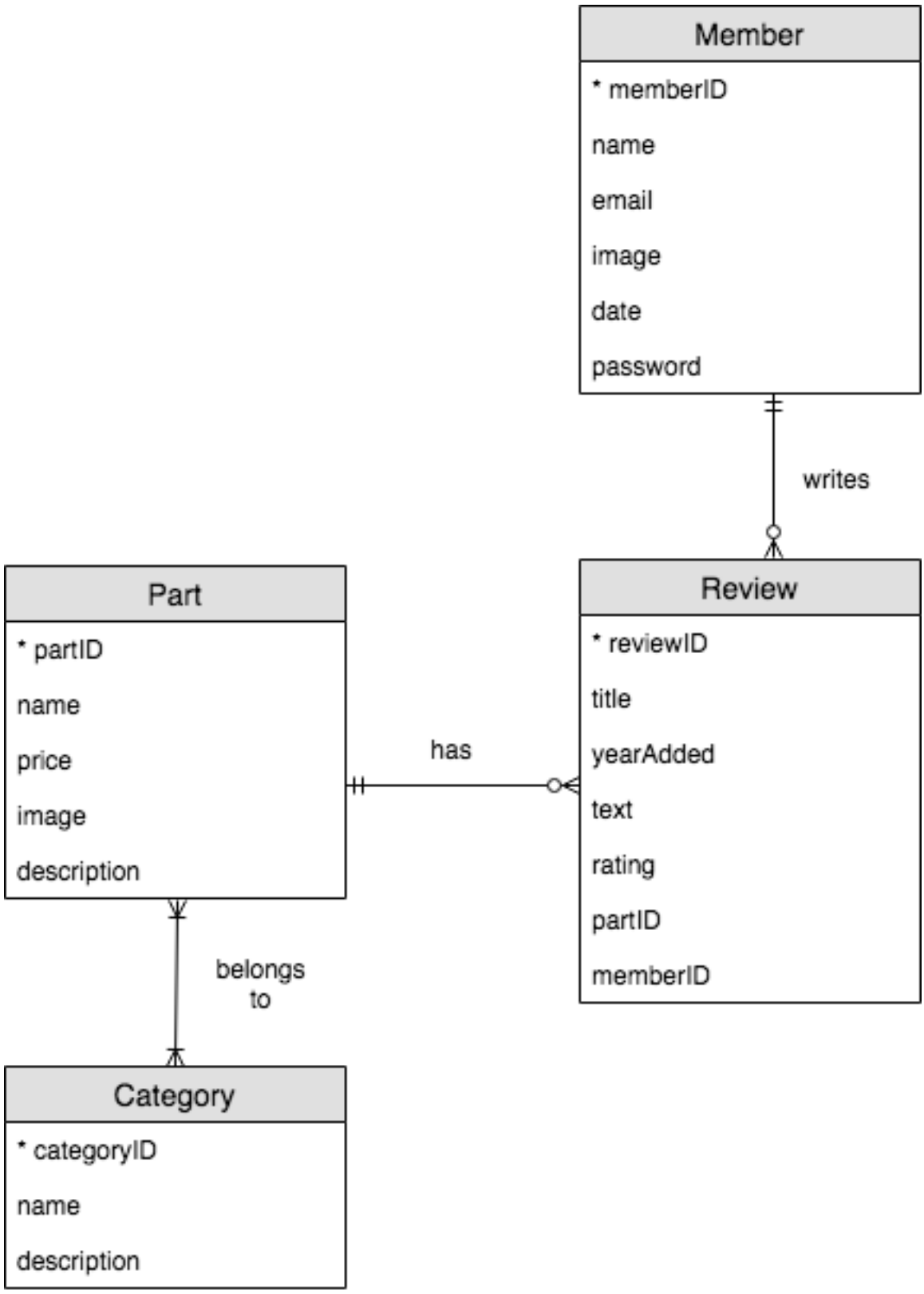
Modelling Exercise Candidate number: 97523

1. I chose www.robotshop.com for this exercise. It is a website that specializes in the sale of robot parts for professional, amateur enthusiasts, and schools. The following explains some of the features I chose to highlight in this exercise. Users may become members, with a username, email, and password. Members can write reviews about specific robot parts and rank them. The rankings are constrained to be within the range of 0-5 (inclusive). Members may optionally upload a photo. Parts belong to different categories to make searching easier. These categories include 'toys', 'professional',' DIY' etc and may overlap - ie each part may belong to more than one category. All parts must have an image associated with them for display.

Here is the reasoning behind my diagram. There is a one-to-many relationship between Part and Review. Each part has 0 to many reviews written about it, and each review is about exactly one part. There is a one-to-many relationship between Member and Review. Each member writes between 0 and many reviews, and each review is written by exactly one member. There is a many-to-many relationship between Part and Category. Each part belongs in from 1 to many categories (DIY, professional, toy, and these fields may overlap), and each category contains from 1 to many parts. This requires a join table using the primary keys from both tables.

2. ER Diagram:



3. See included modelling.sql script

4. Two uses of the described Database that use JOIN are:

    a. Get the top 10 most popular parts based on review ratings

On the sidebar, a list of the most popular parts may be displayed. Popularity of a part can be determined by averaging the ratings from member reviews for each part. To find the average ratings, the Review and Part tables are joined using the partID foreign key, and the aggregate average of ratings is taken, grouping by partID. Further, reviews are filtered by date to keep the ratings current. Only the top ten results will be shown to the user.

The Part.partID = Review.partID limits the join to valid combinations. The GROUP BY is used to find the average rating of rows only with the same partID, so that the ratings of different parts are not averaged together also.

```
SELECT AVG(rating) AS average, Part.partID
FROM Review
JOIN Part ON Part.partID = Review.partID
WHERE Review.yearAdded > '2014'
GROUP BY Part.partID
HAVING average > 3
ORDER BY average DESC
LIMIT 10;
```

b. Targeted product selection. The sidebar can display suggestions of which parts a user may like to purchase based on positive reviews he/she has written. Similarity of products is determined by the category. For example, if a member gives the highest review to a part in the DIY category, DIY parts will be suggested. All data being queried have a NOT NULL constraint, so NULL values need not be considered. If the user has not written any reviews, this section will be blank.

Returns 5 part Id's of parts to suggest to the user, by finding the highest rated review by that member (if there is a review) and selects 5 others parts in the same category as that part

The ? in the query is the memberID of the currently logged in user

```
SELECT Part.partID FROM Part
JOIN CategoryChart ON Part.partID = CategoryChart.partID
JOIN Category ON Category.categoryID = CategoryChart.categoryID
WHERE Category.categoryID = (SELECT Category.categoryID FROM CategoryChart
        WHERE CategoryChart.partID = (SELECT Review.partID FROM Review
                WHERE Review.memberID = ?
                ORDER BY Review.rating DESC LIMIT 1) )
ORDER BY RAND()
LIMIT 5;
```