***495 Project2 finished by XinTONG***

***Q1:***

***Enforce the constraint that categories of a movie must be either “Romantic”, “Comedy”, “Drama”, or “Action”. Suppose, the default value for the Category field is “Action”. If a non-allowed value is inserted/updated, the category for that tuple must be changed to the default value.***

DROP TRIGGER if exists m\_category1;

delimiter //

CREATE TRIGGER m\_category1

BEFORE UPDATE ON made\_money

FOR EACH ROW

BEGIN

IF new.category=NULL OR

(new.category<> 'Romantic' AND new.category<> 'Comedy' AND new.category<>'Drama' AND new.category<>'Action')

THEN

SET new.category='Action';

END IF;

END;

//

delimiter ;

DROP TRIGGER if exists m\_category2;

delimiter //

CREATE TRIGGER m\_category2

BEFORE INSERT ON made\_money

FOR EACH ROW

BEGIN

IF new.category=NULL OR

(new.category<> 'Romantic' AND new.category<> 'Comedy' AND new.category<>'Drama' AND new.category<>'Action')

THEN

SET new.category='Action';

END IF;

END;

//

delimiter ;

***Q2:***

***Enforce the following condition: A star can only be a part of a “Comedy” movie, only if he/she has performed in at least one “Romantic”, “Comedy”, or “Drama” movie previously. Upon insertion of a tuple violating this (e.g., a Comedy movie associated with a star who has previously done only “Action” movies), the category of the movie must be updated to “Drama”.***

DROP TRIGGER if exists star\_movie;

delimiter //

CREATE TRIGGER star\_movie AFTER INSERT ON appeared\_in

FOR EACH ROW BEGIN

IF (NOT EXISTS (SELECT \*

FROM made\_money, appeared\_in WHERE made\_money.movie=appeared\_in.movie AND new.star=appeared\_in.star

AND new.movie<>appeared\_in.movie

AND (made\_money.category ='Romantic'

OR made\_money.category ='Comedy'

OR made\_money.category ='Drama')

AND made\_money.day\_opened<

(SELECT made\_money.day\_opened from made\_money WHERE made\_money.movie=new.movie)))

THEN UPDATE made\_money SET made\_money.category='DRAMA' WHERE made\_money.category='comedy' AND made\_money.movie=new.movie ;

END IF;

END;

// delimiter ;

***Q3:***

***Enforce the constraint: A star cannot be married to multiple stars simultaneously.***

DROP TRIGGER if exists star\_marriage;

delimiter //

CREATE TRIGGER star\_marriage AFTER INSERT ON married

FOR EACH ROW BEGIN

IF EXISTS(

select distinct \* from in\_couple C left join

married M on C.COUPLE\_NUM = M.COUPLE\_NUM left join divorced D on C.COUPLE\_NUM = D.COUPLE\_NUM,

(select W.STAR from married Ma, in\_couple W, in\_couple H

where new.COUPLE\_NUM = W.COUPLE\_NUM

and new.COUPLE\_NUM = H.COUPLE\_NUM and W.STAR != H.STAR) as FS

where FS.star = C.STAR and C.COUPLE\_NUM != new.COUPLE\_NUM and

new.day >= ifnull(M.DAY, "2020-12-12") and new.day <= ifnull(D.DAY, "2020-12-12"))

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = "ERROR: Check constraint on married.";

END IF;

END;

// delimiter ;

***Q4:***

***Enforce that, a movie must make at least $1,000 in the box office, and cannot make more than 3 billion ($3,000,000,000) in the box office. Also, if a movie  category is “Action”, then it should make at least $10,000, and if category is  “Comedy”, it cannot make more than $1,000,000,000.***

DROP TRIGGER if exists movie\_money;

delimiter //

CREATE TRIGGER movie\_money BEFORE INSERT ON made\_money

FOR EACH ROW

BEGIN

IF EXISTS(

select \* from made\_money A

where new.HOW\_MUCH<1000

or new.how\_much>3000000000

or (new.how\_much<10000 and new.Category='Action')

or (new.how\_much>1000000000 and new.Category='Comedy'))

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = " ERROR: Check constraint on made money.";

END IF;

END;

// delimiter ;

