Database Schema

Team members	Matriculation number
Meshu Deb Nath	618652

Schemas for creating database tables

Film related person

```
CREATE TABLE film_persons(
      person_id SERIAL PRIMARY KEY,
      person_name varchar(100) NOT NULL UNIQUE,
      sex varchar(100) NOT NULL,
      dob varchar(100) NOT NULL
);
```

Film

```
CREATE TABLE films(
      film_id SERIAL PRIMARY KEY,
      film_title varchar(100) NOT NULL,
      release_year varchar(100) NOT NULL,
      production_country varchar(100) NOT NULL,
      age_limit integer
);
```

Genres

```
CREATE TABLE genres(
      genre_id SERIAL PRIMARY KEY,
      genre_title varchar(100) NOT NULL,
      film_title varchar(100) NOT NULL,
      release_year varchar(100) NOT NULL
);
```

Roles

```
CREATE TABLE roles(
      role_id SERIAL PRIMARY KEY,
      role_title varchar(100) NOT NULL,
      film_person_name varchar(100) NOT NULL,
      film_title varchar(100) NOT NULL,
      release_year varchar(100) NOT NULL
);
```

subordinate films

```
CREATE TABLE subordinate_films(
      sub_film_id SERIAL PRIMARY KEY,
      sub_film_title varchar(100) NOT NULL,
      sub_film_release_year varchar(100) NOT NULL,
      film_title varchar(100) NOT NULL,
      release_year varchar(100) NOT NULL
);
```

Users

```
CREATE TABLE users(
      user_id SERIAL PRIMARY KEY,
      user_name varchar(100) NOT NULL UNIQUE
);
```

User Rating

```
CREATE TABLE user_rating(
      rating_id SERIAL PRIMARY KEY,
      rating integer,
      user_name varchar(100) NOT NULL,
      film_title varchar(100) NOT NULL,
      release_year varchar(100) NOT NULL
);
```

PL/pgSQL for Inserting instances into Tables

Add a film related person

```
DROP FUNCTION IF EXISTS add_film_person(text,text,text,text);
CREATE OR REPLACE FUNCTION add_film_person(text, text, text, text) RETURNS
text AS $$
  DECLARE
     person_id ALIAS FOR $1;
     person_name ALIAS FOR $2;
     sex ALIAS FOR $3;
     dob ALIAS FOR $4;
     confirm_insert INTEGER;
  BEGIN
     INSERT INTO film_persons (person_id, person_name, sex, dob)
     VALUES
          (CAST (person_id AS INTEGER),
          CAST (person_name AS VARCHAR),
          CAST (sex AS VARCHAR),
          CAST (dob AS VARCHAR));
     GET DIAGNOSTICS confirm_insert = ROW_COUNT;
       IF confirm_insert > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';
  END;
$$ LANGUAGE plpgsql;
```

Add a Film

```
DROP FUNCTION IF EXISTS add_film(text,text,text,text, text);
CREATE OR REPLACE FUNCTION add_film(text, text, text, text, text) RETURNS
text AS $$
  DECLARE
     film_id ALIAS FOR $1;
     film_title ALIAS FOR $2;
     release_year ALIAS FOR $3;
     production_country ALIAS FOR $4;
     age_limit ALIAS FOR $5;
     confirm_insert INTEGER;
  BEGIN
     INSERT INTO films(film_id, film_title, release_year, production_country,
age_limit)
     VALUES
          (CAST (film_id AS INTEGER),
          CAST (film_title AS VARCHAR),
          CAST (release_year AS VARCHAR),
          CAST (production_country AS VARCHAR),
          CAST (age_limit AS INTEGER));

       GET DIAGNOSTICS confirm_insert = ROW_COUNT;
       IF confirm_insert > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';
  END;
$$ LANGUAGE plpgsql;

SELECT add_film ('1', 'beder meye josna', '2019','USA', '18');
```

