Assignment 3

Project: SDS Resource Management Database Design

Physical Database Design

Table of Contents

Reflection on Assignment Two

Marker's Comments Summarised

Unnecessary relations in mapping stage.

Reflection and Changes

Removed unecessary relations from mapping stage.

Changed phoneNo attribute from multivaried to not as I decided multiple weren't needed.
Removed the table created from the multivaried attribute.
Added phoneNo attribute back to Member. Also changed phoneNo data type from varchar to int.

Increased schoolPosition attribute from varchar(10) to varchar(20) so I could input longer titles.

Couldn't use Boolean, so used BIT data type instead for attributes isAdmin and isCancelled.

Added courseID to CourseOffering as it was needed for a query question.
This caused a transitive dependency in CourseOffering so I normalised it to BCNF.

Changed some PK IDs from data type char to int identity(1,1) as a unique auto incrementing number was more convenient (OfferingID, loanID, reservationID, categoryCode, privilegeID, acquisitionID).

Changed maxBorrowTime to maxBorrowHours as most things are borrowed in hours, and if days are needed, I can divide by 24. Also changed the data type form varchar to int.

Changed semesterOffered from multivaried to not (was accidentally set to yes in data dictionary).

Changed Privileges and CourseOffering_Privileges to non plural.

Requirement Specifications

Data Requirements

Catalogue

The School of Data Science at the University of Northwest Technology maintains information about the physical resources it makes accessible to members of the school. This is done through recording data about individual resources and assigning them to a category for easy searchability.

Resource

This will include all Movable and Immovable resources that are able to be borrowed or reserved. Each resource will have a unique resource ID, a description, and a present status (Available, Borrowed, Reserved, Lost, or Maintenance). There are two types/subclasses of resource: Movable and Immovable.

Movable: Maintains information on movable resources such as the name, manufacturer, model, year made, resource asset value, and the building BDS (to identify where the item is stored). Movable resources include items like cameras, microphones, etc. This type of resource can be reserved by members for future use, or immediately loaned to a member if available.

Immovable: Maintains information on immovable resources such as their capacity (how many people fit inside), room name, building name/location, and campus name/location. Immovable resources are reservable for use and include classrooms, studios, labrooms, etc. This type of resource can be reserved for a specific amount of time, allowing members to use rooms for meetings, lectures, activities or whatever suits their needs.

Category

Every resource belongs to a category. Category information includes a unique code to identify the category, name, description, maximum time (in days and hours) allowed to borrow/reserve a resource from the category, what type of resource is held within the category, and a set of keywords that relate to the category (to improve searchability).

Member

This includes SDS Staff and Students (who have enrolled in a course offering) who have borrowing and reservation rights to the school's resources. Members have a unique ID, name, address, phone numbers (can include alternate number i.e. home phone), email, status (disabled or active), and a comment. There are two types/subclasses of members: Staff and Student (enrolled).

Student: This will contain a student's points. If a student's point count reaches 0, they will have restricted borrowing/reserving privileges. Students can borrow and reserve resources based on the privileges granted by the course offerings they have enrolled in. Students begin with 12 points, and they can lose points in two ways:

1. They don't pickup or cancel a reserved item by the pickup date, it will be automatically cancelled resulting in a single point loss.
2. For each day a resource is overdue, a student will lose 3 points

Staff: This will have a staff members school position, and a flag if they are administrator. It will also contain information relating to any previous employment. Staff do not have a limit on the number of resources they can reserve or borrow at any given time. Staff with the administrator privileges can accept/deny acquisitions, and also modify a student's points if necessary.

Course Offering

These are courses offered by SDS for students to enroll in. Each course offering has a unique offering id, name, semester offered (can be both 1 and 2), year offered, date the course begins, and date the course ends. Students will gain borrowing/reservation privileges based on these courses. The status/privileges of students will be 'active' as long as the current date has not passed the end date of a course they are enrolled in.

Privilege

These are privileges assigned to course offerings and will be granted to students that enroll in certain courses. They allow students to access different resources. Information carried includes a unique privilege ID, privilege name, description of the privilege, and the maximum number of resources that can be borrowed or reserved by from a category at any given time.

Loan

These are loans members take out to borrow movable resources/items. They contain a unique loan ID, and the date and time the loan is made, returned, and due. Loan also references resource ID (from Resource), and member ID (from Member) to maintain information on what resource has been borrowed and by which member.

Reservation

These are reservations members make for the future use of a resource, either movable or immovable. Once a reservation is made the resource will be booked for pickup or use on a requested date and time. Reservation information maintained includes a unique reservation ID, a true/false flag to determine if the reservation is cancelled, date and time the resource is to be picked up/used and the date and time the resource is to be returned/unused. Reservation also references resource ID (from Resource), and member ID (from Member) to maintain information on what resource has been reserved and by which member. To further note, when a movable item has been picked up, it should create a Loan.

Acquisition

These are requests to SDS by members for a new resource needed. Along with a unique acquisition ID, the acquisition information maintained includes the requested items name, manufacturer, model, year made, description, and how urgently the item is needed. Acquisition also references the member ID (from Member) to maintain which member made the request. These requests will be sorted by staff with administrator rights.

Business Rules

Expiration of student member access

Students borrowing privileges are automatically revoked when the current date passes the end date of their course offerings. Student's status will then be set to disabled.

Maximum items borrowed or reserved at any one time

A member cannot borrow or reserve more than the maximum number of items specified in their privileges at any given time. Note, this does not apply to staff members, as there is no limit on the number of resources used by staff.

Penalty for late returns by students

Each student member has a default amount of points (they begin with 12). A penalty of 3 points is incurred for each day a resource is overdue. When the point count is reduced to 0, member status is disabled, prohibiting borrowing and reservation privileges. The administrator has rights to reset/change points.

