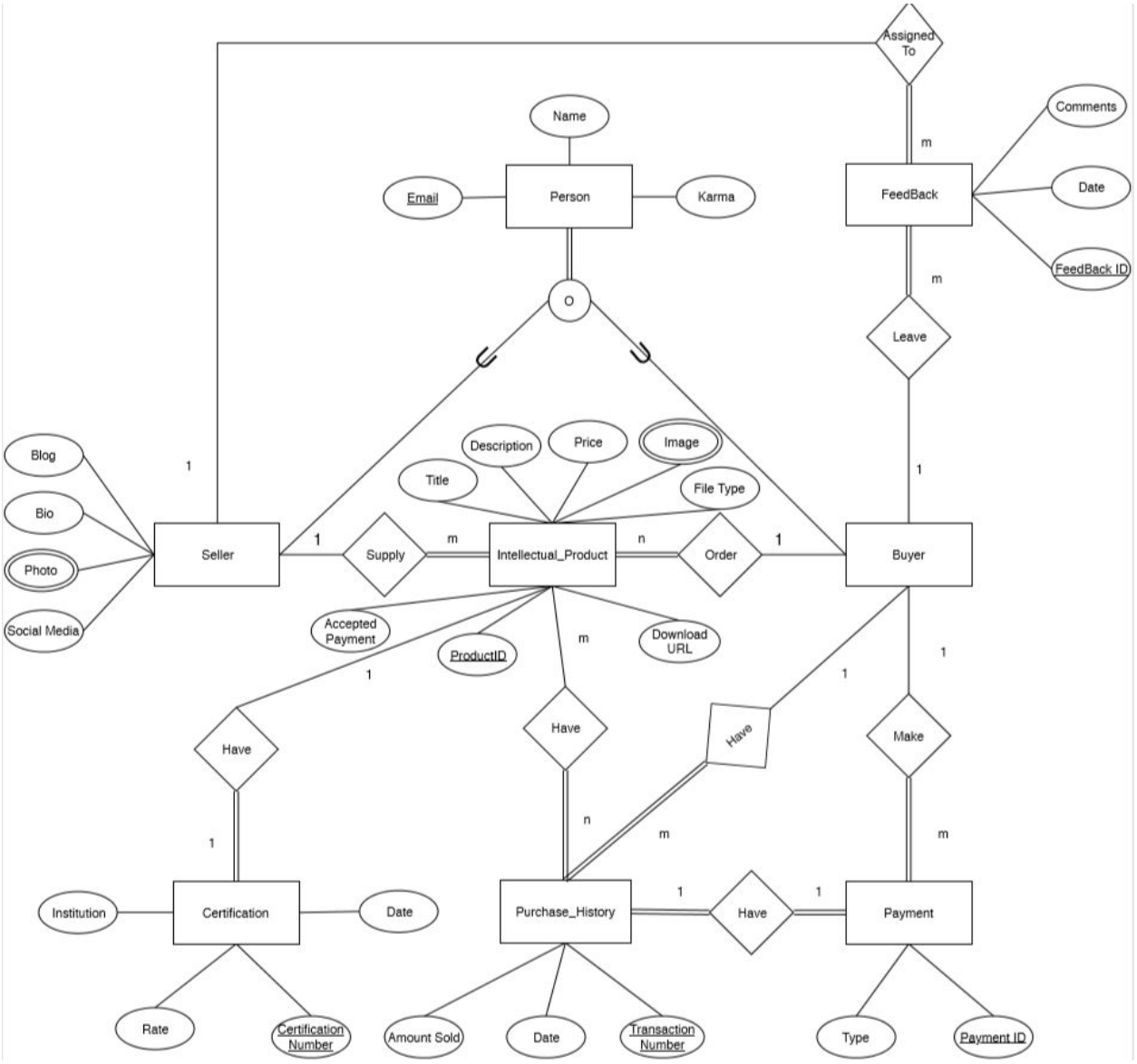
CSE 3241 Final Project

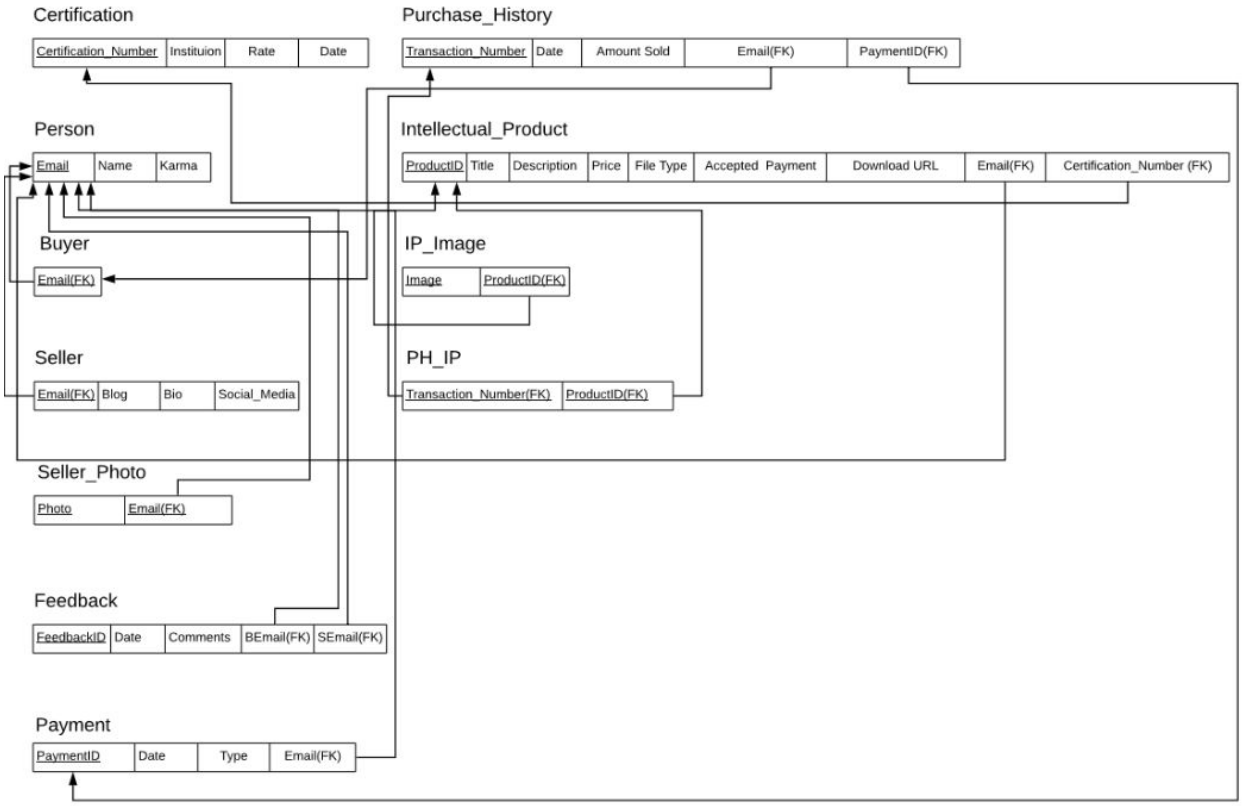
Caleb George
Michael Hayworth
Jianfeng Guo
Zhijian Yao

Section 1 Database Description

1. Entity relational model



2. Relational schema



3. Table Normal Form

Every table is in BCNF. For every dependency X -> Y in a table, Y ⊆ X or X is a super key for that table.

