**-IMDb Management system-**

Praful Praveen Badami

NUID:001839495

INFO 6210-Database Management and

Database design Final Project

**Problem Statement:**

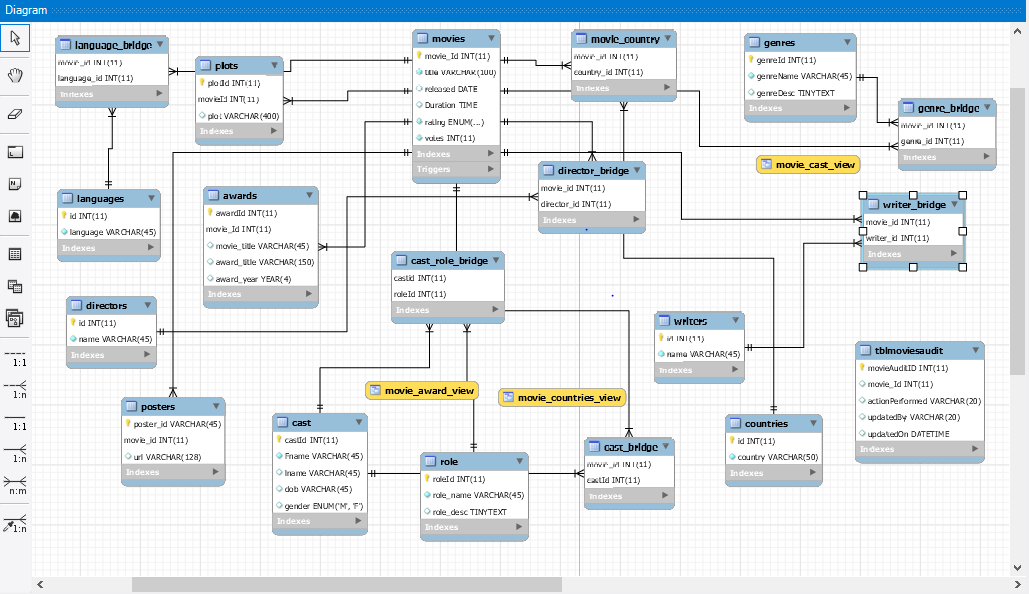
* IMDb is made use by millions of users where there is thousands of movies information stored in one common website.
* IT has all the information regarding Films, TV shows, Cast, reviews, and ratings etc. It stores all the information across all the movies in multiple countries and languages.
* The user can make use of all the information provided to purchase or watch whichever kind of movie they are interested.
* By going through all the reviews, ratings, and information provided any user can make the best decision.
* To maintain all these information, we need a database to store all the information.

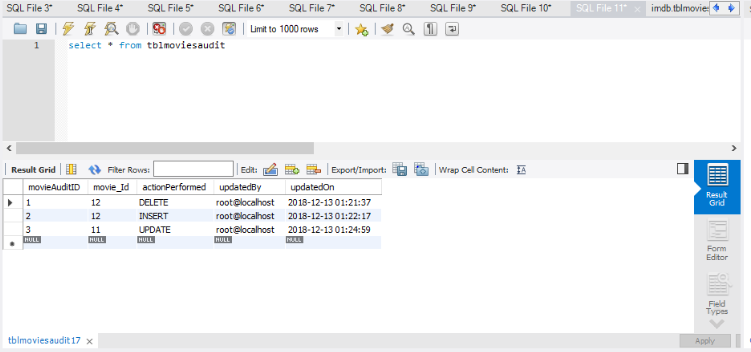
**Entities:**

The Entities that are included in the database are:

* Movies
* Actors
* Director
* Country
* Language
* Awards
* Genre
* Plot
* writer
* cast

**ER Diagram:**

****

**Triggers Implementes: **

**Three** types of **trigger** implemented on one table. They are Insert,Update and Delete.

**1.Update Trigger on Movies:**

DELIMITER //

create trigger tblmoviesauditUpdate

before update

on movies

FOR EACH ROW

Begin

#set @productId= Old.productID;

insert into tblmoviesaudit

(

movie\_Id,

actionPerformed ,

updatedBy,

updatedOn) values (movie\_Id,"UPDATE",user(),now());

END;

//

**2.Delete trigger on Movies:**

DELIMITER //

create trigger tblmoviesauditDelete

before delete

on movies

FOR EACH ROW

Begin

insert into tblmoviesaudit

(

movie\_Id,

actionPerformed ,

updatedBy,

updatedOn) values (OLD.movie\_Id,"DELETE",user(),now());