Add a Genre

```
DROP FUNCTION IF EXISTS add_genre(text,text,text,text);
CREATE OR REPLACE FUNCTION add_genre(text, text, text, text) RETURNS text AS
$$
  DECLARE
     genre_id ALIAS FOR $1;
     genre_title ALIAS FOR $2;
     film_title ALIAS FOR $3;
     release_year ALIAS FOR $4;
     confirm_insert INTEGER;
  BEGIN
     INSERT INTO genres(genre_id, genre_title, film_title, release_year)
     VALUES
          (CAST (genre_id AS INTEGER),
          CAST (genre_title AS VARCHAR),
          CAST (film_title AS VARCHAR),
          CAST (release_year AS VARCHAR));

       GET DIAGNOSTICS confirm_insert = ROW_COUNT;
       IF confirm_insert > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT add_genre('1', 'horror', 'beder meye josna','2019');
```

Add a Role

```
DROP FUNCTION IF EXISTS add_role(text,text,text,text, text);
CREATE OR REPLACE FUNCTION add_role(text, text, text, text, text) RETURNS
text AS $$
  DECLARE
     role_id ALIAS FOR $1;
     role_title ALIAS FOR $2;
     film_person_name ALIAS FOR $3;
     film_title ALIAS FOR $4;
     release_year ALIAS FOR $5;
     confirm_insert INTEGER;
  BEGIN
     INSERT INTO roles(role_id, role_title, film_person_name, film_title,
release_year)
     VALUES
          (CAST (role_id AS INTEGER),
          CAST (role_title AS VARCHAR),
          CAST (film_person_name AS VARCHAR),
          CAST (film_title AS VARCHAR),
          CAST (release_year AS VARCHAR));

       GET DIAGNOSTICS confirm_insert = ROW_COUNT;
       IF confirm_insert > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT add_role('1', 'actor', 'Meshu','beder meye josna', '2019');
```

Add a Subordinate Film

```
DROP FUNCTION IF EXISTS add_subordinate_film(text,text,text,text, text);
CREATE OR REPLACE FUNCTION add_subordinate_film(text, text, text, text, text)
RETURNS text AS $$
  DECLARE
     sub_film_id ALIAS FOR $1;
     sub_film_title ALIAS FOR $2;
     sub_film_release_year ALIAS FOR $3;
     film_title ALIAS FOR $4;
     release_year ALIAS FOR $5;
     confirm_insert INTEGER;
  BEGIN
     INSERT INTO subordinate_films(sub_film_id , sub_film_title ,
sub_film_release_year , film_title, release_year)
     VALUES
          (CAST (sub_film_id AS INTEGER),
          CAST (sub_film_title AS VARCHAR),
          CAST (sub_film_release_year AS VARCHAR),
          CAST (film_title AS VARCHAR),
          CAST (release_year AS VARCHAR));

       GET DIAGNOSTICS confirm_insert = ROW_COUNT;
       IF confirm_insert > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT add_subordinate_film('1', 'Rose', '2019','Jesmin', '2019');
```

Add a User

```
DROP FUNCTION IF EXISTS add_user(text,text);
CREATE OR REPLACE FUNCTION add_user(text, text) RETURNS text AS $$
  DECLARE
     user_id ALIAS FOR $1;
     user_name ALIAS FOR $2;
     confirm_insert INTEGER;
  BEGIN
     INSERT INTO users(user_id, user_name)
     VALUES
          (CAST (user_id AS INTEGER),
          CAST (user_name AS VARCHAR));
       GET DIAGNOSTICS confirm_insert = ROW_COUNT;
       IF confirm_insert > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT add_user('3', 'Meshux');
```

Add a Rating

```
DROP FUNCTION IF EXISTS add_rating(text,text,text,text, text);
CREATE OR REPLACE FUNCTION add_rating(text, text, text, text, text) RETURNS
text AS $$
  DECLARE
     rating_id ALIAS FOR $1;
     rating ALIAS FOR $2;
     user_name ALIAS FOR $3;
     film_title ALIAS FOR $4;
     release_year ALIAS FOR $5;
     confirm_insert INTEGER;
  BEGIN
     INSERT INTO user_rating(rating_id, rating, user_name, film_title,
release_year)
     VALUES
          (CAST (rating_id AS INTEGER),
          CAST (rating AS INTEGER),
          CAST (user_name AS VARCHAR),
          CAST (film_title AS VARCHAR),
          CAST (release_year AS VARCHAR));

       GET DIAGNOSTICS confirm_insert = ROW_COUNT;
       IF confirm_insert > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT add_rating('1', '5', 'Meshu','beder meye josna', '2019');
```

