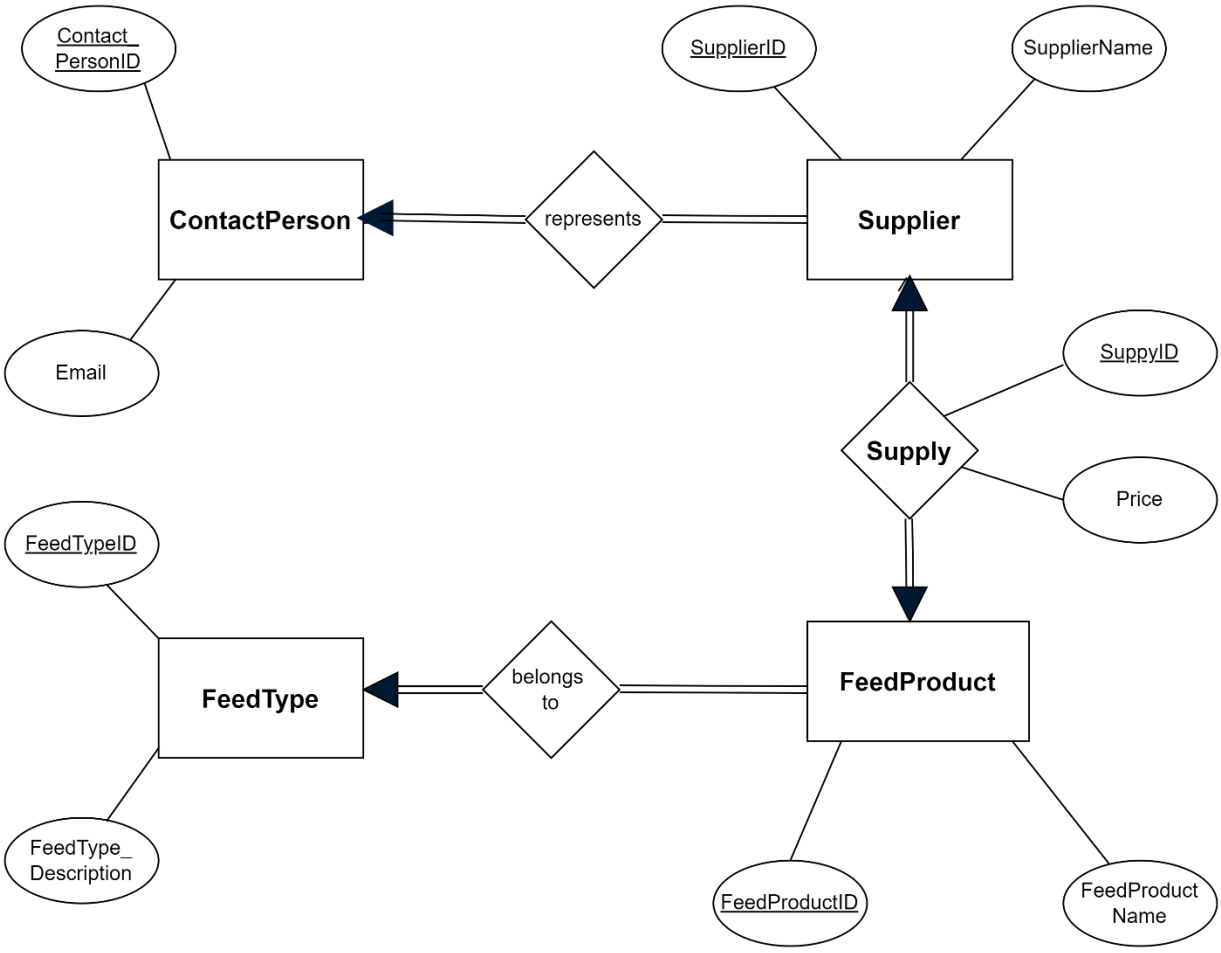
Business rules, ERDs and Relation Schemas examples and solutions

- Using Chen's notation, draw an ERD according to the business rules below. Translate the ERD to a logical level showing the relation schemas – include primary and foreign key fields.
  - **Sheep farm feed supplier business rules:**
    1. All entities must have surrogate primary keys.
    2. A contact person represents one or more suppliers, and each supplier has exactly one contact person.
    3. A supplier supplies many feed products, and every feed product can be supplied by many suppliers.
    4. Each feed product belongs to a specific feed type, and many feed products can have the same feed type.

