**CHECKPOINT 1:**

In a NEATLY TYPED document, provide the following:

1. List names of all your team members. Provide a paragraph explaining how you have been working and plan to work as a team under remote setup, how you plan to communicate with each other, share work, etc. Any issues related to time differences, technology constraints, or any other issues/concerns?

Michael Yamokoski, Daniel Valz, Yusuf Abdirahman, Madeline Woodhull

Under a remote setup, we have been communicating through GroupMe, and using Discord for virtual meetings and sharing work. Documents are being shared on Google Docs.

2. Based on the requirements given in the project overview, list the entities to be modeled in this database. For each entity, provide a list of associated attributes. Make sure that your design allows for proper handling of buyer/seller interactions such as orders, payments, feedback, and karma points.

CUSTOMER - ID, preferred\_payment, email\_address

SELLER - karma\_Points, seller\_id

STOREFRONT - store\_ID, name, description, URLs, seller\_bio, banner, seller\_photo

PRODUCT - product\_ID, Price, keywords, images, description, title, file type

ORDER- order\_ID, order\_Price, date

PAYMENT\_TYPE - method, description

FEEDBACK - feedback\_ID, comment, stars

3. Based on the requirements given in the project overview, what are the various relationships between entities? (For example, “A CUSTOMER places an ORDER”).

CUSTOMER places an ORDER

PRODUCT is ordered by ORDER

STOREFRONT shows PRODUCTS

SELLER sells PRODUCTS

STOREFRONT is front for SELLER

CUSTOMER supplies PAYMENT\_TYPE

ORDER has PAYMENT\_TYPE

ORDER has FEEDBACK

CUSTOMER receives FEEDBACK

SELLER receives FEEDBACK

PRODUCT receives FEEDBACK

4. Propose at least two additional entities that it would be useful for this database to model beyond the scope of the project requirements. Provide a list of possible attributes for the additional entities and possible relationships they may have with each other and the rest of the entities in the database. Give a brief rationale for why adding these entities would be interesting/useful to the stakeholders for this database project.

WISHLIST - list\_ID

* WISHLIST contains PRODUCTs
* WISHLIST is wanted by CUSTOMER

SERVER - server\_ID, capacity

* STOREFRONT is hosted by SERVER

5. Give at least four examples of some informal queries/reports that it might be useful for this database to support. Include one example for each of the additional entities you proposed in question 4 above.

Sales report for different storefronts

Purchase history for a particular user

Feedback report for sellers

Wishlist - what are commonly wished for items

Order history - what items were commonly bought together

Server analytics - which storefronts required the most resources

6. CROSS-CHECK 1: Suppose we want to add a new IP Item to the database. How would we do that given the entities and relationships you have outlined above? Is it possible to add up to five images for the IP Item? Is it possible for the Buyer to pay using more than one payment method such as Karma points and credit card payment combined? Is it possible for the Buyer to purchase multiple IP Items from multiple Sellers at the sometimes part of one order? Can a Buyer leave feedback/rating for an item purchased from a particular Seller/Store? Explain how your model supports these possibilities. If it does not, make changes that allow your design to support all these requirements.

To add a product you would need to add it to the SELLER, which would just mean storing it on the SELLER’s side. Buyers are able to store more than one payment type and use them on orders containing products from more than one seller, leaving feedback for each item and seller, which we have accounted for in our cardinalities and limitations.

7. CROSS-CHECK 2: Refer to the Project Overview document, Main DB functionalities section. Confirm that your model supports ALL listed minimum requirements listed in the table. If it does not, make changes that allow your design to fully support all listed requirements.