PL/pgSQL for Editing Database Schemas

Edit a film related person table

```
DROP FUNCTION IF EXISTS edit_film_person(text,text,text,text);
CREATE OR REPLACE FUNCTION edit_film_person(text, text, text, text) RETURNS
text AS $$
  DECLARE
     person_id_1 ALIAS FOR $1;
     person_name_1 ALIAS FOR $2;
     sex_1 ALIAS FOR $3;
     dob_1 ALIAS FOR $4;
     confirm_update INTEGER;
  BEGIN
     UPDATE film_persons
         SET
            person_name = CAST (person_name_1 AS VARCHAR),
            sex = CAST (sex_1 AS VARCHAR),
            dob = CAST (dob_1 AS VARCHAR)
       WHERE film_persons.person_id = CAST (person_id_1 AS INTEGER);

      GET DIAGNOSTICS confirm_update = ROW_COUNT;
       IF confirm_update > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT edit_film_person('5', 'nafisa', 'female', '05-04-1994');
```

Edit a Film

```
DROP FUNCTION IF EXISTS edit_film(text,text,text,text, text);
CREATE OR REPLACE FUNCTION edit_film(text, text, text, text, text) RETURNS
text AS $$
  DECLARE
     film_id_1 ALIAS FOR $1;
     film_title_1 ALIAS FOR $2;
     release_year_1 ALIAS FOR $3;
     production_country_1 ALIAS FOR $4;
     age_limit_1 ALIAS FOR $5;
     confirm_update INTEGER;
  BEGIN
     UPDATE films
         SET
            film_title = CAST (film_title_1 AS VARCHAR),
            release_year = CAST (release_year_1 AS VARCHAR),
            production_country = CAST (production_country_1 AS VARCHAR),
            age_limit = CAST (age_limit_1 AS INTEGER)
       WHERE films.film_id = CAST (film_id_1 AS INTEGER);

      GET DIAGNOSTICS confirm_update = ROW_COUNT;
       IF confirm_update > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;



SELECT edit_film('1', 'nafisa', '2022', 'sa', '20');
```

Edit a Genre

```
DROP FUNCTION IF EXISTS edit_genre(text,text,text,text);
CREATE OR REPLACE FUNCTION edit_genre(text, text, text, text) RETURNS text AS
$$
  DECLARE
     genre_id_1 ALIAS FOR $1;
     genre_title_1 ALIAS FOR $2;
     film_title_1 ALIAS FOR $3;
     release_year_1 ALIAS FOR $4;
     confirm_update INTEGER;
  BEGIN
     UPDATE genres
         SET
            genre_title = CAST (genre_title_1 AS VARCHAR),
            film_title = CAST (film_title_1 AS VARCHAR),
            release_year = CAST (release_year_1 AS VARCHAR)
       WHERE genres.genre_id = CAST (genre_id_1 AS INTEGER);

      GET DIAGNOSTICS confirm_update = ROW_COUNT;
       IF confirm_update > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT edit_genre('1', 'romantic', 'Jesmin', '2012');
```

Edit a Role

```
DROP FUNCTION IF EXISTS edit_role(text,text,text,text, text);
CREATE OR REPLACE FUNCTION edit_role(text, text, text, text, text) RETURNS
text AS $$
  DECLARE
     role_id_1 ALIAS FOR $1;
     role_title_1 ALIAS FOR $2;
     film_person_name_1 ALIAS FOR $3;
     film_title_1 ALIAS FOR $4;
     release_year_1 ALIAS FOR $5;
     confirm_update INTEGER;
  BEGIN
     UPDATE roles
         SET
            role_title = CAST (role_title_1 AS VARCHAR),
            film_person_name = CAST (film_person_name_1 AS VARCHAR),
            film_title = CAST (film_title_1 AS VARCHAR),
            release_year = CAST (release_year_1 AS VARCHAR)
       WHERE roles.role_id = CAST (role_id_1 AS INTEGER);

     GET DIAGNOSTICS confirm_update = ROW_COUNT;
       IF confirm_update > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';
  END;
$$ LANGUAGE plpgsql;

SELECT edit_role('1', 'writer', 'Meshu', 'jesmin', '2015');
```

