You've started a new movie-rating website, and you've been collecting data on reviewers' ratings of various movies. 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*.

In addition to the base tables, you've created three views:

View **LateRating** contains movie ratings after January 20, 2011. The view contains the movie ID, movie title, number of stars, and rating date.

```
create view LateRating as
  select distinct R.mID, title, stars, ratingDate
  from Rating R, Movie M
  where R.mID = M.mID
  and ratingDate > '2011-01-20'
```

View **HighlyRated** contains movies with at least one rating above 3 stars. The view contains the movie ID and movie title.

```
create view HighlyRated as
  select mID, title
  from Movie
  where mID in (select mID from Rating where stars > 3)
```

View **NoRating** contains movies with no ratings in the database. The view contains the movie ID and movie title.

```
create view NoRating as
  select mID, title
  from Movie
  where mID not in (select mID from Rating)
```

Your exercises will run over a small data set conforming to the schema, with the views predefined. View the database. (You can also download the schema and data.)

Instructions: Each of the problems asks you to enable a certain type of modification to one of the views by writing an "instead-of" trigger. Our back-end creates your trigger using SQLite on the original state of the sample database. It then issues a modification statement on the view, which should activate your trigger to modify the base tables accordingly. Our back-end checks the trigger's base-table modifications, then restores the database to its original state. When you're satisfied with your solution for a given problem, click the "Submit" button to check your answer.

Important Notes:

- Our backend system is SQLite, so you must conform to the instead-of view modification trigger constructs supported by SQLite. A guide to SQLite triggers is here, although you may find it easier to start from the examples in the "View modifications using triggers" video demonstrations.
- *You are to translate the English into a trigger that performs the desired actions for all possible databases and view modifications.* All we actually check is that the verification query gets the right answer on the small sample database. Thus, even if your solution is marked as correct, it is possible that your solution does not correctly reflect the problem at hand. Circumventing the system in this fashion will get you a high score on the exercises, but it won't help you learn about view-modifications. On the other hand, an incorrect attempt at a general solution is unlikely to behave correctly, 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.

problem

0 points possible (ungraded)
We're providing a "sandbox" for experimenting with the data used in this set of exercises.

- Enter any number of queries or modifications (**select**, **insert**, **delete**, or **update** commands) separated by semicolons (;). Your SQL queries and modifications will be executed using SQLite, in order, starting on the initial state of the database. The results will be displayed, then the database restored to its initial state. There are no points awarded for this "problem"; it is provided for your testing purposes only.

```
Enter your SQL command(s) here
```

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

Unanswered

Submit

Q1

1.0/1.0 point (graded)

Write an instead-of trigger that enables updates to the title attribute of view **LateRating**.

Policy: Updates to attribute title in LateRating should update Movie.title for the corresponding movie. (You may assume attribute mID is a key for table Movie.) Make sure the mID attribute of view LateRating has not also been updated -- if it has been updated, don't make any changes. Don't worry about updates to stars or ratingDate.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER LateRatingUpdate
INSTEAD OF UPDATE ON LateRating
FOR EACH ROW
WHEN NEW.mID = OLD.mID
BEGIN
  UPDATE Movie
  SET title = NEW.title
  WHERE mID = OLD.mID;
END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *update LateRating set title = 'Late Favorite' where stars > 2; update LateRating set mID = 100, title = 'Don't change'*.

We then queried the view: *select * from LateRating*
View Result:

101	Late Favorite	2	2011-01-22
101	Late Favorite	4	2011-01-27
103	Late Favorite	3	2011-01-27
104	E.T.	2	2011-01-22
108	Raiders of the Lost Ark	2	2011-01-30

We then ran the following query: *select * from Movie order by mID*
Your Query Result:

101	Late Favorite	1939	Victor Fleming
102	Star Wars	1977	George Lucas
103	Late Favorite	1965	Robert Wise
104	E.T.	1982	Steven Spielberg
105	Titanic	1997	James Cameron
106	Snow White	1937	<NULL>
107	Avatar	2009	James Cameron
108	Raiders of the Lost Ark	1981	Steven Spielberg

Expected Query Result:

101	Late Favorite	1939	Victor Fleming
102	Star Wars	1977	George Lucas
103	Late Favorite	1965	Robert Wise
104	E.T.	1982	Steven Spielberg
105	Titanic	1997	James Cameron
106	Snow White	1937	<NULL>
107	Avatar	2009	James Cameron
108	Raiders of the Lost Ark	1981	Steven Spielberg

Submit

Q2

1.0/1.0 point (graded)

Write an instead-of trigger that enables updates to the stars attribute of view **LateRating**.