Cancellation of reservations

A reserved item is automatically cancelled if it is not picked up a day after the required date. Non cancellation of reservation by a member (before automatic cancellation) results in the loss of 1 point. Administrator has the right to cancel any reservation.

Borrowing/reservation periods

The duration of the borrowing period (which could be days or hours) is determined by the category to which the item belongs. (E.g. Cameras have duration of 2 days). During this period, other members cannot borrow the same item. This also applies to reservations, where you can't borrow or reserve a resource for the same period another member has borrowed/reserved it. If a reservation is cancelled, that reservation period is opened up again.

Priority of borrowing new acquisitions

Members who request a new resource should have a period of time that allows them the first chance to borrow it.

Transaction Requirements

Data Manipulation Operations

- Insert/update/delete an existing loan
- Insert/update/delete a resource
- Insert/update/delete members
- Insert/update/delete a resources present status
- Insert/update/delete reservations
- Insert/update/delete acquisition requests
- Insert/update/delete member comments
- Insert/update/delete movable items location (buildingBDS)
- Insert/update/delete student points
- Insert/update/delete category description
- Insert/update/delete a course offerings semester available

Queries

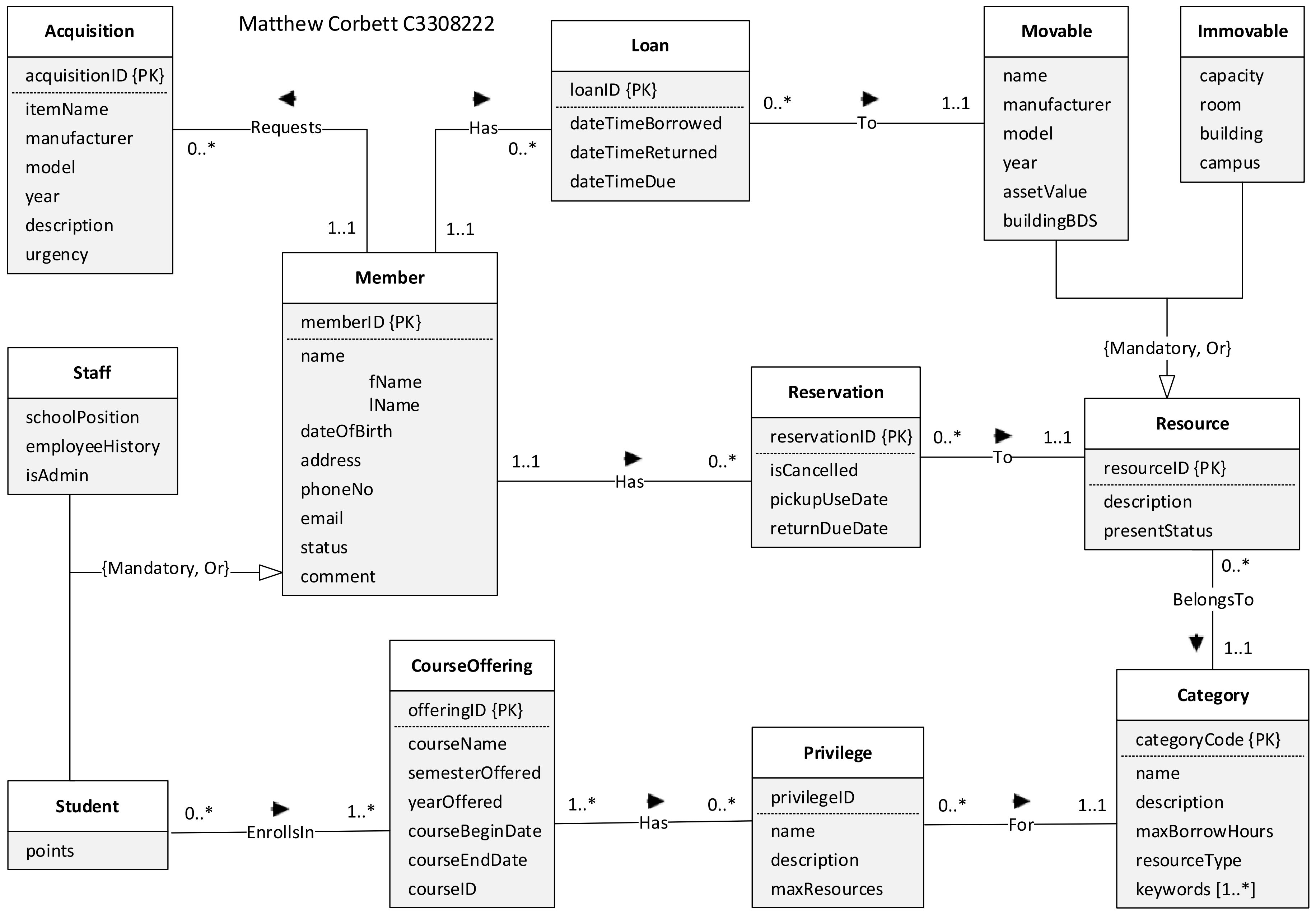
- List present loans taken by a specific member.
- Find the student with a late returned item number.
- Display a student’s current points.
- List items that are often loaned in a semester.
- List all the reservations for a particular item.
- Search for a resource based on model, category keywords, manufacturer, etc.
- Report of points earned/lost for a particular student during a certain period.
- List all privileges a student has access to.
- List the available dates a resource can be reserved/borrowed.
- Provide information on the status of an acquisition.
- Search a loaded item based on an employee number, on a specific date.
- List the names of staff with admin rights.