END;

//

**3.INSERT trigger on Movies:**

DELIMITER //

create trigger tblproductsauditInsert

after insert

on movies

FOR EACH ROW

Begin

insert into tblmoviesaudit

(

movie\_Id,

actionPerformed ,

updatedBy,

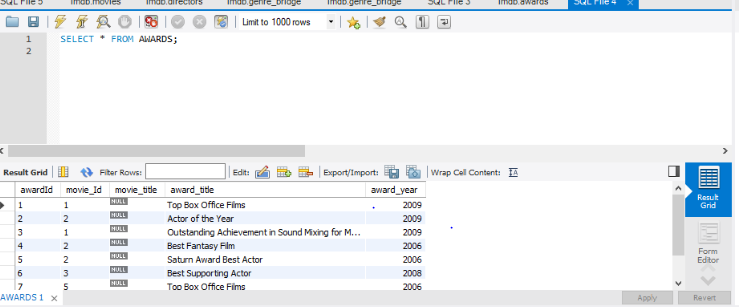
updatedOn) values (NEW.movie\_Id,"DELETE",user(),now());

END;

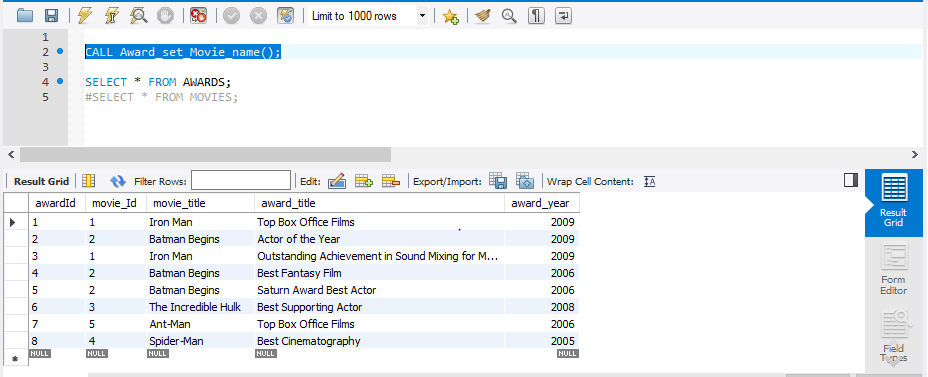
//

**Stored Procedure: using cursors**

**Before calling: procedure Award\_set\_Movie\_name()**

****

**After calling:**

****

DELIMITER //

CREATE PROCEDURE **Award\_set\_Movie\_name()**

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE id INT;

DECLARE set\_title CURSOR FOR SELECT movie\_Id from awards WHERE movie\_title IS NULL;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN set\_title;

search\_loop: LOOP

IF done THEN

LEAVE search\_loop;

END IF;

FETCH set\_title INTO id;

UPDATE awards SET movie\_title = (SELECT title from movies WHERE movie\_Id = id)

WHERE movie\_Id = id;

END LOOP;

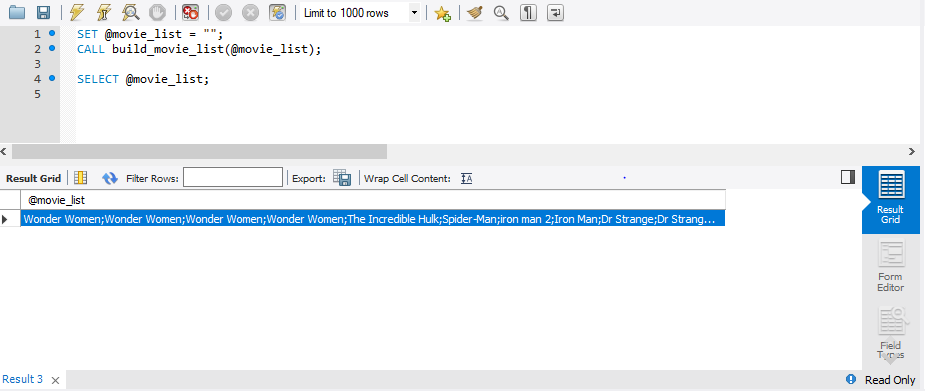
close set\_title;

END;

//

DELIMITER ;

**Stored procedure to generate list of movies using cursor**:



BEGIN

DECLARE done INT DEFAULT 0;

DECLARE movie\_id INT;

DECLARE set\_title CURSOR FOR SELECT awards.movie\_Id from awards WHERE movie\_title IS NULL;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN set\_title;

search\_loop: LOOP

IF done THEN

LEAVE search\_loop;

