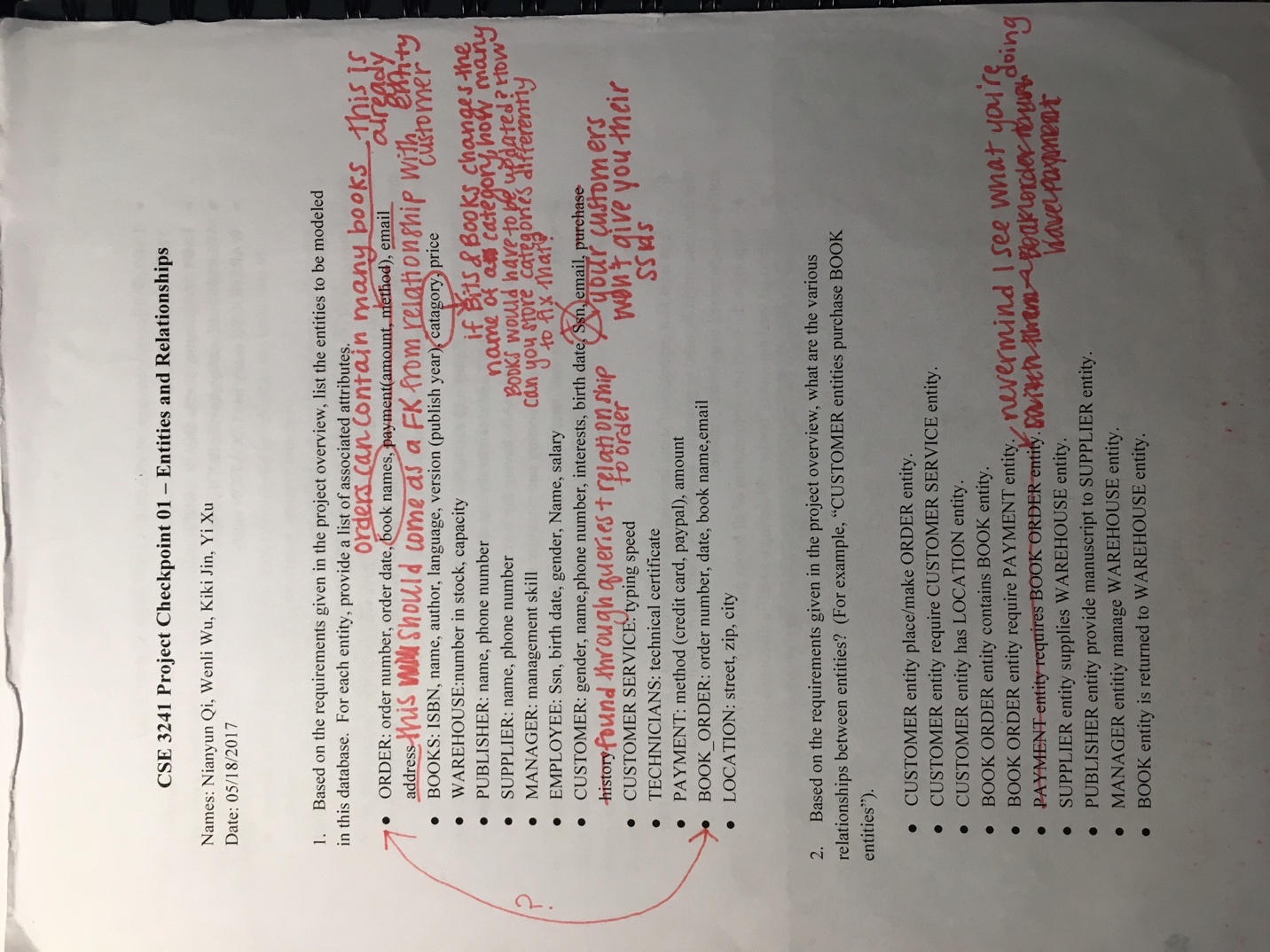
Checkpoint 01:

