Stanford SQL Questions:

Table: Movie (mID, title, year, director)

English: There is a movie with ID number mID, a title, a release year, and a director.

Table: Reviewer (rID, name)

English: The reviewer with ID number rID has a certain name.

Table: Rating (rID, mID, stars, ratingDate)

English: The reviewer rID gave the movie mID a number of stars rating (1-5) on a certain ratingDate.

Q1: Find the titles of all movies directed by Steven Spielberg.

```
SELECT m.title
FROM movie m
WHERE m.director = 'Steven Spielberg';
```

Q2: Find all years that have a movie that received a rating of 4 or 5, and sort them in increasing order.

```
SELECT DISTINCT m.year
FROM movie m
JOIN rating r
ON r.mID=m.mID
WHERE r.stars>=4
ORDER BY m.year;
```

Q3: Find the titles of all movies that have no ratings.

```
SELECT m.title
FROM movie m
WHERE m.mID not in (SELECT r.mID FROM rating r);
```

Q4: Some reviewers didn't provide a date with their rating. Find the names of all reviewers who have ratings with a NULL value for the date.

```
SELECT re.name
FROM reviewer re
WHERE re.rID in (SELECT r.rID FROM rating r WHERE r.ratingDate is NULL);
```

Q5. Write a query to return the ratings data in a more readable format: reviewer name, movie title, stars, and ratingDate. Also, sort the data, first by reviewer name, then by movie title, and lastly by number of stars.

```
SELECT re.name, m.title, r.stars, r.ratingDate
FROM rating r
LEFT JOIN reviewer re ON r.rID = re.rID
LEFT JOIN movie m ON r.mID = m.mID
ORDER BY re.name, m.title, r.stars;
```

Q6. For all cases where the same reviewer rated the same movie twice and gave it a higher rating the second time, return the reviewer's name and the title of the movie.

```
SELECT re.name, m.title
FROM (SELECT r1.rID as rID, r1.mID as mID
      FROM rating r1, rating r2
      WHERE (r1.rID=r2.rID) AND (r1.mID=r2.mID) AND (r1.ratingDate<r2.ratingDate) AND (r1.stars<r2.stars)) AS c
LEFT JOIN reviewer re ON c.rID=re.rID
LEFT JOIN movie m  ON c.mID=m.mID;
```

Q7: For each movie that has at least one rating, find the highest number of stars that movie received. Return the movie title and number of stars. Sort by movie title.

```
SELECT m.title, MAX(r.stars) AS max_rating
FROM rating r
LEFT JOIN movie m ON m.mID=r.mID
GROUP BY m.title;
```

Q8. For each movie, return the title and the 'rating spread', that is, the difference between highest and lowest ratings given to that movie. Sort by rating spread from highest to lowest, then by movie title

```
SELECT m.title, (MAX(r.stars)-MIN(r.stars)) AS rating_spread
FROM rating r
LEFT JOIN movie m ON r.mID=m.mID
GROUP BY m.title
ORDER BY rating_spread DESC;
```

Q9: Find the difference between the average rating of movies released before 1980 and the average rating of movies released after 1980. (Make sure to calculate the average rating for each movie, then the average of those averages for movies before 1980 and movies after. Don't just calculate the overall average rating before and after 1980.)

```
SELECT (t1.avg_b1980 - t2.avg_a1980)
FROM (SELECT AVG(r1.avg_movie) as avg_b1980
      FROM (SELECT r.mID, AVG(r.stars) AS avg_movie
            FROM rating r GROUP BY r.mID) as r1
      LEFT JOIN movie m ON r1.mID=m.mID WHERE m.year<1980) as t1,
     (SELECT AVG(r1.avg_movie) as avg_a1980
     FROM (SELECT r.mID, AVG(r.stars) AS avg_movie
           FROM rating r GROUP BY r.mID) as r1
     LEFT JOIN movie m ON r1.mID=m.mID WHERE m.year>1980) as t2;
```

Q10. Find the names of all reviewers who rated Gone with the Wind.

```
SELECT DISTINCT re.name
FROM rating r
LEFT JOIN reviewer re ON r.rID=re.rID
LEFT JOIN movie m ON r.mID=m.mID
WHERE m.title = 'Gone with the Wind';
```

Q11. For any rating where the reviewer is the same as the director of the movie, return the reviewer name, movie title, and number of stars.

```
SELECT re.name, m.title, r.stars
FROM rating r
LEFT JOIN reviewer re ON r.rID=re.rID
LEFT JOIN movie m ON r.mID=m.mID
WHERE re.name=m.director;
```

