* Step 4.1 - Map the E-R model developed at the end of Chapter 3 to a relational model, using the guidelines presented in Section 4.7 and illustrated in the sample project. Write the schema for the database, but do not create the database at this stage.

Primary keys are underlined, foreign keys are italicized

**Member**(memberID, address, areaCode, city, email, firstName, lastName, phoneNumber, state, zip)

**Play**(playID, author, numberOfActs, title, type, numberOfSets)

**Sponsor** (businessName, income)

**Subscriber** (subscriberID, address, areaCode, city, email, firstName, lastName, pastPlays, phoneNumber, state, zip)

**Production** (playID*,* cost, productionPostions, yearOfTheProduction, seasonStartDate, seasonEndDate)

**Performance** (memberID, date, time, year)

**TicketSale** (ticketIncome)

**DuesPayment (**memberID*,* duesPaid)

**Donation** ( businessName*,* *donationNumber*)

**Ticket﻿﻿﻿﻿﻿﻿﻿** (*qrCode*, playDate, *playNames*, playTime, price, seat, seatNumbers)

**The relationship sets are:**

**Acts (**memberID*, playID****,*** role)

The many to many Acts will be presented by a foreign key. That means many members on the stage can act the plays together. Therefore memberID in the member entity is the primary key and the memberID in the play entity is the foreign key.

**BoughtBy**(subscriberID, *qrCode* )

The BoughtBy relationship connects a Subscriber to a Ticket. Each Subscriber may purchase only one ticket (per our most recent version of assumptions), and a Ticket can only have one Subscriber connected/attached to it. This would be a one-to-one relationship. The subscriberID is a primary key in Subscriber. qrCode is a primary key in Ticket and a foreign key in Subscriber.

**Donates**(businessName*,* donationNumber*,* donationAmount, dateOfDonation)

Donates is a many to many relationship as many businesses can donate as many times as they want. BusinessName and donationNumber are represented as the primary keys in this relationship to connect the business to the donation

**PaysDues**(memberID*,* dateofDuesSubmission, duesAmount)

PaysDues is represented as a many to one relationship as you will have many members paying dues. The memberID connects each individual member to the amount and date of dues paid.

**Performs (***playID, date, year)*

Based on our assumptions, you can only have perform one play in a day creating a one to many relationship. The plays can be performed on more than one date, but one date can not have more than one play.

**Produces**(*memberID*, *playID*, yearOfTheProduction, seasonStartDate, seasonEndDate)

Produces is represented as one to many, because there’s only one producer in the team and he/she needs to be responsible for managing all aspects of that production. Here, the memberID is the primary key in the member entity and the memberID in the production is the foreign key.