/*Import tables from .csv files*/

```
CREATE TABLE Movies (
    movieID INT PRIMARY KEY,
    title VARCHAR(255) NOT NULL
);/*# Rows=85,716*/

CREATE TABLE MoviesSupplement (
    imdb_id VARCHAR(10) PRIMARY KEY,
    release_year INT CHECK (release_year >= 1800),
    director VARCHAR(255),
    language VARCHAR(50),
    poster_link VARCHAR(500)
);/*# Rows=85,716*/

CREATE TABLE Links (
    movieID INT PRIMARY KEY,
    imdb_id VARCHAR(10),
    tmdbId VARCHAR(10),
    FOREIGN KEY (movieID) REFERENCES Movies(movieID) ON DELETE CASCADE,
    FOREIGN KEY (imdb_id) REFERENCES MoviesSupplement(imdb_id) ON DELETE
CASCADE
);/*# Rows=85,716*/

CREATE TABLE MoviesGenres (
    movieID INT,
    genre VARCHAR(255),
    PRIMARY KEY (movieID, genre),
    FOREIGN KEY (movieID) REFERENCES Movies(movieID)
);/*# Rows=150,922*/

CREATE TABLE Ratings (
    movieID INT,
    userID INT,
    rating DECIMAL(2, 1),
    PRIMARY KEY (movieID, userID)
    /*
        FOREIGN KEY (movieID) REFERENCES Movies(movieID)
    FOREIGN KEY (userID) REFERENCES Users(userID)*/
);/*# Rows=33,832,162*/

CREATE TABLE GenomeTags (
    tagID INT PRIMARY KEY,
```

```
    tag VARCHAR(100) NOT NULL
);/*# Rows=1,128*/

CREATE TABLE GenomeScores (
    movieID INT,
    tagID INT,
    relevance DECIMAL(3, 2) CHECK (relevance >= 0.0 AND relevance <= 1.0),
    PRIMARY KEY (movieID, tagID),
    FOREIGN KEY (movieID) REFERENCES Movies(movieID) ON DELETE CASCADE,
    FOREIGN KEY (tagID) REFERENCES GenomeTags(tagID) ON DELETE CASCADE
);/*# Rows=18,309,696*/

CREATE TABLE Users (
    userID VARCHAR(8) PRIMARY KEY,
    password VARCHAR(8) NOT NULL
);/*initial # Rows=0*/

CREATE TABLE Likes (
    userID VARCHAR(8),
    movieID INT,
    PRIMARY KEY (userID, movieID),
    FOREIGN KEY (userID) REFERENCES Users(userID),
    FOREIGN KEY (movieID) REFERENCES Movies(movieID)
);/*initial # Rows=0*/

/*
Schema
a.  Movies (movieID, title)
b.  MoviesSupplement (imdb_id, release_year, director, language, poster_link)
c.  Links (movieID, imdb_id)
d.  MoviesGenres (movieID, genre)
e.  Ratings (movieID, userID, rating, timestamp)
f.  GenomeScores (movieID, tagID, relevance)
g.  GenomeTags (tagID, tag)
h.  Users (userID, password)
i.  Like (userID, movieID)
*/

/*Create indexes and frequently used intermediate table*/

CREATE INDEX index_r    ON Ratings (movieid,userid);
CREATE INDEX index_mr   ON MovieRatings (movieid);
CREATE INDEX index_m    ON Movies (movieid);
```

```
CREATE INDEX index_gs     ON GenomeScores (movieID, tagID);
CREATE INDEX index_mg     ON MoviesGenres(movieID);
CREATE INDEX index_l      ON Links(movieID, imdb_id);
CREATE INDEX index_lk     ON Likes(userID, movieID);
CREATE INDEX index_ms     ON MoviesSupplement_link(movieID);

DROP INDEX index_r;
DROP INDEX index_mr;
DROP INDEX index_m;
DROP INDEX index_gs;
DROP INDEX index_mg;
DROP INDEX index_l;
DROP INDEX index_lk;
DROP INDEX index_ms;
```

/*Prepare precalculated tables*/

```
CREATE TABLE MovieRatings AS
        SELECT movieid,
        ROUND(AVG(rating), 3) AS average_rating,
        COUNT(*) AS n_rating
        FROM ratings
        GROUP BY movieid;

CREATE TABLE Movies_add AS
        SELECT m.movieid,l.tmdbid,m.title,mr.average_rating,mr.n_rating,ms.release_year,
            ms.language,ms.director,ms.poster_link,mg.genre
        FROM Movies m
        LEFT JOIN MovieRatings mr ON m.movieID=mr.movieID
        LEFT JOIN links l ON m.movieID=l.movieID
        LEFT JOIN MoviesSupplement ms ON ms.imdb_id= l.imdb_id
        LEFT JOIN (SELECT movieid, STRING_AGG(genre, ',') AS genre FROM moviesgenres
GROUP BY movieid) mg
   ON m.movieID = mg.movieID
        ORDER BY m.movieID;
```

