A
Project Report
on
Movie Database

Developed by
Parth Shah K. (IT-114) - Department of IT, DD University
Parth Shah N. (IT-115) - Department of IT, DD University

Guided by
Internal Guide:
Sunil K. Vithlani
Department of Information Technology
Faculty of Technology
DD University

Department of Information Technology
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad - 387001
October - 2018

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled "Movie Database" is a bonafide report of the work carried out by

1) Mr. Parth Shah K., Student ID No: 16ITUOS143
2) Mr. Parth Shah N., Student ID No: 16ITUON022

of Department of Information Technology, semester V, under the guidance and supervision for the subject Database Management System. They were involved in Project training during academic year 2018-2019.

Prof. Sunil K. Vithlani
(Project Guide)
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

Prof. Vipul Dabhi
Head, Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher Prof. Sunil K. Vithlani as well as our head of department, Prof. Vipul Dabhi who gave us the golden opportunity to do this project on the topic Movie Database, which also helped us in doing a lot of Research and helped us learn many new things that we are really thankful to.

TABLE OF CONTENTS

SYSTEM OVERVIEW

1.1 CURRENT SYSTEM

This project is a database that stores data for an app that enables users to discover new movies, get information about various movies, search movies using different filters, etc. The database stores information of users such as emails, passwords, favorite genres, country, DOB, identity, name, profile picture path, etc. Moreover, it stores information of movies which include but not limited to movie title, runtime, genres, plot, release date, path of poster and information of people who worked to create the movie. Furthermore the database also stores the data of the reviews and rating that are posted by individual users for any movie.