EER Model with Data Dictionary

EER Model

~View Next Page

Matthew Corbett C3308222



Data Dictionary

Entities

Entity Name	Description	Aliases	Occurrence
Resource	Resources members can borrow/reserve from SDS		All resources SDS makes accessible to members
Movable	Resources that can be borrowed e.g. Camera, microphone, etc.	Item	All resources that can be moved/taken
Immovable	Resources that can be booked for use e.g. classroom, studios, etc.	Room	Rooms that can be reserved/booked
Category	Each resource belongs to a particular category		When a resource has a category it belongs to
Member	People who can borrow/reserve resources	Borrower/User	When a new user joins
Student	Member who enrolls in a course offering		
Staff	Member working in the school		
Course Offering	Courses offered by SDS that students can enroll in		All courses offered by SDS
Privilege	Privileges assigned to course offerings that allow students access to different resources	Privilege Granted	When a course offering/student is granted a privilege
Loan	Member makes a loan to borrow a resource	Borrowing	When a member takes out a loan
Reservation	Reservation member makes to borrow or use a resource	Reserving	When a member makes a reservation
Acquisition	Acquisition to SDS by members for a new resource needed	Request	When a member requests an acquisition

Relationships

Entity Name	Multiplicity	Relationship	Multiplicity	Entity Name
Resource	0..*	BelongsTo	1..1	Category
	(Man,Or)	Generalisation	(Man,Or)	Moveable
	(Man,Or)	Generalisation	(Man,Or)	Immovable
Member	1..1	Has	0..*	Loan
	1..1	Has	0..*	Reservation
	1..1	Requests	0..*	Acquisition
	(Man,Or)	Generalisation	(Man,Or)	Student
	(Man,Or)	Generalisation	(Man,Or)	Staff
Student	0..*	EnrollsIn	1..*	CourseOffering
CourseOffering	1..*	Has	0..*	Privilege
Loan	0..*	To	1..1	Moveable
Reservation	0..*	To	1..1	Resource
Privilege	0..*	For	1..1	Category

Attributes

Entity	Attributes	Description	Data type & Length	Nulls	Multi-Varied	Derived	Default
Resource	resourceID	Uniquely identifies a resource	char(10)	N	N	N	
	description	Description of resource	varchar(50)	N	N	N	
	presentStatus	Current status of resource, either 'Available', 'Lost' 'Borrowed, 'Reserved, or 'Maintenance'.	varchar(15)	N	N	N	'Available'
Movable	name	Name of resource	varchar(20)	N	N	N	
	manufacturer	Manufacturer of resource	varchar(20)	N	N	N	
	model	Model of resource	varchar(15)	N	N	N	
	year	Year resource was made	char(4)	N	N	N	
	assetValue	Value of resource	decimal(6)	N	N	N	
	buildngBDS	Building in which the item is stored	varchar(10)	N	N	N	
Immovable	capacity	How many people can fit in the room	smallint(3)	N	N	N	
	room	Name of room	varchar(10)	N	N	N	
	building	Location of building or building name	varchar(10)	N	N	N	
	campus	Location of campus or campus name	varchar(20)	N	N	N	
Category	categoryCode	Uniquely identifies the category	int identity(1,1)	N	N	N	
	name	Name of category	varchar(20)	N	N	N	
	description	Description of category	varchar(50)	N	N	N	
	maxBorrowHours	Maximum hours allowed to borrow or reserve a resource	int	N	N	N	
	resourceType	Type of resource that belongs to the category	varchar(15)	N	N	N	
	keywords	Keywords that relate to the category	varchar(15)	N	Y	N	
Member	memberID	An id to uniquely identify each member	char(10)	N	N	N	
	name fName lName	 First name of member Last name of member	 varchar(20) varchar(20)	 N N	 N N	 N N	
	dateOfBirth	Date the member was born	date	N	N	N	
	address	The address of the member	varchar(50)	Y	N	N	

	phoneNo	The phone numbers of the member	char(8)	Y	Y	N	
	email	The email address of the member	varchar(50)	Y	N	N	
	status	The status of the member, either ‘active’ or ‘inactive’	varchar(10)	N	N	N	‘Active’
	comment	A comment on the member	varchar(50)	Y	N	N	
Student	points	Points that allow students to reserve or borrow resources	smallint(2)	N	N	N	12
Staff	schoolPosition	Job title of staff member	varchar(20)	N	N	N	
	employeeHistory	History of employee	varchar(50)	Y	N	N	
	isAdmin	Defines whether a staff has admin rights. 1 (True) or 0 (False).	bit	N	N	N	‘0’
Course Offering	offeringID	Uniquely identifies the course offering	int identity(1,1)	N	N	N	
	courseName	Name of Course	varchar(50)	N	N	N	
	semesterOffered	Semester the course is offered	char(1)	N	N	N	
	yearOffered	Year the course is offered	char(4)	N	N	N	
	courseBeginDate	Date the course begins	date	N	N	N	
	courseEndDate	Date the course ends	date	N	N	N	
Privilege	privilegeID	Uniquely identifies the privilege granted	int identity(1,1)	N	N	N	
	name	Name of privileges	varchar(20)	N	N	N	
	description	Description of privileges	varchar(50)	N	N	N	
	maxResources	Max number of resources that the privilege allows you to borrow at any one time	smallint(2)	N	N	N	
Loan	loanID	Uniquely identifies a loan	int identity(1,1)	N	N	N	
	dateTimeBorrowed	Date & time the loan is made	datetime	Y	N	N	
	dateTimeReturned	Date & time the loan is returned	datetime	Y	N	N	
	dateTimeDue	Date and time the loan is due	datetime	Y	N	N	
Reservation	reservationID	Uniquely identifies a reservation	int identity(1,1)	N	N	N	
	isCancelled	Defines whether a reservation is cancelled. 1 (True) or 0 (False).	bit	N	N	N	‘0’

	pickupUseDate	Date and time in which the resource is to be picked up and or used	datetime	Y	N	N	
	returnDueDate	Date and time in which the resource is to be returned and or unused	datetime	Y	N	N	
Acquisition	acquisitionID	Uniquely identifies an acquisition	int identity(1,1)	N	N	N	
	itemName	Name of item requested	varchar(20)	N	N	N	
	manufacturer	Manufacturer of item	varchar(20)	N	N	N	
	model	Model of item	varchar(15)	N	N	N	
	year	Year item was made	char(4)	N	N	N	
	description	Description of item	varchar(50)	N	N	N	
	urgency	How urgently the item is required	varchar(20)	N	N	N	