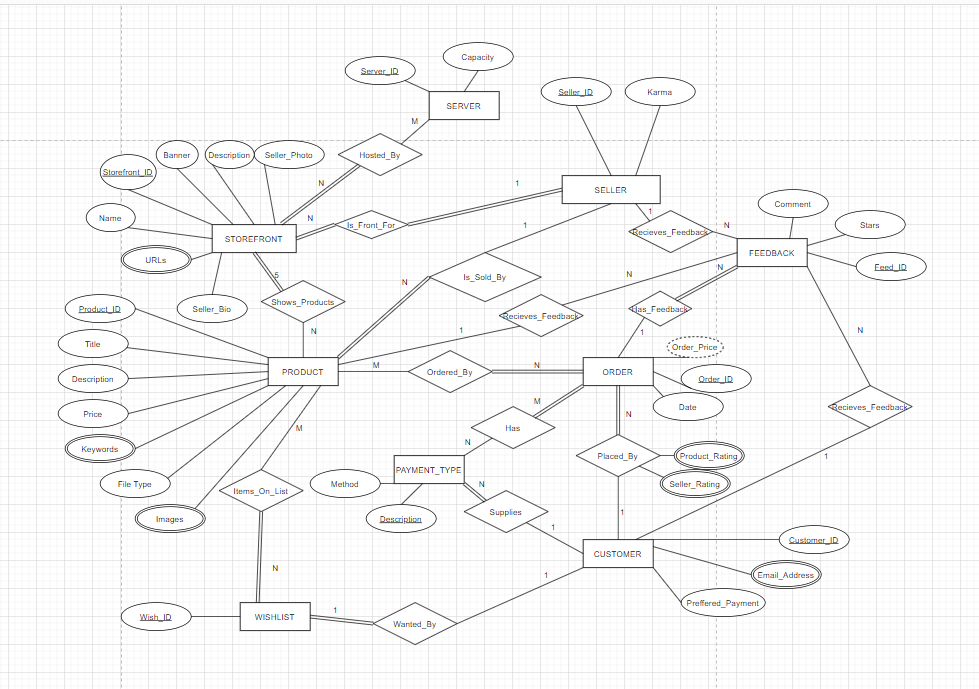
8. Determine at least three other informal update operations and describe what entities would need to have attributes altered and how they would need to be changed given your above descriptions. Include one example for each of the additional entities you proposed in question 4 above. ‘Informal 'means stated in a plain sentence format. ‘Update 'includes making changes, adding, or deleting data to your database that may involve one or more entities.

Adding or removing payment types for the customer. The payment entity would alter the type and info, based on the customer’s supplied information.

Adding a storefront. The storefront would need to be associated with a particular seller, and must have at least 1 product ready listed on it when it is added to the system.

Changing IP products price. The product entity’s price attribute would have to be changed at the seller’s discretion.

9. Provide an ER diagram for your database. Make sure you include ALL the entities and relationships you determined in the questions above INCLUDING the entities for question 4 above. Remember that (E)ER model cannot have any standalone entities. Ensure that you use a proper notation and include a legend. Each entity must have a proper key. All relationships must have cardinality and participation shown. Be mindful about using weak entities. Check for presence and correct identification of all attributes. You can use draw.io for your diagram or other drawing tools. Hand drawn diagrams will not be accepted.



**CHECKPOINT 1 FEEDBACK:**

**Notice: Grader comments will not be exhaustive; it is important to maintain communication with myself and professor Pichkar to ensure that all questions and issues are resolved before the due date of the project.**

**Team 12:**  
Yusuf Abdirahman  
Mike Yamokoski  
Madeline Woodhull  
Daniel Valz

3. Relationships to make sure you’ve considered:

* Buyers purchasing items, from any number of stores
* Sellers listing the items in their stores
* Buyer giving an IP Item a review

ERD:

* Buyer and seller should have overlapping subclasses with a user superclass
* Storefront should have multivalued images attribute as well
* Payment method should be a regular entity with disjoint subclasses (karma, crypto, CC, and bank account)

**This is a solid start, make sure to stay in contact with questions or concerns, as we don’t always catch every required change in the diagram**

**CHECKPOINT 2:**

Relational Algebra – Madeline Woodhull, Mike Yamokoski, Daniel Valz, Yusuf Abdirahman

3.