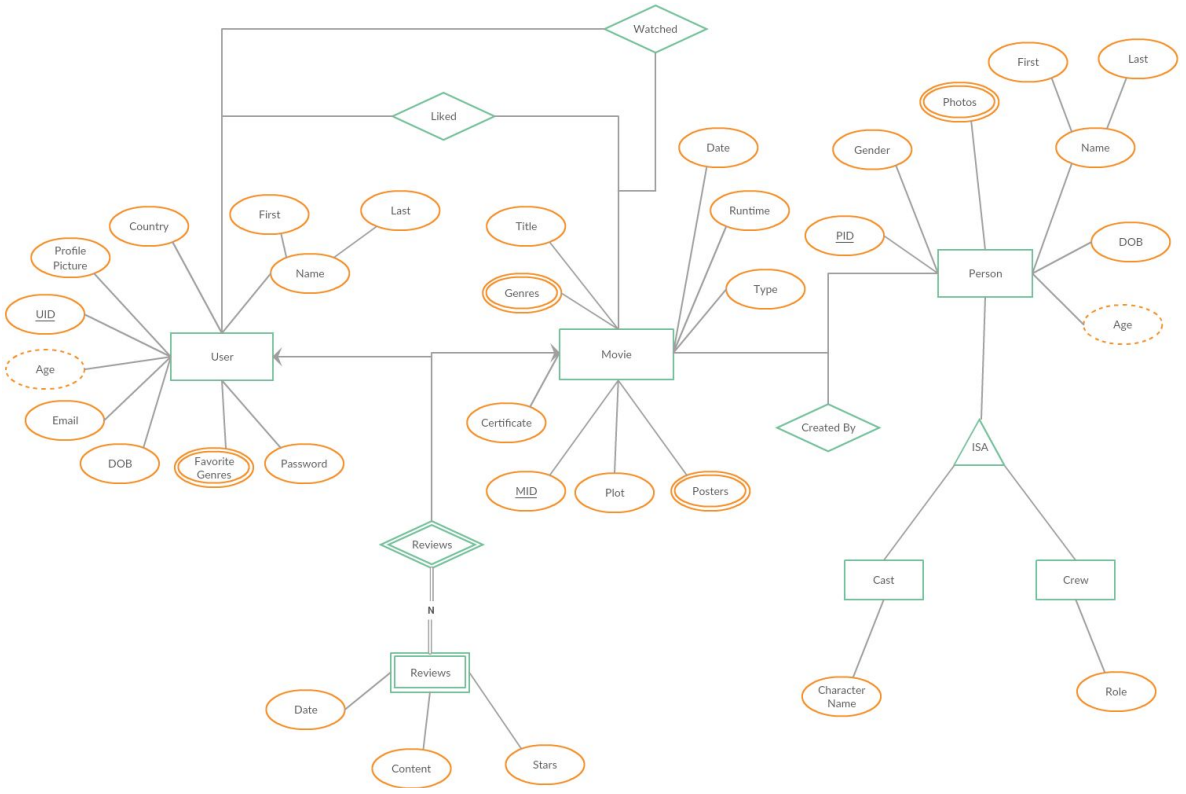
1.2 OBJECTIVES OF THE PROPOSED SYSTEM

The objective of the proposed system should be as follows:
It should allow users to search for movies using detailed and vivid search filters. The application should also recommend the user movies based on their previously watched and like movies, and based on ratings and reviews of other users. These recommendations get better with increase in the use of application by the user. It also allows users to read reviews and ratings posted by other users.

1.3 ADVANTAGES OF CURRENT SYSTEM

The system enables users to discover movies of their choice using the search filters. The recommendations that users get are also helpful in discovering new movies. Moreover, users can read reviews of other movies online. This application can further be scaled to store and retrieve data for music in a similar fashion. All the current uses can be implemented to the newly added music data.

E-R DIAGRAM



2.1 ENTITIES

- User
- Movie
- Review (weak)
- Person
- Cast
- Crew

2.2 RELATIONSHIPS

- Liked (User - Movie)
- Watched (User - Movie)
- Reviews (User - Movie - Review)
- Createdby (Movie - Person)

- ISA (Person - Cast)
- ISA (Person - Crew)

2.3 MAPPING CONSTRAINTS

- Liked :- Many to Many
- Watched :- Many to Many
- Review - Movie :- Many to One
- Review - User :- Many to One
- Createdby :- Many to Many

DATA DICTIONARY

cast

Column	Type	Null	Default	Links to	Comments	MIME
MID *(Primary)*	int(11)	No		movies -> MID		
PID *(Primary)*	int(11)	No		people -> PID		
CharacterName	varchar(255)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	MID	60	A	No	
				PID	60	A	No	
MID	BTREE	No	No	MID	60	A	No	
PID	BTREE	No	No	PID	60	A	No	

createdby

Column	Type	Null	Default	Links to	Comments	MIME
PID *(Primary)*	int(11)	No		people -> PID		
MID *(Primary)*	int(11)	No		movies -> MID		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	PID	45	A	No	
				MID	136	A	No	
PID	BTREE	No	No	PID	45	A	No	
MID	BTREE	No	No	MID	45	A	No	

crew

Column	Type	Null	Default	Links to	Comments	MIME
MID *(Primary)*	int(11)	No		movies -> MID		
PID *(Primary)*	int(11)	No		people -> PID		
Role	varchar(255)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	PID	76	A	No	
				MID	76	A	No	
MID	BTREE	No	No	MID	76	A	No	

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PID	BTREE	No	No	PID	76	A	No	

favgenres

Column	Type	Null	Default	Links to	Comments	MIME
UID *(Primary)*	int(11)	No		users -> UID		
Name *(Primary)*	varchar(255)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	UID	34	A	No	
				Name	34	A	No	
UID	BTREE	No	No	UID	34	A	No	

genres

Column	Type	Null	Default	Links to	Comments	MIME
MID *(Primary)*	int(11)	No		movies -> MID		
Name *(Primary)*	varchar(32)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	MID	42	A	No	
				Name	42	A	No	
MID	BTREE	No	No	MID	42	A	No	

liked

Column	Type	Null	Default	Links to	Comments	MIME
UID *(Primary)*	int(11)	No		users -> UID		
MID *(Primary)*	int(11)	No		movies -> MID		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	UID	36	A	No	
				MID	36	A	No	
UID	BTREE	No	No	UID	36	A	No	
MID	BTREE	No	No	MID	36	A	No	

movies

Column	Type	Null	Default	Links to	Comments	MIME
MID *(Primary)*	int(11)	No				
Title	varchar(255)	No				
ReleaseDate	date	No				
Plot	varchar(1023)	Yes	*NULL*			
Runtime	smallint(6)	Yes	*NULL*			
Type	varchar(255)	Yes	*NULL*			
Certificate	varchar(255)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	MID	20	A	No	

people

Column	Type	Null	Default	Links to	Comments	MIME
PID *(Primary)*	int(11)	No				
FirstName	varchar(255)	Yes	*NULL*			
LastName	varchar(255)	Yes	*NULL*			
Gender	varchar(1)	Yes	*NULL*			
DOB	date	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	PID	20	A	No	