Relational Model (mapped from EER)

Note: {Mandatory,Or} have been replaced with {Optional,Or}.

Member (memberID, fName, lName, dateOfBirth, address, phoneNo, email, status, comment)
Primary Key memberID

Staff (memberID, schoolPosition, employeeHistory, isAdmin)
Primary Key memberID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

Student (memberID, points)
Primary Key memberID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

CourseOffering (offeringID, courseName, semesterOffered, yearOffered, courseBeginDate, courseEndDate, courseID)
Primary Key offeringID

Student_CourseOffering (memberID, offeringID)
Primary Key memberID, offeringID
Foreign Key memberID **references** Student(memberID) on update cascade, on delete cascade
Foreign Key offeringID **references** CourseOffering(OfferingID) on update cascade, on delete cascade

Privilege (privilegeID, name, description, maxResources, categoryCode)
Primary Key privilegeID
Foreign Key categoryCode **references** Category(categoryCode) on update cascade, on delete no action

CourseOffering_Privilege (offeringID, privilegeID)
Primary Key offeringID, privilegeID
Foreign Key offeringID **references** CouseOffering(offeringID) on update cascade, on delete cascade
Foreign Key privilegeID **references** Privilege(privilegeID) on update cascade, on delete cascade

Category (categoryCode, name, description, maxBorrowTime, resourceType)
Primary Key categoryCode

Resource (resourceID, description, presentStatus, categoryCode)
Primary Key resourceID
Foreign Key categoryCode **references** Category(categoryCode) on update cascade, on delete no action

Movable (resourceID, name, manufacturer, model, year, assetValue, buildingBDS)
Primary Key resourceID
Foreign Key resourceID **references** resource(resourceID) on update cascade, on delete no action

Immovable (resourceID, capacity, room, building, campus)
Primary Key resourceID
Foreign Key resourceID **references** resource(resourceID) on update cascade, on delete no action

Loan (loanID, dateTimeBorrowed, dateTimeReturned, dateTimeDue, memberID, resourceID)
Primary Key loanID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action
Foreign Key resourceID **references** Movable(resourceID) on update cascade, on delete no action

Reservation (reservationID, isCancelled, pickupUseDate, returnDueDate, memberID, resourceID)
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

Acquisition (acquisitionID, itemName, manufacturer, model, year, description, urgency, memberID)
Primary Key acquisitionID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

Normalised Relational Schema

Note: For cases of full functional dependency (where determinant is a single attribute) I just state that there are no partial dependencies when checking 2NF.

Member (BCNF)

Member (memberID, fName, lName, dateOfBirth, address, phoneNo, email, status, comment)
Primary Key memberID

FD1: memberID → fName, lName, dateOfBirth, address, phoneNo email, status, comment (PK)

Member relation is well formed and in 1NF.
There are no partial dependencies, therefore member relation is in 2NF.
There are no transitive dependencies, therefore member relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

Staff (BCNF)

Staff (memberID, schoolPosition, employeeHistory, isAdmin)
Primary Key memberID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

FD1: memberID → schoolPosition, employeeHistory, isAdmin (PK)

Staff relation is well formed and in 1NF.
There are no partial dependencies, therefore staff relation is in 2NF.
There are no transitive dependencies, therefore staff relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

Student (BCNF)

Student (memberID, points)
Primary Key memberID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

FD1: memberID → points (PK)

Student relation is well formed and in 1NF.
There are no partial dependencies, therefore student relation is in 2NF.
There are no transitive dependencies, therefore student relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

CourseOffering (BCNF)

R1: **CourseOffering** (offeringID, courseName, semesterOffered, yearOffered, courseBeginDate, courseEndDate, courseID)
Primary Key offeringID

FD1: offeringID → courseName, semesterOffered, yearOffered, courseBeginDate, courseEndDate, courseID
FD2: courseID → courseName (Transitive Dependency)

CourseOffering relation is well formed and in 1NF.
There are no partial dependencies, therefore CourseOffering relation is in 2NF.
There is a transitive dependency (FD2), therefore Acquisition relation is not in 3NF

Normalisation process (decomposing R1 based on FD2)

FD2 can be used to create new relation R2: Course
R2:XY: **Course** (courseID, courseName)
Primary Key courseID

Then remove dependent attribute 'courseName' from R1, and add FK courseID from R2
R3:R1-Y: **CourseOffering** (offeringID, semesterOffered, yearOffered, courseBeginDate, courseEndDate, courseID)
Primary Key offeringID
Foreign Key courseID **references** Course(courseID) on update cascade, on delete no action

Judging normal form of decomposed relations

R2:FD1: courseID → courseName (PK)

R3:FD1: offeringID → semesterOffered, yearOffered, courseBeginDate, courseEndDate, courseID (PK)

R2 and R3 are well formed and in 1NF
There are no partial dependencies in either relation, therefore R2 and R3 are in 2NF
There are no transitive dependencies in either relation, therefore R2 and R3 are in 3NF
The only functional dependency in either relation has a primary key as the determinant.
Therefore the new relations R2 and R3 are in BCNF