Policy: Updates to attribute stars in LateRating should update Rating.stars for the corresponding movie rating. (You may assume attributes [mID,ratingDate] together are a key for table Rating.) Make sure the mID and ratingDate attributes of view LateRating have not also been updated -- if either one has been updated, don't make any changes. Don't worry about updates to title.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER LRupdateR
INSTEAD OF UPDATE ON LateRating
FOR EACH ROW
WHEN NEW.mID = OLD.mID AND NEW.ratingDate = OLD.ratingDate
BEGIN
  UPDATE Rating
  SET stars = NEW.stars
  WHERE mID = OLD.mID AND ratingDate = OLD.ratingDate;
END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *update LateRating set stars = stars - 2 where stars > 2; update LateRating set mID = 100, stars = stars + 2; update LateRating set ratingDate = null, stars = stars + 2*.

We then queried the view: *select * from LateRating*
View Result:

101	Gone with the Wind	2	2011-01-22
101	Gone with the Wind	2	2011-01-27
103	The Sound of Music	1	2011-01-27
104	E.T.	2	2011-01-22
108	Raiders of the Lost Ark	2	2011-01-30

We then ran the following query: *select * from Rating order by mID, stars*
Your Query Result:

201	101	2	2011-01-22
201	101	2	2011-01-27
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	4	2011-01-12
204	101	3	2011-01-09
205	103	1	2011-01-27
205	104	2	2011-01-22
205	108	4	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	5	2011-01-20
208	104	3	2011-01-02

Expected Query Result:

201	101	2	2011-01-22
201	101	2	2011-01-27
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	4	2011-01-12
204	101	3	2011-01-09
205	103	1	2011-01-27
205	104	2	2011-01-22
205	108	4	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	5	2011-01-20
208	104	3	2011-01-02

Submit

Q3

1.0/1.0 point (graded)
Write an instead-of trigger that enables updates to the mID attribute of view **LateRating**.

Policy: Updates to attribute mID in LateRating should update Movie.mID and Rating.mID for the corresponding movie. Update all Rating tuples with the old mID, not just the ones contributing to the view. Don't worry about updates to title, stars, or ratingDate.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER LRupdateMID
INSTEAD OF UPDATE OF mID ON LateRating
FOR EACH ROW
BEGIN
  UPDATE Movie
  SET mID = NEW.mID
  WHERE mID = OLD.mID;
  UPDATE Rating
  SET mID = NEW.mID
  WHERE mID = OLD.mID;
END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *update LateRating set mID = mID+50 where stars = 2*.

We then queried the view: *select * from LateRating*
View Result:

103	The Sound of Music	3	2011-01-27
151	Gone with the Wind	2	2011-01-22
151	Gone with the Wind	4	2011-01-27
154	E.T.	2	2011-01-22
158	Raiders of the Lost Ark	2	2011-01-30

We then ran the following query: *select M.mID, title, stars from Movie M, Rating R where M.mID = R.mID order by M.mID, stars*
Your Query Result:

103	The Sound of Music	2
103	The Sound of Music	3
106	Snow White	4
106	Snow White	5
107	Avatar	3
107	Avatar	5
151	Gone with the Wind	2
151	Gone with the Wind	3
151	Gone with the Wind	4
154	E.T.	2
154	E.T.	3
158	Raiders of the Lost Ark	2
158	Raiders of the Lost Ark	4
158	Raiders of the Lost Ark	4

Expected Query Result:

103	The Sound of Music	2
103	The Sound of Music	3
106	Snow White	4
106	Snow White	5
107	Avatar	3
107	Avatar	5
151	Gone with the Wind	2
151	Gone with the Wind	3
151	Gone with the Wind	4

154	E.T.	2
154	E.T.	3
158	Raiders of the Lost Ark	2
158	Raiders of the Lost Ark	4
158	Raiders of the Lost Ark	4

Submit

Q4

1.0/1.0 point (graded)

Finally, write a single instead-of trigger that combines all three of the previous triggers to enable simultaneous updates to attributes mID, title, and/or stars in view **LateRating**. Combine the view-update policies of the three previous problems, with the exception that mID may now be updated. Make sure the ratingDate attribute of view LateRating has not also been updated -- if it has been updated, don't make any changes.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER LRupdate
INSTEAD OF UPDATE ON LateRating
FOR EACH ROW
WHEN NEW.ratingDate = OLD.ratingDate
BEGIN
  UPDATE Movie
  SET mID = NEW.mID, title = NEW.title
  WHERE mID = OLD.mID;
  UPDATE Rating
  SET mID = NEW.mID
  WHERE mID = OLD.mID;
  UPDATE Rating
  SET stars = NEW.stars
  WHERE mID = NEW.mID AND ratingDate = OLD.ratingDate;
END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *update LateRating set mID = mID+50, title = 'Worth seeing', stars = 5 where stars >= 3; update LateRating set title = 'Mediocre', ratingDate = null where stars = 2*.