photos

Column	Type	Null	Default	Links to	Comments	MIME
PID *(Primary)*	int(11)	No		people -> PID		
FileName *(Primary)*	varchar(255)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	PID	40	A	No	
				FileName	80	A	No	
FileName	BTREE	Yes	No	FileName	80	A	No	
PID	BTREE	No	No	PID	40	A	No	

posters

Column	Type	Null	Default	Links to	Comments	MIME
MID *(Primary)*	int(11)	No		movies -> MID		
FileName *(Primary)*	varchar(255)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	MID	40	A	No	
				FileName	80	A	No	
FileName	BTREE	Yes	No	FileName	80	A	No	
MID	BTREE	No	No	MID	40	A	No	

reviews

Column	Type	Null	Default	Links to	Comments	MIME
UID *(Primary)*	int(11)	No		users -> UID		
MID *(Primary)*	int(11)	No		movies -> MID		
Date	date	No				
Stars	tinyint(4)	No				
Content	varchar(1023)	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	UID	46	A	No	
				MID	46	A	No	
UID	BTREE	No	No	UID	46	A	No	
MID	BTREE	No	No	MID	46	A	No	

users

Column	Type	Null	Default	Links to	Comments	MIME
UID *(Primary)*	int(11)	No				
FirstName	varchar(255)	No				
LastName	varchar(255)	No				
Country	varchar(2)	Yes	*NULL*			
ProfilePicture	varchar(255)	Yes	*NULL*			
Email	varchar(255)	No				
Password	varchar(255)	No				
DOB	date	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	UID	20	A	No	
ProfilePicture	BTREE	Yes	No	ProfilePicture	20	A	Yes	

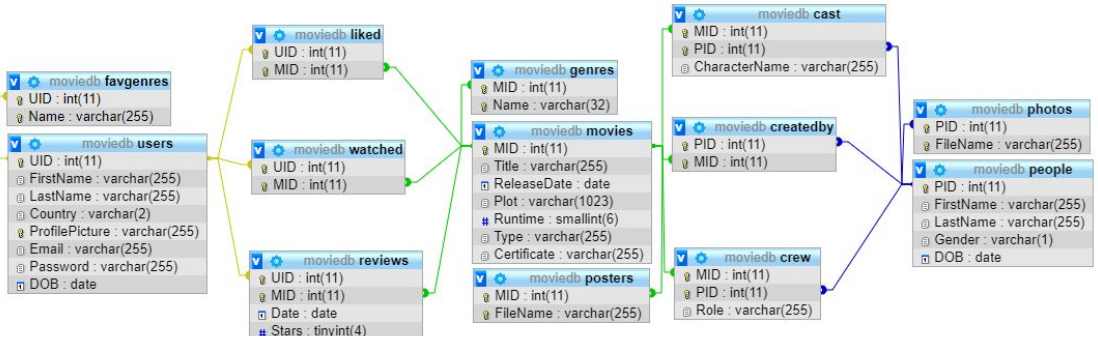
watched

Column	Type	Null	Default	Links to	Comments	MIME
UID *(Primary)*	int(11)	No		users -> UID		
MID *(Primary)*	int(11)	No		movies -> MID		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	UID	42	A	No	
				MID	84	A	No	
UID	BTREE	No	No	UID	42	A	No	
MID	BTREE	No	No	MID	42	A	No	

