* Check if account exist using email. (Query)
* SELECT COUNT(\*) FROM Account WHERE Email = ‘MAIL’;
* Signup and create new account. (Query)
* INSERT INTO Account VALUES(, Name, Email, Username, Password, Birthday, Acc\_Status, Payment\_Method);
* Access available profiles options. (Query)
* SELECT Name, Number FROM Profile WHERE Account\_Id = `ID`;
* Fetch Movie contents for Home Page. (Procedure)
* DELIMITER $$
* CREATE PROCEDURE fetchTop50Movies()
* BEGIN
* SELECT Title, Genre, Description, Director, Actors, Year, Runtime, Rating, isAgeRestricted FROM Movies LIMIT 50;
* END $$
* call fetchTop50Movies()
* Search movies content via options. (Procedure)

- DELIMITER $$

CREATE PROCEDURE findMovie(Search VARCHAR(50))

BEGIN

SELECT Movie\_ID, Title FROM Movies WHERE Title Like CONCAT("%",Search,"%");

END $$

call findMovie("Land")

* Filter Movies based on ratings and genre. (Procedure)
* DELIMITER $$
* CREATE PROCEDURE filterMovie(SGenre VARCHAR(50), SRating INT)
* BEGIN
* SELECT Movie\_ID, Title FROM Movies WHERE Rating >= SRating AND Genre Like CONCAT("%",SGenre,"%");
* END $$
* call filterMovie("Action", 1.0)
* Open Movie Page for Full information of the Movie. (Query)
* SELECT Title, Genre, Description, Director, Actors, Year, Runtime, Rating, isAgeRestricted FROM Movies WHERE Movie\_ID = `MovieID`;
* Movie Playing using given content URL. (Query)
* SELECT Content\_URL FROM Movies WHERE Movie\_ID = `MovieID`;
* Account info showing user’s data including all profiles. (Query)
* SELECT Name, Email, UserName, Birthday, Account\_Status FROM Account WHERE Account\_ID = `AccountID`;
* SELECT Name, Number, isAgeRestricted FROM Profiles WHERE Account\_ID = `AccountID`;
* Subscription page showing Plan information. (Query)
* SELECT P.Name, P.Price, P.Features, P.Profile\_Limit FROM Plan P JOIN Subscription S on S.Plan\_ID = P.Plan\_ID JOIN Account A ON A.Account\_ID = S.Account\_ID WHERE A.Account\_ID = `Account\_ID` ORDER BY S.Subscription\_ID LIMIT 1;
* Billing page showing payment information. (Query)
* SELECT B.Bill\_id, B.Bill\_Amount, A.Payment\_Method FROM Account A JOIN Subscription S ON A.Account\_ID = S.Account\_ID JOIN Billing B ON S.Bill\_ID = B.Bill\_ID WHERE A.Account\_ID = `AccountID` ORDER BY B.Statement\_Date LIMIT 1;
* Statement page showing current month’s Statement. (Query)
* SELECT B.Bill\_id, B.Bill\_Amount, A.Payment\_Method, B.Payment\_Status, B.Statement\_Date FROM Account A JOIN Subscription S ON A.Account\_ID = S.Account\_ID JOIN Billing B ON S.Bill\_ID = B.Bill\_ID WHERE S.isActive = 1 AND A.Account\_ID = `AccountID`;
* Like and Rating for particular watched movie. (Query)
* SELECT M.Title, M.Genre, W.Likes, W.Rating FROM Account A JOIN Profiles P ON P.Account\_ID = A.Account\_ID JOIN Watch\_History W ON W.Profile\_ID = P.Profile\_ID JOIN Movies M ON M.Movie\_ID = W.Movie\_Id WHERE A.Account\_ID = `AccountID` AND M.Title Like CONCAT(“%”,`MovieName`,”%”);
* Showing total movies count and their average time by genre. (Procedure)
* DELIMITER $$
* CREATE PROCEDURE genreStats(SGenre VARCHAR(50))
* BEGIN
* SELECT COUNT(M.Movie\_ID) AS "Movie Count", AVG(M.Runtime) AS "Average Runtime" FROM Movies M WHERE M.Genre Like CONCAT("%",SGenre,"%");
* END $$
* call genreStats (`GENRE`);
* Showing Average rating based on viewer’s rating and IMDB rating. (Procedure)
* DELIMITER $$
* CREATE PROCEDURE movieRatings(SMovie VARCHAR(50))
* BEGIN
* SELECT M.Rating AS "IMDB Rating", W.Rating AS "Viewer's Rating", ROUND((M.Rating + W.Rating) / 2, 1) AS "Average Rating" FROM Movies M JOIN Watch\_History W ON W.Movie\_Id = M.Movie\_ID WHERE M.Title Like CONCAT("%",SMovie,"%");
* END $$
* call movieRatings(`MOVIE`);
* Watch history page showing which movie users have seen till now. (Procedure)
* DELIMITER $$
* CREATE PROCEDURE watchHistory(SProfile INT)
* BEGIN
* SELECT M.Title, M.Genre, W.Likes, W.Rating, W.Comment, W.Timeline, W.Last\_Watched FROM Profiles P JOIN Watch\_History W ON W.Profile\_ID = P.Profile\_ID JOIN Movies M ON M.Movie\_ID = W.Movie\_Id WHERE P.Profile\_ID = SProfile;
* END $$
* call watchHistory(`Profile`);
* Customer Service Request page to send issue about account or movie.(Procedure)
* DELIMITER $$
* CREATE PROCEDURE sendComplain(SNumber INT, SAccount INT, SComplain VARCHAR(200))
* BEGIN
* INSERT INTO Service\_Request (`Service\_Num`, `Account\_ID`, `Complains`, `isResolved`) VALUES (SNumber, SAccount, SComplain, 0);
* END $$
* call sendComplain(`ServiceNum`, `Account`, `Complain`);
* Customer Service Request page to send and track issues about account and movies. (Query)
* SELECT Service\_Num, Complains, isResolved FROM Service\_Request WHERE Service\_Num = `ServiceNum`;