Person: BCNF

FD1: Email -> {Name, Karma}

Seller: BCNF

FD1: Email -> {Blog, Bio, Social_Media}

Feedback: BCNF

FD1: FeedbackID -> {Date, Comments, Email}

Intellectual_Product: BCNF

FD1: ProductID -> {Title, Description, Price, File Type, Accepted Payment, Download URL, Email, Transaction_Number}

IP_Image: BCNF

FD1: Image -> {Title}

Payment: BCNF

FD1: PaymentID -> {Date, Type, Email}

Purchase_History: BCNF

FD1: Transaction_Number -> {Date, Email, Amount Sold, PaymentID}

Certification: BCNF

FD1: Certification_Number -> {Institution, Rate, Date}

Seller_Photo: BCNF

FD1: Photo -> {Email}

4. Two views

View One:

Create a view that displays all the Paypal transaction in the database as well as total price:

$$R1 \leftarrow Payment \bowtie_{payment.paymentid=purchase_{Hisotry}.paymentid} Purchase\_History$$

$$R2 \leftarrow R1 \bowtie_{R1.transaction_{Number}=PH_{IP}.transaction\_Number} PH\_IP$$

$$R3 \leftarrow R2 \bowtie_{R2.productid=Intellectual_{Product}.productid} Intellectual\_Product$$

$$\pi_{TransactionNumber,\ Amount*Price}(\rho_{tpye='Paypal'}(R3))$$

```
CREATE VIEW Paypal_Purchase
AS SELECT PH.Transaction_Number, SUM(PH.Amount * IP.Price) AS Total
       FROM Purchase_History AS PH,
            Payment AS Pay,
            Intellectual_Product AS IP,
            PH_IP AS PI
       WHERE Pay.paymentid = PH.paymentid AND
                     pay.Type = 'Paypal'AND
```

```
                    PI.productid = IP.productid AND
                    PI.transaction_number = PH.transaction_number
    GROUP BY PH.Transaction_Number
```

Results:

Transaction_Number	Total
1546310	15931.039999999999
3905761	296.32
4905074	7500.960000000001
5156926	1108.8999999999999
6168862	1612.5
6950117	5166.36
8212396	7960.68
8580703	2718.3799999999997

View Two:

Create a view that displays the certification whose rate is above average as well as its product ID:

$$R1 \leftarrow \gamma AVG\ rate(Certification)$$

$$R2 \leftarrow Certification \bowtie_{Certification.certificationNumber=IntellectualProduct.certificationNumber} Intellectual\_Product$$

$$\pi_{rate,\ productid}(\rho_{rate>R1}(R2))$$

```
CREATE VIEW GoodCertification
AS SELECT C.rate, IP.productid
      FROM Certification AS C, Intellectual_Product as IP
      WHERE C.certification_number = IP.certification_number AND
                  C.rate > (SELECT AVG(C.rate)
                                 FROM Certification AS C)
```

Partial Results:

Rate	ProductID
3.4	97621584
3.8	13131683
4.9	96282540
3.5	95213183
3.7	87815876
4.3	86723500
4	90787186
3.8	53661036
4.6	42554156
4.5	78265256

5. **Two indexes**

```
Index on Intellectual Product:
CREATE INDEX IP_Index
ON Intellectual_Product (ProductID, Title, Email);
```

Rationale: This table will likely be accessed more frequently since it is the main component of the store. Additionally, it will not be updated as frequently as other tables so the index won't require as much maintenance. We used Title, ProductID, and Email for the indexing since they will likely be the most frequently queried columns.

```
Index on Person:
CREATE INDEX Person_Index
ON Person (Email);
```

Rationale: This table is important for being referenced by so many tables, such as buyer, seller, feedback, payment, etc. Since it is such a central table, speeding up access to it will have significant gains. It may require some more maintenance compared but we believe the increased access speed will be worthwhile. We used Email for the indexing since it will likely be by far the most queried since it is the foreign key.

6. **Two transactions**

This transaction adds a new user into the list of Buyers.

The unit of work for the transaction is two since the user is new and has to be added to the Person table as well. If an error is encountered the transaction rolls back.

```
BEGIN TRANSACTION;
      INSERT OR ROLLBACK INTO Person
      VALUES (“testemail@gmail.com”, “Bobby Tables”, 500);

      INSERT OR ROLLBACK INTO Buyer
      VALUES(“testemail@gmail.com”);
COMMIT;
```

This transaction adds a new item to the IP

The unit of work for the transaction is also one. If intellectual product is being added there must already be an associated user and so nothing else needs to be updated.

```
BEGIN TRANSACTION;
      INSERT OR ROLLBACK INTO Intellectual_Product
      VALUES (3829563, “lorem ipsum”, “item desc here”, 24.99,
“cad”, “PayPal”, “https://google.com/lorem-ipsum/dolor-sit/”,
“testemail@gmail.com”, 2850);
COMMIT;
```

Section 2 User Manual

1. **Explanation of each table**

Person

Represents personal information. A person's email is used as an identifier across the database. Karma is used for ?

Email	Name	Karma
varchar	varchar	int

Constraints:

- Primary Key on email. Email must be unique.

Buyer

An intermediate table in order to connect buyer specific relationships to a person.

Email
varchar

Constraints:

- Primary Key on email. Email must be unique.
- Foreign Key on email to Person.Email. (A buyer must be a person).

Seller

A table to store seller specific information, and to act as an intermediate seller specific relationships to a person. Blog and social_media should be url links to the seller's blog and social media profile respectively. The Bio should be a small description.

Email	Blog	Bio	Social_Media
varchar	varchar	varchar	varchar

Constraints:

- Primary Key on email. Email must be unique.
- Foreign Key on email to Person. (A seller must be a person).

Seller_Photo

Table to link a profile photo to a seller. Photos are optional, so a separate table is used to store them.

Photo	Email(FK)
varbinary	varchar

Constraints:

- Primary Key on email. Email must be unique.
- Foreign Key on email. Email must be an existing Seller.Email

Feedback

Feedback of sellers by buyers. Review system to allow buyers to express seller reputation.

FeedbackID	Date	Comment	BEmail(FK)	SEmail(FK)
int	datetime	varchar	varchar	varchar

Constraints:

- Primary Key on FeedbackID. FeedbackID must be unique.
- Foreign Key on BEmail to Buyer.Email.
- Foreign Key on SEmail to Seller.Email.

Payment

Payment information for a transaction.

PaymentID	Date	Type	Email(FK)
int	datetime	varchar	varchar

Constraints:

- Primary Key on PaymentId. PaymentID must be unique.
- Foreign Key on Email. Email must be a valid Buyer.email

Purchase_History

Transaction history of a customer purchase. A single transaction can include many purchased products, and an IP can be purchased by multiple purchases. PH_IP acts as the intermediate table to enable the many to many relationship and is used to construct an entire order.