Edit a Subordinate Film

```
DROP FUNCTION IF EXISTS edit_subordinate_film(text,text,text,text, text);
```

```
CREATE OR REPLACE FUNCTION edit_subordinate_film(text, text, text, text,
text) RETURNS text AS $$
  DECLARE
     sub_film_id_1 ALIAS FOR $1;
     sub_film_title_1 ALIAS FOR $2;
     sub_film_release_year_1 ALIAS FOR $3;
     film_title_1 ALIAS FOR $4;
     release_year_1 ALIAS FOR $5;
     confirm_update INTEGER;
  BEGIN
     UPDATE subordinate_films
         SET
            sub_film_title = CAST (sub_film_title_1 AS VARCHAR),
            sub_film_release_year = CAST (sub_film_release_year_1 AS
VARCHAR),
            film_title = CAST (film_title_1 AS VARCHAR),
            release_year = CAST (release_year_1 AS VARCHAR)
       WHERE subordinate_films.sub_film_id = CAST (sub_film_id_1 AS INTEGER);

       GET DIAGNOSTICS confirm_update = ROW_COUNT;
       IF confirm_update > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT edit_subordinate_film('1', 'Rose', '2019', 'jesmin', '2015');
```

Edit a User

```
DROP FUNCTION IF EXISTS edit_user(text,text);
CREATE OR REPLACE FUNCTION edit_user(text, text) RETURNS text AS $$
  DECLARE
     user_id_1 ALIAS FOR $1;
     user_name_1 ALIAS FOR $2;
     confirm_update INTEGER;
  BEGIN
     UPDATE users
         SET
            user_name = CAST (user_name_1 AS VARCHAR)
       WHERE users.user_id = CAST (user_id_1 AS INTEGER);

     GET DIAGNOSTICS confirm_update = ROW_COUNT;
       IF confirm_update > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT edit_user('1', 'Mishty');
```

Edit a Rating

```
DROP FUNCTION IF EXISTS edit_rating (text,text,text,text, text);
CREATE OR REPLACE FUNCTION edit_rating (text, text, text, text, text) RETURNS
text AS $$
  DECLARE
     rating_id_1 ALIAS FOR $1;
     rating_1 ALIAS FOR $2;
     user_name_1 ALIAS FOR $3;
     film_title_1 ALIAS FOR $4;
     release_year_1 ALIAS FOR $5;
     confirm_update INTEGER;
  BEGIN
     UPDATE user_rating
         SET
            rating = CAST (rating_1 AS INTEGER),
            user_name = CAST (user_name_1 AS VARCHAR),
            film_title = CAST (film_title_1 AS VARCHAR),
            release_year = CAST (release_year_1 AS VARCHAR)
       WHERE user_rating.rating_id = CAST (rating_id_1 AS INTEGER);

     GET DIAGNOSTICS confirm_update = ROW_COUNT;
       IF confirm_update > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

       EXCEPTION
              WHEN OTHERS THEN
                   return '0';

  END;
$$ LANGUAGE plpgsql;

SELECT edit_rating ('1', '6', 'Meshu', 'jesmin', '2015');
```

PL/pgSQL for Deleting Database Schema

Delete a Film related Person

```
DROP FUNCTION IF EXISTS remove_film_person(text,text);
CREATE OR REPLACE FUNCTION remove_film_person(text, text) RETURNS text AS $$
  DECLARE
     person_id_1 ALIAS FOR $1;
     person_name_1 ALIAS FOR $2;
     _result INTEGER;
     _film_title VARCHAR[];
     _is_film_deleted INTEGER;
     array_len INTEGER;
  BEGIN
    _film_title := ARRAY(
       SELECT film_title FROM roles INNER JOIN film_persons ON
roles.film_person_name = film_persons.person_name
                and roles.film_person_name = CAST(person_name_1 AS VARCHAR)
     );

    array_len = array_length(_film_title, 1);

    IF array_len <= 0 THEN
      return '0';
    END IF;

    IF array_len IS NOT NULL THEN
       FOR index in 1..array_len LOOP
          DELETE FROM films WHERE films.film_title = _film_title[index];
          DELETE FROM genres WHERE genres.film_title = _film_title[index];
          DELETE FROM subordinate_films WHERE subordinate_films.film_title =
_film_title[index];
       END LOOP;

        DELETE FROM roles
       WHERE roles.film_person_name = CAST (person_name_1 AS VARCHAR);
    END IF;

    DELETE FROM film_persons
       WHERE film_persons.person_id = CAST (person_id_1 AS INTEGER)
              and film_persons.person_name = CAST (person_name_1 AS VARCHAR);
       GET DIAGNOSTICS _result = ROW_COUNT;
```

```
       IF _result > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

  END;
$$ LANGUAGE plpgsql;

SELECT remove_film_person('1644448256', 'jiki');
```

