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.

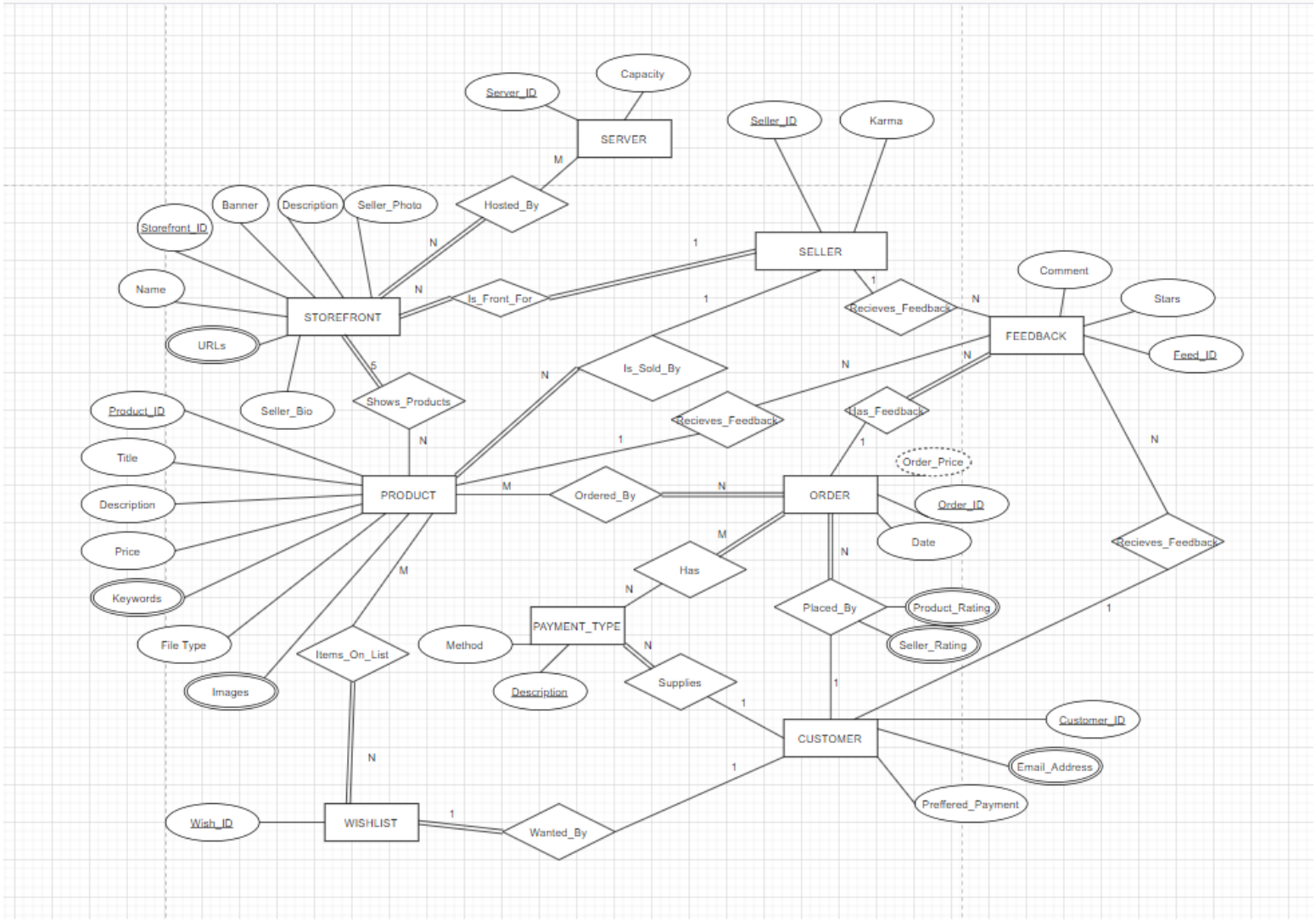
8. Determine at least three other informal update operations and describe what entities would need to have attributes altered and how they would need to be changed given your above descriptions. Include one example for each of the additional entities you proposed in question 4 above. 'Informal 'means 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.


Capacity
Server_ID
SERVER
Seller_ID
Karma
M
Hosted_By
Banner
Description
Seller_Photo
Storefront_ID
N
1
SELLER
Comment
Name
N
Is_Front_For
1
Recieves_Feedback
N
Stars
STOREFRONT
FEEDBACK
URLs
Feed_ID
N
N
Is_Sold_By
5
Shows_Products
N
Product_ID
Seller_Bio
Recieves_Feedback
Has_Feedback
Title
1
1
N
Order_Price
PRODUCT
M
Ordered_By
N
ORDER
Description
Order_ID
Price
M
Date
Recieves_Feedback
Has
N
Keywords
M
N
Placed_By
Product_Rating
1
PAYMENT_TYPE
Seller_Rating
File Type
Items_On_List
Method
N
Supplies
Images
Description
1
1
CUSTOMER
Customer_ID
N
Email_Address
1
Preffered_Payment
Wish_ID
WISHLIST
1
Wanted_By

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.

