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*.  
  
Q1

Find the titles of all movies directed by Steven Spielberg.

Q2

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

Q3

Find the titles of all movies that have no ratings.

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.

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.

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.

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.

Q8

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.

Q9

Find the names of all reviewers who have contributed three or more ratings. (As an extra challenge, try writing the query without HAVING or without COUNT.)

Q10

Find the names of all reviewers who rated 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.

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".)

Q13

Find the titles of all movies not reviewed by 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.

Q15

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

Q16

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.

Q17

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.)

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. (As an extra challenge, try writing the query both with and without COUNT.)

Q19

Find the movie(s) with the highest average rating. Return the movie title(s) and average rating. (Hint: This query is more difficult to write in SQLite than other systems; you might think of it as finding the highest average rating and then choosing the movie(s) with that average rating.)

Q20

Find the movie(s) with the lowest average rating. Return the movie title(s) and average rating. (Hint: This query may be more difficult to write in SQLite than other systems; you might think of it as finding the lowest average rating and then choosing the movie(s) with that average rating.)

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.

MULTIPLE CHOICE

[Q1] Consider the following SQL table declaration:

CREATE TABLE R (a INT, b INT, c INT, CHECK( [fill-in] ));

Currently R contains the tuples (1,4,14), (2,3,15), and (3,3,16). Which of the following tuple-based CHECK constraints will cause the following insertion to be rejected?

INSERT INTO R VALUES (4,4,9);

Note: When a tuple-based check is invoked for an insert and includes a subquery over the same table, the subquery is evaluated on the table *including* the inserted tuple.

a <= ALL (SELECT c - b FROM R)

b > (SELECT AVG(a) FROM R)

c > ALL (SELECT a + b FROM R)

c >= (SELECT SUM(b) FROM R)

[Q2] Consider the following trigger over a table R(a,b), specified using the trigger language of the SQL standard:

CREATE TRIGGER Rins

AFTER INSERT ON R

REFERENCING NEW ROW AS new

FOR EACH ROW

INSERT INTO R(a,b)

(SELECT DISTINCT R.a, new.b

FROM R

WHERE R.b = new.a)

EXCEPT

(SELECT DISTINCT a,b FROM R)

Suppose table R is empty initially. We issue three commands to insert tuples into R: first we insert (1,2), then we insert (2,3), then we insert (3,4). After some of these inserts, trigger **Rins** may insert further tuples into R, which may activate the trigger recursively. After all the inserts are done, which of these tuples is NOT in table R?

(2,4)

(1,4)

(3,1)

(1,3)

[Q3] Consider the following SQL table declaration:

CREATE TABLE Emps (id INT, ssNo INT, name CHAR(20), managerID INT);

We would like to extend the table declaration to enforce that each of **id** and **ssNo** is a key (by itself), and each value of **managerID** must be one of the values that appears in the **id** attribute of the same table. Which of the following is *not* a legal addition of SQL standard key and/or foreign-key constraints? Note: The addition does not have to achieve all of the stated goals; it only must result in legal SQL.

Add "PRIMARY KEY" after the first INT, and add "REFERENCES Emps(id)" before the closing parenthesis.

Add ", FOREIGN KEY (managerID) REFERENCES Emps(id)" before the closing parenthesis.

 Add "UNIQUE" after each of the first two INT's.

 Add "PRIMARY KEY" after the first INT, and add ", FOREIGN KEY (managerID) REFERENCES Emps(id)" before the closing parenthesis.

[Q4] Here are SQL declarations for two tables S and T:

CREATE TABLE S(c INT PRIMARY KEY, d INT);

CREATE TABLE T(a INT PRIMARY KEY, b INT REFERENCES S(c));

Suppose S(c,d) contains four tuples: (2,10), (3,11), (4,12), (5,13). Suppose T(a,b) contains four tuples: (0,4), (1,5), (2,4), (3,5). As a result of the constraints in the table declarations, certain insertions, deletions, and/or updates on S and T are disallowed. Which of the following modifications will *not* violate any constraint?

Deleting (5,13) from S

Inserting (1,2) into T

Inserting (6,1) into T

Inserting (6,6) into S

[Q5] The following SQL statement over tables R(a,b), S(b,c), and T(a,c) requires certain privileges to execute:

UPDATE R

SET a = 10

WHERE b IN (SELECT c FROM S)

AND NOT EXISTS (SELECT a FROM T WHERE T.a = R.a)

Which of the following privileges is **not** useful for execution of this SQL statement?

SELECT ON R(b)

INSERT ON R(a)

SELECT ON R(a)

SELECT ON S

[Q6] Consider a set of users A, B, C, D, E. Suppose user A creates a table T and thus is the owner of T. Now suppose the following set of statements is executed in order:

1. User A: grant update on T to B,C with grant option

2. User B: grant update on T to D with grant option

3. User C: grant update on T to D with grant option

4. User D: grant update on T to E

5. User A: revoke update on T from C cascade

After execution of statement 5, which of the following is true?

B no longer has privilege UPDATE ON T

A no longer has privilege UPDATE ON T

D and E do not have privilege UPDATE ON T, but B does

Both D and E have privilege UPDATE ON T, but C doesn't

[Q7] The following SQL statement over tables R(c,d), S(f,g), and T(a,b) requires certain privileges to execute:

UPDATE T

SET a=1, b=2

WHERE a <= ALL (SELECT d FROM R)

OR EXISTS (SELECT f FROM S WHERE f > T.a)

Which of the following privileges is **not** useful for execution of this SQL statement?

UPDATE ON T

SELECT ON T

SELECT ON S(g)

SELECT ON R

[Q8] Consider a set of users U, V, W, X, and Y. Suppose user U creates a table T and thus is the owner of T. Now suppose the following set of statements is executed in order:

1. User U: grant select on T to V,W with grant option

2. User V: grant select on T to W

3. User W: grant select on T to X,Y

4. User U: grant select on T to Y

5. User U: revoke select on T from V restrict

6. User U: revoke select on T from W cascade

Which of the following statements is true?

Y has privilege SELECT ON T after statement 6

Y does not have privilege SELECT ON T after statement 6

V has privilege SELECT ON T after statement 5

X does not have SELECT ON T privilege after statement 5