SCHEMA DIAGRAM



DATABASE IMPLEMENTATIONS

5.1 CREATE SCHEMA

```
CREATE TABLE Users (
   UID int NOT NULL,
   FirstName varchar(255) NOT NULL,
   LastName varchar(255) NOT NULL,
   Country varchar(2),
   ProfilePicture varchar(255),
   Email varchar(255) NOT NULL,
   Password varchar(255) NOT NULL,
   DOB Date,
   PRIMARY KEY (UID),
   UNIQUE(ProfilePicture)
);

CREATE TABLE Movies (
   MID int NOT NULL,
   Title varchar(255) NOT NULL,
   ReleaseDate Date NOT NULL,
   Plot varchar(1023),
   Runtime smallint,
   Type varchar(255),
   Certificate varchar(255) NOT NULL,
   PRIMARY KEY (MID)
);

CREATE TABLE People (
   PID int NOT NULL,
   FirstName varchar(255),
   LastName varchar(255),
   Gender varchar(1),
   DOB Date,
   PRIMARY KEY (PID)
);

CREATE TABLE Genres (
   MID int NOT NULL,
   Name varchar(32) NOT NULL,
   FOREIGN KEY (MID) REFERENCES movies(MID),
```

```
   PRIMARY KEY (MID, Name)
);

CREATE TABLE Posters (
   MID int NOT NULL,
   FileName varchar(255) NOT NULL,
   FOREIGN KEY (MID) REFERENCES movies(MID),
   UNIQUE(FileName),
   PRIMARY KEY (MID, FileName)
);

CREATE TABLE Photos (
   PID int NOT NULL,
   FileName varchar(255) NOT NULL,
   FOREIGN KEY (PID) REFERENCES people(PID),
   UNIQUE(FileName),
   PRIMARY KEY (PID, FileName)
);

CREATE TABLE FavGenres (
   UID int NOT NULL,
   Name varchar(255) NOT NULL,
   FOREIGN KEY (UID) REFERENCES Users(UID),
   PRIMARY KEY (UID, Name)
);

CREATE TABLE Watched (
   UID int NOT NULL,
   MID int NOT NULL,
   FOREIGN KEY (UID) REFERENCES Users(UID),
   FOREIGN KEY (MID) REFERENCES Movies(MID),
   PRIMARY KEY (UID, MID)
);

CREATE TABLE Liked (
   UID int NOT NULL,
   MID int NOT NULL,
   FOREIGN KEY (UID) REFERENCES Users(UID),
   FOREIGN KEY (MID) REFERENCES Movies(MID),
   PRIMARY KEY (UID, MID)
);
```

```
CREATE TABLE CreatedBy (
   PID int NOT NULL,
   MID int NOT NULL,
   FOREIGN KEY (PID) REFERENCES People(PID),
   FOREIGN KEY (MID) REFERENCES Movies(MID),
   PRIMARY KEY (PID, MID)
);

CREATE TABLE Reviews (
   UID int NOT NULL,
   MID int NOT NULL,
   Date Date NOT NULL,
   Stars tinyint NOT NULL,
   Content varchar(1023),
   FOREIGN KEY (UID) REFERENCES Users(UID),
   FOREIGN KEY (MID) REFERENCES Movies(MID),
   PRIMARY KEY (UID, MID)
);

CREATE TABLE Cast (
   MID int NOT NULL,
   PID int NOT NULL,
   CharacterName varchar(255) NOT NULL,
   FOREIGN KEY (MID) REFERENCES Movies(MID),
   FOREIGN KEY (PID) REFERENCES People(PID),
   PRIMARY KEY (MID, PID)
);

CREATE TABLE Crew (
   MID int NOT NULL,
   PID int NOT NULL,
   Role varchar(255) NOT NULL,
   FOREIGN KEY (MID) REFERENCES Movies(MID),
   FOREIGN KEY (PID) REFERENCES People(PID),
   PRIMARY KEY (MID, PID)
);
```

5.2 INSERT DATA VALUES

1. ADD NEW USER

```
INSERT INTO
users (UID, FirstName, LastName, Country, ProfilePicture,
Email, Password, DOB)
VALUES (1, 'Ryan', 'Mollin', 'AL', 'dQc2XWyAFqxvlQXp.jpg',
'rmollin0@xrea.com', 'UBT65gm', '2002-01-16');
```

2. ADD FAVOURITE GENRES OF A USER

```
INSERT INTO
favgenres (UID, Name)
VALUES
(1, 'Horror'),
(1, 'Mystery');
```

3. ADD REVIEW FROM A USER

```
INSERT INTO
reviews (UID, MID, Date, Stars, Content)
VALUES
(2, 4, '2018-08-06', 4, 'Duis aliquam convallis nunc. Proin at
turpis a pede posuere nonummy. Integer non velit.');
```

4. ADD LIKED MOVIES

```
INSERT INTO
liked (UID, MID)
VALUES
(1, 3),
(1, 9);
```

5. ADD WATCHED MOVIES

```
INSERT INTO
watched (UID, MID)
VALUES
(1, 1),
```

```
(1, 3),
(1, 9),
(1, 17);
```

6. ADD PEOPLE

```
INSERT INTO
people (PID, FirstName, LastName, Gender, DOB)
VALUES
(1, 'Bonni', 'Shieldon', 'F', '1989-04-09');
```