Q12. Return all reviewer names and movie names together in a single list, alphabetized. (Sorting by the first name of the reviewer and first word in the title is fine; no need for special processing on last names or removing "The".)

```
SELECT t.list_
FROM (SELECT DISTINCT re.name as list_
    FROM reviewer re
    UNION ALL
    SELECT DISTINCT m.title as list_
    FROM movie m) as t
ORDER BY t.list_;
```

Q13. Find the titles of all movies not reviewed by Chris Jackson.

```
SELECT DISTINCT m.title
FROM movie m
WHERE m.mID not in (SELECT mID FROM rating WHERE rID = (SELECT rID FROM reviewer WHERE name='Chris Jackson'));
```

Q14. For all pairs of reviewers such that both reviewers gave a rating to the same movie, return the names of both reviewers. Eliminate duplicates, don't pair reviewers with themselves, and include each pair only once. For each pair, return the names in the pair in alphabetical order.

```
SELECT DISTINCT t1.name, t2.name
FROM (SELECT r1.*,re1.name
      FROM rating r1
      LEFT JOIN reviewer re1 ON r1.rID=re1.rID
      ORDER BY re1.name) t1,
      (SELECT r1.*,re1.name
      FROM rating r1
      LEFT JOIN reviewer re1 ON r1.rID=re1.rID
      ORDER BY re1.name) t2
 WHERE (t1.rID!=t2.rID) AND (t1.mID=t2.mID) and t1.name<t2.name;
```

Q15. For each rating that is the lowest (fewest stars) currently in the database, return the reviewer name, movie title, and number of stars.

```
SELECT re.name,m.title,r.stars
FROM rating r
LEFT JOIN reviewer re ON r.rID=re.rID
LEFT JOIN movie m ON r.mID=m.mID
WHERE r.stars in (SELECT MIN(stars) FROM rating);
```

Q16. List movie titles and average ratings, from highest-rated to lowest-rated. If two or more movies have the same average rating, list them in alphabetical order.

```
SELECT m.title, t1.avg_movie
FROM (SELECT mID, AVG(stars) as avg_movie
      FROM rating
      GROUP BY mID) t1
LEFT JOIN movie m ON t1.mID=m.mID
ORDER BY t1.avg_movie DESC,m.title;
```

Q17. Find the names of all reviewers who have contributed three or more ratings.

```
SELECT re.name
FROM reviewer re
WHERE re.rID in (SELECT rID FROM rating GROUP BY rID HAVING count(rID)>2);
```

Q18. Some directors directed more than one movie. For all such directors, return the titles of all movies directed by them, along with the director name. Sort by director name, then movie title.

```
SELECT m.title, m.director
FROM movie m
WHERE m.director in (SELECT director FROM movie GROUP BY director HAVING COUNT(title)>1)
ORDER BY m.director,m.title;
```

Q19. Find the movie(s) with the highest average rating. Return the movie title(s) and average rating

```
SELECT m.title, t1.avg_movie
FROM (SELECT mID, AVG(stars) AS avg_movie
      FROM rating
      GROUP BY mID) t1
LEFT JOIN movie m ON t1.mID=m.mID
WHERE t1.avg_movie = (SELECT AVG(stars) AS avg_movie
                      FROM rating
                      GROUP BY mID
                      ORDER BY avg_movie DESC
                      LIMIT 1);
```

Q20. Find the movie(s) with the lowest average rating. Return the movie title(s) and average rating.

```
SELECT m.title, t1.avg_movie
FROM (SELECT mID, AVG(stars) AS avg_movie
      FROM rating
      GROUP BY mID) t1
LEFT JOIN movie m ON t1.mID=m.mID
WHERE t1.avg_movie = (SELECT AVG(stars) AS avg_movie
                      FROM rating
                      GROUP BY mID
                      ORDER BY avg_movie
                      LIMIT 1);
```

Q21. For each director, return the director's name together with the title(s) of the movie(s) they directed that received the highest rating among all of their movies, and the value of that rating. Ignore movies whose director is NULL.

```
SELECT m1.director,m1.title,t2.max_stars
FROM movie m1
LEFT JOIN (SELECT mID, MAX(stars) as max_stars FROM rating GROUP BY mID) t2 ON m1.mID=t2.mID
WHERE m1.mID in (SELECT m.mID
                 FROM movie m
                 LEFT JOIN (SELECT mID, MAX(stars) as max_stars FROM rating GROUP BY mID) t1 ON m.mID=t1.mID
                 GROUP BY m.director
                 HAVING MAX(max_stars)) AND m1.director is not NULL
ORDER BY m1.director;
```