, name, address, phone, email) values('M9777777', 'Mic', '99 port close', '885935672', 'Mic@gmail.com')

insert into resource(ResourceID, status) values('R9876543', 'In Use')

insert into resource(ResourceID, description) values('R1313131', '10k resistor')

insert into resource(ResourceID, description, status) values('R1245789', 'bunsen burner used for chemistry', 'Lost')

insert into resource(ResourceID, description) values('R6543782', 'industrial oven')

insert into resource(ResourceID, description ,status) values('R6879348', 'computer', 'Maintenance')

insert into resource(ResourceID) values('R5789343')

insert into acquisition(AcquisitionID, MemberID, name, make, manufacturer, model, year, description, Status, FundCode, VendorCode, Price, Notes) values('A7657474', 'M7654321', 'jabulani soccer ball', '2010 nike', 'nike', 'jabulani', '2010', 'A soccer ball used in the 2010 world cup', 'Pending', 'FC121856', 'VC587463', '80', 'A replica version')

insert into acquisition(AcquisitionID, MemberID, name, make, manufacturer, year, urgency, Status, FundCode, VendorCode, Price) values('A5434331', 'M7654321', 'cricket gloves', '2012 adidas', 'adidas', '2012', 'Urgent', 'Acquired', 'FC000123', 'VC768549', '56.89')

insert into acquisition(AcquisitionID, MemberID, name, description, Status, FundCode, VendorCode, Price) values('A5438455', 'M9999999', '10 NPN transistors', 'any 10 pk of NPN transistors', 'Pending', 'FC567492', 'VC121256', '20.05')

insert into student(StudentID, MemberID) values('STU00012', 'M1234567')

insert into student(StudentID, MemberID, Points) values('STU00111', 'M7654321', '9')

insert into student(StudentID, MemberID, Points) values('STU00237', 'M1357911', '9')

insert into staff(StaffID, MemberID) values('STA13013', 'M9999999')

insert into staff(StaffID, MemberID) values('STA14014', 'M9888888')

insert into staff(StaffID, MemberID) values('STA15015', 'M9777777')

insert into course(OfferingID, StudentID, courseID, name, semesterOffered, yearOffered, dateBegun, dateCourseEnd) values('OFF12458', 'STU00012', 'COUR0001', 'Introduction to Computer Science', '1', '2020', '2020-01-14', '2020-05-14')

insert into course(OfferingID, StudentID, courseID, name, semesterOffered, yearOffered, dateBegun, dateCourseEnd) values('OFF75835', 'STU00111', 'COUR0002', 'Introduction to Electrical Engineering', '1', '2020', '2020-02-12', '2020-06-12')

insert into course(OfferingID, StudentID, courseID, name, semesterOffered, yearOffered, dateBegun, dateCourseEnd) values('OFF58705', 'STU00237', 'COUR0003', 'Introduction to Physics', '2', '2020', '2020-06-10', '2020-10-10')

insert into privilege(PrivilegeID, OfferingID, name, description, category, maxResourceBorrow) values('PR000012', 'OFF12458', 'Science storage', 'access to science storage', 'science', '5')

insert into privilege(PrivilegeID, OfferingID, name, description, category, maxResourceBorrow) values('PR000013', 'OFF75835', 'engineering storage', 'access to engineering storage', 'engineering', '10')

insert into privilege(PrivilegeID, OfferingID, name, description, category, maxResourceBorrow) values('PR000014', 'OFF58705', 'physics storage', 'access to physics storage', 'physics', '2')

insert into category(CategoryID, PrivilegeID, ResourceID, name, description, MaxBorrowTime) values('CAT56789', 'PR000012', 'R9876543', 'speaker', 'speaker for music', '2020-12-30 11:59:59')

insert into category(CategoryID, PrivilegeID, ResourceID, name, MaxBorrowTime) values('CAT55555', 'PR000013', 'R5789343', 'camera', '2020-12-30 11:59:59')

insert into category(CategoryID, PrivilegeID, ResourceID, name, MaxBorrowTime) values('CAT57495', 'PR000014', 'R1245789', 'physics category', '2020-12-30 11:59:59')

insert into immovable(ImmovableID, ResourceID, Capacity) values('IM564784', 'R6543782', '2')

insert into immovable(ImmovableID, ResourceID, Capacity) values('IM754835', 'R6879348', '1')

insert into immovable(ImmovableID, ResourceID, Capacity) values('IM478239', 'R5789343', '5')

insert into movable(MovableID, ResourceID, name, manufacturer, model, year, assetValue) values('MO654378', 'R5789343', 'HD camera', 'HD', 'pinpoint', '2020', '150.67')

insert into movable(MovableID, ResourceID, name, assetValue) values('MO467285', 'R1313131', '10k resistor', '0.69')

insert into movable(MovableID, ResourceID, name, make, manufacturer, year, assetValue) values('MO463526', 'R1245789', 'Bunsen Burner', 'flamer', 'flameCo', '2016', '45.12')

insert into location(LocationID, ResourceID, Room, Building, Campus) values('LOC74657', 'R9876543', 'ES105', 'ES', 'Winchester')

insert into location(LocationID, ResourceID, Room, Building, Campus) values('LOC42653', 'R1313131', 'AH684', 'AH', 'Lincoln')

insert into location(LocationID, ResourceID, Room, Building, Campus) values('LOC36847', 'R1245789', 'HU483', 'HU', 'Winchester')

insert into location(LocationID, ResourceID, Room, Building, Campus) values('LOC47386', 'R6543782', 'HU387', 'HU', 'Winchester')