/*Create shared user account for group members and grant access*/

```
CREATE USER grp15_share WITH PASSWORD 'share_PW5500';

GRANT ALL PRIVILEGES ON Movies                  TO grp15_share;
GRANT ALL PRIVILEGES ON MoviesSupplement  TO grp15_share;
GRANT ALL PRIVILEGES ON Links                   TO grp15_share;
```

```
GRANT ALL PRIVILEGES ON MoviesGenres      TO grp15_share;
GRANT ALL PRIVILEGES ON Ratings           TO grp15_share;
GRANT ALL PRIVILEGES ON GenomeTags              TO grp15_share;
GRANT ALL PRIVILEGES ON GenomeScores      TO grp15_share;
GRANT ALL PRIVILEGES ON Users                   TO grp15_share;
GRANT ALL PRIVILEGES ON Likes                   TO grp15_share;
GRANT ALL PRIVILEGES ON links             TO grp15_share;
```

/*Queries*/

/*1. Home Page: Display the top 10 highest-rated movies with names and posters*/

/*After optimization*/

```
/*Prepare precalculated tables*/
CREATE TABLE MovieRatings AS
        SELECT movieid,
        ROUND(AVG(rating), 3) AS average_rating,
        COUNT(*) AS n_rating
        FROM ratings
        GROUP BY movieid;

CREATE TABLE Movies_add AS
        SELECT m.movieid,l.tmdbid,m.title,mr.average_rating,mr.n_rating,ms.release_year,
             ms.language,ms.director,ms.poster_link,mg.genre
        FROM Movies m
        LEFT JOIN MovieRatings mr ON m.movieID=mr.movieID
        LEFT JOIN links l ON m.movieID=l.movieID
        LEFT JOIN MoviesSupplement ms ON ms.imdb_id= l.imdb_id
        LEFT JOIN (SELECT movieid, STRING_AGG(genre, ',') AS genre
                FROM moviesgenres GROUP BY movieid) mg
   ON m.movieID = mg.movieID
        ORDER BY m.movieID;
SELECT
        m.movieID,
        m.title,
        m.average_rating,
        m.tmdbid
FROM Movies_add as m
WHERE m.n_rating>=100
```

```
        AND m.average_rating IS NOT NULL
ORDER BY m.average_rating DESC
LIMIT 10;

/*Before optimization*/
/*
WITH rankratings AS(
   SELECT
      m.movieID,
      mr.average_rating,
      mr.n_rating
   FROM
      movies as m
   LEFT JOIN
      (SELECT movieid,
                  ROUND(AVG(rating), 3) AS average_rating,
                  COUNT(*) AS n_rating
                  FROM ratings
                  GROUP BY movieid) as mr
         ON m.movieID=mr.movieID
   WHERE average_rating IS NOT NULL
   ORDER BY
      mr.average_rating DESC)
SELECT
   rr.*,
   m.title,
   ms.poster_link
FROM rankratings as rr
LEFT JOIN
   movies as m ON rr.movieID=m.movieID
LEFT JOIN links k ON m.movieID=k.movieID
LEFT JOIN MoviesSupplement ms ON k. imdb_id = ms.imdb_id
WHERE rr.n_rating >=100
ORDER BY average_rating DESC
LIMIT 10;
*/
```

/*2. Home Page(Filter by genre): users can select a genre to display the top 10 highest-rated movies for that specific genre*/

```
/*After optimization*/
SELECT
```

```
    m.movieID,
    m.title,
    m.average_rating,
    m.tmdbid
FROM movies_add as m
WHERE m.genre ILIKE '%Comedy%'
          AND m.average_rating IS NOT NULL
          AND m.n_rating >=100
ORDER BY
    m.average_rating DESC
LIMIT 10;

/*Before optimization*/
/*
WITH rankratings AS(
    SELECT
        mg.movieID,
        mg.genre,
        mr.average_rating,
        mr.n_rating
    FROM
        Moviesgenres as mg
    LEFT JOIN
        (SELECT movieid,
                    ROUND(AVG(rating), 3) AS average_rating,
                    COUNT(*) AS n_rating
                    FROM ratings
                    GROUP BY movieid) as mr ON mg.movieID=mr.movieID
    WHERE mg.genre in ('Comedy') AND average_rating IS NOT NULL
    ORDER BY
        mr.average_rating DESC)
SELECT
    rr.*,
    m.title
FROM rankratings as rr
LEFT JOIN
    movies as m ON rr.movieID=m.movieID
WHERE rr.n_rating >=100
ORDER BY average_rating DESC
LIMIT 10;
*/
```