7. ADD PHOTOS OF PEOPLE

```
INSERT INTO
photos (PID, FileName)
VALUES
(1, 'dVvC6FYupGDt69iV.jpg'),
(1, 'JRl5Ash7ioWrMrK5.jpg'),
(1, 'Pcg1CLV9aakEHjvs.jpg'),
(1, 'rtfVnMVse4Tj23lO.jpg'),
(1, 'tXH7mMbCVSJLqPd9.jpg'),
(1, 'wzUh2EDruHhHplDm.jpg'),
(1, 'YTGCH7eRMSI3gnHj.jpg');
```

8. ADD NEW MOVIES

```
INSERT INTO
movies (MID, Title, ReleaseDate, Plot, Runtime, Type,
Certificate)
VALUES
(1, 'The Avengers', '2016-10-26', 'Duis aliquam convallis
nunc. Proin at turpis a pede posuere nonummy. Integer non
velit.', 139, 'Feature', 'PG-13');
```

9. INSERT GENRES OF MOVIES

```
INSERT INTO
genres (MID, Name)
VALUES
(1, 'Drama'),
(1, 'Horror');
```

10. ADD POSTERS OF A MOVIE

```
INSERT INTO
posters (MID, FileName)
VALUES
(1, 'dVvC6FYupGDt69iV.jpg'),
(1, 'JRl5Ash7ioWrMrK5.jpg'),
(1, 'Pcg1CLV9aakEHjvs.jpg'),
(1, 'rtfVnMVse4Tj23lO.jpg'),
(1, 'tXH7mMbCVSJLqPd9.jpg'),
(1, 'wzUh2EDruHhHplDm.jpg'),
(1, 'YTGCH7eRMSI3gnHj.jpg');
```

11. ADD CREDITS TO A MOVIE

```
INSERT INTO
createdby (PID, MID)
VALUES
(2, 1),
(9, 1),
(12, 1),
(13, 1),
(15, 1),
(16, 1),
(17, 1),
(19, 1);
```

12. DEFINE CAST MEMBERS OF A MOVIE

```
INSERT INTO
cast (MID, PID, CharacterName)
VALUES
(1, 2, 'Joshua'),
(1, 15, 'Brooke');
```

13. DEFINE CREW MEMBERS OF A MOVIE

```
INSERT INTO
Crew (MID, PID, Role)
VALUES
(1, 9),
```

```
(1, 12),
(1, 13),
(1, 16),
(1, 17),
(1, 19);
```

5.3 QUERIES

1. FIND MOVIES OF A PARTICULAR GENRE

```
SELECT m.Title, m.ReleaseDate, m.Plot, m.Runtime
FROM movies m
INNER JOIN genres g ON m.MID=g.MID
WHERE g.Name='Horror'
```

2. GET ALL REVIEWS OF A PARTICULAR MOVIE

```
SELECT CONCAT(u.FirstName, ' ', u.LastName) AS User, r.Stars
AS Rating, r.Date, r.Content AS Review
FROM reviews r
INNER JOIN movies m ON r.MID=m.MID
INNER JOIN users u ON u.UID=r.UID
WHERE m.Title='The Secret Life'
ORDER BY r.Date DESC
```

3. LIST ALL CAST MEMBERS OF A MOVIE

```
SELECT CONCAT(p.FirstName, " ", p.LastName) AS `Actor Name`,
c.CharacterName AS `Character Name` FROM createdby cb
INNER JOIN movies m ON cb.MID=m.MID
INNER JOIN cast c ON c.MID=cb.MID AND c.PID=cb.PID
INNER JOIN people p ON cb.PID=p.PID
WHERE m.title='The Avengers'
```

4. LIST ALL CREW MEMBERS OF A MOVIE

```
SELECT CONCAT(p.FirstName, " ", p.LastName) AS `Crew Member
Name`, c.Role AS `Character Name`
FROM createdby cb
INNER JOIN movies m ON cb.MID=m.MID
INNER JOIN crew c ON c.MID=cb.MID AND c.PID=cb.PID
INNER JOIN people p ON cb.PID=p.PID
WHERE m.title='The Avengers'
```

