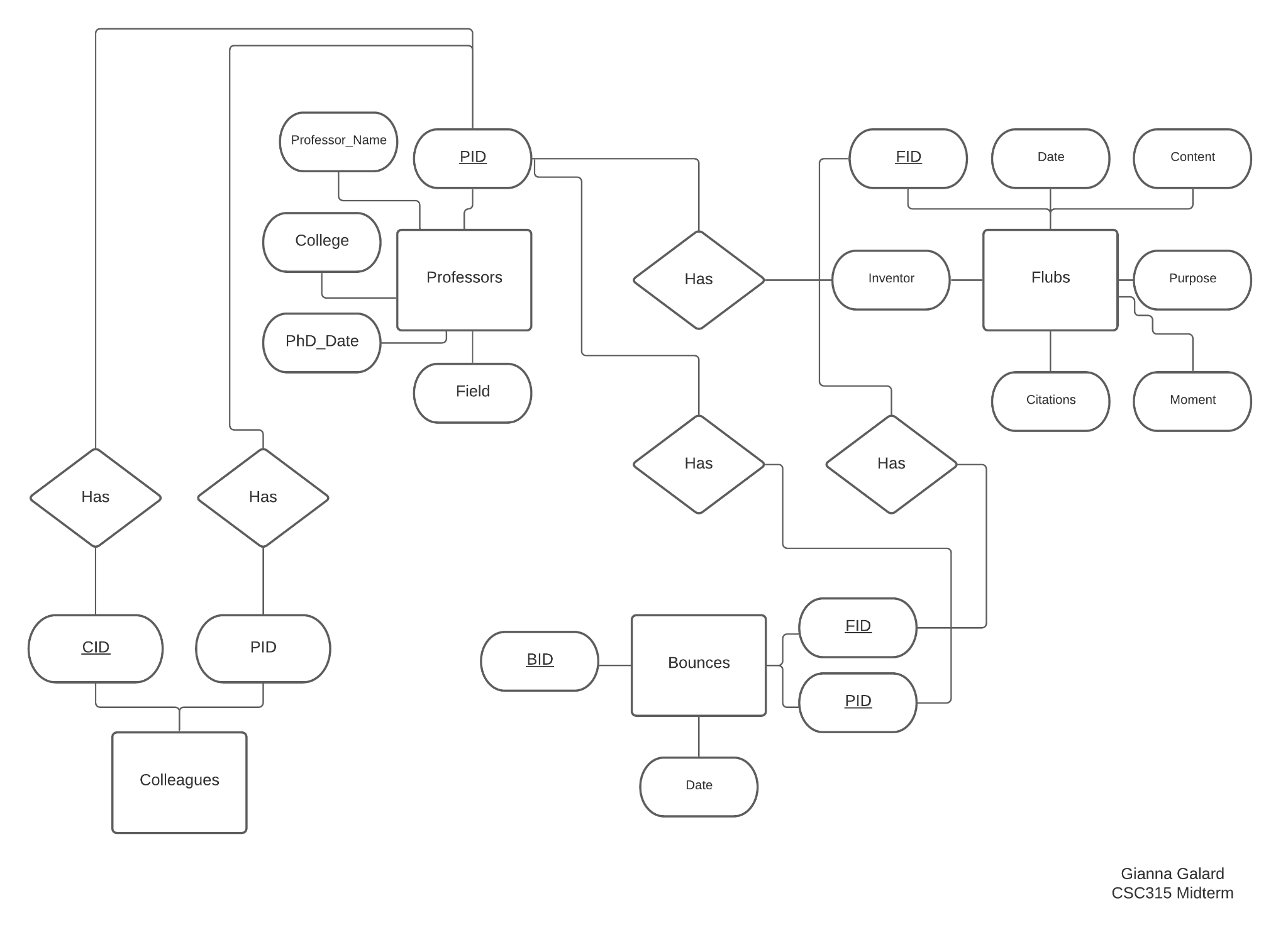
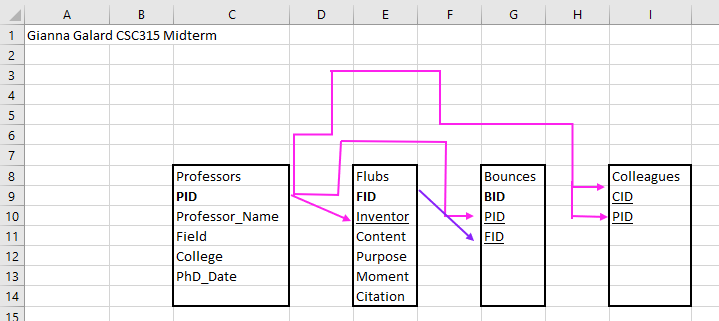
You are asked to design and implement a database for a social media project themed upon the movies The Absent Minded Professor and/or Flubber. (You don't need to watch either movie, apologies in advance if you do, but can visualize this like Twitter or TikTok or any other popular social media with posts, repost/shares, likes, friends/followers, etc) There are 4 steps, each relating to the major topics covered so far in class.