Transaction_Number	Date	Amount	Email(FK)	PaymentID(FK)
varchar	datetime	int	varchar	int

Constraints:

- Primary Key on Transaction_Number. Must be unique.
- Foreign Key on Email. Email must be valid Buyer.Email.
- Foreign Key on PaymentID. PaymendID must be a valid Payment.PaymentID

Intellectual_Product

Data particular to an intellectual product. Includes item description, price, type, accepted payment, url, and certification/seller references.

ProductID	Title	Description	Price	FileType
int	varchar	varchar	float	varchar

AcceptedPayment	DownloadURL	Email(FK)	Certification_Number(FK)
varchar	varchar	varchar	int

Constraints:

- Primary Key on ProductID. Must be unique.
- Foreign Key on Email. Must be a valid Seller.Email.
- Foreign Key on Certification_Number. Must be a valid Certification.Certification_Number.

IP_Image

Table for pictures of intellectual property. Many photos are associated with 1 product.

Image	ProductID(FK)
varbinary	int

Constraints:

- Primary Keys on Image and ProductID. Must be unique pairs.
- Foreign Key on ProductID. Must be a valid Intellectual_Product.ProductID.

PH_IP

Intermediate table to establish a many to many relationship between Purchase History and Intellectual Products. A transaction can have many products purchased. All entries with the same transaction number can be considered part of a single "purchase."

Transaction_Number(FK)	ProductID(FK)
int	int

Constraints:

- Primary Key on both Transaction_Number and ProductID. Pairs must be unique.
- Foreign Key on ProductID. Must be a valid Intellectual_Property.ProductID.
- ForeignKey on Transaction_Number. Must be a valid Purchase_History.Transaction_Number.

Certification

Describes certification for an intellectual product. Gives the certifying institution, the score, and date. There can be many certifications for one product.

Certification_Number	Institution	Rate	Date
varchar	varchar	int	datetime

Constraints:

- Primary Key on Certification_Number. Must be unique.

2. Sample query from CK02 & CK03

Simple Query

A)

(i) Find the titles of all IP Items by a given Seller that cost less than $10 (you choose how to designate the seller)

(ii) $R1 \leftarrow Intellectutal\_Product \bowtie_{Intellectual\_Product.Email=Person.Email} Person$

$R2 \leftarrow R1 \bowtie_{R1.Email=Seller.Email} Seller$

$R3 \leftarrow \sigma_{Price < 10} R2$

$\pi_{Title} R3$

(iii)
```
SELECT IP.title
FROM Intellectual_Product AS IP, Person AS P, Seller
WHERE IP.Email= P.Email
      AND P.Email = Seller.email
      AND IP.Price <10;
```

B)

(i) Give all the titles and their dates of purchase made by given buyer (you choose how to designate the buyer)

(ii) $R1 \leftarrow Person \bowtie_{Person.Email=Buyer.Email} Buyer$

$R2 \leftarrow Payment \bowtie_{Payment.Email=R1.Email} R1$

$R3 \leftarrow \sigma_{Buyer.Email = 'skedwell3@tmall.com'} R2$

$R4 \leftarrow Intellectual\_Product \bowtie_{IP.Email=R3.Email} R3$

$\pi_{Title, Payment.Date} R4$

(iii)
```
SELECT IP.Title, Payment.Date
From Intellectual_Product AS IP, Payment, Person, Buyer
WHERE IP.Email = Person.Email
      AND Payment.Email = Buyer.email
      AND Payment.Email = Person.email
      AND Buyer.email = "skedwell3@tmall.com";
```

C)

(i) Find the seller names for all sellers with less than 5 IP Items for sale

(ii) $R1 \leftarrow Intellectual\_Product \bowtie_{IP.Email=Person.Email} Person$

$R2 \leftarrow {}_{Email,\,Name}\gamma_{SUM(IP.Title)} R1$

$R3 \leftarrow \sigma_{sum\_title<5} R2$

$\pi_{Name} R3$

(iii)
```
SELECT P.Name
FROM   Intellectual_Product AS IP, Person AS P
WHERE  IP.Email = P.Email
GROUP BY IP.Title
HAVING SUM(IP.Title) <5;
```

D)

(i) Give all the buyers who purchased an IP Item by a given seller and the names of the IP Items they purchased

(ii) $R1 \leftarrow Purchase\_History \bowtie_{PH.Email=Person.Email} Person$
$R2 \leftarrow R1 \bowtie_{R1.Transaction\_Number=PH\_IP.Transaction\_Number} PH\_IP$
$R3 \leftarrow Intellectual\_Product \bowtie_{IP.ProductID=R2.ProductID} R2$

$\pi_{Person.Name,\,IP.Title} R3$

(iii)
```
SELECT Person.Name, IP.Title
FROM Intellectual_Product AS IP, Person, Purchase_History AS
PH, PH_IP
```

```
WHERE PH.Email = Person.Email
      AND PH.Transaction_Number = PH_IP.Transaction_Number
      AND PH_IP.ProductID = IP.ProductID;
```

E)

(i) Find the total number of IP Items purchased by a single buyer (you choose how to designate the buyer)

(ii) $R1 \leftarrow Purchase\_History \bowtie_{PH.Email=Person.Email} Person$

$R2 \leftarrow R1 \bowtie_{R1.Email=Buyer.Email} Buyer$

$\gamma_{COUNT(R3.Transaction\_Number)} R2$

(iii)
```
SELECT COUNT(PH.transaction_number)
FROM   Person AS PB, Purchase_History AS PH, Buyer
WHERE  PB.Email = PH.Email AND
       PB.Email = Buyer.Email
```

F)

(i) Finds the buyer who has purchased the most IP Items and the total number of IP Items they have purchased

(ii) $R1 \leftarrow Purchase\_History \bowtie_{PH.Email=Person.Email} Person$

$R2 \leftarrow_{Email,\ Name} \gamma_{SUM(PH.Amount)} R1$

$R3 \leftarrow_{Email,\ Name} \gamma_{MAX(R3.sumAmount)} R2$

$\pi_{Name,\ max\_sum\_amount} R3$

(iii)
```
SELECT MAX(TotalItem), Name
FROM
 (SELECT SUM(PH.Amount) AS TotalItem, P.Name AS Name
  FROM Purchase_History AS PH, Person AS P
  WHERE PH.Email= P.Email
 );
```

Advanced query

A)

(i) Provide a list of buyer names, along with the total dollar amount each buyer has spent.

(ii) $$Name\gamma(AVG(Price), Name)(PurchaseHistory \bowtie_{PH.Email=P.Email} Person)$$

(iii)
```
SELECT P.Name, SUM(IP.Price)

FROM Person AS P, Purchase_History AS PH,
Intellectual_Product AS IP

WHERE P.Email = PH.Email

GROUP BY P.Name;
```

B)

(i) Provide a list of buyer names and e-mail addresses for buyers who have spent more than the average buyer.

$$R1 \leftarrow Name\gamma(SUM(price * amount))(PurchaseHistory \bowtie_{PH.Email=P.Email} Person)$$

$$R2 \leftarrow \gamma AVG(R1)$$

(ii) $$\pi_{name,email}(\rho_{R1.sum(price*amount)>R2.avg(sum(price*amount))}(R1))$$

(iii)
```
SELECT P.Name, P.Email

FROM Person AS P, Purchase_History AS PH,
Intellectual_Product AS IP

WHERE P.Email = PH.Email

GROUP BY P.Name

HAVING SUM(IP.Price*PH.Amount) >

          (SELECT AVG(Price)

           FROM

             (SELECT SUM(IP.Price*PH.Amount) As Price
```

```
                    From  Person AS P, Purchase_History AS PH,
.                   Intellectual_Product AS IP
                    WHERE P.Email = PH.Email
                    GROUP BY P.Name)
                )
```

C)

(i) Provide a list of the IP Item names and associated total copies sold to all buyers, sorted from the IP Item that has sold the most individual copies to the IP Item that has sold the least.