5. LIST ALL MOVIES OF FAVORITE GENRES

```
SELECT m.Title AS Movie, g.Name AS `Genre Match`
FROM movies m
INNER JOIN genres g ON m.MID=g.MID
INNER JOIN favgenres f ON f.Name=g.Name
INNER JOIN users u ON f.UID=u.UID
WHERE u.UID=1
```

6. GET ALL INFORMATION OF A MOVIE

```
SELECT m.Title, m.ReleaseDate AS 'Release Date', m.Plot,
m.runtime, GROUP_CONCAT(g.Name) AS Genres
FROM movies m
INNER JOIN genres g ON m.MID=g.MID
WHERE m.Title = 'The Avengers'
GROUP BY m.MID
```

7. GET MOVIES RELEASED IN A PARTICULAR DATE RANGE

```
SELECT m.Title, m.ReleaseDate AS 'Release Date'
FROM movies m
WHERE m.ReleaseDate BETWEEN '2017-01-01' AND '2018-01-01'
ORDER BY m.ReleaseDate ASC
```

8. FIND ALL MOVIES A PERSON HAS STARRED IN

```
SELECT m.Title AS Movie, c.CharacterName as 'Character Name'
FROM movies m
INNER JOIN cast c ON m.MID=c.MID
INNER JOIN people p ON c.PID=p.PID
```

```
WHERE p.PID=1
```

9. FIND ALL MOVIES A PERSON HAS WORKED IN AS CREW MEMBER

```
SELECT m.Title AS Movie, c.Role
FROM movies m
INNER JOIN crew c ON m.MID=c.MID
INNER JOIN people p ON c.PID=p.PID
WHERE p.PID=1
```

10. ALL LIKED MOVIES OF A USER

```
SELECT m.Title
FROM movies m
INNER JOIN liked l ON m.MID=l.MID
INNER JOIN users u ON u.UID=l.UID
WHERE u.UID=1
```

11. ALL WATCHED MOVIES OF A USER

```
SELECT m.Title
FROM movies m
INNER JOIN watched w ON m.MID=w.MID
INNER JOIN users u ON u.UID=w.UID
WHERE u.UID=1
```

12. LIST ALL REVIEWS BY A USER

```
SELECT m.Title AS Movie, r.Date, r.Stars AS Rating, r.Content
AS Review
FROM reviews r
INNER JOIN movies m ON r.MID=m.MID
WHERE r.UID=2
ORDER BY r.Date DESC
```

13. ALL CAST MEMBERS OF USER'S LIKED MOVIES

```
SELECT CONCAT(p.FirstName, ' ', p.LastName) AS Actor, m.Title
AS Movie, c.CharacterName AS 'Character Name'
FROM people p
INNER JOIN cast c ON p.PID=c.PID
```

```
INNER JOIN liked l ON l.MID=c.MID
INNER JOIN movies m ON c.MID=m.MID
WHERE l.UID=2
```

14. LIST ALL MOVIES WITH A RATING GREATER THAN A PARTICULAR RATING

```
SELECT DISTINCT m.Title, r.Stars
FROM movies m
INNER JOIN reviews r ON r.MID=m.MID
WHERE r.Stars > 4
```

15. LIST ALL PHOTOS OF A PERSON

```
SELECT p.FileName
FROM photos p
WHERE p.PID=1
```

16. LIST ALL FAVORITE GENRE

```
SELECT f.Name AS Genres
FROM favgenres f
WHERE f.UID=1
```

17. LIST ALL GENRES OF A MOVIE

```
SELECT g.Name AS Genres
FROM genres g
WHERE g.MID=1
```

18. LIST ALL POSTER OF A MOVIE

```
SELECT p.FileName
FROM posters p
WHERE MID=1
```

19. LIST OF NUMBER OF MOVIES FOR EACH CERTIFICATE

```
SELECT COUNT(MID) AS 'Number of Movies', Certificate
FROM movies
```

```
GROUP BY Certificate
```

20. WHICH TYPE OF CERTIFICATE OF MOVIES IS AVAILABLE IN MAJORITY

```
SELECT COUNT(MID) AS 'Number of Movies', Certificate
FROM movies
GROUP BY Certificate
HAVING COUNT(MID) > (SELECT (COUNT(MID)/2)-1 FROM movies)
```