* 1. π Product\_ID, Name(STOREFRONT \* SHOWS\_PRODUCT \* PRODUCT)
  2. πTitle(σPrice<10PRODUCT)
  3. BOB\_ORDERS <- σname = “Bob” (ORDER x CUSTOMER)

π Product\_ID, Title, Date(PRODUCT \* ORDERED\_BY \* BOB\_ORDERS)

* 1. STORE\_PRODS<- (PRODUCT \* SHOWS\_PRODUCTS \* (σStorefront\_ID = ‘7’ (STOREFRONT))

PROD\_ORDERS<- (STORE\_PRODS \* ORDERED\_BY \* ORDER)

π Customer\_ID, Product\_ID (PROD\_ORDERS \* PLACED\_BY \* CUSTOMER)

* 1. CUST\_PRODUCT\_LIST <- PRODUCT \* ORDERED\_BY \* ORDER \* PLACED\_BY \* CUSTOMER

COUNTS\_LIST <- Name F COUNT Product\_ID

π Name F MAX Count\_Name COUNTS\_LIST)

* 1. π Storefront\_ID (STOREFRONT \* σProd\_Count < 5 (Storefront\_ID F COUNT (SHOWS\_PRODUCTS \* PRODUCT)))
  2. SALES <- PRODUCT \* ORDERED\_BY

TOP\_SELL <- (F MAXCount\_Product\_ID(F COUNTProduct\_ID SALES) \* F SUMPrice SALES)

πTitle, Count\_Product\_ID, Sum\_Price,Seller\_Id TOP\_SELL

* 1. π Payment\_Type, Payment\_Count, Payment\_Sum (PAYMENT\_TYPE \* F COUNT Payment\_ID \* F SUM Order\_Price)
  2. π Customer\_ID, Email\_Address (CUSTOMER \* F MAX KARMA)

4.

1. Retrieve the Product IDs of all items on a Customer’s Wishlist.

π Product\_ID(CUSTOMER \* WISHLIST)

1. Create a list of all Storefronts and their Servers.

π Storefront\_ID, Server\_ID(HOSTED\_BY)

1. Retrieve the Storefront with the most Sellers.

π Storefront\_ID (F MAXCount\_Seller\_ID (F COUNT Seller\_ID (STOREFRONT \* SELLER)))

Graphical user interface, text

Description automatically generated

**CHECKPOINT 2 FEEDBACK:**

**Notice: Grader comments will not be exhaustive; it is important to maintain communication with myself and professor Pichkar to ensure that all questions and issues are resolved before the due date of the project.**

**Team 12:**  
Yusuf Abdirahman  
Mike Yamokoski  
Madeline Woodhull  
Daniel Valz

ERD:

Storefront should have multivalued images attribute

Numbers other than 1 should not be used when utilizing Chen’s notation

User superclass should be overlapping and total with buyer and seller subclasses

User superclass should have a key as it does not inherit the keys in its subclass

User superclass should connect to payment so it can be utilized by either type of user

Schema:

Didn’t see a relation for multivalued keywords attribute but the document formatting may have omitted it 🡪 Ensure all multivalued attributes are in specialized relations

Incorporate foreign keys into relations with N:1 and 1:1 relations (i.e. key on 1 side would be FK on many side)

Ensure all M:N relationships are documented, most appear to be but hard to tell with formatting

Make sure work isn’t cropped off page in the future

4.

One query needs to use an outer join

**CHECKPOINT 3:**

Checkpoint 3 – Madeline Woodhull, Mike Yamokoski, Daniel Valz, Yusuf Abdirahman

**CreateQueries.txt:**

CREATE SCHEMA CP03

CREATE TABLE STOREFRONT

(Storefront\_id INT NOT NULL,

Name VARCHAR(40) NOT NULL,

Banner ?image? NOT NULL,

Description VARCHAR(1000) NOT NULL,

Seller\_photo ?image? NOT NULL,

Seller\_bio VARCHAR(500) NOT NULL,

Product\_ID INT NOT NULL,

SERVER\_ID INT NOT NULL,

SELLER\_ID INT NOT NULL,

PRIMARY KEY(Storefront\_ID),

Foreign KEY(Product\_ID) REFERENCES PRODUCT,

Foreign KEY(Seller\_ID) REFERENCES USER,

FOREIGN KEY(Server\_ID) REFERENCES SERVER);

CREATE TABLE STOREFRONT\_IMAGES

(Storefront\_id INT NOT NULL,

Image ?image? NOT NULL,

PRIMARY KEY(Storefront\_ID, Image),

FOREIGN KEY(Storefront\_ID) REFERENCES STOREFRONT);