Student_CourseOffering (BCNF)

Student_CourseOffering (memberID, offeringID)
Primary Key memberID, offeringID
Foreign Key memberID **references** Student(memberID) on update cascade, on delete cascade
Foreign Key offeringID **references** CourseOffering(OfferingID) on update cascade, on delete cascade

FD1: memberID, offeringID → (PK)

Student_CourseOffering relation is well formed and in 1NF.
There are no non-prime attributes, therefore the relation is in 3NF and BCNF

Privilege (BCNF)

Privilege (privilegeID, name, description, maxResources, categoryCode)
Primary Key privilegeID
Foreign Key categoryCode **references** Category(categoryCode) on update cascade, on delete no action

FD1: privilegeID → name, description, maxResources, categoryCode (PK)

Privilege relation is well formed and in 1NF.
There are no partial dependencies, therefore Privilege relation is in 2NF.
There are no transitive dependencies, therefore Privilege relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

CourseOffering_Privilege (BCNF)

CourseOffering_Privilege (offeringID, privilegeID)
Primary Key offeringID, privilegeID
Foreign Key offeringID **references** CouseOffering(offeringID) on update cascade, on delete cascade
Foreign Key privilegeID **references** Privilege(privilegeID) on update cascade, on delete cascade

FD1: offeringID, privilegeID → (PK)

CourseOffering_Privilege relation is well formed and in 1NF.
There are no non-prime attributes, therefore the relation is in 3NF and BCNF

Category (BCNF)

Category (categoryCode, name, description, maxBorrowTime, resourceType)
Primary Key categoryCode

FD1: categoryCode → name, description, maxBorrowTIme, resourceType (PK)

Category relation is well formed and in 1NF.
There are no partial dependencies, therefore Category relation is in 2NF.
There are no transitive dependencies, therefore Category relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

CategoryKeywords (BCNF)

CategoryKeywords (categoryCode, keywords)
Primary Key categoryCode, keywords
Foreign Key categoryCode **references** Category(categoryCode) on update cascade on delete cascade

FD1: categoryCode, keywords → (PK)

CategoryKeywords relation is well formed and in 1NF.
There are no non-prime attributes, therefore the relation is in 3NF and BCNF

Resource (BCNF)

Resource (resourceID, description, presentStatus, categoryCode)
Primary Key resourceID
Foreign Key categoryCode **references** Category(categoryCode) on update cascade, on delete no action

FD1: resourceID → description, presentStatus, categoryCode (PK)

Resource relation is well formed and in 1NF.
There are no partial dependencies, therefore Resource relation is in 2NF.
There are no transitive dependencies, therefore Resource relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

Movable (BCNF)

Movable (resourceID, name, manufacturer, model, year, assetValue, buildingBDS)
Primary Key resourceID
Foreign Key resourceID **references** resource(resourceID) on update cascade, on delete no action

FD1: resourceID → name, manufacturer, model, year, assetValue, buildingBDS (PK)

Movable relation is well formed and in 1NF.
There are no partial dependencies, therefore Movable relation is in 2NF.
There are no transitive dependencies, therefore Movable relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

Immovable (BCNF)

R1: **Immovable** (resourceID, capacity, room, building, campus)
Primary Key resourceID
Foreign Key resourceID **references** resource(resourceID) on update cascade, on delete no action

FD1: resourceID → capacity, room, building, campus (PK)

Immovable relation is well formed and in 1NF.
There are no partial dependencies, therefore Immovable relation is in 2NF.
There are no transitive dependencies, therefore Immovable relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

Loan (BCNF)

Loan (loanID, dateTimeBorrowed, dateTimeReturned, dateTimeDue, memberID, resourceID)
Primary Key loanID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action
Foreign Key resourceID **references** Movable(resourceID) on update cascade, on delete no action

FD1: loanID → dateTimeBorrowed, dateTimeReturned, dateTimeDue, memberID, resourceID (PK)

Loan relation is well formed and in 1NF.
There are no partial dependencies, therefore Loan relation is in 2NF.
There are no transitive dependencies, therefore Loan relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

Reservation (BCNF)

Reservation (reservationID, isCancelled, pickupUseDate, returnDueDate, memberID, resourceID)
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

FD1: reservationID → isCancelled, pickupUseDate, returnDueDate, memberID, resourceID (PK)

Reservation relation is well formed and in 1NF.
There are no partial dependencies, therefore Reservation relation is in 2NF.
There are no transitive dependencies, therefore Reservation relation is in 3NF.
The only functional dependency has a primary key as the determinant, so the relation is in BCNF

Acquisition (2NF to BCNF)

R1: **Acquisition** (acquisitionID, itemName, manufacturer, model, year, description, urgency, memberID)
Primary Key acquisitionID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

FD1: acquisitionID → itemName, manufacturer, model, year, description, urgency, memberID (PK)
FD2: model → description (Transitive Dependency)

Acquisition relation is well formed and in 1NF.
There are no partial dependencies, therefore Acquisition relation is in 2NF.
There is a transitive dependency (FD2), therefore Acquisition relation is not in 3NF.