21. AVERAGE RUNTIME OF MOVIES

```
SELECT AVG(Runtime) AS 'Average Runtime of Movies'
FROM MOVIES
```

22. SEARCH MOVIE BY SUBSTRING OF TITLE

```
SELECT *
FROM movies
WHERE Title
LIKE '%ave%'
```

23. LIST MOVIES OF PARTICULAR CERTIFICATES

```
SELECT *
FROM movies
WHERE Certificate
IN ('PG-13', 'PG')
```

24. LIST ALL NON-ADULT MOVIES

```
SELECT *
FROM movies
WHERE Certificate
NOT IN ('R')
```

25. LIST ALL GENRES THAT NO USER AS LIKED A MOVIE OF

```
SELECT Name AS Genre
FROM genres
```

```
WHERE Name
NOT IN (
    SELECT Name
    FROM favgenres
)
```

26. LIST OF ALL CAST AND CREW MEMBERS

```
SELECT CONCAT(p.FirstName, " ", p.LastName) AS `Person Name`
FROM createdby cb
INNER JOIN movies m ON cb.MID=m.MID
INNER JOIN cast c ON c.MID=cb.MID AND c.PID=cb.PID
INNER JOIN people p ON cb.PID=p.PID
WHERE m.title='The Avengers'
UNION
SELECT CONCAT(p.FirstName, " ", p.LastName) AS `Crew Member
Name`
FROM createdby cb
INNER JOIN movies m ON cb.MID=m.MID
INNER JOIN crew c ON c.MID=cb.MID AND c.PID=cb.PID
INNER JOIN people p ON cb.PID=p.PID
WHERE m.title='The Avengers'
```

27. RUNTIME GREATER THAN A PARTICULAR VALUE

```
SELECT Title
FROM movies
WHERE Runtime =
ANY (
    SELECT Runtime
    FROM movies
    WHERE Runtime>100
);
```

28. AVERAGE RATING OF A MOVIE

```
SELECT AVG(Stars) AS Rating
FROM reviews
WHERE MID = 2
```

5.4 PL / SQL (PROCEDURES and FUNCTIONS)

1. GET ALL MOVIES WITH RUNTIME GREATER THAN THE GIVEN VALUE

```
DELIMITER //
CREATE PROCEDURE MovieFromRuntime(IN rt SMALLINT)
    BEGIN
        SELECT Title
        FROM movies
        WHERE Runtime =
        ANY (
            SELECT Runtime
            FROM movies
            WHERE Runtime>rt
        );
    END //
DELIMITER ;
```

2. COUNT NUMBER OF MALE AND FEMALE FROM PEOPLE

```
DELIMITER //
CREATE PROCEDURE CountGender()
    BEGIN

        DECLARE Male INT DEFAULT 0;
        DECLARE Female INT DEFAULT 0;

        SELECT COUNT(PID) INTO Male FROM people WHERE Gender =
'M';
        SELECT COUNT(PID) INTO Female FROM people WHERE Gender
= 'F';

        SELECT Male, Female;
    END //
DELIMITER ;
```

3. INSERT INTO CAST TABLE, EXCEPTION IF DUPLICATE KEY

```
DELIMITER //
```

```
CREATE PROCEDURE InsertCast(IN mi INT, IN pi INT, IN cn
VARCHAR(255))
    BEGIN

        DECLARE CONTINUE HANDLER FOR 1062
        SELECT CONCAT('Duplicate Keys (',mi,',',pi,') Found')
AS Error;

        INSERT INTO cast VALUES (mi, pi, cn);

    END //
DELIMITER ;
```

4. INSERT INTO REVIEWS WITH VALID RATING VALUE

```
DELIMITER //
CREATE PROCEDURE AddReview(IN ui INT, IN mi INT, IN d date, IN
s tinyint, IN c VARCHAR(1023))
    BEGIN

        IF(s > 5) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Rating should be less than or
equal to 5';
        ELSE
            INSERT INTO Reviews VALUES (ui, mi, d, s, c);
        END IF;

    END //
DELIMITER ;
```

5. CHECKS IF THE MOVIE IS AN ADULT MOVIE OR NOT

```
DELIMITER //
CREATE FUNCTION AdultMovieCheck(name varchar(255)) RETURNS
VARCHAR(255)
    DETERMINISTIC
BEGIN

    DECLARE cert VARCHAR(255);
    DECLARE ans BOOLEAN;
```

```
    SELECT Certificate INTO cert FROM movies WHERE Title =
name;

    IF cert = 'R' OR cert = 'NC-17' THEN
        SET ans = TRUE;
    ELSE
        SET ans = FALSE;
    END IF;

    RETURN (ans);
END //
DELIMITER ;
```