Delete a Film

```
DROP FUNCTION IF EXISTS remove_film(text,text);
CREATE OR REPLACE FUNCTION remove_film(text, text) RETURNS text AS $$
  DECLARE
     film_title_1 ALIAS FOR $1;
     release_year_1 ALIAS FOR $2;
     _is_film_deleted INTEGER;
     _is_subfilm INTEGER;
  BEGIN
    SELECT count(film_title) INTO _is_subfilm FROM subordinate_films
           WHERE  subordinate_films.film_title = CAST (film_title_1 AS
VARCHAR)
              and subordinate_films.release_year = CAST (release_year_1 AS
VARCHAR);

     IF _is_subfilm > 0 THEN

     DELETE FROM subordinate_films
       WHERE subordinate_films.film_title = CAST (film_title_1 AS VARCHAR)
              and subordinate_films.release_year = CAST (release_year_1 AS
VARCHAR);
     END IF;

     SELECT count(film_title) INTO _is_subfilm FROM user_rating
           WHERE  user_rating.film_title = CAST (film_title_1 AS VARCHAR)
              and user_rating.release_year = CAST (release_year_1 AS
VARCHAR);

     IF _is_subfilm > 0 THEN

     DELETE FROM user_rating
       WHERE user_rating.film_title = CAST (film_title_1 AS VARCHAR)
                  and user_rating.release_year = CAST (release_year_1 AS
VARCHAR);
     END IF;

    --Checking for films in Roles table and then deleted it.

     SELECT count(film_title) INTO _is_subfilm FROM roles
           WHERE  roles.film_title = CAST (film_title_1 AS VARCHAR)
              and roles.release_year = CAST (release_year_1 AS VARCHAR);
```

```
     IF _is_subfilm > 0 THEN

     DELETE FROM roles
       WHERE roles.film_title = CAST (film_title_1 AS VARCHAR)
                  and roles.release_year = CAST (release_year_1 AS VARCHAR);
     END IF;

    --Checking for films in Genres table and then deleted it.

     SELECT count(film_title) INTO _is_subfilm FROM genres
           WHERE  genres.film_title = CAST (film_title_1 AS VARCHAR)
              and genres.release_year = CAST (release_year_1 AS VARCHAR);

     IF _is_subfilm > 0 THEN

     DELETE FROM genres
       WHERE genres.film_title = CAST (film_title_1 AS VARCHAR)
                  and genres.release_year = CAST (release_year_1 AS VARCHAR);
     END IF;

    DELETE FROM films
       WHERE films.film_title = CAST (film_title_1 AS VARCHAR)
                and films.release_year = CAST (release_year_1 AS VARCHAR);
       GET DIAGNOSTICS _is_film_deleted= ROW_COUNT;
       IF _is_film_deleted > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

  END;
$$ LANGUAGE plpgsql;

SELECT remove_film('testing4','2022');
```

Delete a Rating

```
DROP FUNCTION IF EXISTS remove_rating(text);
CREATE OR REPLACE FUNCTION remove_rating(text) RETURNS text AS $$
  DECLARE
     rating_id_1 ALIAS FOR $1;
     _is_rating_deleted INTEGER;
  BEGIN
    DELETE FROM user_rating
       WHERE user_rating.rating_id = CAST (rating_id_1 AS INTEGER);
       GET DIAGNOSTICS _is_rating_deleted = ROW_COUNT;
       IF _is_rating_deleted > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

  END;
$$ LANGUAGE plpgsql;

SELECT remove_rating('1');
```