1.) Create an *Entity-Relationship Diagram* with the following requirements:

* There are Professors (ie users) with the changeable attributes Professor\_Name, Field, College, PhD\_Date
* There are Flubs with the unchangeable attributes Content, Purpose, Moment, Inventor (which is the creating Professor)
* There are Bounces where a Professor can share another Professor's Flub
* Add ID attributes as necessary
* Content only needs to be a text of fixed length
* Professors can have/be Colleagues
* A Flub can get Citations by other Professors



2.) Create the *Relational Model* for the Flubber database based upon your ER Diagram in the previous question, meeting all the same requirements. Include all necessary Primary Key and Foreign Key constraints. This should be syntactically correct to test in a MySQL database. (if using a different database, specify which)



| # Gianna Galard, CSC315 Midterm  # init db  CREATE DATABASE CSC315Midterm;  USE CSC315Midterm;  CREATE TABLE Professors (  PID int auto\_increment PRIMARY KEY, -- auto generate key  Field varchar(100),  Professor\_Name varchar(100),  College varchar(100),  PhD\_Date date  );  CREATE TABLE Bounces (  BID int auto\_increment PRIMARY KEY,  PID int,  FID int,  Date date,  FOREIGN KEY (FID) REFERENCES Flubs(FID),  FOREIGN KEY (PID) REFERENCES Professors(PID)  );  CREATE TABLE Flubs (  FID int auto\_increment PRIMARY KEY,  Inventor int,  Date date,  Content varchar(100),  Purpose varchar(100),  Moment varchar(100),  Citations int,  FOREIGN KEY (Inventor) REFERENCES Professors(PID)  );  CREATE TABLE Colleagues(  CID int NOT NULL auto\_increment PRIMARY KEY,  PID int,  FID int,  FOREIGN KEY (PID) REFERENCES Professors (PID),  FOREIGN KEY (FID) REFERENCES Professors (PID)  ); |
| --- |

3.) Write the necessary *SQL Queries* for the following functions.

* Adding a new Professor
* Changing a specific Professor's Professor\_Name
* Removing a Flub
* Show a portfolio of the Flubs by a Professor in reverse chronological order
* Show a Flub along with how many Bounces and Citations it has
* Show a portfolio of all Flubs and Bounces (the Flubs bounced) by all of a Professor's Colleagues in reverse chronological order

| -- adding a new professor  INSERT INTO Professors (Professor\_Name, Field, College, PhD\_Date) VALUES ('Firstname Lastname', 'Computer Science', 'NYU', '2019-6-09');  -- changing a specific professors Professor\_Name  UPDATE Professors SET Professor\_Name = 'Lastname Firstname'  WHERE Professor\_Name = 'Firstname Lastname';  -- removing a flub  DELETE FROM Flubs  WHERE FID = 1;  -- show a portfolio of the flubs by a professor in reverse chronological order  SELECT \* FROM Flubs  WHERE Inventor = 1  ORDER BY Date ASC;  -- show a flub along with how many bounces and citations it has  SELECT COUNT(Bounces.BID) AS Bounces, Flubs.\*  FROM Flubs, Bounces  WHERE Flubs.FID = 1 and BOUNCES.FID = 1;  -- show a portfolio of all flubs and bounces (the flubs bounced) by all of a professors colleagues in reverse chronological order  SELECT Bounces.\*, Flubs.\*  FROM Flubs, Bounces  WHERE Flubs.fid IN(SELECT PID FROM Colleagues WHERE CID = 1)  AND Bounces.FID IN(SELECT PID FROM Colleagues WHERE CID = 1)  ORDER BY Bounces.Date ASC, Flubs.Date ASC; |
| --- |

4.) Show the *Relational Algebra* AND *Domain Relational Calculus* formulas for each. This one may be a photo/scan of hand written answers if you have trouble entering the necessary symbols in an editor.

* Show a portfolio of the Flubs by a Professor

| SELECT \* FROM Flubs ORDER BY PID;  Relational Algebra:  Domain Relational Calculus: |
| --- |

* Show a portfolio of all Flubs and Bounces (the Flubs bounced) by all of a Professor's Colleagues

| SELECT \* FROM Flubs UNION SELECT \* FROM Bounces ORDER BY PID;  Relational Algebra:  Domain Relational Calculus: |
| --- |

5.) Create the SQL for a single *View* called "Flubbergasting" of a Professor which includes all attributes in question part 1-A as well as how many Flubs, the average number of Bounces the Flubs had, and total Citations across all Flubs by that professor.

| /\* Create the SQL for a single View called "Flubbergasting" of a Professor which includes all  attributes in question part 1-A as well as how many Flubs, the average number of Bounces the Flubs had, and total Citations across all Flubs by that professor. \*/  CREATE VIEW Flubbergasting AS  SELECT Professor\_Name, Field, College, PhD\_Date,  (SELECT COUNT(\*) FROM Flubs WHERE Flubs.Inventor = P.PID) AS FlubsCount,  (SELECT AVG (BounceCount) FROM  (SELECT COUNT(\*) AS BounceCount, Flubs.FID FROM Flubs INNER JOIN Bounces ON Flubs.FID = Bounces.FID WHERE Flubs.Inventor = P.PID  GROUP BY Flubs.FID) AS temp) AS AvgBounceCount,  (SELECT COUNT(\*) FROM Flubs WHERE Flubs.FID IN  (SELECT Flubs.FID FROM Flubs WHERE Flubs.inventor = P.PID)) AS CitationCount  FROM Professors P; |
| --- |