Normalisation process (decomposing R1 based on FD2)

FD2 can be used to create new relation R2: AcquisitionModel
R2:XY: **AcquisitionModel** (model, description)
Primary Key model

Then remove dependent attribute 'description' from R1, and add FK model from R2
R3:R1-Y: **Acquisition** (acquisitionID, itemName, manufacturer, year, urgency, model, memberID)
Primary Key acquisitionID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action
Foreign Key model **references** AcquisitionModel(model) on update cascade, on delete no action

Judging normal form of decomposed relations

R2:FD1: model → description (PK)

R3:FD1: acquisitionID → itemName, manufacturer, year, urgency, model, memberID (PK)

R2 and R3 are well formed and in 1NF
There are no partial dependencies in either relation, therefore R2 and R3 are in 2NF
There are no transitive dependencies in either relation, therefore R2 and R3 are in 3NF
The only functional dependency in either relation has a primary key as the determinant.
Therefore the new relations R2 and R3 are in BCNF

Assumed Relation (to demonstrate normalising from 1NF to BCNF)

This is a bad example, but I couldn't think of a better one relevant to my EER to show partial dependency. In this example you will have to assume that a member cannot reserve the same resource ever again, as then then composite key of memberID + resourceID would not uniquely identify the pickupDate and returnDate (because one member would have multiple pickup/return dates).

Reservation (1NF to BCNF)

R1: **Reservation** (memberID, resourceID, memberName, pickupDate, returnDate, resourceStatus)
Primary Key memberID, resourceID

FD1: memberID → memberName (Partial Dependency)
FD2: resourceID → resourceStatus (Partial Dependency)
FD3: memberID, resourceID → pickupDate, returnDate

Reservation (R1) relation is well formed and in 1NF.
There are two partial dependencies (FD1, FD2), therefore relation is not in 2NF.

Normalisation process (decomposing R1 based on FD1 and FD2)

FD1 can be used to create new relation R2: MemberInfo (X1=memberID, Y1=memberName)
R2:XY1: **MemberInfo** (memberID, memberName)
Primary Key memberID

FD2 can be used to create new relation R3: ResourceInfo (X2=resourceID, Y2=resourceStatus)
R3:XY2: **ResourceInfo** (resourceID, resourceStatus)
Primary Key resourceID

We then remove the dependents Y1 and Y2 from R1, and add X1 and X2 as foreign keys in the parent
R4: R1-(Y1+Y2): **Reservation** (memberID, resourceID, pickupDate, returnDate)
Primary Key memberID, resourceID
Foreign Key memberID **references** MemberInfo(memberID) on update cascade, on delete no action
Foreign Key resourceID **references** ResourceInfo(resourceID) on update cascade, on delete no action

Judging normal form of result (R2, R3, R4)

R2:FD1: memberID → memberName

R3:FD1: resourceID → resourceStatus

R4:FD1: memberID, resourceID → pickupDate, returnDate

R2, R3, and R4 are well formed and in 1NF
There are no partial dependencies in any relation, therefore R2, R3, and R4 are in 2NF
There are no transitive dependencies in any relation, therefore R2, R3, and R4 are in 3NF
The only functional dependency in either relation has a primary key as the determinant.
Therefore the new relations R2 (MemberInfo), R3(ResourceInfo), and R4(Reservation) are in BCNF

Complete List of BCNF Relations in DBDL

Member (memberID, fName, lName, dateOfBirth, address, phoneNo, email, status, comment)
Primary Key memberID

Staff (memberID, schoolPosition, employeeHistory, isAdmin)
Primary Key memberID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

Student (memberID, points)
Primary Key memberID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

Course (courseID, courseName)
Primary Key courseID

CourseOffering (offeringID, semesterOffered, yearOffered, courseBeginDate, courseEndDate, courseID)
Primary Key offeringID
Foreign Key courseID **references** Course(courseID) on update cascade, on delete no action

Student_CourseOffering (memberID, offeringID)
Primary Key memberID, offeringID
Foreign Key memberID **references** Student(memberID) on update cascade, on delete cascade
Foreign Key offeringID **references** CourseOffering(OfferingID) on update cascade, on delete cascade

Category (categoryCode, name, description, maxBorrowTime, resourceType)
Primary Key categoryCode

CategoryKeywords (categoryCode, keywords)
Primary Key categoryCode, keywords
Foreign Key categoryCode **references** Category(categoryCode) on update cascade on delete cascade

Privilege (privilegeID, name, description, maxResources, categoryCode)
Primary Key privilegeID
Foreign Key categoryCode **references** Category(categoryCode) on update cascade, on delete no action

CourseOffering_Privilege (offeringID, privilegeID)
Primary Key offeringID, privilegeID
Foreign Key offeringID **references** CouseOffering(offeringID) on update cascade, on delete cascade
Foreign Key privilegeID **references** Privilege(privilegeID) on update cascade, on delete cascade

Resource (resourceID, description, presentStatus, categoryCode)
Primary Key resourceID
Foreign Key categoryCode **references** Category(categoryCode) on update cascade, on delete no action

Movable (resourceID, name, manufacturer, model, year, assetValue, buildingBDS)
Primary Key resourceID
Foreign Key resourceID **references** resource(resourceID) on update cascade, on delete no action

Immovable (resourceID, capacity, room, building, campus)
Primary Key resourceID
Foreign Key resourceID **references** resource(resourceID) on update cascade, on delete no action

Loan (loanID, dateTimeBorrowed, dateTimeReturned, dateTimeDue, memberID, resourceID)
Primary Key loanID
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action
Foreign Key resourceID **references** Movable(resourceID) on update cascade, on delete no action

Reservation (reservationID, isCancelled, pickupUseDate, returnDueDate, memberID, resourceID)
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

AcquisitionModel (model, description)
Primary Key model

Acquisition (acquisitionID, itemName, manufacturer, year, urgency, model, memberID)
Primary Key acquisitionID
Foreign Key model **references** AcquisitionModel(model) on update cascade, on delete no action
Foreign Key memberID **references** Member(memberID) on update cascade, on delete no action