ERD from the Sheep farm business rules.



Relation schema from the Sheep farm ERD

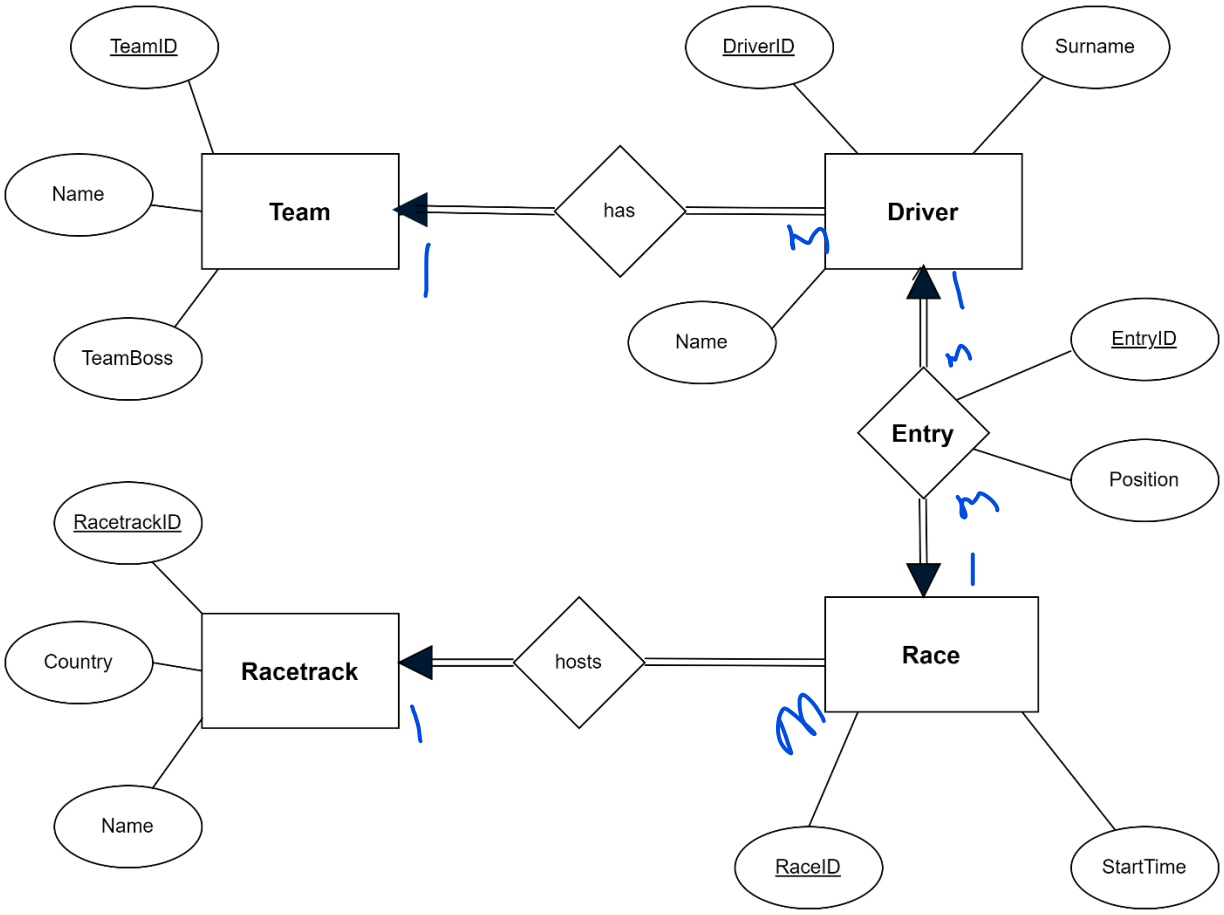
- ContactPerson (**ContactPersonID,** Email)
- Supplier (**SupplierID**, SupplierName, ContactPersonID)
- Supply (**SupplyID**, Price, SupplierID, FeedProductID)
    - *(Note: Supply is a bridge entity with **SupplyID** as surrogate PK as it is indicated in the business rules. Otherwise, if surrogate keys are not required, the PK would be **SupplierID + FeedProductID**)*
- FeedPoduct (**FeedProductID**, FeedProductName, FeedTypeID)
- FeedType (**FeedTypeID**, FeedTypeDescription)

