You've started a new movie-rating website, and you've been collecting data on reviewers' ratings of various movies. There's not much data yet, but you can still try out some interesting queries. Here's the schema:

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

Reviewer (rID, name)
English: The reviewer with ID number *rID* has a certain *name*.

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

Your queries will run over a small data set conforming to the schema. View the database. (You can also download the schema and data.)

Instructions: Each problem asks you to write a query in SQL. To run your query against our back-end sample database using SQLite, click the "Submit" button. You will see a display of your query result and the expected result. If the results match, your query will be marked "correct". You may run as many queries as you like for each question.

Important Notes:

- Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.
- Unless a specific result ordering is asked for, you can return the result rows in any order.
- *You are to translate the English into a SQL query that computes the desired result over all possible databases.* All we actually check is that your query gets the right answer on the small sample database. Thus, even if your solution is marked as correct, it is possible that your query does not correctly reflect the problem at hand. (For example, if we ask for a complex condition that requires accessing all of the tables, but over our small data set in the end the condition is satisfied only by Star Wars, then the query "select title from Movie where title = 'Star Wars'" will be marked correct even though it doesn't reflect the actual question.) Circumventing the system in this fashion will get you a high score on the exercises, but it won't help you learn SQL. On the other hand, an incorrect attempt at a general solution is unlikely to produce the right answer, so you shouldn't be led astray by our checking system.

You may perform these exercises as many times as you like, so we strongly encourage you to keep working with them until you complete the exercises with full credit.

Q1

1.0/1.0 point (graded)
Find the titles of all movies directed by Steven Spielberg.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
SELECT title
FROM Movie
WHERE director = "Steven Spielberg"
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

E.T.
Raiders of the Lost Ark

Expected Query Result:

E.T.
Raiders of the Lost Ark

Submit

Q2

1.0/1.0 point (graded)

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

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
SELECT DISTINCT year
FROM Movie, Rating
WHERE Movie.mID = Rating.mID
    AND stars >= 4
ORDER BY year
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

1937
1939
1981
2009

Expected Query Result:

1937
1939

1981
2009

(Order matters)

Submit

Q3

1.0/1.0 point (graded)

Find the titles of all movies that have no ratings.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
1 SELECT title
2 FROM Movie
3 WHERE mID NOT IN (SELECT mID FROM Rating)
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

Star Wars
Titanic

Expected Query Result:

Star Wars
Titanic

Submit

Q4

1.0/1.0 point (graded)

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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
1 SELECT name
2 FROM Reviewer
3 WHERE rID in
4     (SELECT rID
5     FROM Rating
6     WHERE ratingDate IS NULL)
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

Chris Jackson
Daniel Lewis

Expected Query Result:

Chris Jackson
Daniel Lewis

Submit

Q5

1.0/1.0 point (graded)

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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
SELECT name, title, stars, ratingDate
FROM Reviewer, Movie, Rating
WHERE Reviewer.rID = Rating.rID AND Movie.mID = Rating.mID
ORDER BY name, title, stars
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

Ashley White	E.T.	3	2011-01-02
Brittany Harris	Raiders of the Lost Ark	2	2011-01-30
Brittany Harris	Raiders of the Lost Ark	4	2011-01-12
Brittany Harris	The Sound of Music	2	2011-01-20
Chris Jackson	E.T.	2	2011-01-22

Chris Jackson	Raiders of the Lost Ark	4	<NULL>
Chris Jackson	The Sound of Music	3	2011-01-27
Daniel Lewis	Snow White	4	<NULL>
Elizabeth Thomas	Avatar	3	2011-01-15
Elizabeth Thomas	Snow White	5	2011-01-19
James Cameron	Avatar	5	2011-01-20
Mike Anderson	Gone with the Wind	3	2011-01-09
Sarah Martinez	Gone with the Wind	2	2011-01-22
Sarah Martinez	Gone with the Wind	4	2011-01-27

Expected Query Result:

Ashley White	E.T.	3	2011-01-02
Brittany Harris	Raiders of the Lost Ark	2	2011-01-30
Brittany Harris	Raiders of the Lost Ark	4	2011-01-12
Brittany Harris	The Sound of Music	2	2011-01-20
Chris Jackson	E.T.	2	2011-01-22
Chris Jackson	Raiders of the Lost Ark	4	<NULL>
Chris Jackson	The Sound of Music	3	2011-01-27
Daniel Lewis	Snow White	4	<NULL>
Elizabeth Thomas	Avatar	3	2011-01-15
Elizabeth Thomas	Snow White	5	2011-01-19
James Cameron	Avatar	5	2011-01-20
Mike Anderson	Gone with the Wind	3	2011-01-09
Sarah Martinez	Gone with the Wind	2	2011-01-22
Sarah Martinez	Gone with the Wind	4	2011-01-27

(Order matters)

Submit

Q6

1.0/1.0 point (graded)

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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
SELECT name, title
FROM Reviewer, Movie
WHERE rID IN (SELECT R1.rID
                FROM Rating AS R1, Rating AS R2
                WHERE R1.rID = R2.rID AND R1.mID = R2.mID AND R1.ratingDate < R2.ratingDate AND R1.stars < R2.stars)
AND mID IN  (SELECT R1.mID
                FROM Rating AS R1, Rating AS R2
                WHERE R1.rID = R2.rID AND R1.mID = R2.mID AND R1.ratingDate < R2.ratingDate AND R1.stars < R2.stars)
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

Sarah Martinez	Gone with the Wind

Expected Query Result:

Sarah Martinez	Gone with the Wind

Submit

Q7

1.0/1.0 point (graded)

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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
SELECT title, maxStars
FROM Movie, (SELECT mID, MAX(stars) AS maxStars
            FROM Rating
            GROUP BY mID) AS MaxRates
WHERE Movie.mID = MaxRates.mID
ORDER BY title
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

Avatar	5
E.T.	3
Gone with the Wind	4
Raiders of the Lost Ark	4
Snow White	5
The Sound of Music	3

Expected Query Result:

Avatar	5
E.T.	3
Gone with the Wind	4
Raiders of the Lost Ark	4
Snow White	5
The Sound of Music	3

(Order matters)

Submit

Q8

1.0/1.0 point (graded)

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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
SELECT title, spread
FROM Movie, (SELECT mID, MAX(stars) - MIN(stars) AS spread FROM Rating GROUP BY mID) AS RateSpread
WHERE Movie.mID = RateSpread.mID
ORDER BY spread DESC, title
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

Avatar	2
Gone with the Wind	2
Raiders of the Lost Ark	2

E.T.	1
Snow White	1
The Sound of Music	1

Expected Query Result:

Avatar	2
Gone with the Wind	2
Raiders of the Lost Ark	2
E.T.	1
Snow White	1
The Sound of Music	1

(Order matters)

Submit

Q9

1.0/1.0 point (graded)

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.)

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
SELECT AVG(finalRatingPre80) - AVG(finalRatingPost80)
FROM (SELECT mID, year, AVG(stars) AS finalRatingPre80
        FROM (SELECT * FROM Rating NATURAL JOIN Movie WHERE year < 1980)
        GROUP BY mID),
     (SELECT mID, year, AVG(stars) AS finalRatingPost80
        FROM (SELECT * FROM Rating NATURAL JOIN Movie WHERE year > 1980)
        GROUP BY mID)
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

0.0555555555556

Expected Query Result:

0.0555555555556

Submit