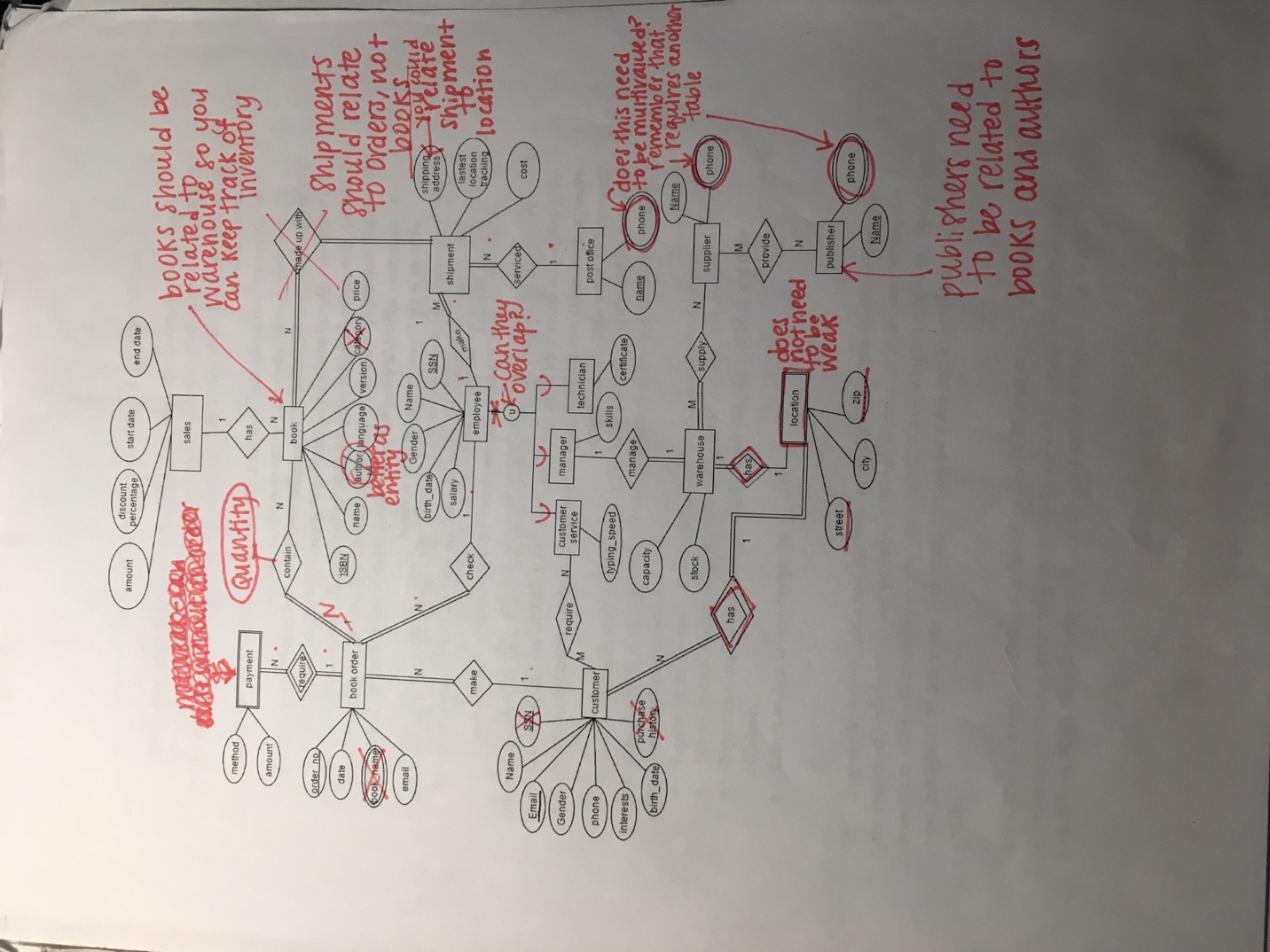
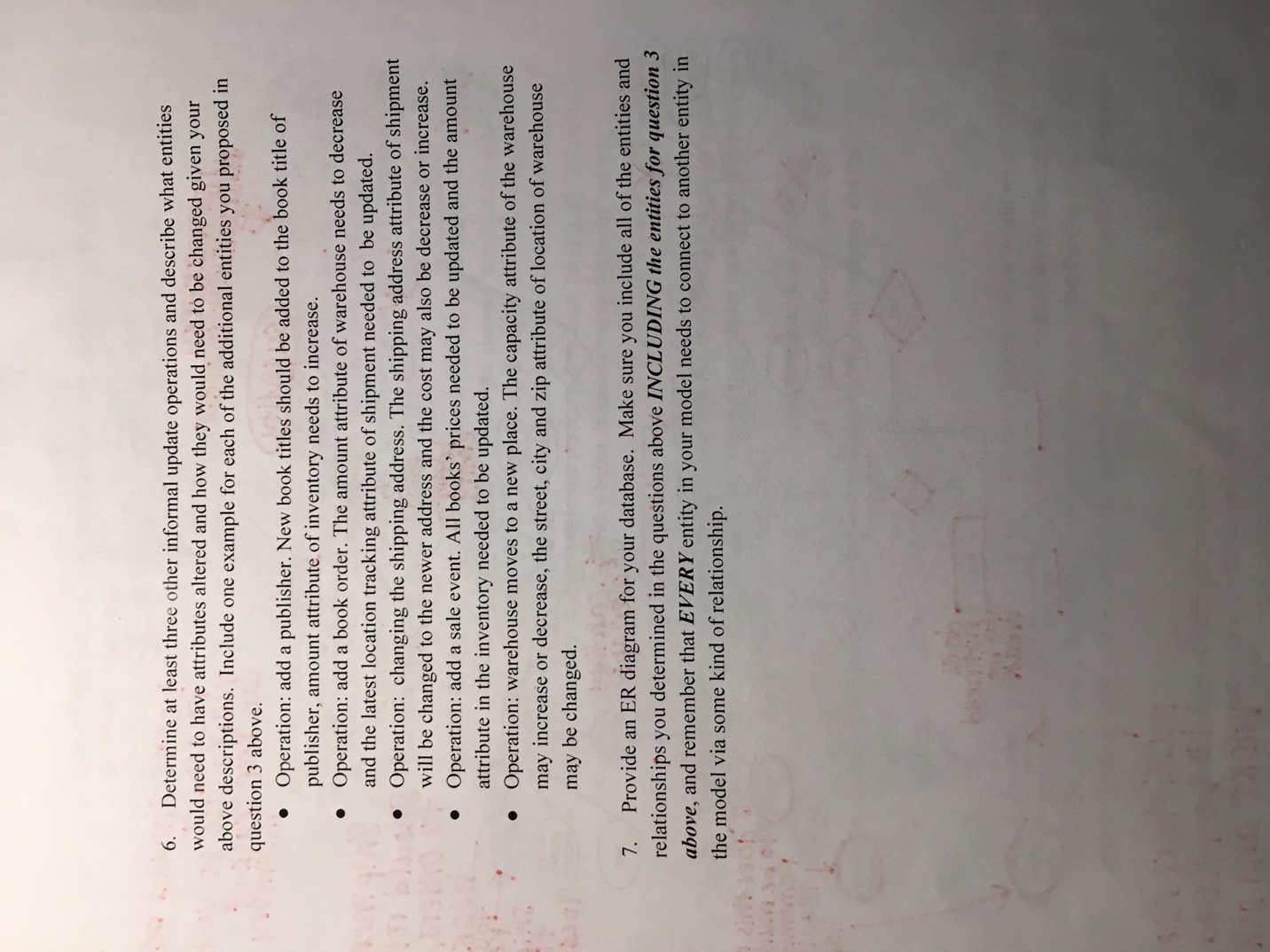


1. Since the book data provided in the course’s webpage has an attribute named category, we do not remove category from the book table.

