Assignment 3
COMPSCI 2DB3
400451379
saxenp4

Part 1:

We start with the User entity, as it is not a weak entity and does not partake in any one-to-many relationships. User is also the root entity of a ISA-Hierarchy. Translating this entity yields the following relationship schema:

User(<u>mid</u> : INT, email : VARCHAR(200), password : VARCHAR(100)).

Furthermore, mid is the primary key of the table and has the datatype INT, the email and password attribute must be unique. We have used VARCHAR for both password and email instead of CLOB as both attributes have constraints on them (which CLOB is not guaranteed to support). The Subscriber schema translates to the following SQL statement:

```
CREATE TABLE user(
mid INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
email VARCHAR(200) NOT NULL UNIQUE ,
password VARCHAR(100) NOT NULL
)
```

Moving on, we translate the Consumer entity. For the ISA-Hierarchy, I chose to go with the ER method. The NULL method is inefficient as the weak entity Offer require a Producer table. The OO method is inefficient as it would not be optimized for fetching all entities in the hierarchy. For example, to get all the consumers we would have to go through both the consumer and consumer_and_producer tables.

Next, we translate the entity Consumer using the ER method into the following relational schema:

Consumer(<u>mid</u> : INT, address : VARCHAR(200))

In which mid is a foreign key that refers to the primary key of the User table. The address of the consumer does not need to be unique as multiple consumers can have the same address for example, members of a family. Also, the address attribute cannot be NULL as the produce has to be delivered somewhere.

```
CREATE TABLE consumer(
mid INT NOT NULL PRIMARY KEY REFERENCES user(mid),
address VARCHAR(200) NOT NULL
)
```

Next, we translate the Producer entity:

Producer(<u>mid</u> : INT, name : CLOB)

The table uses mid as a foreign key being referenced from the user table. Every producer has a name which is not unique and cannot be NULL, also the datatype is CLOB as the Name attribute is just stored and does not need to be accessed.

```
CREATE TABLE producer(
mid INT NOT NULL PRIMARY KEY REFERENCES user(mid),
name CLOB NOT NULL
)
```

We move onto the Produce entity. It has a pid which is auto generated and unique and it cannot be NULL as it is a primary key. The produce also has a name and description, the name cannot be NULL however, the description in my schema can be allowed to be NULL as some produce are very self explanatory and the description does not give us any restrictions. The name and description are both CLOB as they are just data being stored.

Produce(<u>pid</u>: INT, name: CLOB, description: CLOB)

```
CREATE TABLE produce(
pid INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
name CLOB NOT NULL,
description CLOB
)
```

Next is the Sale entity. It has a sid which is auto generated and unique and it cannot be NULL since it is the primary key. It also has a date with datatype DATE. I will make a new column called BuyerID instead of naming it Consumer, this column will contain the consumer id which is retrieved from the user table.

Sale(<u>sid</u>: INT, date: DATE, buyerid: INT)

```
CREATE TABLE sale(
sid INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
date DATE,
buyerid INT NOT NULL REFERENCES consumer(mid)
)
```

For the Offer entity, since the week attribute needs to a count down for how much longer the offer is valid for I use the INTERVAL datatype however, DBeaver does not accept INTERVAL as a datatype so I just used DATE. For price attribute since the amount can be in dollars and cents I use the DECIMAL datatype. For the primary key of this entity I use the week, producer's mid, produce's pid attributes.

Offer(price: DECIMAL, mid: INT, pid: INT, week: DATE)

```
CREATE TABLE offer(
mid INT NOT NULL REFERENCES producer(mid),
pid INT NOT NULL REFERENCES produce(pid),
price DECIMAL NOT NULL,
week DATE NOT NULL,
PRIMARY KEY (mid,pid,week)
)
```

Lastly, the BOffer relationship. I will make a table for this relationship since it has an attribute called amount. This table has columns: amount of datatype INT, sid fom the Sale table and, the primary key of the Offer table (mid, pid, week). Since there is a constraint on this relationship and the Sale entity, the amount attribute cannot be NULL since, a sale cannot take place is the amunt of product bought is zero.

BOffer(amount: INT, mid: INT, pid: INT, week: DATE, sid: INT)

```
CREATE TABLE boffer(
mid INT NOT NULL REFERENCES offer(mid),
pid INT NOT NULL REFERENCES offer(pid),
week DATE NOT NULL REFERENCES offer(week),
sid INT NOT NULL REFERENCES sale(sid),
amount INT NOT NULL,
PRIMARY KEY(mid,pid,week,sid)
)
```

Part 2

Reveiw table has a review ID, datatype INT. It also has the sid which will be referenced from the Sale table. The uid is the user ID of the user who took part in the sale, now this can either be a consumer or a producer who **partook in the sale** so I get the uid from the sale table attribute buyerid instead of the uid from the user table as the sale table gives us uid of the consumers who were involved in the sale whereas, uid from user table would give us uid of any user irrespective of whether they partook in a sale or not.

Review(rid: INT, sid: INT, uid: INT, score: INT, date: DATE, description: CLOB)

```
CREATE TABLE review(
rid INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
sid INT NOT NULL REFERENCES sale(sid),
uid INT NOT NULL REFERNECES sale(buyerid),
score INT NOT NULL,
date DATE,
description CLOB
)
```

For the ReviewVote table, I use the rid from the review table and uid from the user table instead of sale as done in the last table since **any** user can vote on a review and not just the ones who partook in a sale. For the VoteType column, I assign its datatype as VARCHAR and add a CHECK statement so that users can only enter one of the following values:

a. Upvote
b. Downvote
c. Neutral

ReviewVote (rid: INT, uid: INT, voteType: VARCHAR(8))

```
CREATE TABLE reviewvote(
rid INT NOT NULL REFERENCES review(rid),
uid INT NOT NULL REFERNECES user(mid),
votetype VARCHAR(8),
CHECK (votetype = ‘upvote’ OR votetype = ‘downvote’ OR votetype = ‘neutral’),
PRIMARY KEY (uid,rid)
)
```

Next is SellerScore, first we find the reviews for a specific sale using AVG(review.score) gives us the average of all the scores from the review table for a specific sale using ‘WHERE sale.sid = review.sid’

```
CREATE VIEW sellerscore
AS SELECT COUNT(r.score), AVG(review.score), producer.mid
FROM sale, review, boffer, producer
WHERE sale.sid = review.sid AND boffer.sid = sale.sid
GROUP BY producer.mid
```

(Incomplete, I could’nt solve this **insert sad emoji**)

Lastly comments, I make a unique comment ID for every comment which is auto generated, reaction, description and title are both datatype CLOB since, they only store values. Description and titles have been set to NOT NULL since it would’nt be a comment without a comment title and comment description. Lastly, the date of the comment is also noted.

Comments(<u>commentid</u>: INT, title: CLOB, description: CLOB, uid: INT, date: DATE, reaction: CLOB)

```
CREATE VIEW comments(
cid INT NOT NULL GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
description CLOB NOT NULL,
title CLOB NOT NULL,
uid INT NOT NULL REFERNECES user(mid),
date DATE NOT NULL,
reaction CLOB
)
```