***Q5：***

***Using a trigger, ensure that the divorce date of a couple is at least the same or  after their marriage date. If this is violated, set the divorce date to be the same as the marriage date.***

DROP TRIGGER if exists Divorce\_condition;

delimiter //

CREATE TRIGGER Divorce\_condition BEFORE INSERT ON divorced

FOR EACH ROW BEGIN

IF EXISTS(

select \* from married

where married.couple\_num=new.couple\_num

and new.day<married.day)

THEN

set new.day=(select married.day from married where married.COUPLE\_NUM = new.COUPLE\_NUM);

END IF;

END;

// delimiter ;

***Q6:***

***We want to keep a log file containing data (movie & category) from rows that have been inserted into “MADE\_MONEY” table into the given “LOG\_DATA” table. Use a trigger to accomplish this goal.***

DROP TRIGGER if exists log\_data;

delimiter //

CREATE TRIGGER log\_data AFTER INSERT ON made\_money

FOR EACH ROW

BEGIN

insert into log\_data

values (new.MOVIE, new.Category);

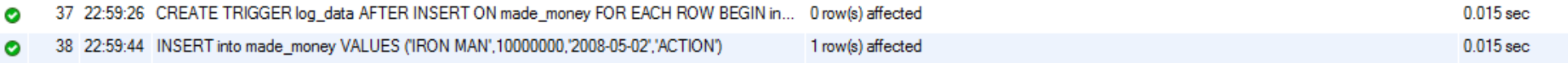
END;

// delimiter ;

1. ***Insert a new movie, with values (“IRON MAN”, 1000000, 2008-05-02,  
   “ACTON”) in MADE\_MONEY table.***

INSERT into made\_money

VALUES ('IRON MAN',10000000,'2008-05-02','ACTION')





***b) Update the CATEGORY of the movie “Fight Club” to “Horror in  
MADE\_MONEY table.***

UPDATE made\_money

SET category='Horror' WHERE movie='Fight Club'





***c) Insert a new tuple in APPEARED\_IN table, with values (“Matt Damon”, “Bruce  
Almighty”).***

INSERT INTO APPEARED\_IN  
VALUES('Matt Damon','Bruce Almighty')





***d) Insert a new tuple in MARRIED, with values (1, 2015-06-26).***

INSERT INTO MARRIED

VALUES(1, '2015-06-26')



***e) Insert two new tuples in MADE\_MONEY, having values (“Most Welcome”,  
8000, 2012-07-07, “Action”) and (“Speed”, 9000, 2010-03-28, “Comedy”).***

INSERT INTO made\_money

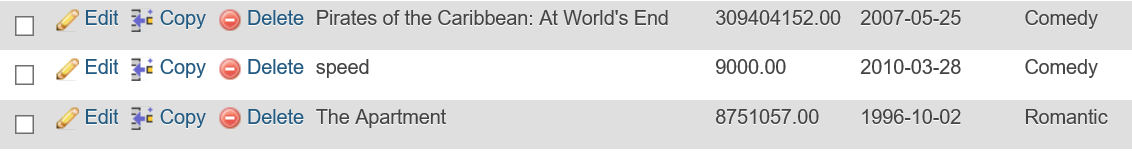
VALUES ('Most Welcome', 8000, '2012-07-07', 'Action')



INSERT INTO made\_money

VALUES ('speed', 9000, '2010-03-28', 'Comedy')





***f) Insert a new tuple in MADE\_MONEY, having values (“Hangover”, 1500000000,  
2011-03-05, “Comedy”).***

INSERT INTO made\_money

VALUES('Hangover', 1500000000, '2011-03-05', 'Comedy')

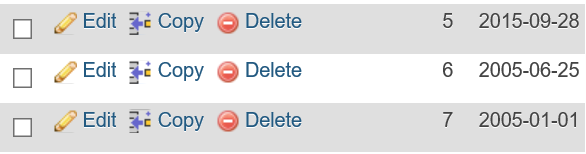


***g) Insert a new tuple in DIVORCED, with values (6, 2004-01-01)***

INSERT INTO divorced

VALUES (6,'2004-01-01')





***Test Q6:***

insert into MADE\_MONEY

values('newmovie', 56667870, '2013-11-01', 'action')