CREATE TABLE STOREFRONT\_URLS

(Storefront\_id INT NOT NULL,

URL VARCHAR(100) NOT NULL,

PRIMARY KEY(Storefront\_ID, URL),

FOREIGN KEY(Storefront\_ID) REFERENCES STOREFRONT);

CREATE TABLE SERVER

( Server\_id INT NOT NULL,

Capacity INT NOT NULL,

Storefront\_Id INT NOT NULL,

PRIMARY KEY(Server\_id),

FOREIGN KEY(Storefront\_id) REFERENCES STOREFRONT);

CREATE TABLE PRODUCT

(Product\_ID INT NOT NULL,

Title VARCHAR(20) NOT NULL,

DESCRIPTION VARCHAR(200) NOT NULL,

Price INT NOT NULL,

File\_Type VARCHAR(5) NOT NULL,

Storefront\_Id INT NOT NULL,

Order\_Id INT NOT NULL,

Wish\_Id INT NOT NULL,

Feed\_Id INT NOT NULL,

Seller\_Id INT NOT NULL,

PRIMARY KEY(Product\_ID),

FOREIGN KEY(Storefront\_id) REFERENCES STOREFRONT,

FOREIGN KEY(Order\_id) REFERENCES ORDER,

FOREIGN KEY(Wish\_Id) REFERENCES WISH,

FOREIGN KEY(Feed\_Id) REFERENCES FEEDBACK,

CREATE TABLE SHOWS\_PRODUCT

Storefront\_ID INT NOT NULL,

Product\_ID INT NOT NULL,

Capacity INT. NOT NULL,

PRIMARY KEY(Storefront\_ID, Product\_ID);

FORIEGN KEY(Storefront\_ID) REFRENCES STOREFRONT

FORIEGN KEY(Product\_ID) REFRENCES PRODUCT;

CREATE TABLE HOSTED\_BY

Storefront\_ID INT NOT NULL,

Server\_id INT NOT NULL,

Capacity INT. NOT NULL,

PRIMARY KEY(Storefront\_ID,Server\_ID);

FORIEGN KEY(Storefront\_ID) REFRENCES STOREFRONT

FORIEGN KEY(Server\_ID) REFRENCES SERVER;

CREATE TABLE ORDER\_HAS

Order-ID INT NOT NULL,

Payment\_ID INT NOT NULL,

PRIMARY KEY(Order\_ID, Payment\_ID);

FORIEGN KEY(Order\_ID) REFRENCES ORDER

CREATE TABLE KARMA

Payment\_ID INT NOT NULL,

PRIMARY KEY( Payment\_ID);

CREATE TABLE CRYPTO

Payment\_ID INT NOT NULL,

PRIMARY KEY( Payment\_ID);

CREATE TABLE ORDER

Order-ID INT NOT NULL,

DATE VARCHAR(20) NOT NULL,

Order\_Price INT NOT NULL,

PRIMARY KEY(Order\_ID);

CREATE TABLE ORDER\_BY

Order-ID INT NOT NULL,

Product\_ID INT NOT NULL,

DATE VARCHAR(20) NOT NULL,

PRIMARY KEY(Order\_ID, PRODUCT\_ID);

FORIEGN KEY(Order\_ID) REFRENCES ORDER

FORIEGN KEY(Product\_ID) REFRENCES PRODUCT;

CREATE TABLE FEEDBACK

(Feed-ID INT NOT NULL,

Stars INT NOT NULL,

Comment VARCHAR(200) NOT NULL,

PRIMARY KEY(Feed\_ID);