SQL Database Creation

```
--DROP DATABASE IF EXISTS SDS_Database

--create database SDS_Database

use SDS_Database

go


--drop tables

drop table Acquisition

drop table AcquisitionModel

drop table Reservation

drop table Loan

drop table Immovable

drop table Movable

drop table Resource

drop table CourseOffering_Privilege

drop table Privilege

drop table CategoryKeywords

drop table Category

drop table Student_CourseOffering

drop table CourseOffering

drop table Course

drop table Student

drop table Staff

drop table Member

go


--MEMBER TABLE

CREATE TABLE Member(

memberID char(10) NOT NULL,

fName           varchar(20) NOT NULL,

lName           varchar(20) NOT NULL,

dateOfBirth     date NOT NULL,

address         varchar(50),

phoneNo         int,

email           varchar(50),

status          varchar(10) DEFAULT 'Active' CHECK (Status IN ('Active', 'Inactive')) NOT NULL,
comment                 varchar(50),
PRIMARY KEY(memberID));

--STAFF TABLE
CREATE TABLE Staff(
memberID                char(10) NOT NULL,
schoolPosition          varchar(20) NOT NULL,
employeeHistory         varchar(50),
isAdmin                 bit DEFAULT '1' NOT NULL, -- 1 = true, 0 = false
PRIMARY KEY(memberID),
FOREIGN KEY(memberID) references Member(memberID) on update cascade on delete no action);

--STUDENT TABLE
CREATE TABLE Student(
memberID                char(10) NOT NULL,
points                  smallint DEFAULT '12',
PRIMARY KEY(memberID),
FOREIGN KEY(memberID) references Member(memberID) on update cascade on delete no action);

--COURSE TABLE
CREATE TABLE Course(
courseID                char(8) NOT NULL,
courseName              varchar(50) NOT NULL,
PRIMARY KEY(courseID));

--COURSEOFFERING TABLE
CREATE TABLE CourseOffering(
offeringID              int identity(1,1) NOT NULL,
semesterOffered          char(1) NOT NULL,
yearOffered             char(4) NOT NULL,
courseBeginDate         date NOT NULL,
courseEndDate           date NOT NULL,
courseID                char(8) NOT NULL,
PRIMARY KEY(offeringID),
FOREIGN KEY(courseID) references Course(courseID) on update cascade on delete no action);

--STUDENT_COURSEOFFERING TABLE
CREATE TABLE Student_CourseOffering(
memberID            char(10) NOT NULL,
offeringID          int NOT NULL,
PRIMARY KEY(memberID, offeringID),
FOREIGN KEY(memberID) references Student(memberID) on update cascade on delete cascade,
FOREIGN KEY(offeringID) references CourseOffering(offeringID) on update cascade on delete cascade);

--CATEGORY TABLE
CREATE TABLE Category(
categoryCode                 int identity(1,1) NOT NULL,
name                         varchar(20) NOT NULL,
description                  varchar(50) NOT NULL,
maxBorrowHours               int NOT NULL,
resourceType                 varchar(15) NOT NULL
PRIMARY KEY(categoryCode));

--CATEGORYKEYWORDS TABLE
CREATE TABLE CategoryKeywords(
categoryCode        int NOT NULL,
keywords            varchar(15)            NOT NULL,
PRIMARY KEY(categoryCode, keywords),
FOREIGN KEY(categoryCode) references Category(categoryCode) on update cascade on delete cascade);

--PRIVILEGE TABLE
CREATE TABLE Privilege(
privilegeID                  int identity(1,1) NOT NULL,
name                         varchar(20) NOT NULL,
description                  varchar(50) NOT NULL,
maxResources                 smallint NOT NULL,
categoryCode                 int NOT NULL,
PRIMARY KEY(privilegeID),
FOREIGN KEY(categoryCode) references Category(categoryCode) on update cascade on delete no action);

--COURSEOFFERING_PRIVILEGE TABLE
CREATE TABLE CourseOffering_Privilege(
offeringID          int NOT NULL,
privilegeID         int NOT NULL,
PRIMARY KEY(offeringID, privilegeID),
FOREIGN KEY(offeringID) references CourseOffering(offeringID) on update cascade on delete cascade,
FOREIGN KEY(privilegeID) references Privilege(privilegeID) on update cascade on delete cascade);


--RESOURCES TABLE

CREATE TABLE Resource(

resourceID          char(10) NOT NULL,

description         varchar(50) NOT NULL,

presentStatus       varchar(15) DEFAULT 'Available' CHECK (presentStatus IN ('Available', 'Lost', 'Borrowed', 'Reserved', 'Maintenance')) NOT NULL,

categoryCode        int,

PRIMARY KEY(resourceID),

FOREIGN KEY(categoryCode) references Category(categoryCode) on update cascade on delete no action);


--MOVABLE TABLE

CREATE TABLE Movable(

resourceID                    char(10) NOT NULL,

name                          varchar(20) NOT NULL,

manufacturer                  varchar(20) NOT NULL,

model                         varchar(15) NOT NULL,

year                          char(10) NOT NULL,

assetValue                    decimal(6) NOT NULL,

buildingBDS                   varchar(10) NOT NULL,

PRIMARY KEY(resourceID),

FOREIGN KEY(resourceID) references Resource(resourceID) on update cascade on delete no action);


--IMMOVABLE TABLE

CREATE TABLE Immovable(

resourceID          char(10) NOT NULL,

capacity            smallint NOT NULL,

room                varchar(10) NOT NULL,

building            varchar(10) NOT NULL,

campus              varchar(20) NOT NULL,

PRIMARY KEY(resourceID),

FOREIGN KEY(resourceID) references Resource(resourceID) on update cascade on delete no action);


--LOAN TABLE

CREATE TABLE Loan(

loanID                        int identity(1,1) NOT NULL,

dateTimeBorrowed              datetime,

dateTimeReturned              datetime,
dateTimeDue                 datetime,
memberID                    char(10) NOT NULL,
resourceID                  char(10) NOT NULL,
PRIMARY KEY(loanID),
FOREIGN KEY(memberID) references Member(memberID) on update cascade on delete no action,
FOREIGN KEY(resourceID) references Movable(resourceID) on update cascade on delete no action);

--RESERVATION TABLE
CREATE TABLE Reservation(
reservationID       int identity(1,1) NOT NULL,
isCancelled         bit DEFAULT '0' NOT NULL, -- 1 = true, 0 = false
pickupUseDate       datetime,
returnDueDate       datetime,
memberID            char(10) NOT NULL,
resourceID          char(10) NOT NULL,
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);

--ACQUISITIONMODEL TABLE
CREATE TABLE AcquisitionModel(
model               varchar(15) NOT NULL,
description         varchar(50) NOT NULL,
PRIMARY KEY(model));

--ACQUISITION TABLE
CREATE TABLE Acquisition(
acquisitionID        int identity(1,1) NOT NULL,
itemName            varchar(20) NOT NULL,
manufacturer        varchar(20) NOT NULL,
year                char(4) NOT NULL,
urgency             varchar(20) NOT NULL,
model               varchar(15) NOT NULL,
memberID            char(10) NOT NULL,
PRIMARY KEY(acquisitionID),
FOREIGN KEY(model) references AcquisitionModel(model) on update cascade on delete no action,
FOREIGN KEY(memberID) references Member(memberID) on update cascade on delete no action);
go
```