We made changes to our ER diagram according to the advices TA gave us. Our fixed ER diagram is available in part a in Section 1 of Part 1(same folder). We changed the primary key of CUSTOMER table to email, deleted purchase history and made email address a foreign key of BOOK\_ORDER table.

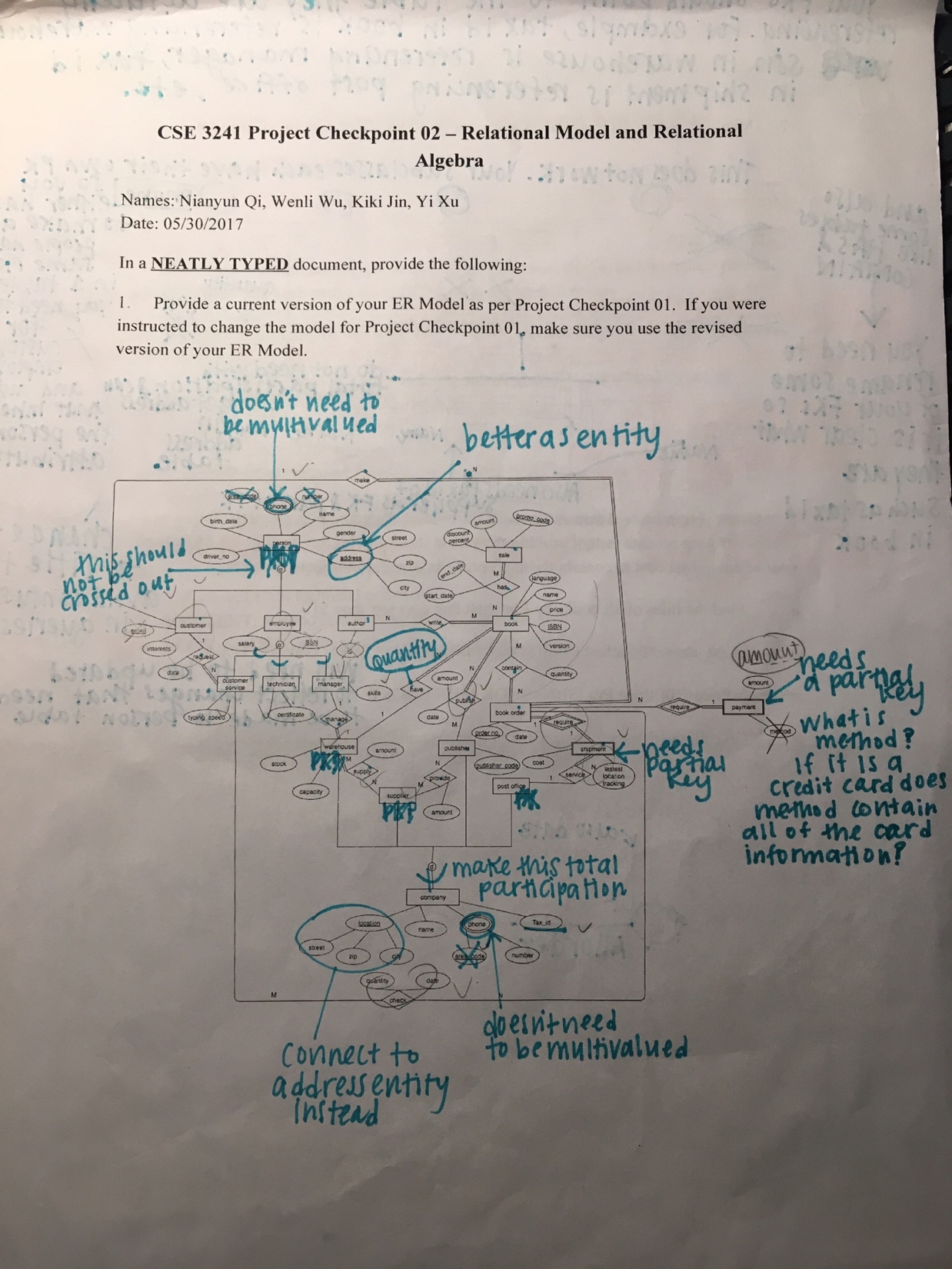


we made publisher an entity which can be found in ER diagram in part a in Section 1 of Part 1(same folder).

