1. Enforce a unique constraint on movie title and release year to prevent duplicate entries:

Trigger 1:

DELIMITER $$

CREATE TRIGGER enforce\_unique\_title\_year

BEFORE INSERT ON movie\_metadata

FOR EACH ROW

BEGIN

DECLARE existing\_movie INT;

SELECT COUNT(\*) INTO existing\_movie

FROM movie\_metadata

WHERE title = NEW.title AND release\_date = NEW.release\_date;

IF existing\_movie > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Duplicate movie entry!';

END IF;

END

$$

DELIMITER ;

INSERT INTO movie\_metadata (tmdb\_id,title,release\_date) VALUES (2,"Ariel","1988-10-21");

Screenshot 1:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Maintain a table with the latest average rating for each movie (instead of updating the movie table directly):

Trigger 2:

DELIMITER $$

CREATE TRIGGER update\_average\_rating\_log

AFTER INSERT ON ratings

FOR EACH ROW

BEGIN

INSERT INTO movie\_ratings (movie\_id, rating\_date, average\_rating)

VALUES (NEW.movie\_id, CURDATE(), (

SELECT AVG(rating) FROM ratings WHERE movie\_id = NEW.movie\_id

));

END

$$

INSERT INTO ratings(user\_id,movie\_id,rating) values (1,2,5);

SELECT \* from movie\_ratings;

Screenshot 2:

A screenshot of a computer

Description automatically generated

1. Trigger to prevent negative ratings:

Trigger 3:

DELIMITER $$

CREATE TRIGGER prevent\_negative\_rating

BEFORE INSERT ON ratings

FOR EACH ROW

BEGIN

DECLARE rating\_valid INT;

SET rating\_valid = (NEW.rating >= 0 AND NEW.rating <= 5);

IF NOT rating\_valid THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Rating must be between 0 and 5!';

END IF;

END

$$

INSERT INTO ratings(user\_id,movie\_id,rating) values (1,2,-5);

Screenshot:

A screenshot of a computer

Description automatically generated

1. Trigger to keep track of popularity score updation

DELIMITER //

CREATE TRIGGER log\_popularity\_update

AFTER UPDATE ON movie\_metadata

FOR EACH ROW

BEGIN

IF OLD.popularity <> NEW.popularity THEN

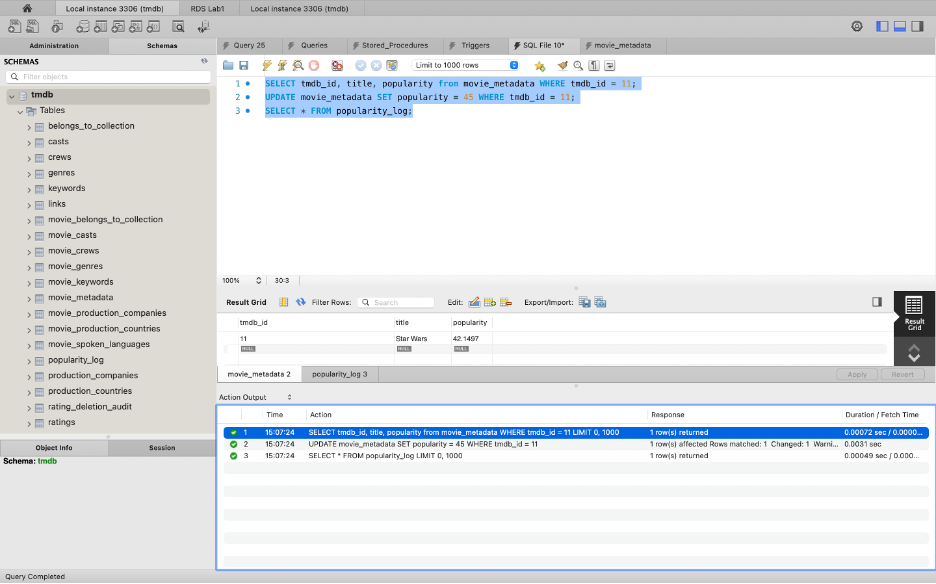
INSERT INTO popularity\_log (tmdb\_id, old\_popularity, new\_popularity, updated\_at)