insert into location(LocationID, ResourceID, Room, Building, Campus) values('LOC78966', 'R6879348', 'AH789', 'AH', 'Lincoln')

insert into location(LocationID, ResourceID, Room, Building, Campus) values('LOC96785', 'R5789343', 'ES107', 'ES', 'Winchester')

insert into loan(LoanID, MemberID, MovableID, dateTimeBorrowed, dateTimeReturned, dateTimeDue) values('LOAN5673', 'M1234567', 'MO654378', '2020-06-10 05:45:56', '2020-07-10 09:55:56', '2020-08-10 05:45:56')

insert into loan(LoanID, MemberID, MovableID, dateTimeBorrowed, dateTimeReturned, dateTimeDue) values('LOAN4565', 'M9999999', 'MO467285', '2020-04-13 08:15:43', '2020-06-14 07:34:51', '2020-09-13 08:15:43')

insert into loan(LoanID, MemberID, MovableID, dateTimeBorrowed, dateTimeReturned, dateTimeDue) values('LOAN5473', 'M1357911', 'MO463526', '2020-11-01 09:32:12', '2020-11-03 05:58:12', '2020-11-15 09:32:12')

insert into loan(LoanID, MemberID, MovableID, dateTimeBorrowed, dateTimeReturned, dateTimeDue) values('LOAN5474', 'M1357911', 'MO463526', '2020-11-14 09:32:12', '2020-11-16 05:58:12', '2020-11-18 09:32:12')

insert into loan(LoanID, MemberID, MovableID, dateTimeBorrowed, dateTimeReturned, dateTimeDue) values('LOAN5475', 'M1357911', 'MO463526', '2020-11-20 09:32:12', '2020-11-23 05:58:12', '2020-11-27 09:32:12')

insert into reservation(ReservationID, MemberID, ResourceID, dateTimeReserved, dateTimeDue) values('RES35487', 'M9999999', 'R5789343', '2019-04-14 10:34:56', '2019-04-18 10:34:56')

insert into reservation(ReservationID, MemberID, ResourceID, dateTimeReserved, dateTimeDue) values('RES43827', 'M9999999', 'R5789343', '2019-05-13 09:39:32', '2019-05-15 09:39:32')

insert into reservation(ReservationID, MemberID, ResourceID, dateTimeReserved, dateTimeDue) values('RES23786', 'M9999999', 'R6879348', '2020-08-28 06:45:19', '2020-09-01 06:45:19')

insert into reservation(ReservationID, MemberID, ResourceID, dateTimeReserved, dateTimeDue) values('RES75654', 'M9999999', 'R6879348', '2020-05-01 06:45:19', '2020-09-01 06:45:19')

insert into reservation(ReservationID, MemberID, ResourceID, dateTimeReserved, dateTimeDue) values('RES56464', 'M9999999', 'R6879348', '2020-06-05 06:45:19', '2020-09-01 06:45:19')

insert into reservation(ReservationID, MemberID, ResourceID, dateTimeReserved, dateTimeDue) values('RES64564', 'M9999999', 'R6879348', '2020-09-19 06:45:19', '2020-09-25 06:45:19')

insert into reservation(ReservationID, MemberID, ResourceID, dateTimeReserved, dateTimeDue) values('RES64565', 'M9999999', 'R6879348', '2020-09-19 06:45:19', '2020-09-25 06:45:19')

**SQL Transactions:**

--Q1

select m.name Student\_who\_enrolled\_in\_COUR0001 from course c join student s

on c.StudentID = s.StudentID join member m on s.MemberID = m.MemberID

where courseID like 'COUR0001'

--Q2

select m.name, p.maxResourceBorrow maximum\_number\_of\_speakers\_can\_borrow from category ca join privilege p

on ca.PrivilegeID = p.PrivilegeID join course c on p.OfferingID = c.OfferingID

join student s on c.StudentID = s.StudentID join member m on s.MemberID = m.MemberID

where courseID like 'COUR0001'

--Q3

select m.name, m.phone, count(a.MemberID) as number\_of\_acquisition\_requests, count(r.MemberID) as number\_of\_reservations\_in\_2019 from reservation r join member m

on r.MemberID = m.MemberID join acquisition a on m.MemberID = a.MemberID

join staff s on m.MemberID = s.MemberID

where s.StaffID like 'STA13013'

group by m.name, m.phone

--Q4

select m.name name\_of\_student\_who\_has\_borrowed\_camera from loan l join member m

on l.MemberID = m.MemberID join movable mo on l.MovableID = mo.MovableID

join category c on mo.ResourceID = c.ResourceID join student st on m.MemberID = st.MemberID

where c.name like 'camera'

group by m.name, mo.model, mo.year

having mo.model like 'pinpoint' and mo.year like '2020'

--Q5

select r.ResourceID, count(r.description) as name\_of\_resource\_most\_loaned\_in\_this\_month from loan l join movable mo

on l.MovableID = mo.MovableID join resource r on mo.ResourceID = r.ResourceID

where month(getdate()) = month(l.dateTimeBorrowed)

group by r.ResourceID, r.description

order by name\_of\_resource\_most\_loaned\_in\_this\_month desc

--Q6

select convert(varchar(10), r.dateTimeReserved, 111) as date, loc.Room, count(r.dateTimeReserved) as amount\_of\_reservations\_on\_date from reservation r join location loc

on r.ResourceID = loc.ResourceID

where convert(varchar(10), r.dateTimeReserved, 111) like '2020/06/05' or convert(varchar(10), r.dateTimeReserved, 111) like '2020/05/01' or convert(varchar(10), r.dateTimeReserved, 111) like '2020/09/19'

group by r.dateTimeReserved, loc.Room