(ii) $R1 \leftarrow Intellectual\_Product \bowtie_{IP.ProductID=PH\_IP.ProductID} PH\_IP$

$R2 \leftarrow Purchase\_History \bowtie_{PH.Transaction\_Number=PH\_IP.Transaction\_Number} PH\_IP$

$R3 \leftarrow {}_{ProductID}\gamma_{SUM(R2.Amount)} R2$

$R4 \leftarrow R1 \bowtie_{R1.ProductID=R3.ProductID} R3$

$R5 \leftarrow \pi_{Title,\ Sum\_Amount} R4$

(iii)
```
SELECT IP.Title, SUM(PH.Amount)
FROM Purchase_History AS PH, Intellectual_Product AS IP,
PH_IP
WHERE IP.ProductID = PH_IP.ProductID
      AND PH.Transaction_Number = PH_IP.Transaction_Number
GROUP BY IP.Title
ORDER BY SUM(PH.Amount) DESC
```

D)

(i) Provide a list of the IP Item names and associated dollar totals for copies sold to all buyers, sorted from the IP Item that has sold the highest dollar amount to the IP Item that has sold the smallest.

(ii) $R1 \leftarrow Intellectual\_Product \bowtie_{IP.ProductID=PH\_IP.ProductID} PH\_IP$

$R2 \leftarrow Purchase\_History \bowtie_{PH.Transaction\_Number=PH\_IP.Transaction\_Number} PH\_IP$

$R3 \leftarrow {}_{ProductID}\gamma_{SUM(R2.Amount)} R2$

$R4 \leftarrow R1 \bowtie_{R1.ProductID=R3.ProductID} R3$

$R5 \leftarrow {}_{Title}\gamma_{Sum\_Amount * R4.Price} R4$

(iii)
```
SELECT IP.Title, SUM(PH.Amount)*IP.Price
FROM Purchase_History AS PH, Intellectual_Product AS IP,
PH_IP
WHERE IP.ProductID = PH_IP.ProductID
      AND PH.Transaction_Number = PH_IP.Transaction_Number
GROUP BY IP.Title
ORDER BY SUM(PH.Amount)*IP.Price DESC
```

E)

(i) Find the most popular Seller (i.e. the one who has sold the most IP Items)

(ii) $R1 \leftarrow Intellectual\_Product \bowtie_{IP.ProductID=PH\_IP.ProductID} PH\_IP$

$R2 \leftarrow Purchase\_History \bowtie_{PH.Transaction\_Number=PH\_IP.Transaction\_Number} PH\_IP$

$R3 \leftarrow Seller \bowtie_{Seller.Email=Person.Email} Person$

$R4 \leftarrow Intellectual\_Product \bowtie_{IP.Email=Person.Email} Person$

$R5 \leftarrow R4 \bowtie_{R4.Email=R5.Email} R5$

$R6 \leftarrow R5 \bowtie_{R5.ProductID=R1.ProductID} R1$

$R7 \leftarrow R6 \bowtie_{R6.ProductID=R2.ProductID} R2$

$R8 \leftarrow {}_{Email,\ Transaction\_Number}\gamma_{SUM(R2.Amount)} R7$

$R9 \leftarrow {}_{Email}\gamma_{SUM(sum\_ammount)} R8$

$R10 \leftarrow {}_{Email}\gamma_{MAX(TOTAL.sum\_ammount)} R9$

```
SELECT Email, MAX(Total)
FROM(
     SELECT SUM(total) AS Total, Email
     FROM(
        SELECT S.email, PH.transaction_number, PH.amount as
.       total
        FROM Purchase_History as PH, Intellectual_Product as
                IP, Person AS P, Seller AS S, PH_IP as PI
```

```
            WHERE PI.Transaction_Number = PH.transaction_number
                    AND PI.productid = IP.productid
                    AND IP.email = P.email
                    AND S.email = P.email
            GROUP by PH.transaction_number )
    GROUP BY Email)
```

F)

(i) Find the most profitable seller (i.e. the one who has brought in the most money)

(ii) $R1 \leftarrow Intellectual\_Product \bowtie_{IP.ProductID=PH\_IP.ProductID} PH\_IP$

$R2 \leftarrow Purchase\_History \bowtie_{PH.Transaction\_Number=PH\_IP.Transaction\_Number} PH\_IP$

$R3 \leftarrow Seller \bowtie_{Seller.Email=Person.Email} Person$

$R4 \leftarrow Intellectual\_Product \bowtie_{IP.Email=Person.Email} Person$

$R5 \leftarrow R4 \bowtie_{R4.Email=R2.Email} R2$

$R6 \leftarrow {}_{Email,PH.Transaction\_Number,}\gamma_{SUM(IP.Price*PH.Amount)} R5$

$R7 \leftarrow {}_{Email}\gamma_{SUM(Total)} R6$

$R7 \leftarrow {}_{Email}\gamma_{Max(Total)} R7$

(iii)

```
SELECT Email, MAX(Total)
FROM(
    SELECT SUM(total) AS Total, Email
    FROM(
         SELECT S.email, PH.transaction_number,
                (SUM(IP.price * PH.amount)) as total
         From Purchase_History as PH, Intellectual_Product
              as IP, Person AS P, Seller AS S, PH_IP as PI
```

```
            WHERE PI.Transaction_Number =
                  PH.transaction_number AND
                  PI.productid = IP.productid AND
                  IP.email = P.email AND
                  S.email = P.email
            GROUP by PH.transaction_number )
      GROUP BY Email)
```

G)

(i) Provide a list of buyer names for buyers who purchased anything listed by the most profitable Seller.

