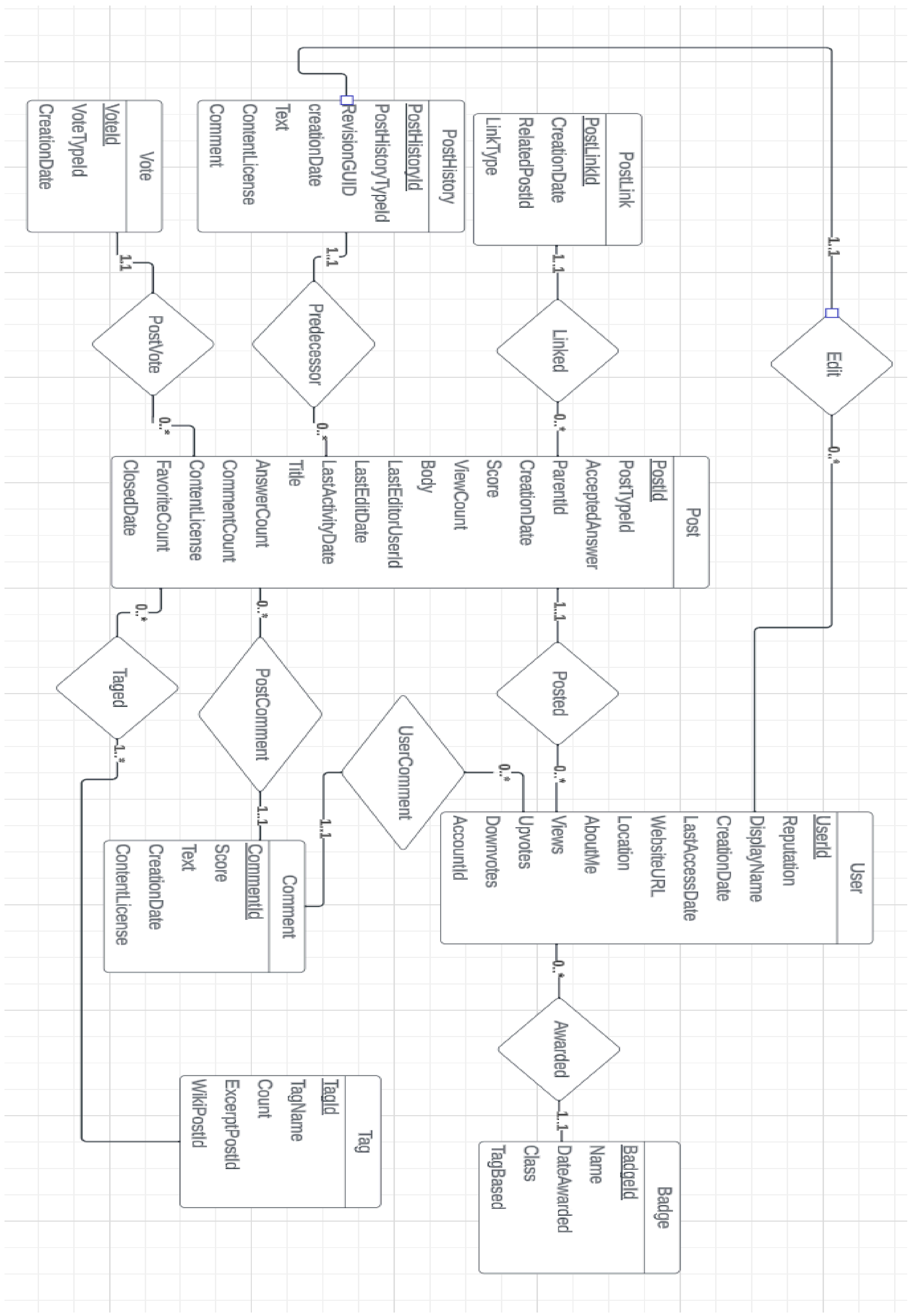
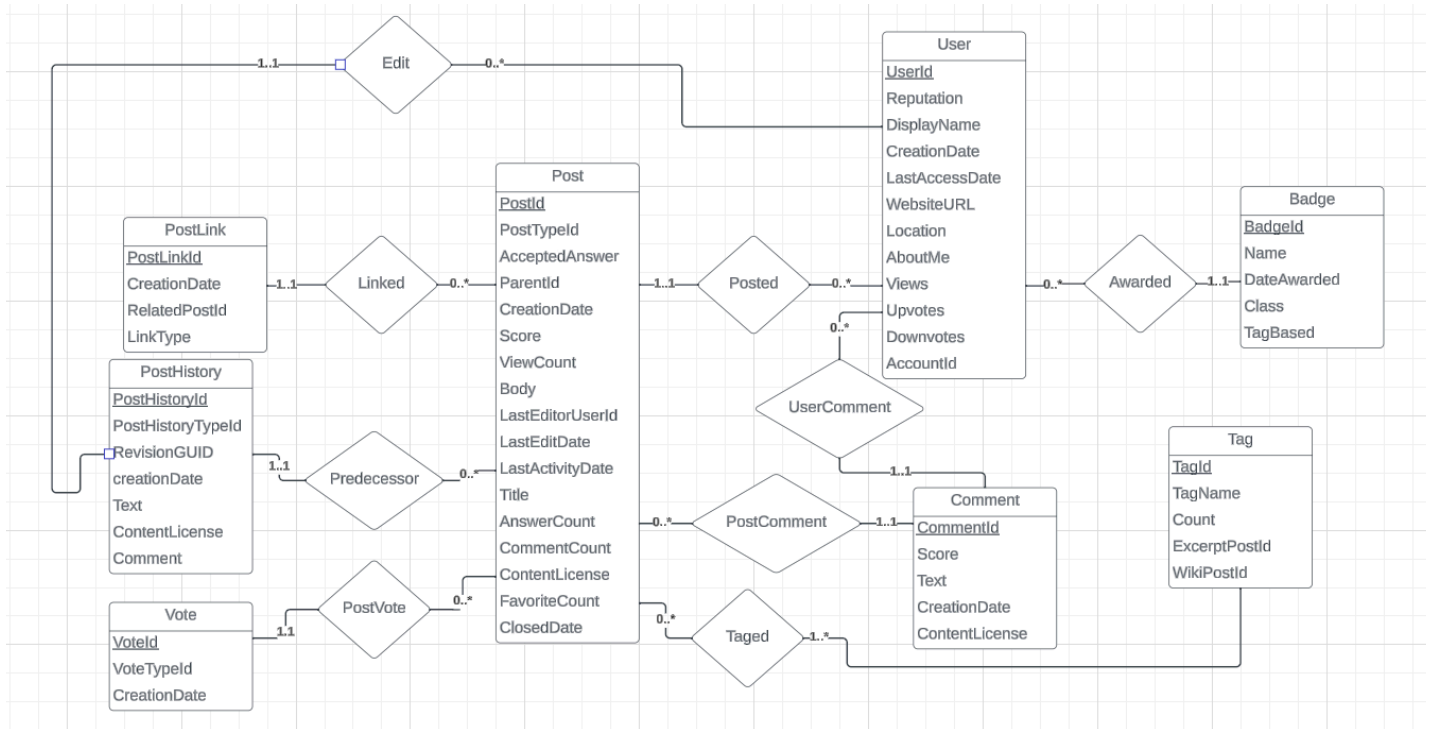
1. -The following image is the ER diagram made to represent the Stack Exchange data:



The image is repeated here again in landscape for the convenience of not hurting your neck

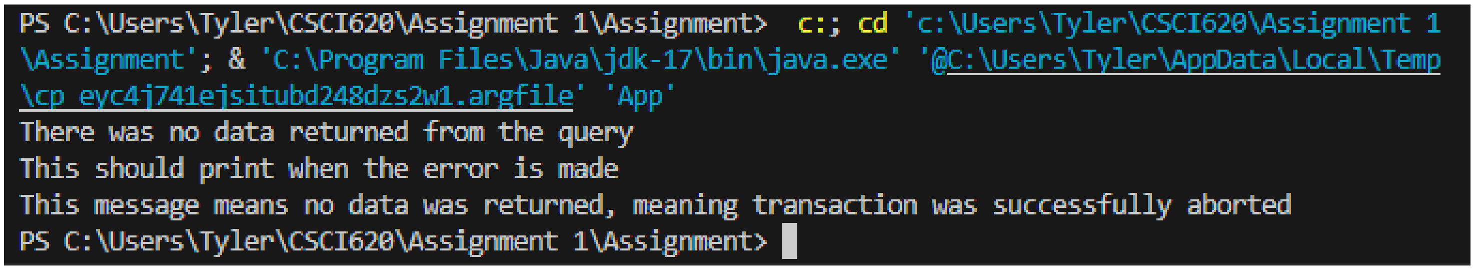


2. As discussed in class with Professor Mior, Question 2, like Question 4, could be just a set of SQL DDL statements to represent the relational model. The DDL statements are located in the code zip file, as Professor Mior mentioned they should be. The following paragraph is the brief description of solving the DDL issues as Question 2 requires.
    a. Firstly, figuring out what tables I needed was easy. I needed one for every strong entity type I had and one for every strong relationship I had. The attributes for the tables that related to entities were the attributes listed under them in the ER diagram. For the relationship tables, they only needed the two foreign keys of the two entities that they related. All Ids in the data files were integers, so those types were easy to decide. Additionally, all dates/times listed were a full day and time, so Timestamp was used for all of them. Any block of text that was longer and could have a large number of chars (ie. body of a post, about me, etc.) I used a TEXT type because they do not require a max number of characters to be listed. The smaller strings (ie. title, display name, tag name, etc.) were given a VARCHAR type. The max number of characters was determined by looking at the data files and estimating the max length set by the website and setting a value above that amount. Lastly, I gave an attribute/column the NOT NULL keywords if I could not find a piece of data that did not have that information.
    b. I should also note, that to make the tables work, I had to change the name of the entity "User" to "Youser" because Postgres doesn't like people making a table with the name "User"
3.
    a. Bages.xml
        i. The purpose of the Badges file is to keep track of all earned badges of all users of the website. It contains the ID of the badge, the userId of the person who earned the badge, which badge it is, the date the badge was earned, what type of badge it was, and if the badge was "TagBased"
    b. Comments.xml
        i. The purpose of this file is to keep track of all the smaller comments that people can make on a post. It doesn't contain too much information because the comments are smaller and just meant to be a little bit of text. The file contains the commentId, the Id of the post the comment is attached to, the score of the comment, the body text of the comment, the creation date of the comment, the user id of who made the comment, and the content license information.
    c. PostHistory.xml
        i. The purpose of this file is to keep information about the past history of posts. It tracks all pasts edits of all posts The file contains the id of the posthistory (which is not the same as the post id), the type of post history it is, the id of the post the history belongs to, the revision GUID string (most likely an internal reference string), the creation date, the user id of the person who made the change, the text of the old version of the post, and the content license information.

d. PostLinks.xml
    i. The purpose of this file is to keep track of every time a post links to another post or is linked to another post. It contains the Id of the link, the creation date of the link, the Id of the post that has the link, the Id of the post that is being linked to, and data on the type of link it is
e. Posts.xml
    i. This is the file with the most amount of columns/attributes. The purpose of this file is to keep track of all posts on the website and all the necessary information about them. It keeps track of the following items: the post id, the post type (initial post or reply post), id of the post that is the accepted answer if one exists, the id of the parent post if it exists, the creation date of the post, the post’s score, the view count of the post, the body text of the post, the user id of the owner of the post, the id of the last editor of the post, the date of the last edit of the post, the date of the last activity to be done on the post, the title of the post if it exists, the tags of the post if they exist, the number of answers if it was a parent post, the number of comments on the post, the content license information, the number of favorites the post has if any exist, and the date the post was closed if it exists
f. Tags.xml
    i. The purpose of this file is to keep track of the tag information. It has the id of the tag, the name of the tag, the count of the number of posts that use the tag, the id of the post that is the “Excerpt Post”, and the id of the post that is the “Wiki Post:
g. Users.xml
    i. The purpose of this file is to keep track of all the data of all the users of the website. It contains the id of the user, the reputation value of the user, the date the account was created, the display name of the user, the date the account was last accessed, the user’s website URL if they put one, the user’s location if they put one, the text of an “about me” section if the user filled it out, the number of views the user’s account has, the number of upvotes the user has,the number of downvotes the user has, and the id that connects the user to other accounts across other websites.
h. Votes.xml
    i. The purpose of this file is to track the info of the votes made on posts. It contains the id of the vote, the id of the post the vote was made on, the type of vote that was made, and the creation date of the vote.

4.
    a. After connecting to the database, inputting the data should have been fairly easy. However, after putting a good effort into this section of the project, I could not get any efficient parsing method to work. I have not worked much with .xml files before and this threw me off. So, I did what I thought was the next best thing and decided to parse the data manually, via reading the files line by line and completing string operations on them to extract the data. This severely slowed down the code. While the SQL and database connections all work, I have not had the chance to know the full timing of my program. Through my testing and timing of individual sections, I am, assuming that it is somewhere in the 6-8 hours range, over the 4-hour time limit that was given.

5.
    a. Below is the picture of the output when doing this test. For the code, check the submitted zipped code file.



```
PS C:\Users\Tyler\CSCI620\Assignment 1\Assignment>  c:; cd 'c:\Users\Tyler\CSCI620\Assignment 1\Assignment'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '@C:\Users\Tyler\AppData\Local\Temp\cp_eyc4j741ejsitubd248dzs2w1.argfile' 'App'
There was no data returned from the query
This should print when the error is made
This message means no data was returned, meaning transaction was successfully aborted
PS C:\Users\Tyler\CSCI620\Assignment 1\Assignment> 
```