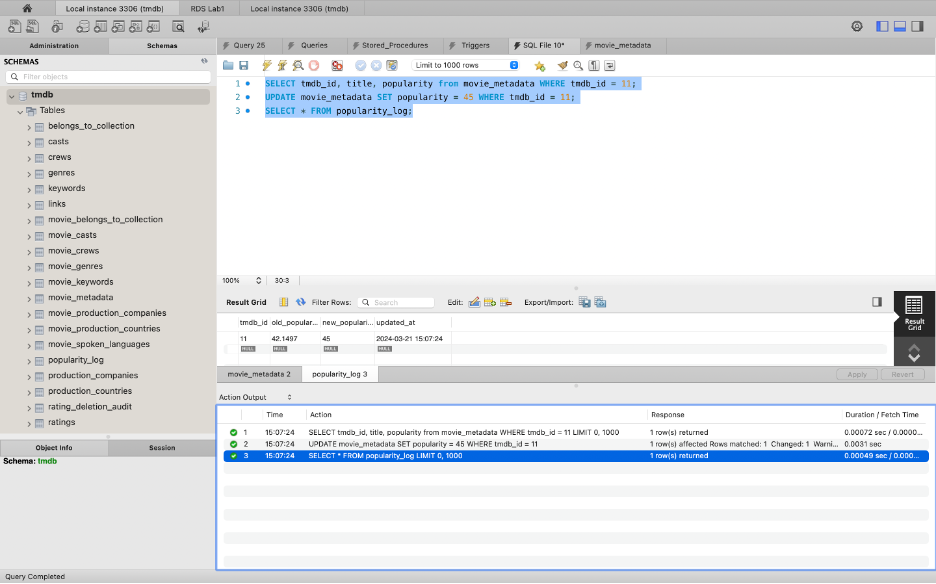
VALUES (NEW.tmdb\_id, OLD.popularity, NEW.popularity, NOW());

END IF;

END //

DELIMITER ;





1. Trigger to add limit number of Genres linked to a movie

DELIMITER //

CREATE TRIGGER check\_genre\_limit

BEFORE INSERT ON movie\_genres

FOR EACH ROW

BEGIN

DECLARE genre\_count INT;

SELECT COUNT(\*) INTO genre\_count

FROM movie\_genres

WHERE tmdb\_id = NEW.tmdb\_id;

IF genre\_count >= 10 THEN

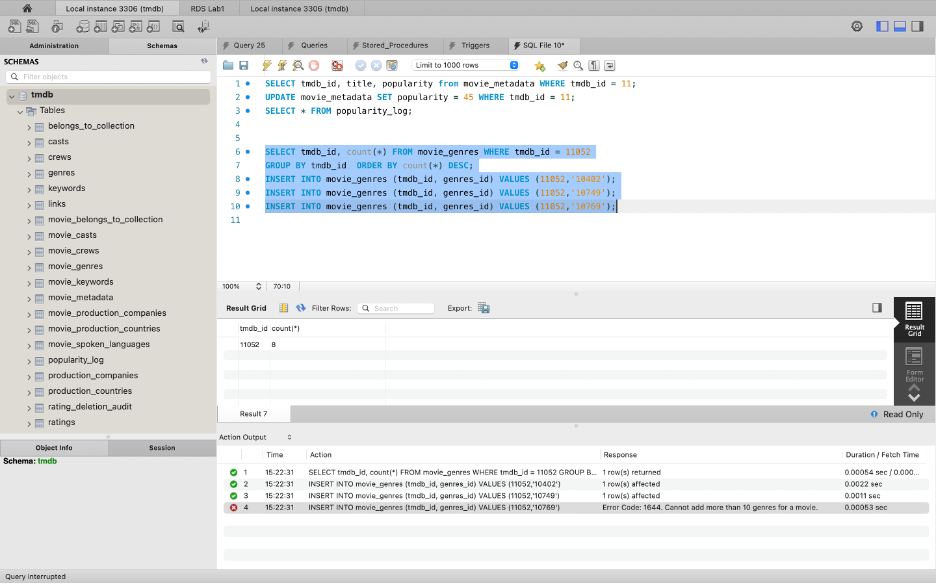
SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot add more than 10 genres for a movie.';

END IF;

END //

DELIMITER ;



1. Trigger to keep track of ratings deletion

DELIMITER //

CREATE TRIGGER rating\_deletion\_audit

AFTER DELETE ON ratings

FOR EACH ROW

BEGIN

INSERT INTO rating\_deletion\_audit (user\_id, movie\_id, rating, deleted\_at)

VALUES (OLD.user\_id, OLD.movie\_id, OLD.rating, NOW());

END //

DELIMITER ;