(ii) $R1 \leftarrow Intellectual\_Product \bowtie_{IP.ProductID=PH\_IP.ProductID} PH\_IP$

$R2 \leftarrow Purchase\_History \bowtie_{PH.Transaction\_Number=PH\_IP.Transaction\_NumberEmail} PH\_IP$

$R3 \leftarrow R1 \bowtie_{R1.ProductID=R2.ProductID} R2$

$R4 \leftarrow R3 \bowtie_{IP.Email=Person.Email} Person$

$SELLER \leftarrow Seller \bowtie_{Seller.Email=R4.Email} R4$

$SellerProducts \leftarrow SELLER \bowtie_{SELLER.ProductID=IP.ProductID} Intellectual\_Product$

$SellerTotalSold \leftarrow {}_{Email}\gamma_{SUM(SellerProducts.Amount)} SellerProducts$

$MaxSold \leftarrow {}_{Email}\gamma_{MAX(sum\_amount)} SellerTotalSold$

$R5 \leftarrow R4 \bowtie_{R4.Email=MaxSold.Email} MaxSold$

$\pi_{Name} R5$

(iii)

```
SELECT P.Name
FROM Person AS P, Purchase_History AS PH,
Intellectual_Product AS IP, PH_IP
WHERE P.Email = PH.Email
      AND PH_IP.ProductID = IP.productid
      AND PH.Transaction_Number = PH_IP.Transaction_Number
```

```
     AND IP.Email =
(SELECT email
 FROM (SELECT Email, MAX(Total)
       FROM(
              SELECT SUM(total) AS Total, Email
              FROM(
                   SELECT S.email, PH.transaction_number,
                          PH.amount as total
                   From Purchase_History as PH,
                        Intellectual_Product as IP,
                        Person AS P, Seller AS S, PH_IP
                        as PI
                   WHERE PI.Transaction_Number =
                         PH.transaction_number AND
                         PI.productid = IP.productid AND
                         IP.email = P.email AND
                         S.email = P.email
                   GROUP by PH.transaction_number )
             GROUP BY Email)))
```

H)

(i) Provide the list of sellers who listed the IP Items purchased by the buyers who have spent more than the average buyer.

(ii) $R1 \leftarrow Intellectual\_Product \bowtie_{IP.ProductID=PH\_IP.ProductID} PH\_IP$

$R2 \leftarrow Purchase\_History \bowtie_{PH.Transaction\_Number=PH\_IP.Transaction\_NumberEmail} PH\_IP$

$R3 \leftarrow R1 \bowtie_{R1.ProductID=R2.ProductID} R2$

$SellerHistory0 \leftarrow R3 \bowtie_{IP.Email=Person.Email} Person$

$SellerHistory \leftarrow SellerHistory0 \bowtie_{person\_Email=Seller.Email} Seller$

$BuyerHistory0 \leftarrow R3 \bowtie_{PH.Email=Person.Email} Person$

$$BuyerHistory \leftarrow BuyerHistory 0 \bowtie_{Person.Email=Buyer.Email} Buyer$$

$$BuyerSpent \leftarrow_{Email} \gamma_{SUM(Ip.Price * PH.amount)} BuyerHistory$$

$$AvgBuyerSpent \leftarrow \gamma_{Avg(sum\_spent)} BuyerSpent$$

$$R4 \leftarrow BuyerSpent \bowtie_{BuyerSpent.sum\_spent > AvgBuyerSpent.avg\_sum\_spent} AvgBuyerSpent$$

$$AboveAverageSpent \leftarrow R4$$

$$SellersWithAboveAverageCustomers \leftarrow SellerHistory \bowtie_{SellerHistory.PH\_Email = R4.Email} R4$$

(iii)

```
SELECT sellerEmail
      FROM (SELECT Buyer.email AS buyerEmail, IP.email as
                  sellerEmail
            FROM Purchase_History as PH, PH_IP,
                 Intellectual_Product AS IP, Person AS PB,
                 Buyer
            Where PH.transaction_number =
                  PH_IP .transaction_number AND
                  IP.ProductID = PH_IP.productid AND
                  PH.email = PB.email AND
                 PB.email = Buyer.email
           GROUP BY Buyer.email
    HAVING SUM(IP.price * PH.amount) >
     (SELECT AVG(total)
     FROM( SELECT Buyer.email, SUM(IP.price * PH.amount)
                 AS total
           FROM Purchase_History as PH, PH_IP,
                Intellectual_Product AS IP, Person AS P,
                Buyer
           WHERE PH.transaction_number =
                 PH_IP .transaction_number AND
                  IP.ProductID = PH_IP.productid AND
                 PH.email = P.email AND
```

```
                P.email = Buyer.email
        GROUP BY Buyer.email)))
```

Extra query

A)

i) A List of all sellers and everything that they sell.

ii) $\pi_{Name,title}(Person \bowtie_{Person.Email=IP.Email} Intellectual\_Product)$

iii)
```
SELECT P.Name, IP.Title
FROM Person AS P, Intellectual_Product AS IP
WHERE P.Email = IP.email
GROUP by P.Name
```

B)

1) The seller name whose average IP price is greater than 80
2) $R1 \leftarrow Person.Name \curlyvee AVG(Price)(Person \bowtie_{Person.Email=IP.Email} Intellectual\_Product)$
3) $\pi_{Name,AVG(Price)}(\rho_{AVG(Price)>80}(R1))$

```
SELECT P.Name, avg(IP.Price)
FROM Person AS P, Intellectual_Product AS IP
WHERE P.Email = IP.Email
GROUP BY P.Name
HAVING avg(IP.Price) > 80
```

C)

i) Select all purchases made with Paypal

ii) $\pi_{transactionNumber}(\rho_{type='paypal'}(PurchaseHistory \bowtie_{ph.paymentID=p.paymentID} Payment))$

iii)
```
SELECT PH.Transaction_Number
```

```
FROM Purchase_History AS PH, Payment
WHERE PH.PaymentID = Payment.PaymentID
     AND Payment.Type  = 'PayPal';
```

3. Insert samples

1. Insert a new seller

   need to preserve referential integrity, since Seller has Person email as foreign key.

```
INSERT INTO Person
VALUES('yao.578@osu.edu', 'zhijian', 600);

INSERT into Seller
VALUES('yao.578@osu.edu', 'blog', 'bio', 'socialmedia');
```

2. Insert a new intellectual product

   need to preserve referential integrity, since IP has Certification number as foreign key.

```
INSERT INTO Certification
VALUES ('8888', 'OSU', 4.5, '12/3/2019');

INSERT INTO Intellectual_Product
VALUES ('88888888', 'IP', 'example', 12.5, 'pdf', 'Paypal',
        'URL', 'yao.578@osu.edu', '8888');
```

3. Insert a new buyer

```
INSERT INTO Person
VALUES('yao.777@osu.edu', 'Allen', 666);

INSERT INTO Buyer
VALUES('yao.777@osu.edu');
```

4. Insert a new order

   first insert payment, then insert purchase history last insert PH_IP to preserve the referential integrity

```
INSERT into Payment
VALUES('666666', '12/3/2019', 'Paypal', 'yao.777@osu.edu');

INSERT INTO Purchase_History
VALUES('7777777', '12/3/2019', 1, 'yao.777@osu.edu', '666666');

INSERT INTO PH_IP
VALUES('7777777', '88888888');
```

4. **Delete samples**

1. Delete a order

The order of deleting is the opposite sequence of inserting

```
DELETE FROM PH_IP
WHERE transaction_number = '7777777';

DELETE from Purchase_History
WHERE transaction_number = '7777777';

DELETE FROM Payment
WHERE paymentid = '666666';
```

2. Delete an intellectual product

```
DELETE FROM Intellectual_Product
WHERE productid = '88888888';

DELETE FROM Certification
WHERE certification_number = '8888';
```

3. Delete a seller

```
DELETE from Seller
WHERE email = 'yao.578@osu.edu';

DELETE FROM Person
WHERE email = 'yao.578@osu.edu';
```

