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

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our project guide, Prof. Sunil K. Vithlani, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report. Furthermore we would also like to acknowledge with much appreciation the role of our head of department Prof. Vipul Dabhi, for giving us this opportunity which enabled us in doing a lot of research on this subject.

TABLE OF CONTENTS

I. Certificate **1**

II. Acknowledgement **2**

1. SYSTEM OVERVIEW **4**

1.1 Current system 4
1.2 Objectives of the Proposed System 4
1.3 Advantages of the Proposed system (over current) 4

2. E-R DIAGRAM **5**

2.1 Entities 5
2.2 Relationships 5
2.3 Mapping Constraints 6

3. DATA DICTIONARY **7**

4. SCHEMA DIAGRAM **12**

5. DATABASE IMPLEMENTATION **13**

5.1 Create Schema 13
5.2 Insert Data values 17
5.3 Queries (Based on functions, group by, having, joins, subquery etc.) 21
5.4 PL/SQL Blocks (Procedures and Functions) 32
5.5 Views 36
5.6 Triggers 37
5.7 Cursors. 38

6. FUTURE ENHANCEMENTS OF THE SYSTEM **40**

7. BIBLIOGRAPHY **41**

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.

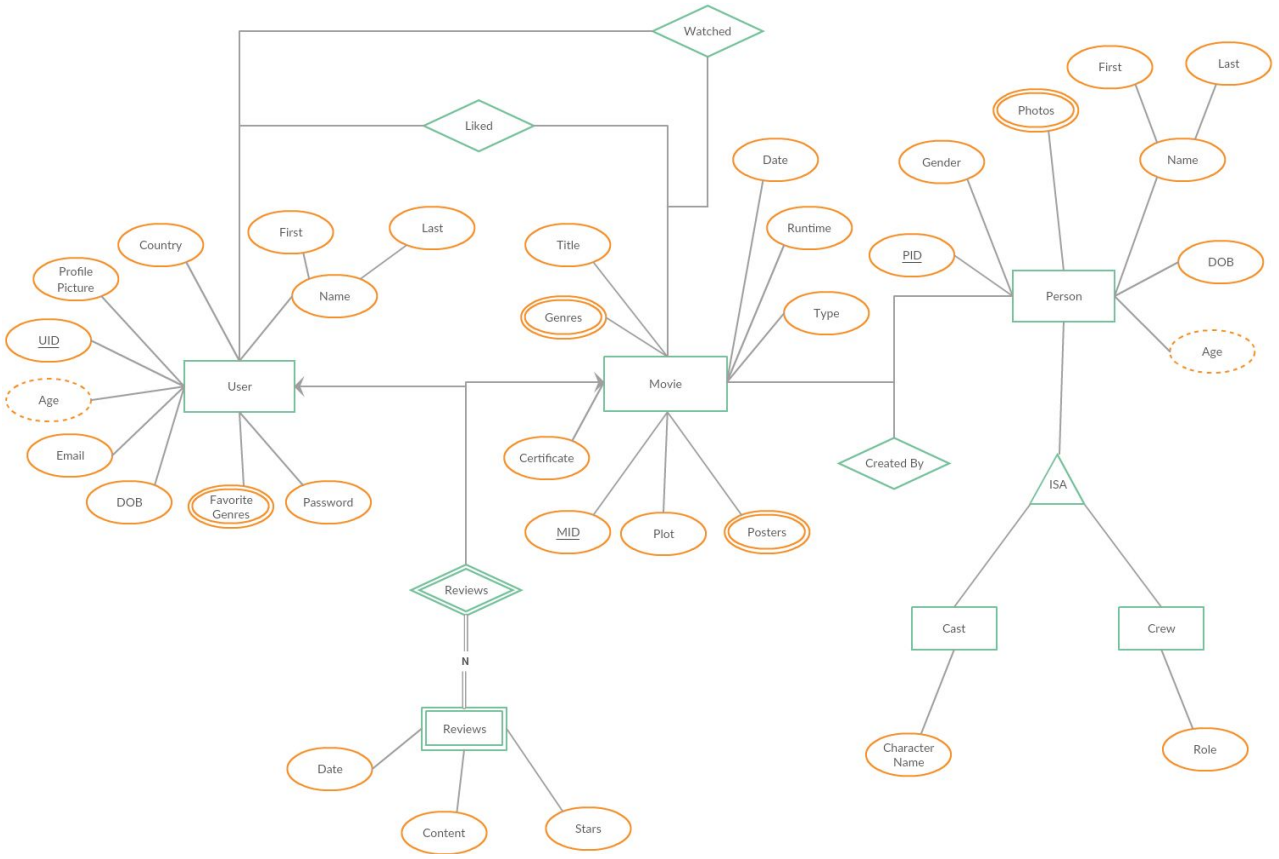
1.2 OBJECTIVES OF THE PROPOSED SYSTEM

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	

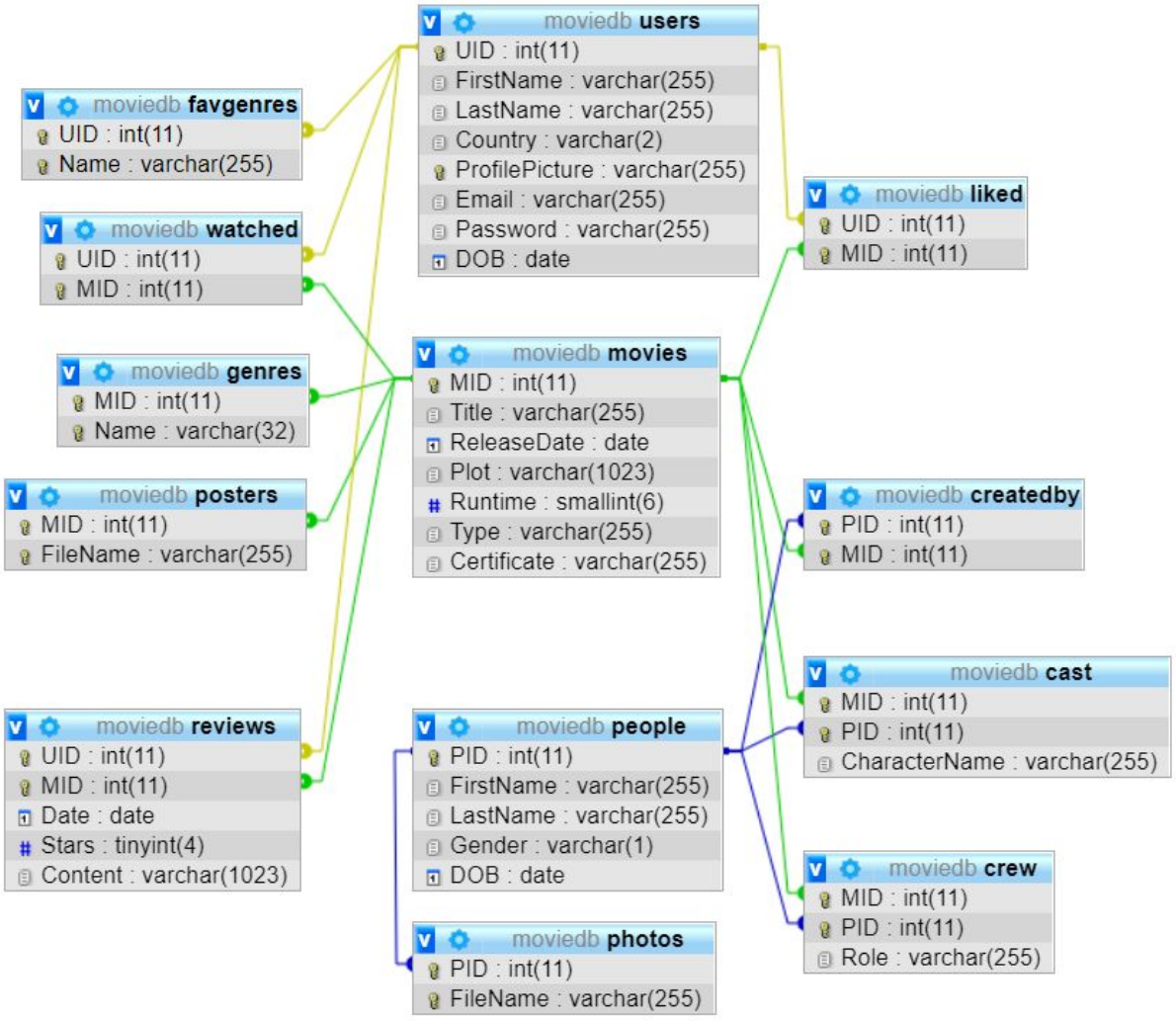
watched

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

1. USER TABLE

CREATE QUERY:
```
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)
);
```

2. MOVIES TABLE

CREATE QUERY:
```
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)
);
```

3. PEOPLE TABLE

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

4. GENRES TABLE

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

5. POSTERS TABLE