We then queried the view: *select * from LateRating*
View Result:

104	E.T.	2	2011-01-22
108	Raiders of the Lost Ark	2	2011-01-30
151	Worth seeing	2	2011-01-22
151	Worth seeing	5	2011-01-27
153	Worth seeing	5	2011-01-27

We then ran the following query: *select M.mID, title, stars from Movie M, Rating R where M.mID = R.mID order by M.mID, stars*
Your Query Result:

104	E.T.	2
104	E.T.	3
106	Snow White	4
106	Snow White	5
107	Avatar	3
107	Avatar	5
108	Raiders of the Lost Ark	2
108	Raiders of the Lost Ark	4
108	Raiders of the Lost Ark	4
151	Worth seeing	2
151	Worth seeing	3
151	Worth seeing	5

153	Worth seeing	2
153	Worth seeing	5

Expected Query Result:

104	E.T.	2
104	E.T.	3
106	Snow White	4
106	Snow White	5
107	Avatar	3
107	Avatar	5
108	Raiders of the Lost Ark	2
108	Raiders of the Lost Ark	4
108	Raiders of the Lost Ark	4
151	Worth seeing	2
151	Worth seeing	3
151	Worth seeing	5
153	Worth seeing	2
153	Worth seeing	5

Submit

Q5

1.0/1.0 point (graded)

Write an instead-of trigger that enables deletions from view **HighlyRated**.

Policy: Deletions from view HighlyRated should delete all ratings for the corresponding movie that have stars > 3.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER HRdel
INSTEAD OF DELETE ON HighlyRated
FOR EACH ROW
BEGIN
  DELETE FROM Rating
  WHERE mID = OLD.mID AND stars > 3;
END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *delete from HighlyRated where mID > 106*.

We then queried the view: *select * from HighlyRated*
View Result:

101	Gone with the Wind
106	Snow White

We then ran the following query: *select * from Rating order by mID desc*
Your Query Result:

201	101	2	2011-01-22

201	101	4	2011-01-27
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
206	106	5	2011-01-19
206	107	3	2011-01-15
208	104	3	2011-01-02

Expected Query Result:

201	101	2	2011-01-22
201	101	4	2011-01-27
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
206	106	5	2011-01-19
206	107	3	2011-01-15
208	104	3	2011-01-02

Submit

Q6

1.0/1.0 point (graded)

Write an instead-of trigger that enables deletions from view **HighlyRated**.

Policy: Deletions from view HighlyRated should update all ratings for the corresponding movie that have stars > 3 so they have stars = 3.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER HRdel2
INSTEAD OF DELETE ON HighlyRated
FOR EACH ROW
BEGIN
  UPDATE Rating
  SET stars = 3
  WHERE mID = OLD.mID AND stars > 3;
END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *delete from HighlyRated where mID > 106*.

We then queried the view: *select * from HighlyRated*
View Result:

101	Gone with the Wind
106	Snow White

We then ran the following query: *select * from Rating order by mID desc*
Your Query Result:

201	101	2	2011-01-22
201	101	4	2011-01-27
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	3	2011-01-12
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
205	108	3	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	3	2011-01-20
208	104	3	2011-01-02

Expected Query Result:

201	101	2	2011-01-22
201	101	4	2011-01-27
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	3	2011-01-12
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
205	108	3	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	3	2011-01-20
208	104	3	2011-01-02

Submit

Q7

1.0/1.0 point (graded)
Write an instead-of trigger that enables insertions into view **HighlyRated**.

Policy: An insertion should be accepted only when the (mID,title) pair already exists in the Movie table. (Otherwise, do nothing.) Insertions into view HighlyRated should add a new rating for the inserted movie with rID = 201, stars = 5, and NULL ratingDate.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER HRinsert
INSTEAD OF INSERT ON HighlyRated
```

```
3 FOR EACH ROW
4 WHEN NEW.mID IN (SELECT mID FROM Movie) AND NEW.title IN (SELECT title FROM Movie)
5 BEGIN
6   INSERT INTO Rating VALUES (201, NEW.mID, 5, NULL);
7 END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *insert into HighlyRated values (104, 'E.T.'); insert into HighlyRated values (105, 'Titanic 2')*.