END IF;

FETCH set\_title INTO movie\_id;

UPDATE awards SET movie\_title = (SELECT title from movies WHERE movies.movie\_Id = ovie\_id)

WHERE movies.movie\_Id = movie\_id;

END LOOP;

close set\_title;

END;

//

DELIMITER ;

**Procedure to create user:**

DELIMITER //

CREATE PROCEDURE **create\_user**

(IN user\_nm CHAR(20), IN user\_ps CHAR(20))

BEGIN

SET @host\_name := 'localhost';

SET @user\_name := user\_nm;

SET @pass := user\_ps;

SET @db\_nm:='imdb';

SET @sql:= CONCAT("CREATE USER ", QUOTE(@user\_name), "@", QUOTE(@host\_name), "IDENTIFIED BY ", QUOTE(@pass));

PREPARE stmt FROM @sql;

EXECUTE stmt;

SET @GRANT\_PERMISSIONS = CONCAT("GRANT ALL ON ", @db\_nm, ".\* TO ", QUOTE(@user\_name), "@",QUOTE(@host\_name));

PREPARE statement FROM @GRANT\_PERMISSIONS;

EXECUTE statement;

END //

DELIMITER ;

**Procedure to find movies:**

CREATE INDEX by\_movies\_name ON movies (title);

DELIMITER $$

CREATE PROCEDURE Search\_movie\_bytitle(IN m varchar(45))

BEGIN

SELECT title,rating

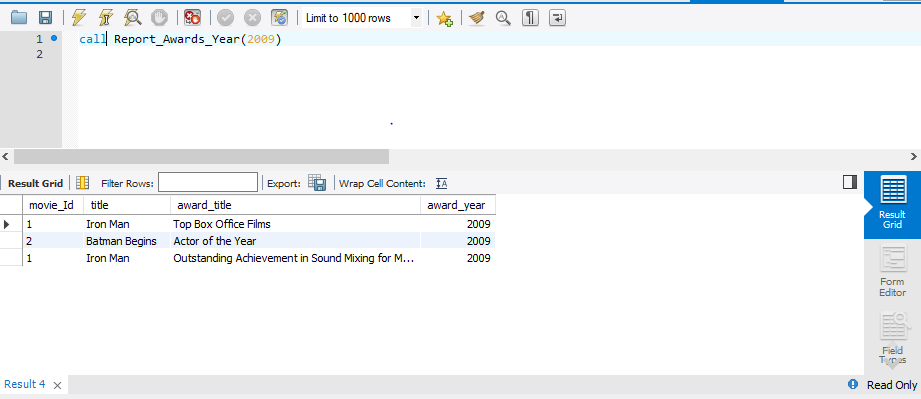
FROM movies

WHERE title = m;

END $$

DELIMITER ;

**Procedure for analysis:**

****

DELIMITER $$

CREATE PROCEDURE Report\_Awards\_Year(IN yr year)

BEGIN

SELECT movies.movie\_Id, movies.title, awards.award\_title,awards.award\_year

FROM movies

INNER JOIN awards

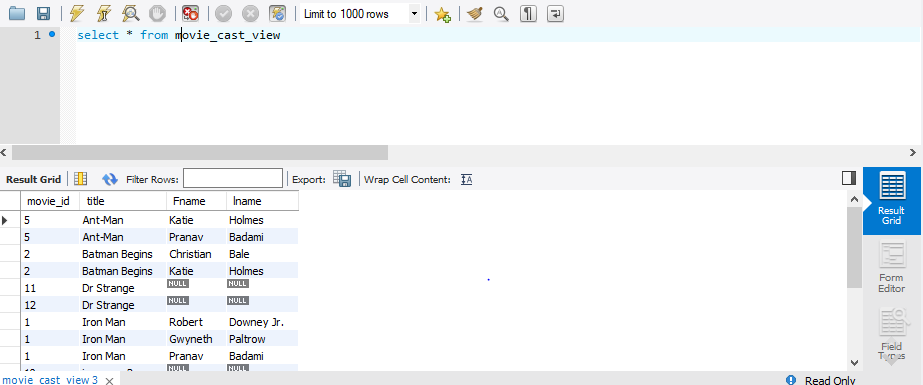
on movies.movie\_Id = awards.movie\_Id where award\_year=yr;

END $$

DELIMITER ;

**VIEWS:**

**movie\_cast\_view-**



create view movie\_cast\_view as SELECT movies.movie\_id, movies.title, cast.Fname,cast.lname