More examples on business rule, ERD and Relation schemas

Formula One racing – business rules:

- For each team that takes part in Formula One races, you need to store the team name and the name of the team boss.
- A team has exactly two drivers, and a driver can only be part of one team.
- The name and surname of each driver needs to be stored in the database.
- Each racetrack that is used for Formula One racing has a name and country that needs to be recorded.
- Every race that takes place is hosted by exactly one racetrack, and a race has a start time (which should include the date).
- Drivers can enter many races during their careers, and the position that they place in each race needs to be stored for statistical purposes.
- Many drivers enter the same race.

ERD from Formula One racing business rules.



Relation schema from Formula One racing business rules.

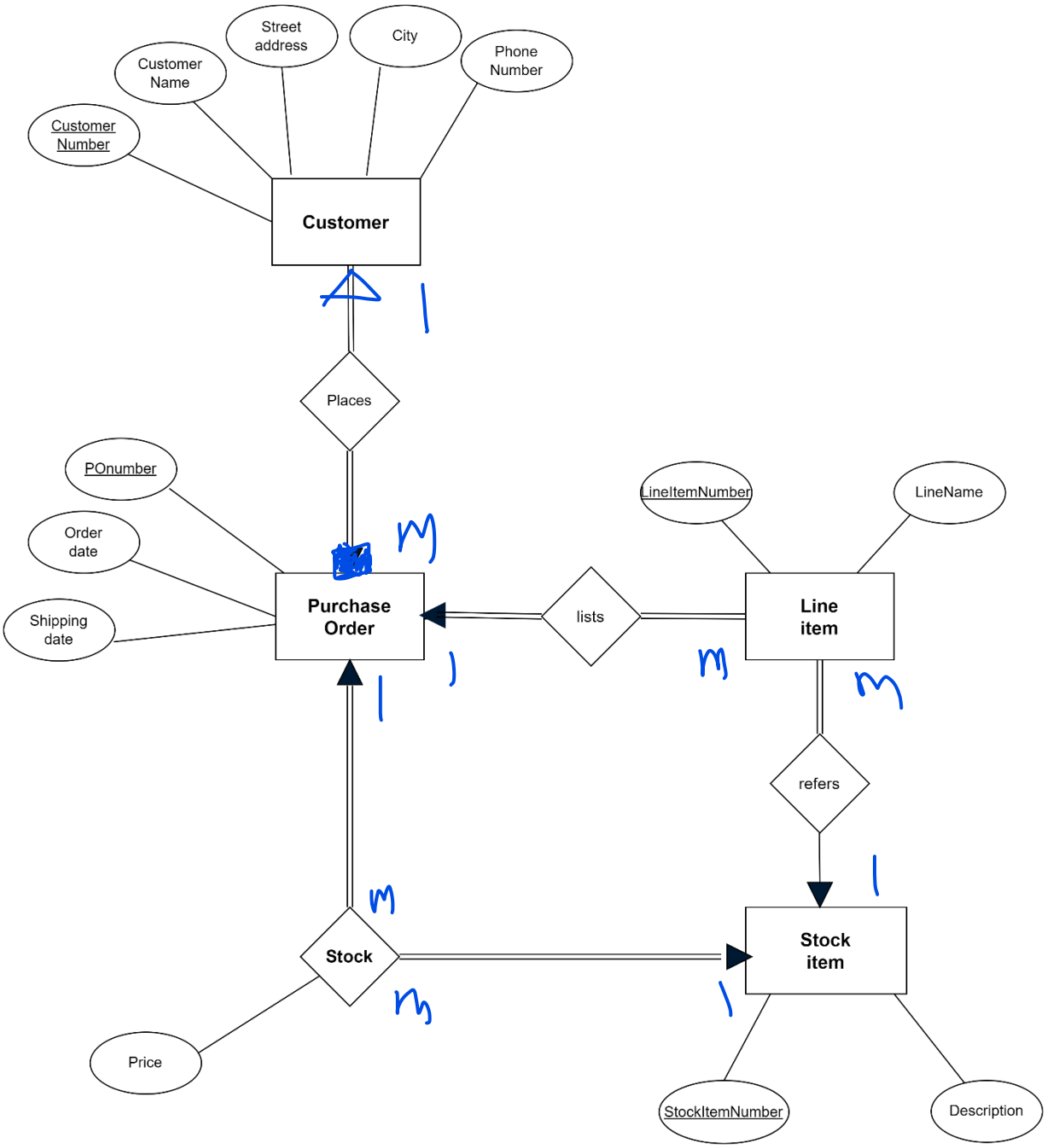
- Team (**TeamID**, Name, TeamBoss)
- Driver (**DriverID**, Surname, Name, TeamID)
- Entry (**EntryID**, Position, DriverID, RaceID)
- Race (**RaceID**, StartTime, RacetrackID)
- Racetrack (**RacetrackID**, Name, Country)