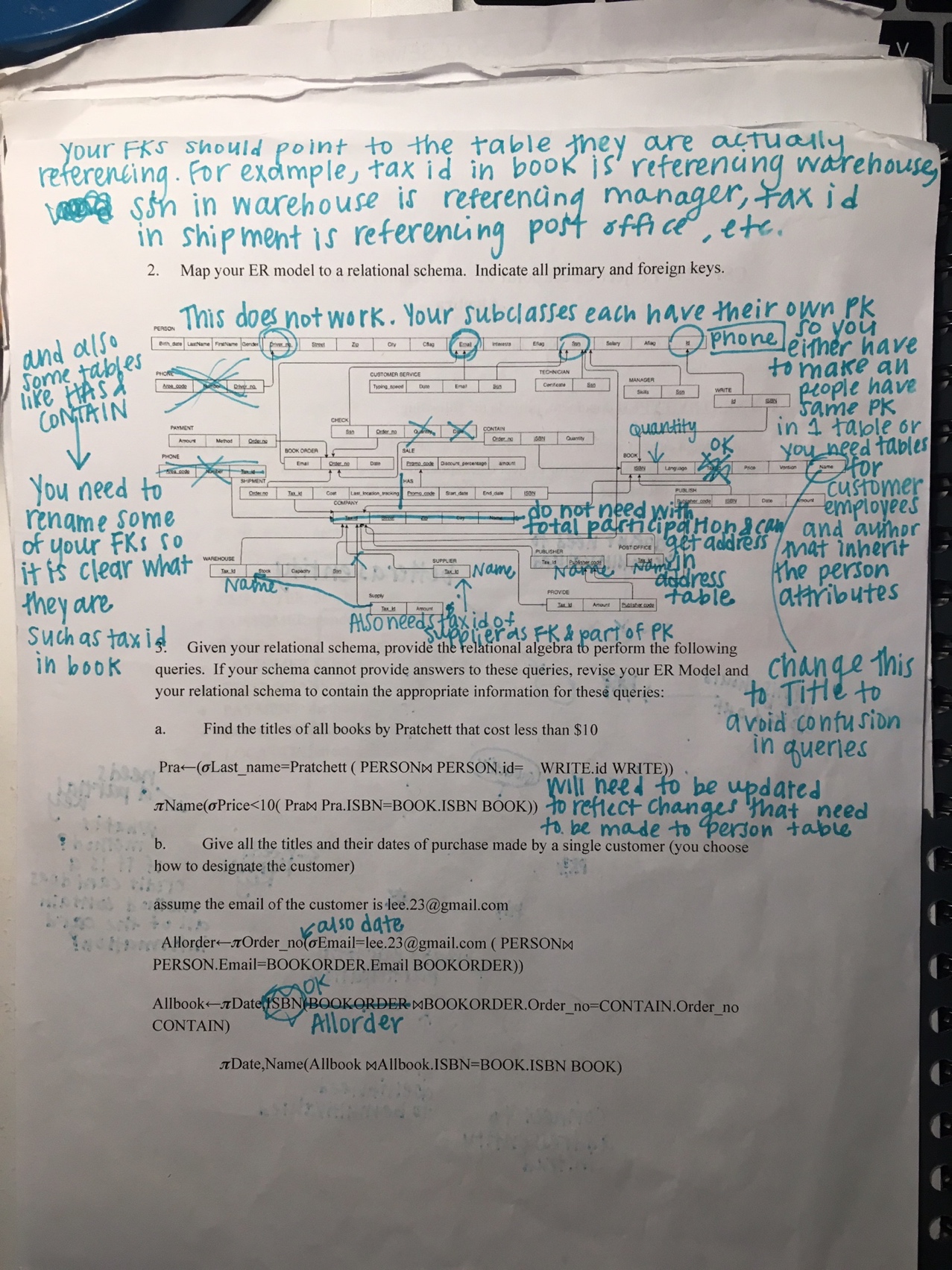


we made changes to our ER model according to the feedback, and we added a person entity which was further divided into customer, employee, and author. We also added a company entity which was divided into post office, warehouse, supplier, publisher. By doing so, some common attributes like phone could be inherited. Our fixed ER diagram is available in part a in Section 1 of Part 1(same folder).