FROM movies

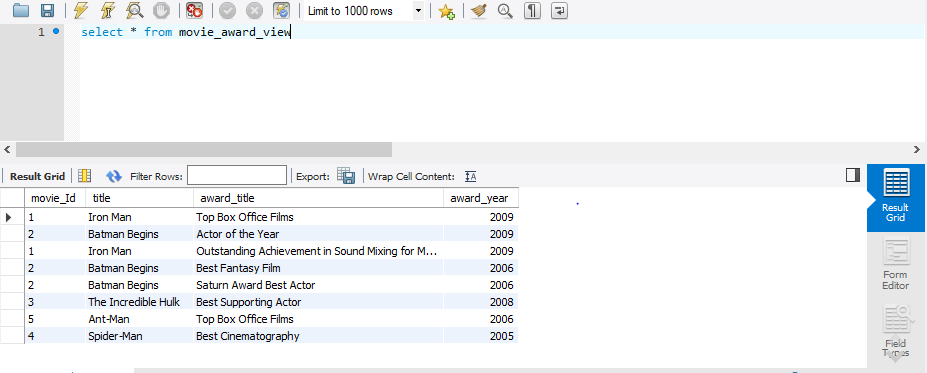
LEFT OUTER JOIN cast\_bridge

ON movies.movie\_id = cast\_bridge.movie\_id

LEFT OUTER JOIN cast

ON cast\_bridge.castId = cast.castId;

**movie\_award\_view--**



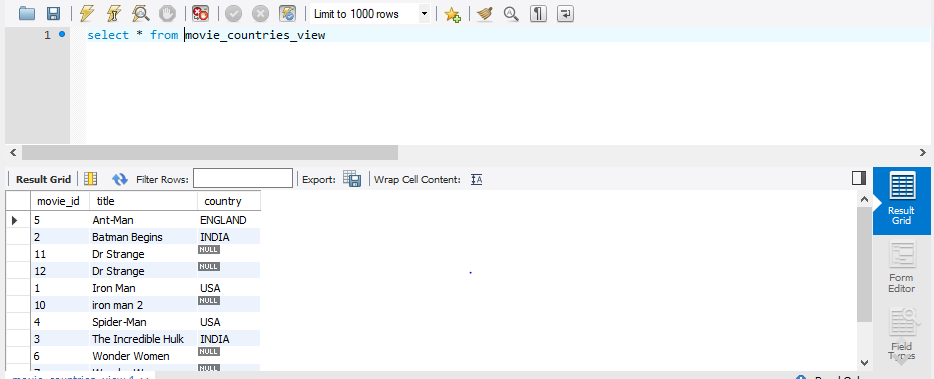
Create View movie\_award\_view as SELECT movies.movie\_Id, movies.title, awards.award\_title,awards.award\_year

FROM movies

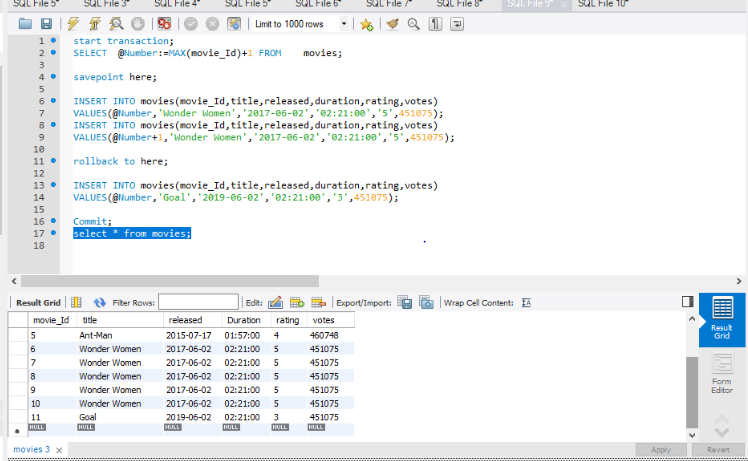
INNER JOIN awards

ON movies.movie\_Id = awards.movie\_Id;