More examples on business rule, ERD and Relation schemas

Purchase order business rules:

- A purchase order can list many line items, but a given line item can be listed by only one purchase order.
- Purchases order details that can be considers are purchase order number, customer number, order date, shipping date, street address, city.
- A line item can refer to only one stock item, but a given stock item can be referred to by many line items. The relationship is optional because zero line items might refer to a given stock item.
- Line item details can be as follows line item number, purchase order number, quantity and discount.
- Purchase order can contain many stock items, and a stock item can be contained in many purchase orders.
- Stock can be stock item number, description and price.
- Customer can place many orders, but a given purchase order can be placed by only one customer.
- Customer details will contain the following, customer number, customer name , street address, city and phone number.

ERD from Purchase order business rules.



Relation schema from Purchase order business rules

- Customer (**CustomerNumber**, CustomerName, StreetAddress, City, PhoneNumber, PONumber)
- PurchaseOrder (**PONumber**, OrderDate, ShippingDate)
- LineItem (**LineItemNumber**, LineName, PONumber, StockItemNumber)
- StockItem (**StockItemNumber**, Description)

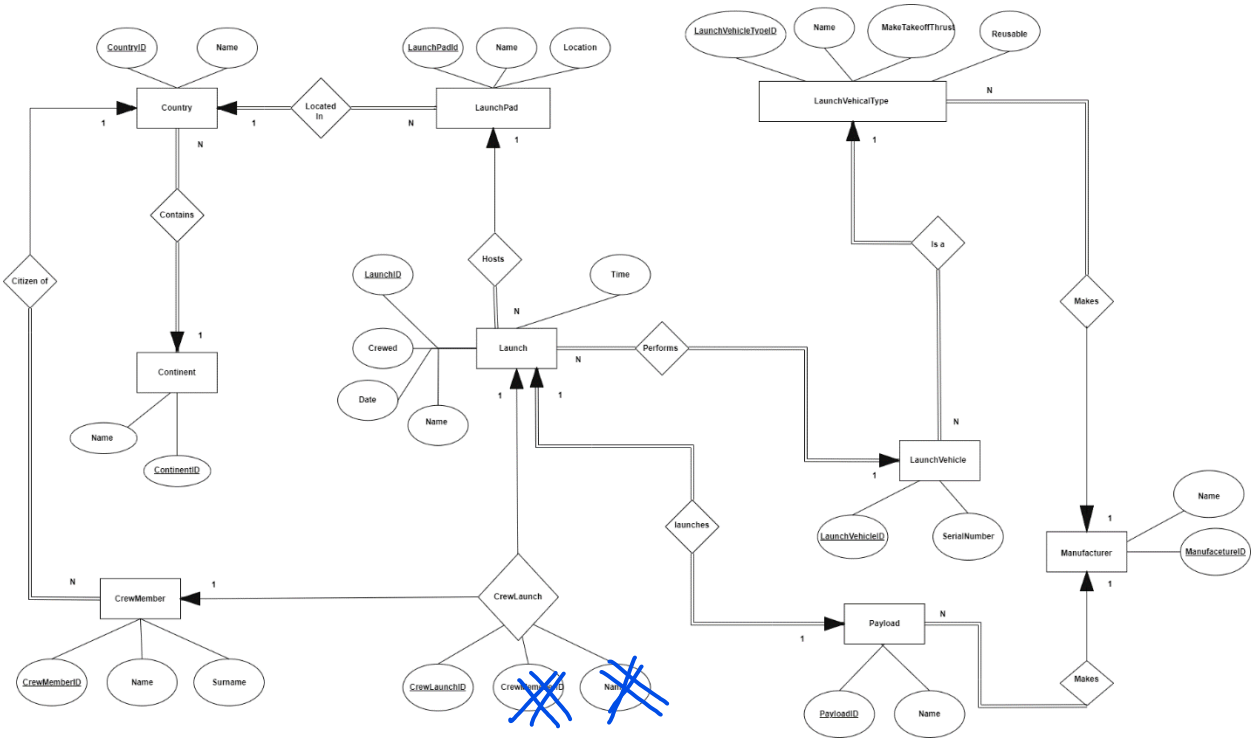
- Stock (**PONumber, StockITemNumber,** Price)
    - (Stock is the bridge entity between Purchase order and Stock item. In this case: the PK is **PONumber, StockITemNumber**, while we have two FKs i.e., FK1: **PONumber** and FK2: **StockITemNumber**).

More examples on business rule, ERD and Relation schemas

- **Launch Pad business rules:**