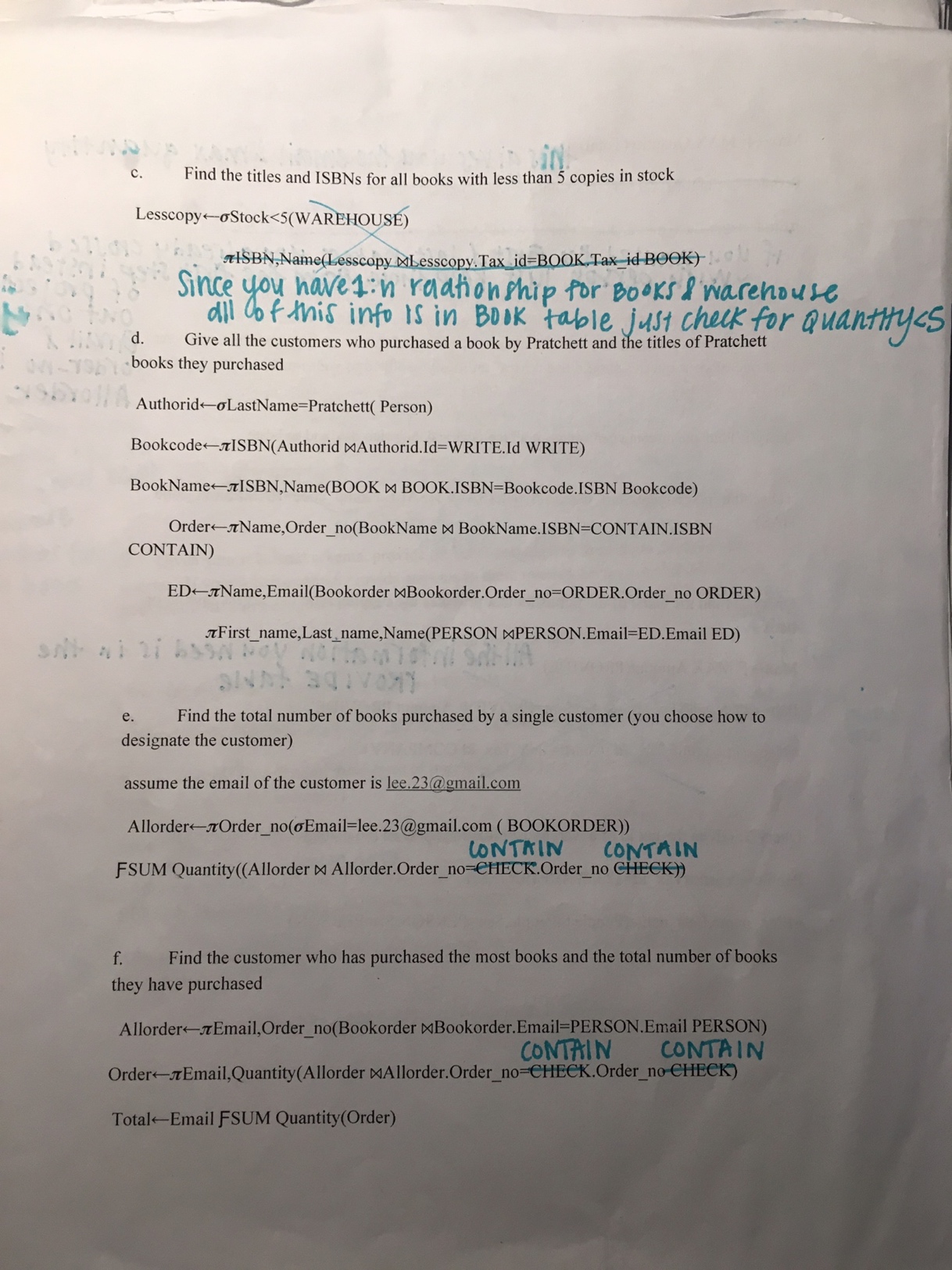
Checkpoint 2



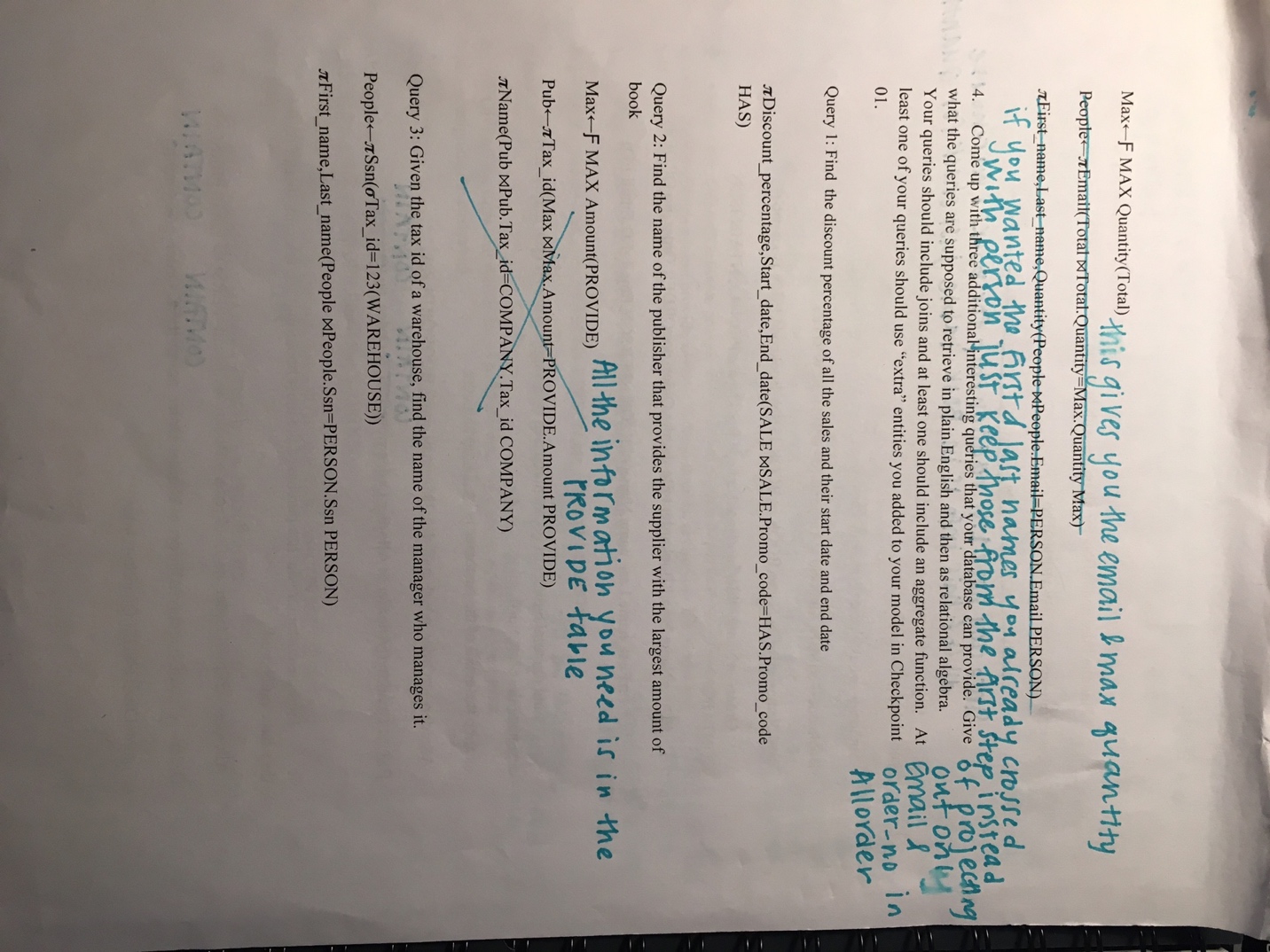
we made some changes to our ER diagram according to the feedback, for example, we made the address as an entity and we added card number, billing zip, security code as payment’s attributes. Our fixed ER diagram is available in part a in Section 1 of Part 1(same folder).



we made changes to our relational mode according to the feedback, for example, we changed the pointers to make them point to the primary keys they actually referred to and we renamed some attributes to avoid ambiguity when writing queries. Our fixed relational model is available in part b in Section 1 of Part 1(same folder).