CREATE TABLE WISHLIST

(Wish\_ID INT NOT NULL,

Customer\_ID INT NOT NULL,

PRIMARY KEY( Wish\_ID),

FORIEGN KEY(Customer\_ID) REFRENCES USER

CREATE TABLE ITEMS\_ON\_LIST

(Product\_ID INT NOT NULL,

Wishlist\_ID INT NOT NULL,

Description VARCHAR(200) NOT NULL,

PRIMARY KEY(Product\_ID, Wishlist\_ID),

FOREIGN KEY(Product\_ID) REFERENCES PRODUCT,

FOREIGN KEY(Wishlist\_ID) REFERENCES WISHLIST,

CREATE TABLE USER

(Name VARCHAR(30) NOT NULL,

Email VARCHAR(20) NOT NULL,

Karma INT NOT NULL,

Customer\_ID INT NOT NULL,

Seller\_ID INT NOT NULL,

Prefered\_Payment VARCHAR(15) NOT NULL,

PRIMARY KEY(Customer\_ID, Seller\_ID);

**InsertQueries.txt:**

**INSERT INTO SERVER**

**VALUES (1, 5000);**

**INSERT INTO HOSTED\_BY**

**VALUES(1, 1, 1) INSERT INTO STOREFRONT**

**VALUES(1, "J Store", 23, "This is a store", 23, "I'm a store", 2, 3)**

**INSERT INTO STOREFRONT**

**VALUES(1, "J's Store", 2423, "This is a website", 23, "I'm a person", 1, 1);**

**INSERT INTO STOREFRONT**

**VALUES(5, "F Store", 23, "This is a store", 23, "I'm a store", 2, 3)**

**SimpleQueries.txt:**

1. SELECT Product\_ID, Name

FROM STOREFRONT, PRODUCT;

1. SELECT Title

FROM PRODUCT

WHERE Price<10;

1. SELECT Product\_ID, Title, Date

FROM PRODUCT, ORDER, CUSTOMER

WHERE CUSTOMER.name = “Bob”;

1. SELECT Customer\_ID, Product\_ID

FROM PRODUCT, STOREFRONT, ORDER

WHERE STOREFRONT.Storefront\_ID = ‘7’;

1. SELECT Name, Max(\*)

FROM PRODUCT, ORDER, CUSTOMER

WHERE (SELECT max(SELECT COUNT(Product\_ID) FROM PRODUCT) AS Max(\*) FROM PRODUCT);

1. SELECT Storefront\_ID

FROM STOREFRONT

WHERE Prod\_Count < 5;

1. SELECT Title, Count\_Product\_ID (\*), Sum\_Price (\*), Seller\_ID

FROM PRODUCT, SALES

WHERE (SELECT max(SELECT count(Product\_ID) FROM SALES) AS Count\_Product\_ID (\*) FROM PRODUCT) AND (SELECT sum(Price) AS Sum\_Price (\*) FROM SALES);

1. SELECT Payment\_Type, Payment\_Count (\*), Payment\_Sum (\*)

FROM PAYMENT, ORDER

WHERE (SELECT sum(Order\_Price) AS Payment\_Sum (\*) FROM ORDER) AND (SELECT count(Payment\_ID) AS Payment\_Count (\*) FROM PAYMENT);

1. SELECT Customer\_ID, Email\_Address

FROM CUSTOMER

WHERE (SELECT max(Karma) FROM CUSTOMER) ;

**ExtraQueries.txt:**

1. SELECT Storefront\_ID

FROM STOREFRONT

WHERE (SELECT max(SELECT count(Seller\_ID) FROM SELLER) FROM STOREFRONT);

1. SELECT Storefront\_ID, Server\_ID

FROM HOSTED\_BY;

1. SELECT Customer\_ID, Seller\_ID

FROM CUSTOMER C FULL OUTER JOIN SELLER S ON c.Customer\_ID = s.Seller\_ID;

**AdvancedQueries.txt:**

1. SELECT Name, Spent (\*)

FROM CUSTOMER

WHERE (SELECT sum(Price) AS Spent (\*) FROM ORDER);

1. SELECT Name, Email

FROM CUSTOMER

WHERE (SELECT sum(PRICE) FROM ORDER)>(SELECT avg(Price) FROM ORDER);

1. SELECT Title, Sold (\*)

FROM PRODUCT, ORDER\_BY

WHERE (SELECT sum(Product\_ID) AS Sold (\*) FROM ORDER\_BY)

ORDER\_BY Sold DESC;

1. SELECT Title, Sold (\*)

FROM PRODUCT, ORDER\_BY

WHERE (SELECT sum(Price) AS Sold (\*) FROM ORDER WHERE ORDER\_BY.Product\_ID = PRODUCT.Product\_ID)