6. RATES THE MOVIE BASED ON THE STARS IT REVEIVED

```
DELIMITER //
CREATE FUNCTION AverageRating(mi INT) RETURNS VARCHAR(255)
BEGIN

    DECLARE r INT;
    DECLARE ans VARCHAR(255);

    SELECT AVG(Stars) INTO r FROM reviews WHERE MID = mi;

    IF r = 5 THEN
        SET ans = 'Excellent';
    ELSEIF (r >= 4 AND r < 5) THEN
        SET ans = 'GOOD';
    ELSEIF (r >= 3 AND r < 4) THEN
        SET ans = 'Average';
    ELSEIF (r >= 2 AND r < 3) THEN
        SET ans = 'Poor';
    ELSEIF r < 2 THEN
        SET ans = 'Very Bad';
    END IF;

    RETURN (ans);
END // DELIMITER ;
```

5.5 Views

1. LIST ALL MOVIE DETAILS IN ONE TOUPLE

```
CREATE VIEW movie_details AS
SELECT m.Title, m.ReleaseDate AS 'Release Date', m.Plot,
m.Runtime, m.Type, m.Certificate, GROUP_CONCAT(g.Name) AS
Genres
FROM movies m
INNER JOIN genres g ON m.MID=g.MID
GROUP BY m.MID
```

2. LIST RATINGS OF MOVIES

```
CREATE VIEW ratings AS
SELECT m.Title, AVG(r.Stars) AS Rating
FROM reviews r
INNER JOIN movies m ON r.MID = m.MID
GROUP BY r.MID
```

5.6 Triggers

1. CHECK MOVIE TYPE WITH RUNTIME

```
DELIMITER //
CREATE TRIGGER MovieTypeCheck BEFORE INSERT ON movies
    FOR EACH ROW
        IF (NEW.Runtime < 40) THEN
            SET NEW.Type = 'Short';
        ELSEIF (NEW.Runtime >= 40) THEN
            SET NEW.Type = 'Feature';
        END IF; //
DELIMITER ;
```

5.7 Cursors

1. COUNT THE NUMBER OF MOVIES OF A PARTICULAR GENRE

```
DELIMITER //
CREATE PROCEDURE CountGenres(IN gen VARCHAR(255))
    BEGIN

        DECLARE GenreCount VARCHAR(255) DEFAULT FALSE;
        DECLARE finished INT DEFAULT 0;
        DECLARE x VARCHAR(255);

        DECLARE genreCounter CURSOR FOR SELECT genres.Name
FROM moviedb.genres WHERE genres.Name = gen;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished =
TRUE;

        OPEN genreCounter;

        label1: LOOP
            FETCH genreCounter INTO x;
            IF finished THEN
                LEAVE label1;
            END IF;
            SET GenreCount = GenreCount + 1;
        END LOOP label1;

        CLOSE genreCounter;

        SELECT GenreCount;

    END //
DELIMITER ;
```

2. PREPARE MAILING LIST

```
DELIMITER $$

CREATE PROCEDURE MailingList (INOUT email_list varchar(4000))
BEGIN
```

```
    DECLARE v_finished INTEGER DEFAULT FALSE;
    DECLARE v_email varchar(100) DEFAULT "";


    DEClARE email_cursor CURSOR FOR SELECT Email FROM users;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished =
TRUE;

    OPEN email_cursor;

    get_email: LOOP

        FETCH email_cursor INTO v_email;

        IF v_finished THEN
            LEAVE get_email;
        END IF;

        SET email_list = CONCAT(v_email,";",email_list);

    END LOOP get_email;

    CLOSE email_cursor;
END$$
DELIMITER ;
SET @email_list = "";
CALL MailingList(@email_list);
SELECT @email_list;
```

FUTURE ENHANCEMENTS OF THE SYSTEM

This system can further be scaled to store and retrieve data for music in a similar fashion. All the current uses can be implemented to the newly added music data. Moreover it can be scaled to store more information for individual movies.

BIBLIOGRAPHY

- Database System Concepts - Fourth Edition by Silberschatz-Korth-Sudarshan