We then queried the view: *select * from HighlyRated*
View Result:

101	Gone with the Wind
104	E.T.
106	Snow White
107	Avatar
108	Raiders of the Lost Ark

We then ran the following query: *select * from Rating order by stars desc, mID*
Your Query Result:

201	101	2	2011-01-22
201	101	4	2011-01-27
201	104	5	<NULL>
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	4	2011-01-12
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
205	108	4	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	5	2011-01-20
208	104	3	2011-01-02

Expected Query Result:

201	101	2	2011-01-22
201	101	4	2011-01-27
201	104	5	<NULL>
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	4	2011-01-12
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
205	108	4	<NULL>
206	106	5	2011-01-19

206	107	3	2011-01-15
207	107	5	2011-01-20
208	104	3	2011-01-02

Submit

Q8

1.0/1.0 point (graded)

Write an instead-of trigger that enables insertions into view **NoRating**.

Policy: An insertion should be accepted only when the (mID,title) pair already exists in the Movie table. (Otherwise, do nothing.) Insertions into view NoRating should delete all ratings for the corresponding movie.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER NRinsert
INSTEAD OF INSERT ON NoRating
FOR EACH ROW
WHEN NEW.mID IN (SELECT mID FROM Movie) AND NEW.title IN (SELECT title FROM Movie)
BEGIN
  DELETE FROM Rating WHERE mID = NEW.mID;
END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *insert into NoRating values (104, 'E.T.'); insert into NoRating values (110, 'Avatar')*.

We then queried the view: *select * from NoRating*
View Result:

102	Star Wars
104	E.T.
105	Titanic

We then ran the following query: *select * from Rating order by mID*
Your Query Result:

201	101	2	2011-01-22
201	101	4	2011-01-27
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	4	2011-01-12
204	101	3	2011-01-09
205	103	3	2011-01-27
205	108	4	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	5	2011-01-20

Expected Query Result:

201	101	2	2011-01-22

201	101	4	2011-01-27
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	4	2011-01-12
204	101	3	2011-01-09
205	103	3	2011-01-27
205	108	4	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	5	2011-01-20

Submit

Q9

1.0/1.0 point (graded)

Write an instead-of trigger that enables deletions from view **NoRating**.

Policy: Deletions from view NoRating should delete the corresponding movie from the Movie table.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
1 CREATE TRIGGER NRdel
2 INSTEAD OF DELETE ON NoRating
3 FOR EACH ROW
4 BEGIN
5   DELETE FROM Movie WHERE mID = OLD.mID;
6 END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *delete from NoRating where title = 'Titanic'*.

We then queried the view: *select * from NoRating*
View Result:

102	Star Wars

We then ran the following query: *select * from Movie order by title*
Your Query Result:

101	Gone with the Wind	1939	Victor Fleming
102	Star Wars	1977	George Lucas
103	The Sound of Music	1965	Robert Wise
104	E.T.	1982	Steven Spielberg
106	Snow White	1937	<NULL>
107	Avatar	2009	James Cameron
108	Raiders of the Lost Ark	1981	Steven Spielberg

Expected Query Result:

101	Gone with the Wind	1939	Victor Fleming
102	Star Wars	1977	George Lucas
103	The Sound of Music	1965	Robert Wise
104	E.T.	1982	Steven Spielberg
106	Snow White	1937	<NULL>
107	Avatar	2009	James Cameron
108	Raiders of the Lost Ark	1981	Steven Spielberg

Submit

Q10

1.0/1.0 point (graded)

Write an instead-of trigger that enables deletions from view **NoRating**.

Policy: Deletions from view NoRating should add a new rating for the deleted movie with rID = 201, stars = 1, and NULL ratingDate.

- Remember you need to use an instead-of trigger for view modifications as supported by SQLite.

```
CREATE TRIGGER NRdel2
INSTEAD OF DELETE ON NoRating
FOR EACH ROW
BEGIN
  INSERT INTO Rating VALUES (201, OLD.mID, 1, NULL);
END;
```

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

Correct

Correct

Trigger command was executed.

To check your trigger, we first ran the following data modification statement(s): *delete from NoRating*.

We then queried the view: *select * from NoRating*
View Result: *Empty result*

We then ran the following query: *select * from Rating order by stars*
Your Query Result:

201	101	2	2011-01-22
201	101	4	2011-01-27
201	102	1	<NULL>
201	105	1	<NULL>
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	4	2011-01-12
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
205	108	4	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	5	2011-01-20
208	104	3	2011-01-02

Expected Query Result:

201	101	2	2011-01-22
201	101	4	2011-01-27
201	102	1	<NULL>
201	105	1	<NULL>
202	106	4	<NULL>
203	103	2	2011-01-20
203	108	2	2011-01-30
203	108	4	2011-01-12
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
205	108	4	<NULL>
206	106	5	2011-01-19
206	107	3	2011-01-15
207	107	5	2011-01-20
208	104	3	2011-01-02

Submit