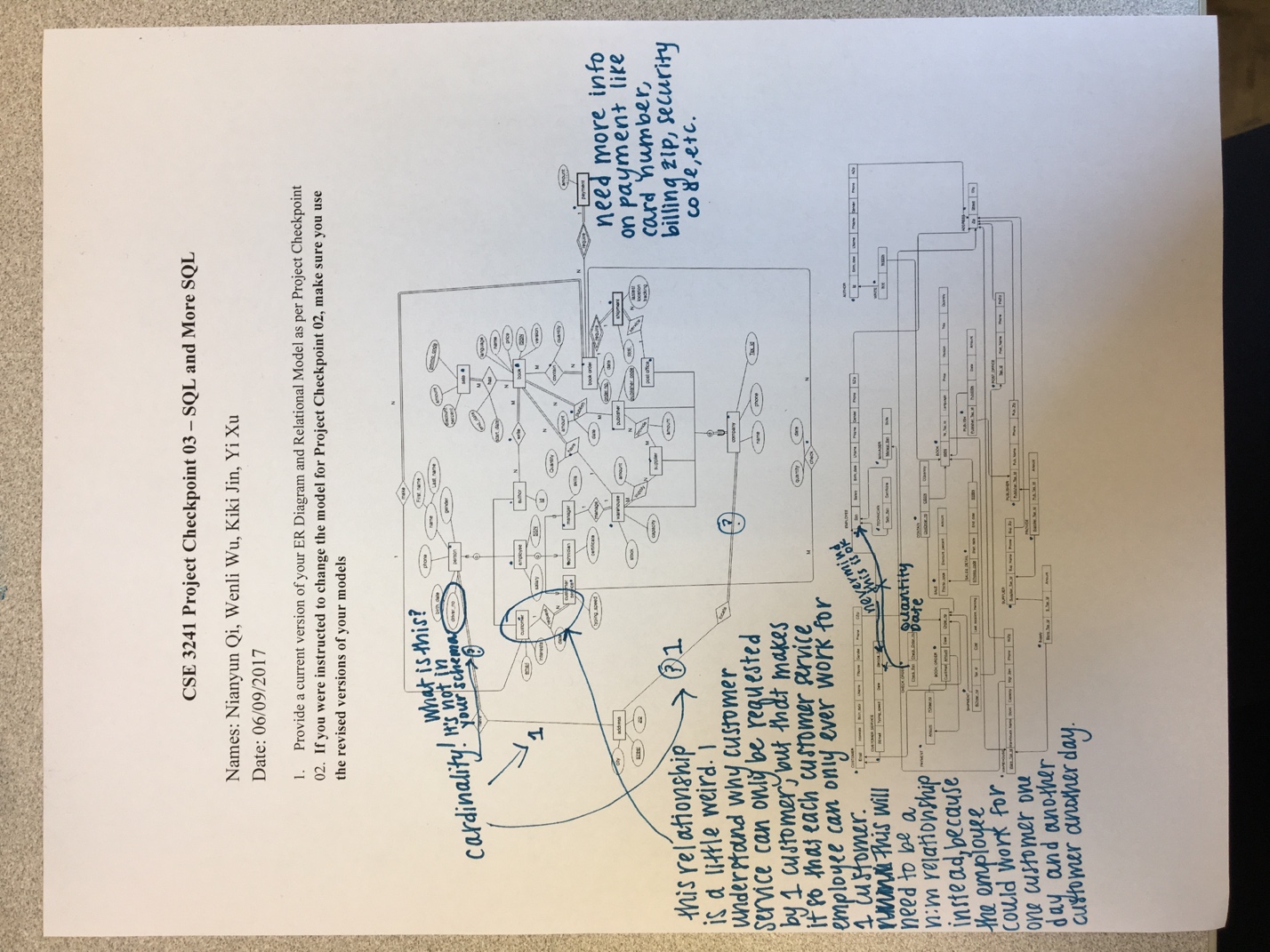


changes to those queries could be found in part B in Section 2 of Part 1(same folder)