  1. A launchpad hosts multiple launches over time, and each launch is hosted by exactly one launchpad. Therefore, Launchpads will be created in the database even before the first launch is hosted there.
  2. A launchpad is located in a country, and a country, in turn, is located on a continent.
  3. Each launchpad has a name and a location.
  4. A launch requires a launchpad that hosts it, a launch vehicle (the rocket), and the payload.
  5. For a launch, the date, time and name should be recorded as well as whether it is a crewed launch or not.
  6. Each launch is performed by a launch vehicle. The serial number for the launch vehicle should be recorded. A launch vehicle can perform multiple launches over time.
  7. Each launch vehicle is of a specific launch vehicle type. The name, maximum thrust and whether it is reusable need to be recorded for the launch vehicle type.
  8. A specific manufacturer makes each launch vehicle type, and a manufacturer can make many different launch vehicle types.
  9. Each launch carries one payload, and a payload is only carried by one launch.
  10. A manufacturer also makes each payload, and a manufacturer can make many different payloads over time.
  11. Multiple crew members can be carried on a crewed launch, and each crew member can be carried on multiple launches over time.
  12. Each crew member has a nationality (the country they come from), and their name and surname also need to be recorded.
  13. Crew members can appear in the database before they launch for the first time.

- NB: Draw an ERD representing these business rules.
- You are required to show relation schemas that include surrogate primary key and foreign key fields.

ERD extracted from the Launch Pad business rules.