SQL Database Queries

```
--SQL QUERIES


-- Q1: Name of students who have enrolled in courseID INFT1060

select m.fname as 'First Name of Student in INFT1060', m.lname as 'Last Name of Student in INFT1060'

from member m, course c, courseoffering co, Student_CourseOffering sc

where m.memberID = sc.memberID

        and c.courseID = co.courseID

        and co.offeringID = sc.offeringID

        and co.courseID = 'INFT1060';



--cleaner version, but also prints memberID

select m.memberID as 'Students enrolled in INFT1060', m.fname, m.lname

from member m, course c, courseoffering co, Student_CourseOffering sc

where m.memberID = sc.memberID

        and c.courseID = co.courseID

        and co.offeringID = sc.offeringID

        and co.courseID = 'INFT1060';




-- Q2: Maximum number of speakers that student Rolland Owens can borrow

-- V1: using sum to add maxResources. (change to distinct for if not adding them)

select sum(p.maxResources) as 'Max number of speakers Rolland Owens can borrow'

from category c, member m, privilege p, Student_CourseOffering sco, CourseOffering_Privilege cop

where c.name = 'speaker'

        and m.fname = 'rolland'

        and m.lname = 'owens'

        and p.categoryCode = c.categoryCode

        and sco.offeringID = cop.offeringID

        and p.privilegeID = cop.privilegeID

        and m.memberID = sco.memberID;
```

```
-- V2: showing each maxResources and the course that grants them. (note added courseOffering)
select (p.maxResources) as 'Max number of speakers Rolland Owens can borrow', co.courseID as 'granted by'
from category c, member m, privilege p, Student_CourseOffering sco, CourseOffering_Privilege cop, CourseOffering co
where c.name = 'speaker'
        and m.fname = 'rolland'
        and m.lname = 'owens'
        and p.categoryCode = c.categoryCode
        and sco.offeringID = cop.offeringID
        and p.privilegeID = cop.privilegeID
        and m.memberID = sco.memberID
        and co.offeringID = sco.offeringID; --added line
```

```
-- Q3: For Staff with ID number M0004, print name, phone, total reservations made in 2022
select m.fname, m.lname, m.phoneNo, count(r.reservationID) as 'total reservations in 2022'
from member m, reservation r
where m.memberID = 'M0004'
        and m.memberID = r.memberID
        and r.pickupUseDate between '2022-01-01' and '2022-12-31'
group by m.fname, m.lname, m.phoneNo;
```

```
-- Q4: Names of students who have borrowed from category camera with model E-M10 this year.
select m.fname, m.lname
from member m, loan l, category c, movable mr, resource r
where m.memberID = l.memberID
        and mr.resourceID = l.resourceID
        and r.resourceID = l.resourceID
        and r.categoryCode = c.categorycode
        and c.name = 'camera'
        and mr.model = 'E-M10'
        and year(l.dateTimeBorrowed) = year(SYSDATETIME());
```

```
-- Q5: Find movable resource that is the most loaned in current month. Print name/ID
select mr.resourceID, mr.name
from movable mr, loan l
where mr.resourceID = l.resourceID
        and month(l.dateTimeBorrowed) = month(SYSDATETIME()) --can also use getdate()
        and year(l.dateTimeBorrowed) = year(SYSDATETIME())
group by mr.resourceID, mr.name
having count(*) >= All
        (Select count(*)
        from movable mr, loan l
        where mr.resourceID = l.resourceID
        and month(l.dateTimeBorrowed) = month(SYSDATETIME())
        and year(l.dateTimeBorrowed) = year(SYSDATETIME())
        group by mr.resourceID);



-- Q6: For the three days (01-May-2022, 05-June-2022, 19-Sep-2022), print date, room name, and
-- total reservations made for room ICT-325 on each day.
select cast(r.pickupUseDate as date) as 'Reservation date', i.room, count(r.pickupUseDate) as '# of reservations'
from reservation r left join immovable i on (r.resourceID = i.resourceID)
where i.room = '325'
        --cast datetime to date
        and cast(r.pickupUseDate as date) in ('2022-05-01', '2022-06-05', '2022-09-19')
        group by cast(r.pickupUseDate as date), i.room;
go
```