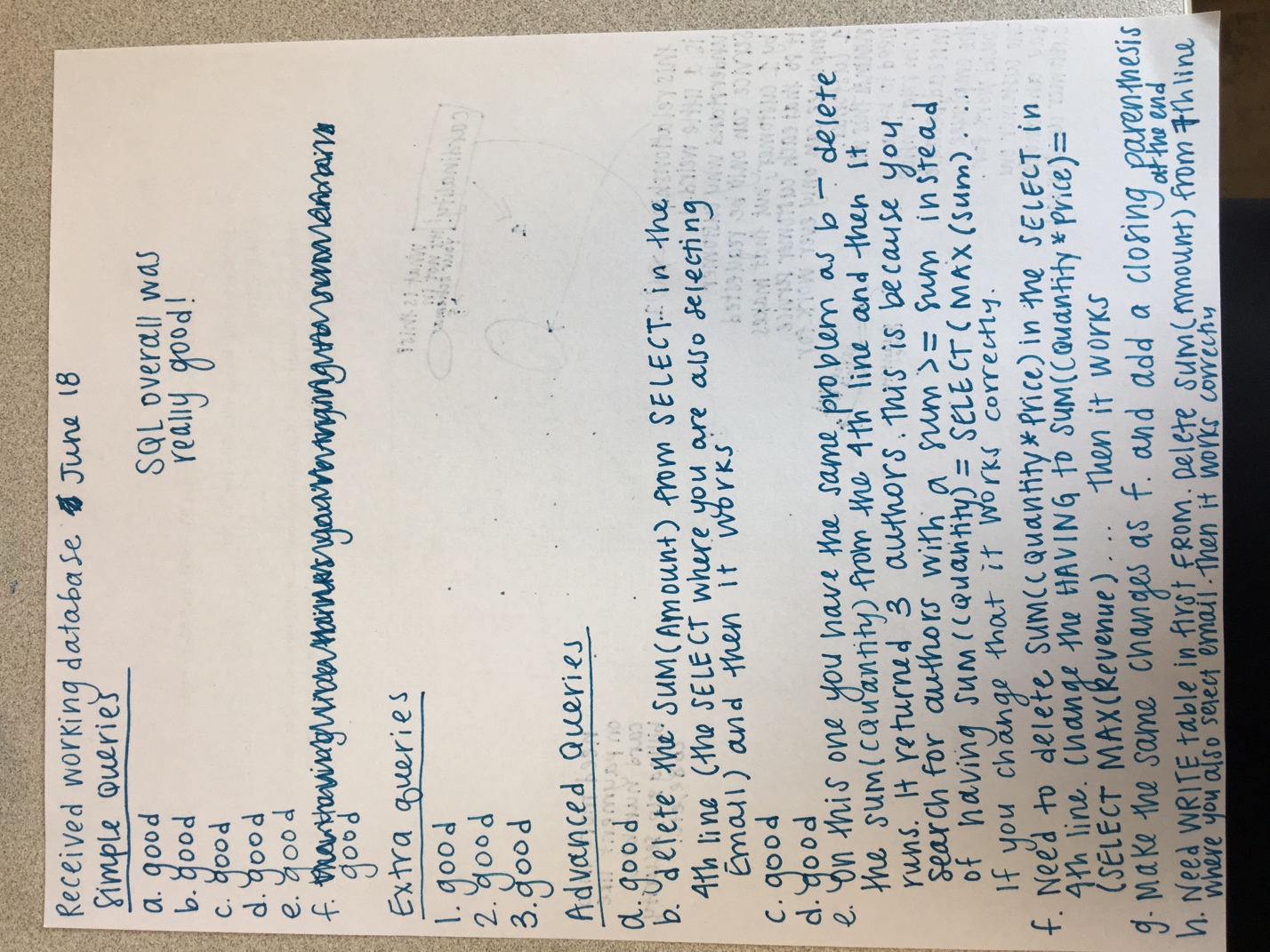


changes to those queries is available in part B in Section 2 of Part 1(same folder)

Checkpoint 3



we added cardinality to the relationships between address and person, and between address and company. We also changed the cardinality of request between customer and customer service to be many to many and added service code as the relationship attribute since the primary key ssn of customer service and email of customer combined cannot uniquely identify a service, we created a surrogate key named service code. We fixed our ER diagram according to the feedback, Our fixed ER diagram is available in part a in Section 1 of Part 1(same folder).



we made changes to our query and the resulting query is found below.

Provide a list of customer names, along with the total dollar amount each customer has spent.

SELECT FName,LName, SUM(Amount)

FROM CUSTOMER, BOOK\_ORDER,PAYMENT

WHERE Email=CusEmail

GROUP BY Email;

Provide a list of customer names and e-mail addresses for customers who have spent more than the average customer.

SELECT Email,FName,LName

FROM CUSTOMER

WHERE Email IN

(SELECT Email

FROM(SELECT Email, SUM(Amount)

            FROM CUSTOMER, BOOK\_ORDER,PAYMENT

            WHERE Email=CusEmail AND POrder\_no=Order\_no

             GROUP BY Email

            HAVING SUM(Amount)> (SELECT AVG(sum)

                                                         FROM (SELECT SUM(Amount) AS sum

                                                                                     FROM CUSTOMER, BOOK\_ORDER,PAYMENT

                                                                                 WHERE Email=CusEmail AND POrder\_no=Order\_no

                                                                                GROUP BY  Email))));

    Provide a list of the titles in the database and associated total copies sold to customers, sorted from the title that has sold the most individual copies to the title that has sold the least.

SELECT Title, SUM(CQuantity)

FROM  CONTAIN, BOOK

WHERE CISBN=ISBN

GROUP BY ISBN

ORDER BY SUM(CQuantity) DESC;

Provide a list of the titles in the database and associated dollar totals for copies sold to customers, sorted from the title that has sold the highest dollar amount to the title that has sold the smallest.

SELECT Title, SUM(CQuantity \* Price) AS Revenue

FROM CONTAIN, BOOK

WHERE CISBN=ISBN

GROUP BY ISBN

ORDER BY Revenue DESC;

Find the most popular author in the database (i.e. the one who has sold the most books)

SELECT FName,LName

FROM AUTHOR

WHERE Id IN

(SELECT Id

            FROM   (SELECT Id, SUM(CQuantity)

              FROM CONTAIN, BOOK,WRITE,AUTHOR

             WHERE CISBN=ISBN AND  ISBN=WISBN AND WId=Id

               GROUP BY Id

                 HAVING SUM(CQuantity)=(SELECT MAX(sum)

                                                                FROM(SELECT SUM(CQuantity) AS sum

                                                                 FROM CONTAIN, BOOK,WRITE,AUTHOR

                                                              WHERE CISBN=ISBN AND  ISBN=WISBN AND WId=Id

                                                               GROUP BY Id))));

Find the most profitable author in the database for this store (i.e. the one who has brought in the most money)

SELECT FName, LName

FROM AUTHOR

WHERE Id IN

(SELECT Id

      FROM   (SELECT Id, SUM(CQuantity \* Price)

           FROM CONTAIN, BOOK, AUTHOR,WRITE

          WHERE CISBN=ISBN AND ISBN=WISBN AND WId=Id

           GROUP BY Id

           HAVING SUM(CQuantity \* Price)=(SELECT MAX(Revenue)

                                                             FROM(SELECT SUM(CQuantity \*Price) AS Revenue

                                                               FROM CONTAIN, BOOK, AUTHOR,WRITE

                                                              WHERE CISBN=ISBN AND ISBN=WISBN AND WId=Id

                                                              GROUP BY Id ))));

 Provide a list of customer information for customers who purchased anything written by the most profitable author in the database.

FROM BOOK\_ORDER, CONTAIN, BOOK,WRITE

WHERE Order\_no=ConOrder\_no AND CISBN=ISBN AND ISBN=WISBN AND WId IN(

SELECT Id

FROM AUTHOR

WHERE Id IN

(SELECT Id

       FROM (SELECT Id, SUM(CQuantity \* Price)

           FROM CONTAIN, BOOK, AUTHOR,WRITE

          WHERE CISBN=ISBN AND ISBN=WISBN AND WId=Id

           GROUP BY Id

           HAVING SUM(CQuantity\*Price)=(SELECT MAX(Revenue)

                                                             FROM(SELECT SUM(CQuantity \*Price) AS Revenue

                                                               FROM CONTAIN, BOOK, AUTHOR,WRITE

                                                              WHERE CISBN=ISBN AND ISBN=WISBN AND WId=Id

                                                              GROUP BY Id ))))));

 Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.