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

6. PHOTOS TABLE

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

7. FAVOURITE GENRES TABLE

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

8. WATCHED TABLE

CREATE QUERY:
```
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)
);
```

9. LIKED TABLE

CREATE QUERY:
```
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)
);
```

10. CREATED BY TABLE

CREATE QUERY:
```
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)
);
```

11. REVIEWS TABLE

CREATE QUERY:
```
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)
);
```

12. CAST TABLE

CREATE QUERY:
```
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)
```

```
);
```

13. CREW TABLE

CREATE QUERY:

```
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

QUERY:
```
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

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

3. ADD REVIEW FROM A USER

QUERY:
```
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

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

5. ADD WATCHED MOVIES

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

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

6. ADD PEOPLE

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

7. ADD PHOTOS OF PEOPLE

QUERY:
```
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

QUERY:
```
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

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

10. ADD POSTERS OF A MOVIE

QUERY:
```
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

QUERY:
```
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

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

13. DEFINE CREW MEMBERS OF A MOVIE

QUERY:
```
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

QUERY:
```
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'
```

OUTPUT:

Title	ReleaseDate	Plot	Runtime
The Avengers	2016-10-26	Duis aliquam convallis nunc. Proin at turpis a ped...	139
The Secret Life	2018-06-17	In quis justo. Maecenas rhoncus aliquam lacus. Mor...	120
Shepherd of the Hills, The	2017-10-25	Aliquam quis turpis eget elit sodales scelerisque....	103
GoldenEye	2018-08-24	Maecenas leo odio, condimentum id, luctus nec, mol...	97

2. GET ALL REVIEWS OF A PARTICULAR MOVIE

QUERY:
```
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
```

OUTPUT:

User	Rating	Date ▼ 1	Review
Guido Schrir	1	2018-08-04	Donec diam neque, vestibulum eget, vulputate ut, u...
Desdemona Delahunty	5	2017-09-17	Proin eu mi. Nulla ac enim. In tempor, turpis nec ...

3. LIST ALL CAST MEMBERS OF A MOVIE

QUERY:
```
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'
```

OUTPUT:

Actor Name	Character Name
Brocky Kee	Joshua
Winni Warin	Brooke

4. LIST ALL CREW MEMBERS OF A MOVIE

QUERY:
```
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'
```

OUTPUT:

Crew Member Name	Character Name
Adina Murrill	Editor
Toiboid Lorrimer	Executive Producer
Ange Cruz	Producer
Selinda Leavey	Cinematographer
Ford Franz	Director
Oliver Pedrol	Costume Designer

5. LIST ALL MOVIES OF FAVORITE GENRES

QUERY:
```
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
```

OUTPUT:

Crew Member Name	Character Name
Adina Murrill	Editor
Toiboid Lorrimer	Executive Producer
Ange Cruz	Producer
Selinda Leavey	Cinematographer
Ford Franz	Director
Oliver Pedrol	Costume Designer

6. GET ALL INFORMATION OF A MOVIE

QUERY:
```
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
```

OUTPUT:

Title	Release Date	Plot	runtime	Genres
The Avengers	2016-10-26	Duis aliquam convallis nunc. Proin at turpis a ped...	139	Drama,Horror

7. GET MOVIES RELEASED IN A PARTICULAR DATE RANGE

QUERY:

```
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
```

OUTPUT:

Title	Release Date
Weekender	2017-02-21
Boy Who Could Fly	2017-04-26
May in the Summer	2017-07-02
Trances	2017-07-22
Night of the Demons 2	2017-09-11
Mr. Popper's Penguins	2017-10-06
Jason X	2017-10-08
London Conspiracy	2017-10-16
Shepherd of the Hills, The	2017-10-25
American Pimp	2017-11-27

8. FIND ALL MOVIES A PERSON HAS STARRED IN

QUERY:

```
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
```

OUTPUT:

Movie	Character Name
Trances	Sarah
ZMD: Zombies of Mass Destruction	Kimberly
The Godfather	Savannah
Jason X	Samantha

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

QUERY:
```
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
```

OUTPUT:

Movie	Role
The Secret Life	Editor
London Conspiracy	Director

10. ALL LIKED MOVIES OF A USER

QUERY:
```
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
```

OUTPUT:

Title
Trances
ZMD: Zombies of Mass Destruction

11. ALL WATCHED MOVIES OF A USER

QUERY:
```
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
```

OUTPUT:

Title
The Avengers
Trances
ZMD: Zombies of Mass Destruction
Desperate Journey

12. LIST ALL REVIEWS BY A USER

QUERY:
```
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
```

OUTPUT:

Movie	Date ▼ 1	Rating	Review
Mr. Popper's Penguins	2018-08-06	4	Duis aliquam convallis nunc. Proin at turpis a ped...
May in the Summer	2017-11-08	3	Aliquam quis turpis eget elit sodales scelerisque....
Jason X	2017-09-13	4	Nulla ut erat id mauris vulputate elementum. Nulla...

13. ALL CAST MEMBERS OF USER'S LIKED MOVIES

QUERY:
```
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
```

OUTPUT:

Actor	Movie	Character Name
Brocky Kee	GoldenEye	Isaiah
Lucie Sigmund	GoldenEye	Chloe
Winni Warin	GoldenEye	Jasmine

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

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

OUTPUT:

Title	Stars
The Secret Life	5
ZMD: Zombies of Mass Destruction	5
The Godfather	5
American Pimp	5
GoldenEye	5
Night Flight	5
London Conspiracy	5
Jason X	5

15. LIST ALL PHOTOS OF A PERSON

QUERY:
```
SELECT p.FileName
FROM photos p
WHERE p.PID=1
```
OUTPUT:

FileName
dVvC6FYupGDt69iV.jpg
JRl5Ash7ioWrMrK5.jpg
Pcg1CLV9aakEHjvs.jpg
rtfVnMVse4Tj23lO.jpg
tXH7mMbCVSJLqPd9.jpg
wzUh2EDruHhHplDm.jpg
YTGCH7eRMSI3gnHj.jpg

16. LIST ALL FAVORITE GENRE

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

Genres
Horror
Mystery

17. LIST ALL GENRES OF A MOVIE

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

OUTPUT:

Genres
Drama
Horror

18. LIST ALL POSTER OF A MOVIE

QUERY:
```
SELECT p.FileName
FROM posters p
WHERE MID=1
```
OUTPUT:

FileName
dVvC6FYupGDt69iV.jpg
JRI5Ash7ioWrMrK5.jpg
Pcg1CLV9aakEHjvs.jpg
rtfVnMVse4Tj23IO.jpg
tXH7mMbCVSJLqPd9.jpg
wzUh2EDruHhHplDm.jpg
YTGCH7eRMSI3gnHj.jpg

19. LIST OF NUMBER OF MOVIES FOR EACH CERTIFICATE

QUERY:
```
SELECT COUNT(MID) AS 'Number of Movies', Certificate
FROM movies
GROUP BY Certificate
```
OUTPUT:

Number of Movies	Certificate
1	G
2	NC-17
2	PG
10	PG-13
5	R

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

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

Number of Movies	Certificate
10	PG-13

21. AVERAGE RUNTIME OF MOVIES

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

OUTPUT:

Average Runtime of Movies
103.3000

22. SEARCH MOVIE BY SUBSTRING OF TITLE

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

OUTPUT:

MID	Title	ReleaseDate	Plot	Runtime	Type	Certificate
1	The Avengers	2016-10-26	Duis aliquam convallis nunc. Proin at turpis a ped...	139	Feature	PG-13

23. LIST MOVIES OF PARTICULAR CERTIFICATES

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

OUTPUT:

MID	Title	ReleaseDate	Plot	Runtime	Type	Certificate
1	The Avengers	2016-10-26	Duis aliquam convallis nunc. Proin at turpis a ped...	139	Feature	PG-13
3	Trances	2017-07-22	Nullam porttitor lacus at turpis. Donec posuere me...	91	Feature	PG-13
4	Mr. Popper's Penguins	2017-10-06	Aliquam quis turpis eget elit sodales scelerisque....	30	Short	PG-13
7	May in the Summer	2017-07-02	In hac habitasse platea dictumst. Morbi vestibulum...	80	Feature	PG-13
8	Night of the Demons 2	2017-09-11	Sed sagittis. Nam congue, risus semper porta volut...	142	Feature	PG-13
9	ZMD: Zombies of Mass Destruction	2016-08-20	Lorem ipsum dolor sit amet, consectetuer adipiscin...	20	Short	PG-13
11	Modulations	2018-09-28	In hac habitasse platea dictumst. Etiam faucibus c...	148	Feature	PG-13
12	The Godfather	2018-03-21	Nulla ut erat id mauris vulputate elementum. Nulla...	145	Feature	PG-13
14	GoldenEye	2018-08-24	Maecenas leo odio, condimentum id, luctus nec, mol...	97	Feature	PG
15	Night Flight	2016-03-02	Phasellus in felis. Donec semper sapien a libero. ...	28	Short	PG
16	New York, I Love You	2018-01-27	In hac habitasse platea dictumst. Etiam faucibus c...	131	Feature	PG-13
19	Bewitched	2016-03-05	Quisque porta volutpat erat. Quisque erat eros, vi...	116	Feature	PG-13

24. LIST ALL NON-ADULT MOVIES

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

OUTPUT:

MID	Title	ReleaseDate	Plot	Runtime	Type	Certificate
1	The Avengers	2016-10-26	Duis aliquam convallis nunc. Proin at turpis a ped...	139	Feature	PG-13
3	Trances	2017-07-22	Nullam porttitor lacus at turpis. Donec posuere me...	91	Feature	PG-13
4	Mr. Popper's Penguins	2017-10-06	Aliquam quis turpis eget elit sodales scelerisque....	30	Short	PG-13
5	Boy Who Could Fly	2017-04-26	Suspendisse potenti. In eleifend quam a odio. In h...	126	Feature	NC-17
7	May in the Summer	2017-07-02	In hac habitasse platea dictumst. Morbi vestibulum...	80	Feature	PG-13
8	Night of the Demons 2	2017-09-11	Sed sagittis. Nam congue, risus semper porta volut...	142	Feature	PG-13
9	ZMD: Zombies of Mass Destruction	2016-08-20	Lorem ipsum dolor sit amet, consectetuer adipiscin...	20	Short	PG-13
10	Shepherd of the Hills, The	2017-10-25	Aliquam quis turpis eget elit sodales scelerisque....	103	Feature	NC-17
11	Modulations	2018-09-28	In hac habitasse platea dictumst. Etiam faucibus c...	148	Feature	PG-13
12	The Godfather	2018-03-21	Nulla ut erat id mauris vulputate elementum. Nulla...	145	Feature	PG-13
14	GoldenEye	2018-08-24	Maecenas leo odio, condimentum id, luctus nec, mol...	97	Feature	PG
15	Night Flight	2016-03-02	Phasellus in felis. Donec semper sapien a libero. ...	28	Short	PG
16	New York, I Love You	2018-01-27	In hac habitasse platea dictumst. Etiam faucibus c...	131	Feature	PG-13
18	London Conspiracy	2017-10-16	Aenean lectus. Pellentesque eget nunc. Donec quis ...	35	Short	G
19	Bewitched	2016-03-05	Quisque porta volutpat erat. Quisque erat eros, vi	116	Feature	PG-13