ORDER\_BY Sold DESC;

1. SELECT Name

FROM SELLER AS S FULL OUTER JOIN PRODUCT AS P

WHERE (SELECT max(SELECT sum(Product\_ID) FROM ORDER\_BY WHERE ORDER\_BY.Product\_ID = PRODUCT.Product\_ID) FROM ORDER\_BY WHERE P.Seller\_ID = S.Seller\_ID);

1. SELECT Name

FROM SELLER AS S FULL OUTER JOIN PRODUCT AS P

WHERE (SELECT max(SELECT sum(Price) FROM ORDER) FROM ORDER\_BY WHERE P.Seller\_ID = S.Seller\_ID);

1. SELECT Name

FROM CUSTOMER, PRODUCT, ORDER\_BY

WHERE (SELECT Seller\_ID FROM SELLER AS S FULL OUTER JOIN PRODUCT AS P WHERE (SELECT max(SELECT sum(Price) FROM ORDER) FROM ORDER\_BY WHERE P.Seller\_ID = S.Seller\_ID) = PRODUCT.Seller\_ID AND ORDER\_BY.Product\_ID = PRODUCT.Seller\_ID;

1. SELECT Name

FROM SELLER, ORDER

WHERE (SELECT Name FROM CUSTOMER WHERE (SELECT sum(PRICE) FROM ORDER)>(SELECT avg(Price) FROM ORDER));

1. SELECT Sold (\*), Highest (\*), Lowest (\*), Avg (\*)

FROM PRODUCT

WHERE (SELECT max(Price) AS Highest (\*) FROM ORDER) AND (SELECT min(PRICE) AS Lowest (\*) FROM ORDER) AND (SELECT avg(Price) AS Avg (\*) FROM ORDER) AND (SELECT count(Product\_ID) AS Sold (\*) FROM ORDER\_BY);

NOT WORKING SQL

CREATE TABLE KARMA

Payment\_ID INT NOT NULL,

PRIMARY KEY(Payment\_ID);

CREATE TABLE CRYPTO

Payment\_ID INT NOT NULL,

PRIMARY KEY(Payment\_ID);

CREATE TABLE USER

(Name VARCHAR(30) NOT NULL,

Email VARCHAR(20) NOT NULL,

Karma INT NOT NULL,

Customer\_ID INT NOT NULL,

Seller\_ID INT NOT NULL,

Prefered\_Payment VARCHAR(15) NOT NULL,

PRIMARY KEY(Customer\_ID, Seller\_ID);

**CHECKPOINT 3 FEEDBACK:**

**Team 12:**

Yusuf Abdirahman

Mike Yamokoski

Madeline Woodhull

Daniel Valz

ERD

Relationship lines shouldnt cross

Ensure customer and seller PK’s do not conflict with superclass PK

Schema

File ‘Relational Schema after CP02 Feedback’ is not legible text when opened

CreateQueries

Create storefront table : ‘near “?”: syntax error’

Create storefront\_images table : ‘near “?”: syntax error’

Create table product: ‘near “ORDER”: syntax error’

Create table shows\_product: ‘near “Storefront\_ID”: syntax error’

Create table hosted\_by: ‘near “Storefront\_ID”: syntax error’

Create table order\_has: ‘near “ORDER”: syntax error’

Create table karma: ‘near “payment\_ID”: syntax error’

Create table crypto: ‘near “payment\_ID”: syntax error’

Create table order: ‘near “ORDER”: syntax error’

Create table order\_by: ‘near “ORDER”: syntax error’

Create table feedback: ‘near “ORDER”: syntax error’

Create table wishlist: ‘near “FORIEGN”: syntax error’

Create table items\_on\_list: ‘incomplete input’

Create table user: ‘ near “;” syntax error ’

Because the create and insert statements do not all work / do not exist, I cannot grade the queries for compilation. Please create insert statements and confirm create / insert statements work on Sqlite online, resubmit and I can confirm simple, extra, and advanced queries run correctly.

Note: Grader notes are not exhaustive, stay in communication with the graders and professor Pichkar