Delete a Genre

```
DROP FUNCTION IF EXISTS remove_genre(text);
CREATE OR REPLACE FUNCTION remove_genre(text) RETURNS text AS $$
  DECLARE
     genre_id_1 ALIAS FOR $1;
     _is_genre_deleted INTEGER;
  BEGIN
    DELETE FROM genres
       WHERE genres.genre_id = CAST (genre_id_1 AS INTEGER);
       GET DIAGNOSTICS _is_genre_deleted = ROW_COUNT;
       IF _is_genre_deleted > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

  END;
$$ LANGUAGE plpgsql;

SELECT remove_genre('1');
```

Delete a Role

```
DROP FUNCTION IF EXISTS remove_role(text);
CREATE OR REPLACE FUNCTION remove_role(text) RETURNS text AS $$
  DECLARE
     role_id_1 ALIAS FOR $1;
     _is_role_deleted INTEGER;
  BEGIN
    DELETE FROM roles
       WHERE roles.role_id = CAST (role_id_1 AS INTEGER);
       GET DIAGNOSTICS _is_role_deleted = ROW_COUNT;
       IF _is_role_deleted > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

  END;
$$ LANGUAGE plpgsql;

SELECT remove_role('1');
```

Delete a Sub Film

```
DROP FUNCTION IF EXISTS remove_subordinate_film(text);
CREATE OR REPLACE FUNCTION remove_subordinate_film(text) RETURNS text AS $$
  DECLARE
     sub_film_id_1 ALIAS FOR $1;
     _is_data_deleted INTEGER;
  BEGIN
    DELETE FROM subordinate_films
       WHERE subordinate_films.sub_film_id = CAST (sub_film_id_1 AS INTEGER);
       GET DIAGNOSTICS _is_data_deleted = ROW_COUNT;
       IF _is_data_deleted > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

  END;
$$ LANGUAGE plpgsql;

SELECT remove_subordinate_film('1');
```

Delete a User

```
DROP FUNCTION IF EXISTS remove_user(text);
CREATE OR REPLACE FUNCTION remove_user(text) RETURNS text AS $$
  DECLARE
     user_id_1 ALIAS FOR $1;
     _is_user_deleted INTEGER;
  BEGIN
    DELETE FROM users
       WHERE users.user_id = CAST (user_id_1 AS INTEGER);
       GET DIAGNOSTICS _is_user_deleted = ROW_COUNT;
       IF _is_user_deleted > 0 THEN
          return '1';
       ELSE
          return '0';
       END IF;

  END;
$$ LANGUAGE plpgsql;

SELECT remove_user('1');
```

Film Suggetion

```
DROP FUNCTION IF EXISTS film_suggestion(text);
CREATE OR REPLACE FUNCTION film_suggestion(text) RETURNS text[] AS $$
  DECLARE
     user_name_1 ALIAS FOR $1;
     _film_title text[];
     _suggest_films text[];
     _suggest_genres text[];
     _row_count INTEGER;
     array_len INTEGER;
     _empty text[];
  BEGIN
       _film_title := ARRAY(
       SELECT user_rating.film_title FROM user_rating INNER JOIN films ON
user_rating.film_title = films.film_title
                and user_rating.user_name = CAST(user_name_1 AS VARCHAR)
       );

       array_len = array_length(_film_title, 1);

       IF array_len > 0 THEN
            _suggest_genres := ARRAY(
                 SELECT Distinct genres.genre_title FROM unnest(_film_title)
temp_films INNER JOIN genres ON genres.film_title = temp_films
                );

            --after getting genres from the joined result, I'm going to
search film related to this;
            _suggest_films := ARRAY(
                      SELECT films.film_title FROM unnest(_suggest_genres)
temp_genres
                                INNER JOIN genres ON genres.genre_title =
temp_genres
                                INNER JOIN films ON films.film_title =
genres.film_title
                                WHERE films.film_title != ALL(_film_title)
```

```
                );

            IF _suggest_films = '{}' THEN
                 _suggest_films := ARRAY(
                     SELECT films.film_title FROM films ORDER BY film_title
DESC LIMIT 5
                  );
                  return _suggest_films;
            ELSE
                return _suggest_films;
            END IF;
       ELSE
           _suggest_films := ARRAY(
                     SELECT films.film_title FROM films ORDER BY film_title
DESC LIMIT 5
                );
           return _suggest_films;
       END IF;

  END;
$$ LANGUAGE plpgsql;

SELECT film_suggestion('meshu');
```