Relation schemas for Launch Pad business rules

Country (**CountryID,** Name, ContinentID)

Continent (**ContinentID,** Name)

CrewMember (**CrewMemberID,** Name, Surname, CountryID)

Launch (**LaunchID,** Date, Time, Name, Crewed, LaunchPadID, LaunchVehicleID, PayloadID)

LaunchPad (**LaunchPadID**, Name, Location, CountryID)

CrewLaunch (**CrewLaunchID**, ~~Name,~~ LaunchID, CrewMemberID)

LaunchVehicle (**LaunchVehicleID**, SerialNumber, LaunchVehicleType)

LaunchVehicleType (**LaunchVehicleTypeID**, Name, MaxtakeoffThrust, Reusable, ManufacturerID)

Manufacturer (**ManufacturerID**, Name)

Payload (**PayloadID**, Name, ManufacturerID)

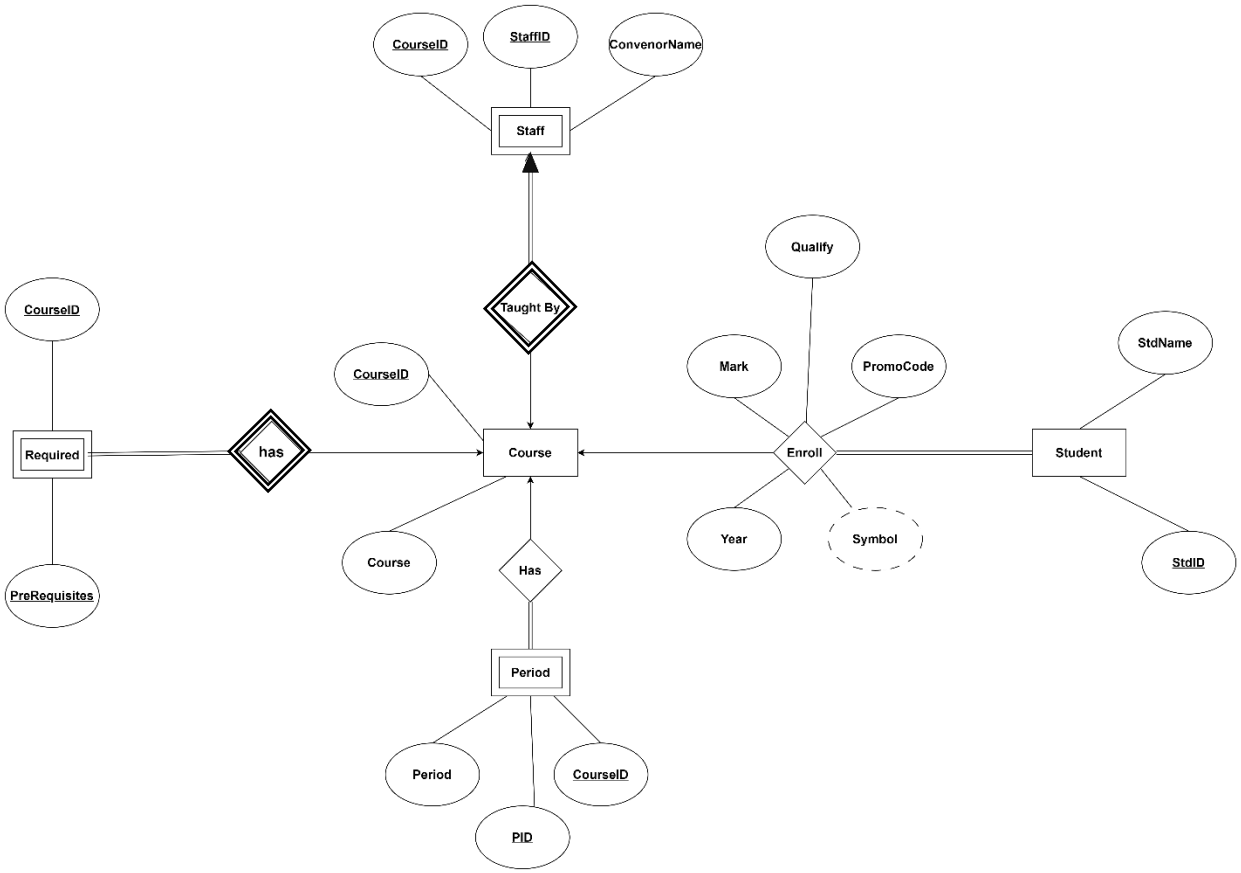
HW1.8 solution

Case: Greendale Community College (see spreadsheet data attached on Blackboard)

- A spreadsheet has been created that stores results of students at Greendale Community College. This includes annual promotion code REN (excluded), CON (can continue) or QUA (qualifies), along with information about the courses; their pre-requisites and lecture periods. You are required to design a relational database that can represent the information contained in the spreadsheet as closely and completely as possible.
  - Draw an ER (entity-relationship) diagram for the information in the spreadsheet, using the notation employed in this course. If necessary, state any assumptions you had to make when modelling this college data, and explain any modelling decisions taken.
  - Represent your ERD as a logical design showing relation schemas of this data. Choose 3 or 4 rows from the spreadsheet and show how that data would be stored in your database. Briefly describe any design decisions taken and any limitations of your schema.
  - Give any one functional dependency that holds for this information, and state in simple words what your functional dependency tells us about the data.
  - Is your relation scheme in 1st normal form or not? Give a reason for your answer.