4. Delete a buyer

```
DELETE from Buyer
WHERE email = 'yao.777@osu.edu';

DELETE FROM Person
WHERE email = 'yao.777@osu.edu';
```

Delete the following entities that are not covered in the previous delete examples:

5. Delete seller photo

```
DELETE FROM Seller_Photo
WHERE email = 'stiron6@ihg.com';
```

6. Delete feedback

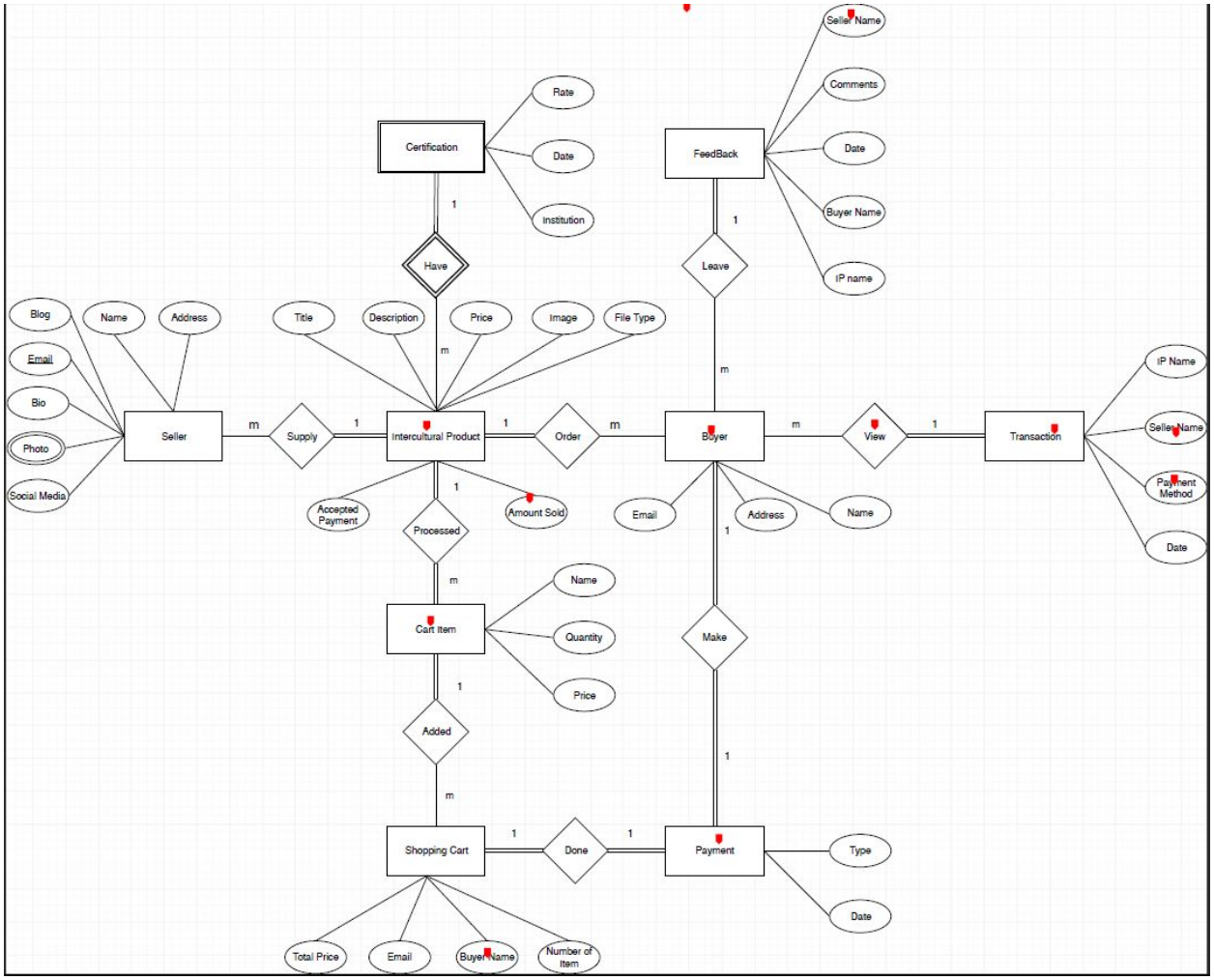
```
DELETE FROM Feedback
WHERE feedbackid = '99825';
```

7. Delete IP image

```
DELETE FROM IP_Image
WHERE productid = '97621584';
```

Section 3 Graded Checkpoint Documents

1. **Check Point 1**
   Original Submission:

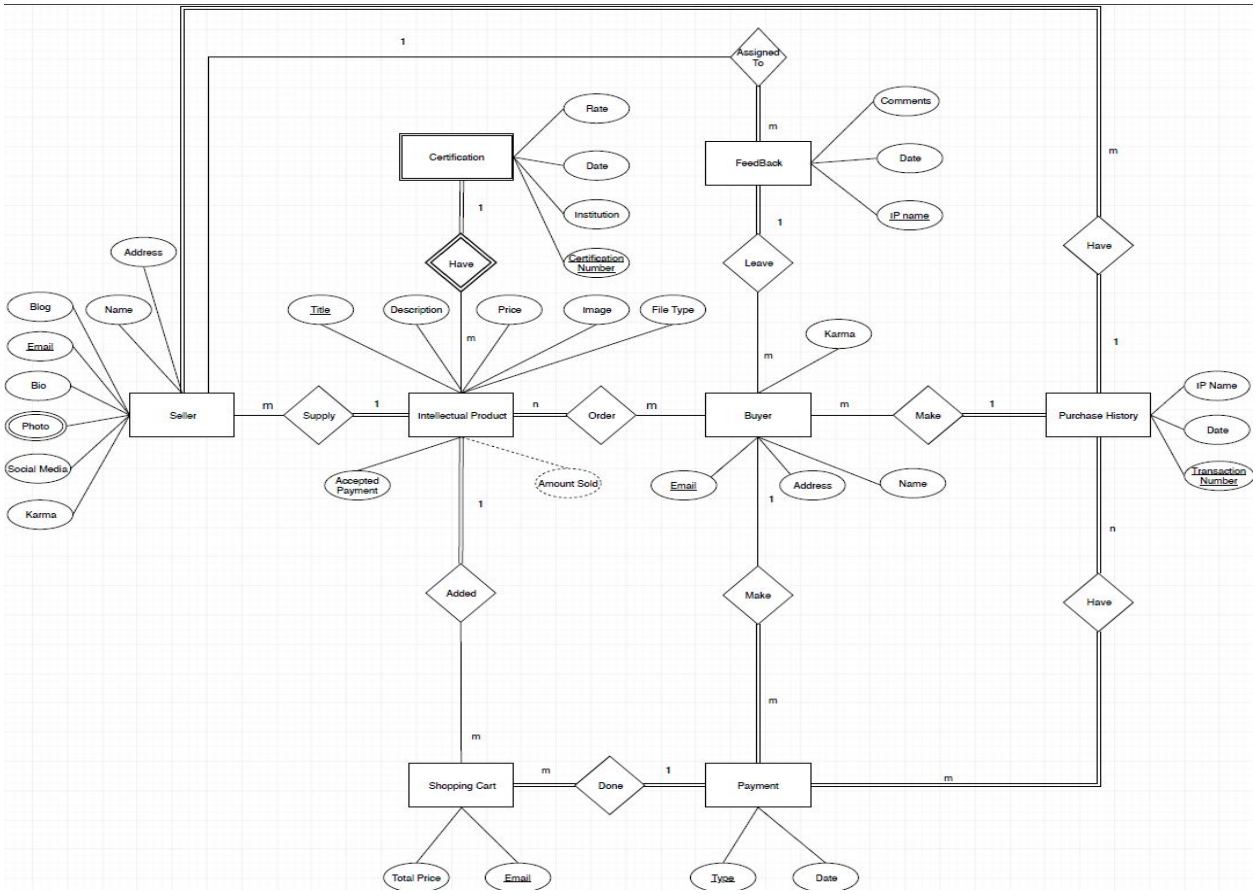


Grader Feedback:

1. You don't wanna have the email and Buyer name attributes in a Shopping cart. Those are foreign attributes and ER diagrams don't have such a concept (only relationships can be used to do something like that)
2. Shopping Cart -> Payment -> Buyer all seem to have total one-to-one relationships. In your relational schema these will all end up being one table, so you want to consider refactoring that.
3. You probably should not have a Cart Item entity, it is storing the same information as an IP (with Quantity as an extra attribute), and adding attributes to this entity will result in redundancy.
4. Amount Sold should be a derived attribute - it can be derived by running queries on Transactions/Payment entities.

5. Payment method seems to be the same thing as the type attribute of a Payment entity. You guys should definitely rethink the dynamics of how this should work.
6. Seller name is a foreign attribute of Seller. This should not be done in ER diagrams, you can only represent constructs like these using relationships here. Having said that, you guys should consider having a relationship between Seller and Transaction.
7. Since this is a solely online platform, you guys might want to have a karma attribute for Buyer and/or Seller.
8. It would make more sense if Transactions also have some relationships with other entities such as Payment. To me it seems intuitive that those are directly related.
9. intellectual*
10. Again, view wouldn't be a great relationship, you guys might wanna change that to something like "make".
11. Feedback should have a relationship with Seller, and not have a Seller name attribute. ERDs do not have a concept of foreign key.
12. Again, you guys MUST identify what primary key(s) each entity in your model has. I don't see any except for Seller email.

Our team resolved the issues raised by the grader through the following diagram:

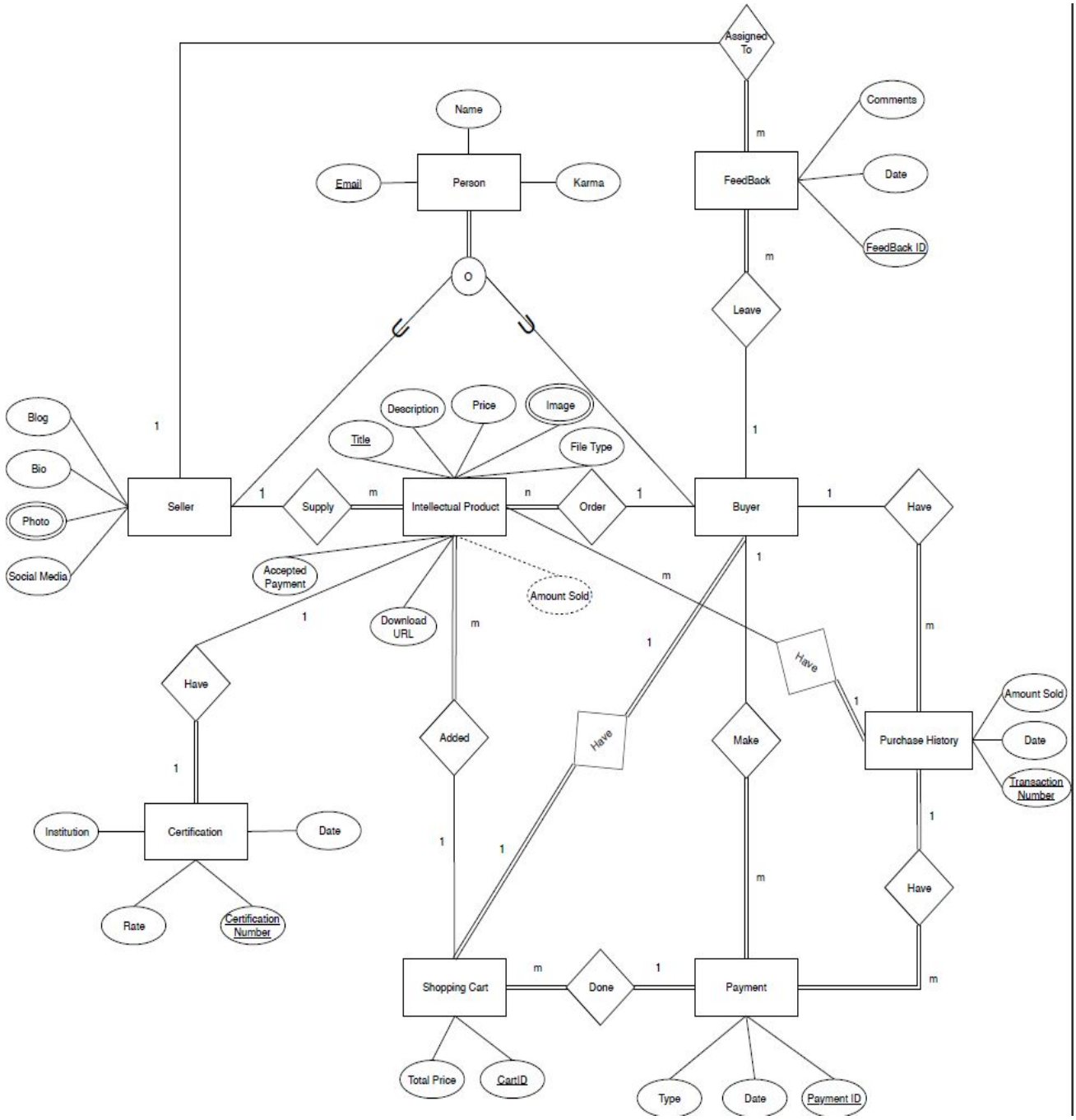


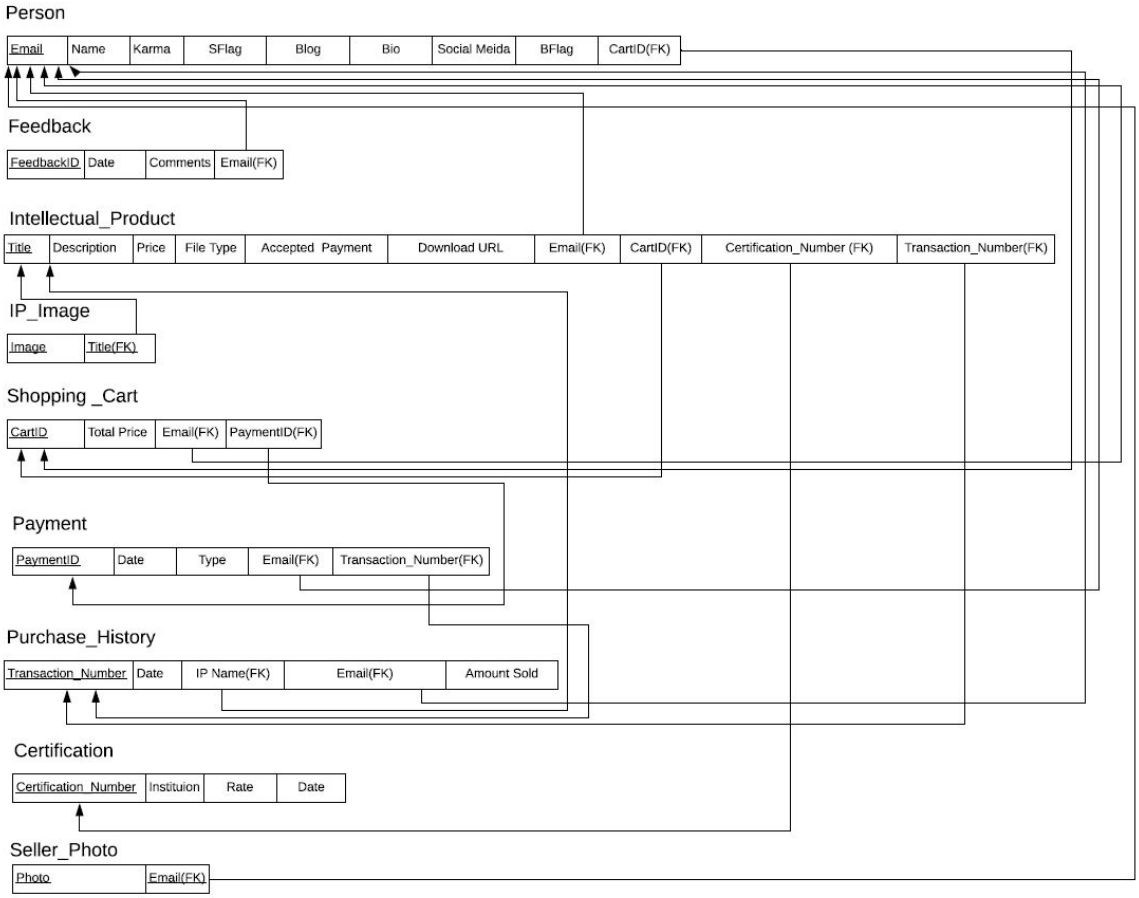
2. **Check Point 2**

   Grader's feedback based on the previous diagram we submitted:

   a. I guess I didn't notice this in the last checkpoint, but you guys mixed up the cardinalities for all relationships. Here, Seller can supply only one IP, and an IP can be supplied by many sellers. It should probably be the other way around. (you guys got some of them right tho, like Feedback <-> Seller)
   b. Certification need not be a weak entity because Certification number is a primary key.
   c. IP name seems like a foreign key, Feedback should not have it as an attribute in the ERD. Instead, you might wanna consider making it a weak entity of Buyer and Seller, or think of another unique attribute.
   d. Perhaps having the physical address of a person does not make sense for this project, since everything is done completely online and physical address doesn't matter
   e. Here you guys have Title as the primary key for IP, but the rest of your checkpoint doesn't reflect that, you also seem to have IpID in the rest of the questions
   f. Image must be a multivalued attribute - upto 5 images should be supported
   g. You guys must have a hierarchy of Person - with Buyer and Seller subclasses, as a total and overlapping specialization.
   h. Each Intellectual Product should have an attribute which stores the download URL for it, otherwise there's no way anyone can access it.
   i. Again, Email is a foreign key here and you might wanna add a relationship between Buyer and Shopping Cart.

Our team resolved the issues raised by the grader through the following diagram:





3. Check Point 3

Grader's feedback based on the previous checkpoint we submitted:

a. ER diagram looks mostly good. However, I feel like there's some redundancy with the triangular relationship between Buyer, Payment, Purchase History, and Shopping cart. I'd suggest removing the Purchase History entity, Buyer-has-ShoppingCart relationship, and make the Payment-done-ShoppingCart relationship one to one (because I assume each time you make a payment it's for only one shopping cart). At the minimum, considering removing at least one of ShoppingCart or Purchase History. It seems like you guys are storing a "singleton" shopping cart for each user, but I think shopping carts are a temporary and not persistent storage (even if they were persistent, they could be stored as Cookies on the client side).

Our solution is to delete the entity ShoppingCart and it does not affect the funcionality of the database.

b. Relational Schema looks decent for the existing ERD, you will have to make changes after updating the ERD. Person and Shopping cart are one-to-one, so they should be combined if you decide to keep the Shopping cart relation. I could not run the SQL create statement file (CK3 SQL Code.txt) on SQLiteOnline, but I was able to run the other insert file (SQLFile.sql).

There are few syntax errors in the CK3 SQL Code.txt file. Our team have corrected them and this file is named as creation source code in the submission file.

c. It also seems like there are minor typos in the queries/insert statements. For example, the SFlag column is misspelled as SFalg, which resulted in the queries failing. Also, Intellectual_Product was written as Intellectual Product (without the underscore), so I had to fix them myself. Bettina will not give you a good grade for the final project if the queries do not run right off the bat, even if they look correct.

d. Same as above for ExtraQueries.txt. For example, the second query seems to be referencing a Price table, which does not exist, you guys must fix that.

e. Same as above for AdvancedQueries.txt. There are too many typos, and even after trying to fix them, I wasn't able to run them. Overall, even though the queries looked good at first glance, it seems like there are too many errors (the most important one being the fact that there are no underscores where they are supposed to be). You guys need to re-do most of the SQL queries, test all of them on your database, and then turn them in. Also, you guys should try to have queries that give results on your database, that is, not return empty data. For example, for the first SimpleQuery, change Seller's email to a real email in your database (something when run will show results).

For feedback c,d,e, we have made the correction and please refer to the user manual section.

4. Check Point 4

Grader's feedback based on the previous checkpoint we submitted:
1) ER Diagram and Relational Schema look same as Checkpoint 3, you guys might want to look at Checkpoint 3's feedback.

Due to late submission of check point 3, our team does not have enough time to go over the errors raised by grader. The correction is made and please refer to the ERD and relational schema in section 1.

2, 3, 4) - Social Media or Blog might not be the best determinant for Person, what if they want to change their social media link? Email on the other hand, can be fixed as a unique constant for each new user. - Similarly, for Intellectual Product, Title might not be the best determinant. - For Follows, you can alternatively write {BuyerEmail, SellerId} -> {BuyerEmailm SellerId} - For each relation, Bettina expects you to provide a short and specific rationalization of why the relation is in that normal form (For example "all of the FDs have the whole PK as the determinant, so the table is in BCNF").

The correction is made and please refer to the normal form in section 1.

5) Views look good. For the first one however, you wanna do SUM(PH.Amount_Sold * IP.Price) instead of just (PH.Amount_Sold * IP.Price). It won't work if you don't have an aggregate.

The correction is made and please refer to the views in section 1.