a. $\pi_{\text{Product\_ID, Name}}$(STOREFRONT * SHOWS_PRODUCT * PRODUCT)

b. $\pi_{\text{Title}}(\sigma_{\text{Price}<10}\text{PRODUCT})$

c. BOB_ORDERS <- $\sigma_{\text{name = "Bob"}}$ (ORDER x CUSTOMER)

$\pi_{\text{Product\_ID, Title, Date}}$(PRODUCT * ORDERED_BY * BOB_ORDERS)

d. STORE_PRODS<- (PRODUCT * SHOWS_PRODUCTS * ($\sigma_{\text{Storefront\_ID = '7'}}$ (STOREFRONT))

PROD_ORDERS<- (STORE_PRODS * ORDERED_BY * ORDER)

$\pi_{\text{Customer\_ID, Product\_ID}}$ (PROD_ORDERS * PLACED_BY * CUSTOMER)

e. CUST_PRODUCT_LIST <- PRODUCT * ORDERED_BY * ORDER * PLACED_BY * CUSTOMER

COUNTS_LIST <- $_{\text{Name}}$ F COUNT Product_ID

π_{Name} F MAX $_{\text{Count\_Name}}$ COUNTS_LIST)

f. $\pi_{\text{Storefront\_ID}}$ (STOREFRONT * $\sigma_{\text{Prod\_Count} < 5}$ ($_{\text{Storefront\_ID}}$ F COUNT (SHOWS_PRODUCTS * PRODUCT)))

g. SALES <- PRODUCT * ORDERED_BY

TOP_SELL <- (F $\text{MAX}_{\text{Count\_Product\_ID}}$(F $\text{COUNT}_{\text{Product\_ID}}$ SALES) * F $\text{SUM}_{\text{Price}}$ SALES)

$\pi_{\text{Title, Count\_Product\_ID, Sum\_Price,Seller\_Id}}$ TOP_SELL

h. $\pi_{\text{Payment\_Type, Payment\_Count,}}$ $\text{Payment}_{\text{Sum}}$ (PAYMENT_TYPE * F COUNT Payment_ID * F SUM Order_Price)

i. $\pi_{\text{Customer\_ID, Email\_Address}}$ (CUSTOMER * F MAX KARMA)

4.

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

$\pi_{\text{Product\_ID}}$(CUSTOMER * WISHLIST)

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

$\pi_{\text{Storefront\_ID, Server\_ID}}$(HOSTED_BY)

c. Retrieve the Storefront with the most Sellers.

$$\pi_{\text{Storefront\_ID}} (F\ MAX_{\text{Count\_Seller\_ID}} (F\ COUNT_{\text{Seller\_ID}} (\text{STOREFRONT} * \text{SELLER})))$$

STOREFRONT

Storefront_ID	Name	Banner	Description	Seller_Photo	Name	Seller_Bio	Product_ID	Seller_ID

STOREFRONT_IMAGES

Storefront_ID	Image

STOREFRONT_URLS

Storefront_ID	URL

SERVER

Server_ID	Capacity

HOSTED_BY

Storefront_ID	Server_ID	Capacity

SHOWS_PRODUCTS

Storefront_ID	Product_ID	Capacity

PRODUCT

Product_ID	Title	Description	Price	File_Type	Storefront_ID	Order_ID	Wish_ID	Feed_ID	Seller_ID

ITEMS_ON_LIST

Product_ID	Wishlist_ID	Description

Order_Has

Order_ID	Payment_ID

KARMA

payment_ID

CRYPTO

payment_ID

CREDIT_CARD

payment_ID	cc_no

BANK

payment_ID	route_no

WISHLIST

Wish_ID

User

Name	Email	Karma	Seller_ID	Customer_ID	Preferred_Payment

Order

Order ID	Date	Order price

Order_by

Order ID	Product ID	Date

Feedback

Feed ID	Stars	Comment

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;
2. SELECT Title

   FROM PRODUCT
   WHERE Price<10;
3. SELECT Product_ID, Title, Date

   FROM PRODUCT, ORDER, CUSTOMER
   WHERE CUSTOMER.name = “Bob”;
4. SELECT Customer_ID, Product_ID

   FROM PRODUCT, STOREFRONT, ORDER
   WHERE STOREFRONT.Storefront_ID = ‘7’;
5. SELECT Name, Max(*)

   FROM PRODUCT, ORDER, CUSTOMER
   WHERE (SELECT max(SELECT COUNT(Product_ID) FROM PRODUCT) AS Max(*) FROM PRODUCT);
6. SELECT Storefront_ID

   FROM STOREFRONT
   WHERE Prod_Count < 5;
7. 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);

8. 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);
9. 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);
2. SELECT Storefront_ID, Server_ID

   FROM HOSTED_BY;
3. 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);
2. SELECT Name, Email

   FROM CUSTOMER
   WHERE (SELECT sum(PRICE) FROM ORDER)>(SELECT avg(Price) FROM ORDER);
3. SELECT Title, Sold (*)

   FROM PRODUCT, ORDER_BY
   WHERE (SELECT sum(Product_ID) AS Sold (*) FROM ORDER_BY)
   ORDER_BY Sold DESC;
4. 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;

5. 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);

6. 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);

7. 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;

8. SELECT Name

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

9. 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