/*3.1 Login/Signup Page: when a user creates a new account, add the username and password to the Users table.*/

```
INSERT INTO Users (userID, password)
VALUES ('Group15', 'Grp15pw');
```

/*3.2 Login/Signup Page: when a user updates the password, update the password in the Users table.*/

```
UPDATE Users
SET password = 'Grp15pw2'
WHERE userID = 'Group15';
```

/*4.1 My List Page: When a user ‘like’ a movie, add the user’s id and the movie to the Like table.*/

```
INSERT INTO likes (userID, movieid)
VALUES ('Group15', '1'),
      ('Group15', '2'),
      ('Group15', '3'),
      ('Group15', '4'),
      ('Group15', '5'),
      ('Group15', '6'),
      ('Group15', '7'),
      ('Group15', '8'),
      ('Group15', '9'),
      ('Group15', '10');
```

/*4.2 My List Page: When a user ‘unlike’ a movie, remove the user’s id and the movie from the Like table.*/

```
DELETE FROM likes
WHERE userID = $1 AND movieID = $2;
```

/*5. Display logged-in user's liked movies info*/

```
/*After optimization*/
SELECT
   m.title,
```

```
    m.tmdbId,
    m.average_rating,
    m.movieID
FROM likes l
JOIN Movies_add m ON m.movieID=l.movieID
WHERE l.userID = $1
ORDER BY m.average_rating DESC;

/*Before optimization*/
/*
SELECT
    m.title,
    ms.poster_link
FROM
    Movies m
LEFT JOIN links k ON m.movieID=k.movieID
LEFT JOIN MoviesSupplement ms ON k. imdb_id = ms.imdb_id
LEFT JOIN
    Movieratings mr ON m.movieID = mr.movieID
WHERE
    m.movieID in
    (SELECT movieid FROM likes
     WHERE likes.userID = $1)
ORDER BY m.movieID;

SELECT
    m.title,
    ms.poster_link,
    mr.average_rating,
    mr.n_rating
FROM Movies m
JOIN likes l ON m.movieID=l.movieID
LEFT JOIN links k ON m.movieID=k.movieID
LEFT JOIN MoviesSupplement ms ON k. imdb_id = ms.imdb_id
LEFT JOIN Movieratings mr ON m.movieID = mr.movieID
WHERE l.userID = $1
ORDER BY mr.average_rating;
*/


/*6. Access movie details: Users can click on a movie to go to the Movie Details Page*/

SELECT
```

```
    m.movieID,
    m.title,
    m.release_year,
    m.director,
    m.language,
    m.genre,
    m.tmdbid,
    m.average_rating
FROM Movies_add m
WHERE m.movieID=$1
ORDER BY average_rating DESC;
```

/*7.1 Recommendation Page: Find top 3 genres in likes and find the top 10 rated movies in those genres.*/

```
/*After optimization*/
WITH TopGenres AS (
    SELECT mg.genre
    FROM Likes l
    JOIN MoviesGenres mg ON l.movieID = mg.movieID
    WHERE l.userID = 'Group15'
    GROUP BY mg.genre
    ORDER BY COUNT(*) DESC
    LIMIT 3
)
SELECT m.movieID, m.title, m.average_rating, m.tmdbid
FROM Movies_add m
JOIN MoviesGenres mg ON m.movieID = mg.movieID
JOIN TopGenres tg ON mg.genre = tg.genre
WHERE m.movieID NOT IN (
    SELECT movieID FROM Likes WHERE userID = 'Group15'
)
AND n_rating>1000
ORDER BY average_rating DESC
LIMIT 10;
```