**movie\_countries\_view--**

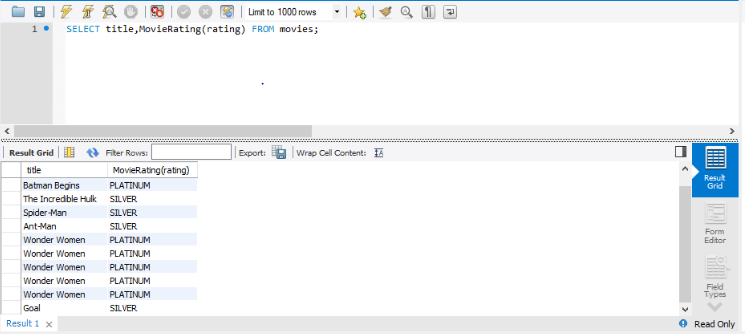


**Transactions:**

****

**User Defined Functions:**

**MovieRating--**

****

DELIMITER $$

CREATE FUNCTION MovieRating(movie\_class ENUM('1','2','3','4','5')) RETURNS VARCHAR(10)

DETERMINISTIC

BEGIN

DECLARE rating varchar(10);

IF movie\_class = '5'THEN

SET rating = 'PLATINUM';

ELSEIF (movie\_class ='4' AND movie\_class = '5') THEN

SET rating = 'GOLD';

ELSE

SET rating = 'SILVER';

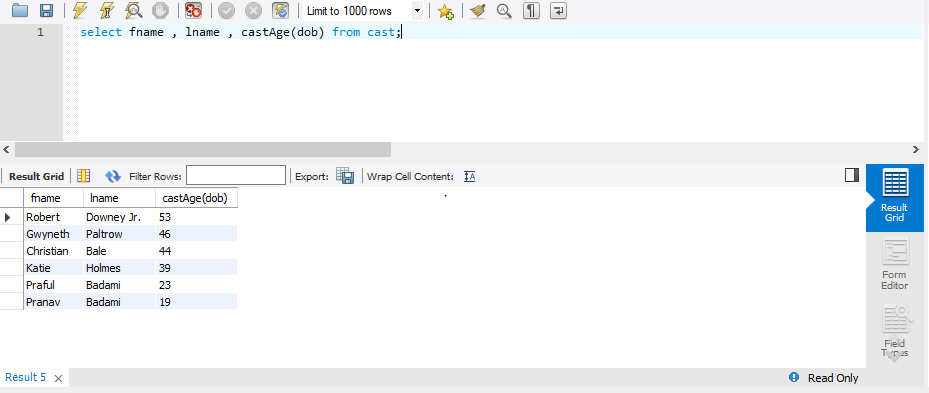
END IF;

RETURN (rating);

END

$$

**castAge--**



CREATE DEFINER=`root`@`localhost` FUNCTION `CastAge`(dob date) RETURNS varchar(10) CHARSET utf8mb4

DETERMINISTIC

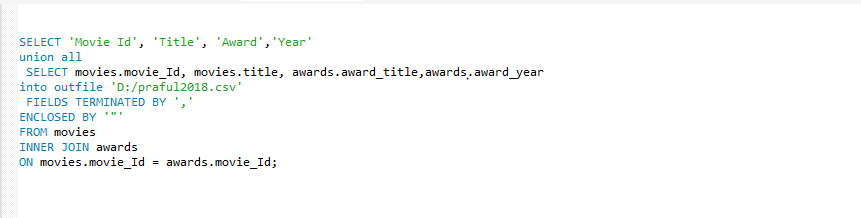
BEGIN

DECLARE age varchar(10);

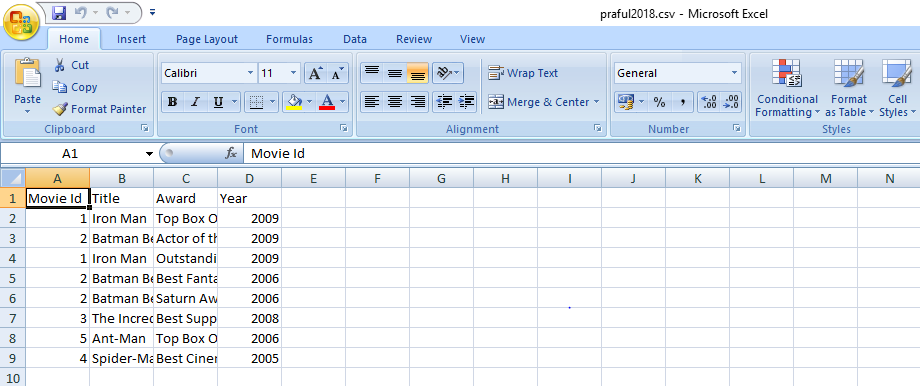
set age=Date\_format( From\_Days( To\_Days(Curdate()) - To\_Days(dob) ), '%Y' ) + 0 ;

RETURN (age);

END

**Out File Report Generation:** 

**File saved in local system:**



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_x\_\_\_\_\_\_\_