  - Is your relation scheme in 3rd normal form or not? Give a reason for your answer.

Normalized

ERD for Greendale Community College



Relation schemas for Greendale Community College

Student (**StdID**, StdName)

Enroll (**StdID, CourseID**, Qualify, Mark, PromoCode, Year)

Course (**CourseID**, Course)

Period (**CourseID**, Period)

Required (**CourseID**, PreRequisites)

Staff (**StaffID, CourseID,** ConvenorName)

Showing data in each relation/table

Table name: **Student**, PK: **StdID**, FK: **None**

StdID	StdName
1	Abbot
2	Adams
3	Barnes

Table name: **Enroll**, PK: **StdID + CourseID**, FK1: **StdID**, FK2: **CourseID**

StdID	**CourseID**	Qualify	Mark	PromoCode	Year
1	BOT202	N	54	CON	2019
1	CSC201	N	55	CON	2019
2	ACC106	N	43	REN	2019
2	ECO110	N	50	REN	2019
3	MAM100	N	73	CON	2019
3	PHY131	N	67	CON	2019

Table name: **Course**, PK: **CourseID**, FK: **None**

CourseID	Course
1	BOT202
2	CSC201
3	ACC106

Table name: **Period**, PK: **PID + CourseID**, FK: **None**

PID	**CourseID**	Period
1	1	5th daily
2	2	2nd daily
3	3	4th daily
4	3	6th daily

Table name: **CourseID**, PK: **CourseID + PreRequisites**, FK: **None**

CourseID	PreRequisites
1	BOT104
1	ZOO103
2	CSC117
2	MAM100
3	

Table name: **Staff**, PK: **StaffID + CourseID**, FK: **None**

StaffID	**CourseID**	ConvenorName
1	BOT202	Marx
2	CSC201	Almeida
3	ACC106	Pavel
4	ECO110	Powers

Recall: Symbol is a derived attribute for which should not be stored in the database. This is why it does not appear anywhere in our relations/tables.

Functional dependencies

StdID → StdName: this can be formally interpreted as StdName is functionally dependent on StdID.

StdID, CourseID → Qualify, Mark, PromoCode, Year

CourseID → Course

CourseID, **PID** → Period)

CourseID → PreRequisites)

StaffID, CourseID → ConvenorName)

Normal Forms of the relations/tables

The relation schemas are in 1NF (each relation has a single data value, **PK** is identified) and 3NF (no partial and transitive dependencies. Each attribute depends on a primary key.)