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

QUERY:
```
SELECT Name AS Genre
FROM genres
WHERE Name
NOT IN (
    SELECT Name
    FROM favgenres
)
```

OUTPUT:

Genre
War
Children
Crime

26. LIST OF ALL CAST AND CREW MEMBERS

QUERY:
```
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'
```

OUTPUT:

Person Name
Brocky Kee
Winni Warin
Adina Murrill
Toiboid Lorrimer
Ange Cruz
Selinda Leavey
Ford Franz
Oliver Pedrol

27. RUNTIME GREATER THAN A PARTICULAR VALUE

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

OUTPUT:

Title
The Avengers
The Secret Life
Boy Who Could Fly
Weekender
Night of the Demons 2
Shepherd of the Hills, The
Modulations
The Godfather
American Pimp
New York, I Love You
Desperate Journey
Bewitched
Jason X

28. AVERAGE RATING OF A MOVIE

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

OUTPUT:

Rating
3.0000

5.4 PL / SQL (PROCEDURES and FUNCTIONS)

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

PL/SQL Program:

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

OUTPUT:

Title
The Avengers
The Secret Life
Trances
Boy Who Could Fly
Weekender
May in the Summer
Night of the Demons 2
Shepherd of the Hills, The
Modulations
The Godfather
American Pimp
GoldenEye
New York, I Love You
Desperate Journey
Bewitched
Jason X

2. COUNT NUMBER OF MALE AND FEMALE FROM PEOPLE

PL/SQL Program:

```
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 ;
```

OUTPUT:

Male	Female
10	10

3. INSERT INTO CAST TABLE, EXCEPTION IF DUPLICATE KEY

PL/SQL Program:

```
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 ;
```

OUTPUT:

Error
Duplicate Keys (1,2) Found

4. INSERT INTO REVIEWS WITH VALID RATING VALUE

PL/SQL Program:

```
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

PL/SQL Program:

```
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 ;
```

OUTPUT:

AdultMovieCheck
1

6. RATES THE MOVIE BASED ON THE STARS IT RECIEVED

PL/SQL Program:

```
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 ;
```

OUTPUT:

AverageRating
Poor

5.5 Views

1. LIST ALL MOVIE DETAILS IN ONE TUPLE

CREATE QUERY:
```
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
```

VIEW OUTPUT:

Title	Release Date	Plot	Runtime	Type	Certificate	Genres
The Avengers	2016-10-26	Duis aliquam convallis nunc. Proin at turpis a ped...	139	Feature	PG-13	Drama,Horror
The Secret Life	2018-06-17	In quis justo. Maecenas rhoncus aliquam lacus. Mor...	120	Feature	R	Comedy,Drama,Horror
Trances	2017-07-22	Nullam porttitor lacus at turpis. Donec posuere me...	91	Feature	PG-13	Drama,Mystery
Mr. Popper's Penguins	2017-10-06	Aliquam quis turpis eget elit sodales scelerisque....	30	Short	PG-13	Documentary,Musical
Boy Who Could Fly	2017-04-26	Suspendisse potenti. In eleifend quam a odio. In h...	126	Feature	NC-17	Action,Documentary
Weekender	2017-02-21	Duis consequat dui nec nisi volutpat eleifend. Don...	146	Feature	R	Drama,Thriller
May in the Summer	2017-07-02	In hac habitasse platea dictumst. Morbi vestibulum...	80	Feature	PG-13	Drama,War
Night of the Demons 2	2017-09-11	Sed sagittis. Nam congue, risus semper porta volut...	142	Feature	PG-13	Comedy,Romance
ZMD: Zombies of Mass Destruction	2016-08-20	Lorem ipsum dolor sit amet, consectetuer adipiscin...	20	Short	PG-13	Drama
Shepherd of the Hills, The	2017-10-25	Aliquam quis turpis eget elit sodales scelerisque....	103	Feature	NC-17	Action,Horror,Thriller
Modulations	2018-09-28	In hac habitasse platea dictumst. Etiam faucibus c...	148	Feature	PG-13	Comedy
The Godfather	2018-03-21	Nulla ut erat id mauris vulputate elementum. Nulla...	145	Feature	PG-13	Adventure,Documentary
American Pimp	2017-11-27	Proin interdum mauris non ligula pellentesque ultr...	125	Feature	R	Children,Comedy
GoldenEye	2018-08-24	Maecenas leo odio, condimentum id, luctus nec, mol...	97	Feature	PG	Action,Horror,Thriller
Night Flight	2016-03-02	Phasellus in felis. Donec semper sapien a libero. ...	28	Short	PG	Comedy,Fantasy
New York, I Love You	2018-01-27	In hac habitasse platea dictumst. Etiam faucibus c...	131	Feature	PG-13	Action,Mystery,Sci-Fi
Desperate Journey	2018-10-23	Phasellus sit amet erat. Nulla tempus. Vivamus in ...	143	Feature	R	Crime,Drama
London Conspiracy	2017-10-16	Aenean lectus. Pellentesque eget nunc. Donec quis ...	35	Short	G	Drama,Sci-Fi
Bewitched	2016-03-05	Quisque porta volutpat erat. Quisque erat eros, vi...	116	Feature	PG-13	Comedy,Fantasy
Jason X	2017-10-08	Integer ac leo. Pellentesque ultrices mattis odio....	101	Feature	R	Romance,Sci-Fi

2. LIST RATINGS OF MOVIES

CREATE QUERY:
```
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
```

VIEW OUTPUT:

Title	Rating
The Secret Life	3.0000
Trances	2.6667
Mr. Popper's Penguins	3.5000
Boy Who Could Fly	2.0000
Weekender	2.0000
May in the Summer	3.0000
Night of the Demons 2	2.0000
ZMD: Zombies of Mass Destruction	3.0000
Shepherd of the Hills, The	2.7500
Modulations	3.0000
The Godfather	3.5000
American Pimp	3.6000
GoldenEye	5.0000
Night Flight	3.6667
Desperate Journey	4.0000
London Conspiracy	5.0000
Bewitched	4.0000
Jason X	4.6667

5.6 Triggers

1. CHECK MOVIE TYPE WITH RUNTIME

PL/SQL Program:
```
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 ;
```

2. DELETE ALL FOREIGN KEY INSTANCES OF A MOVIE

PL/SQL Program:
```
DELIMITER //
CREATE TRIGGER DeleteMovie AFTER DELETE ON movies
    FOR EACH ROW

    BEGIN

        DELETE FROM cast WHERE (MID = OLD.MID);
        DELETE FROM createdby WHERE (MID = OLD.MID);
        DELETE FROM crew WHERE (MID = OLD.MID);
        DELETE FROM genres WHERE (MID = OLD.MID);
        DELETE FROM liked WHERE (MID = OLD.MID);
        DELETE FROM posters WHERE (MID = OLD.MID);
        DELETE FROM reviews WHERE (MID = OLD.MID);
        DELETE FROM watched WHERE (MID = OLD.MID);

    END //
DELIMITER ;
```

5.7 Cursors

1. COUNT THE NUMBER OF MOVIES OF A PARTICULAR GENRE

PL/SQL Program:

```
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 ;
```

OUTPUT:

GenreCount
4

2. PREPARE MAILING LIST

PL/SQL Program:

```
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;
```

OUTPUT:

@email_list
wfearej@123-reg.co.uk;gcapleni@sphinn.com;abatthewh@last.fm;wayarsg@webnode.com;bcrombf@mit.edu;gassantee@businessinsider.com;hstringfellowd@bar

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