Generating Data

```
SELECT add_film_person('1121', 'meshu', 'male','02-05-1994');
SELECT add_film_person('1212', 'johh, 'male','02-05-1994');
SELECT add_film_person('12123', 'wich, 'male','02-05-1994');
SELECT add_film_person('1124', 'naf, 'female','02-05-1994');
SELECT add_film_person('1125', 'ben, 'other','02-05-1994');
SELECT add_film_person('1126', 'Rick', 'male','02-05-1994');
SELECT add_film_person('1127', 'Bren, 'female','02-05-1994');
SELECT add_film_person('1128', 'Michel, 'male','02-05-1994');
SELECT add_film_person('1219', 'Sam', 'male','02-05-1994');
SELECT add_film_person('11102', 'Mishty', 'female','02-05-1994');

SELECT add_film ('1121', 'horror movie 01', '2019','germany', '18');
SELECT add_film ('1122', 'horror movie 02', '2018','germany', '18');
SELECT add_film ('1123', 'horror movie 03', '2017','germany', '19');
SELECT add_film ('1214', 'horror movie 04', '2016','germany', '17');
SELECT add_film ('1215', 'romantic movie 01', '2016','poland', '15');
SELECT add_film ('1216', 'romantic movie 02', '2017','poland', '13');
SELECT add_film ('1127', 'romantic movie 03', '2018','poland', '18');
SELECT add_film ('1129', 'action movie 01', '2016','denmark', '18');
SELECT add_film ('2120', 'action movie 02', '2018','denmark', '20');
SELECT add_film ('1221', 'action movie 03', '2022','denmark', '18');

SELECT add_role('1121', 'actor', 'meshu','horror movie 01', '2019');
SELECT add_role('1122', 'writer', 'meshu','horror movie 01', '2019');
SELECT add_role('1213', 'singer', 'meshu','horror movie 01', '2019');
SELECT add_role('1214', 'actor', 'meshu','horror movie 01', '2019');
SELECT add_role('12155', 'actor', 'bren','horror movie 02', '2018');
SELECT add_role('12156', 'actor', 'mishty','action movie 03', '2022');
SELECT add_role('122157', 'actor', 'ben','horror movie 04', '2016');
SELECT add_role('113358', 'actor', 'michel','romantic movie 01', '2016');

SELECT add_subordinate_film('114411', 'horror movie 01', '2019','horror movie
02', '2018');
SELECT add_subordinate_film('331112', 'horror movie 01', '2019','horror movie
03', '2017');
SELECT add_subordinate_film('441113', 'horror movie 01', '2019','horror movie
04', '2016');

SELECT add_genre('66111', 'horror', 'horror movie 01','2019');
SELECT add_genre('55112', 'crime', 'horror movie 02','2018');
SELECT add_genre('44113', 'action', 'action movie 01','2016');
```

```
SELECT add_genre('33114', 'action', 'action movie 02','2018');
SELECT add_genre('33116', 'adventure', 'action movie 03','2022');

SELECT add_user('1111', 'user01');
SELECT add_user('1113', 'user02');
SELECT add_user('1114', 'user03');
SELECT add_user('12115', 'user04');
SELECT add_user('1116', 'user05');

SELECT add_rating('1121611', '5', 'user01','horror movie 01', '2019');
SELECT add_rating('1121311', '5', 'user01','horror movie 02', '2018');
SELECT add_rating('111d511', '6', 'user01','horror movie 03', '2017');
SELECT add_rating('1121311', '5', 'user01','horror movie 04', '2016');
SELECT add_rating('1181311', '5', 'user02','horror movie 01', '2019');
SELECT add_rating('118911', '5', 'user03','horror movie 01', '2019');
SELECT add_rating('1980311', '5', 'user04','action movie 01', '2016');
SELECT add_rating('1381311', '6', 'user02','action movie 02', '2018');
SELECT add_rating('1992311', '8', 'user03','action movie 03', '2022');
SELECT add_rating('1185611', '7', 'user04','romantic movie 01', '2018');
SELECT add_rating('1144311', '2', 'user03','romantic movie 01', '2018');
SELECT add_rating('113311', '3', 'user01','action movie 01', '2016');
```