SELECT AUTHOR.FName, AUTHOR.LName

FROM AUTHOR, CUSTOMER, BOOK\_ORDER,CONTAIN, BOOK,WRITE

WHERE Email=CusEmail AND Order\_no=ConOrder\_no AND CISBN=ISBN AND WISBN=ISBN                 AND WId=Id AND Email IN (SELECT  Email

                                                FROM CUSTOMER

                                                 WHERE Email IN

                                             (SELECT Email

                                               FROM (SELECT Email, SUM(Amount)

                                                  FROM CUSTOMER, BOOK\_ORDER,PAYMENT

                                                  WHERE Email=CusEmail AND POrder\_no=Order\_no

                                                   GROUP BY Email

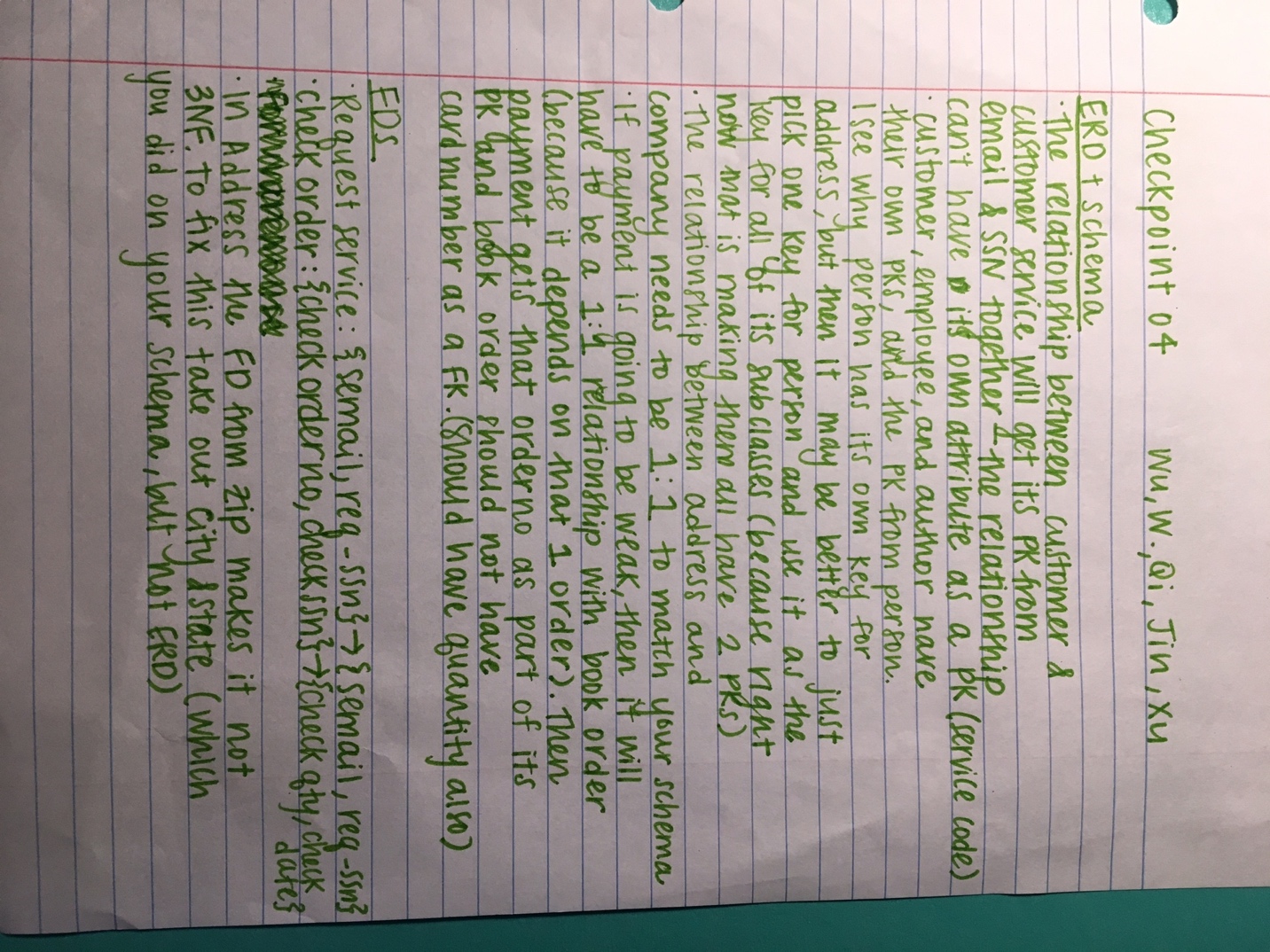
                                                      HAVING SUM(Amount)> (SELECT AVG(sum)

                                                         FROM (SELECT SUM(Amount) AS sum

                                                                                     FROM CUSTOMER, BOOK\_ORDER,PAYMENT

                                                                                    WHERE Email=CusEmail AND POrder\_no=Order\_no

                                                                                         GROUP BY  Email)))));



We cannot find a primary key of Person that is applicable to uniquely identify Author, Employee and Customer, and customer cannot give others their ssn, we kept our original choice of primary key of Person. Since the primary key of Payment table is card number and order number, if we changed the cardinality to 1:1, the table is not in 2nf, so we kept the original cardinality. Because Contain table already has an attribute named quantity, since the quantity in an order payment cannot be found, we do not add the quantity attribute. Our fixed ER diagram can be found in part a in Section 1 of Part 1(same folder). And our fixed functional dependency can be found in part B in Section 2 of Part 1(same folder).