/*7.2 Recommendation Page: Find top 3 directors in likes and find the top 10 rated movies directed by these directors.*/

```
/*After optimization*/
WITH TopDirectors AS (
```

```
   SELECT m.director
   FROM Likes l
   JOIN Movies_add m ON l.movieID = m.movieID
   WHERE l.userID = 'Group15'
   GROUP BY m.director
   ORDER BY COUNT(*) DESC
   LIMIT 3)
SELECT
        m.movieID,
        m.title,
        m.average_rating,
        m.tmdbid
FROM Movies_add m
JOIN TopDirectors td ON m.director = td.director
WHERE m.movieID NOT IN (SELECT movieID FROM Likes WHERE userID = 'Group15')
        /*AND mr.n_rating > 1000*/
ORDER BY m.average_rating DESC
LIMIT 10;

/*Before optimization*/
/*
WITH TopDirectors AS (
   SELECT ms.director
   FROM Likes l
   JOIN Movies m ON l.movieID = m.movieID
   LEFT JOIN links k ON m.movieID=k.movieID
LEFT JOIN MoviesSupplement ms ON k. imdb_id = ms.imdb_id

   WHERE l.userID = 'Group15'
   GROUP BY ms.director
   ORDER BY COUNT(*) DESC
   LIMIT 3)
SELECT
        m.movieID,
        m.title,
        ms.director,
        mr.average_rating,
        mr.n_rating
FROM Movies m
LEFT JOIN MoviesSupplement_link ms ON m.movieID= ms.movieID
JOIN TopDirectors td ON ms.director = td.director
JOIN MovieRatings mr ON m.movieID = mr.movieID
WHERE m.movieID NOT IN (SELECT movieID FROM Likes WHERE userID = 'Group15')
```

```
ORDER BY mr.average_rating DESC
LIMIT 10;
*/
```

/*7.3 Recommendation Page: Find tags in liked movies, rank by frequency, then based on the movie tags from likes, create recommendations.*/

```
/*After optimization*/
WITH top_tags AS (
   SELECT
      gs.tagid
   FROM likes l
   JOIN genomescores gs ON l.movieID = gs.movieID
   WHERE gs.relevance >= 0.5 AND l.userID = 'Group15'
   GROUP BY gs.tagid
   ORDER BY COUNT(*) DESC
   LIMIT 10
),
filtered_movies AS (
   SELECT DISTINCT gm.movieID
   FROM genomescores gm
   JOIN top_tags tt ON gm.tagid = tt.tagid
   WHERE gm.relevance > 0.5
)
SELECT
   m.movieID,
   m.title,
   m.average_rating,
   m.tmdbid
FROM filtered_movies fm
JOIN Movies_add m ON fm.movieID = m.movieID
WHERE NOT EXISTS (
   SELECT 1
   FROM Likes l
   WHERE l.movieID = m.movieID AND l.userID = 'Group15'
)
   AND n_rating>100
ORDER BY m.average_rating DESC
LIMIT 10;

/*Before optimization*/
/*
```

```
WITH top_tags as (
SELECT
tag,
tagid,
count(*) as n_liked_tags
FROM (
   SELECT
      l.*,
      gs.tagid,
      gs.relevance,
      gt.tag
   FROM likes l
   LEFT JOIN genomescores gs ON l.movieID=gs.movieID
   LEFT JOIN genometags gt ON gs.tagid=gt.tagid
   WHERE relevance>=0.5
   ORDER BY l.movieID,gs.relevance DESC) ttt
         GROUP BY tag,tagid
         ORDER BY n_liked_tags DESC
         LIMIT 10)
SELECT
         gs.movieID,
         m.movieID,
         m.title,
         mr.average_rating,
         mr.n_rating
FROM (SELECT DISTINCT movieID
FROM genomescores
WHERE relevance>0.5
         AND tagid IN (SELECT tagid FROM top_tags)) gs
JOIN Movies m ON gs.movieid=m.movieid
LEFT JOIN links k ON m.movieID=k.movieID
LEFT JOIN MoviesSupplement ms ON k. imdb_id = ms.imdb_id
JOIN MovieRatings mr ON m.movieID = mr.movieID
         WHERE m.movieID NOT IN (
   SELECT movieID FROM Likes WHERE userID = 'Group15')
ORDER BY mr.average_rating DESC
LIMIT 10;
*/
```

/*8. Movie Search Page: Search movies by title/tag, genre, director, language, or release year.*/

```
/*After optimization*/
SELECT
        m.movieID,
        m.title,
        m.release_year,
        m.director,
        m.average_rating,
        m.tmdbid
FROM Movies_add m
WHERE (m.title ILIKE '%search_term%' AND
    m.genre ILIKE '%Comedy%' AND
    m.director ILIKE '%search_term%' AND
    m.language ILIKE '%search_term%' AND
    m.release_year = '1990')
ORDER BY m.average_rating DESC, m.release_year DESC;

/*Before optimization*/
/*
SELECT
        m.movieID,
        m.title,
        ms.release_year,
        ms.director,
        ms.language,
        mr.average_rating
FROM Movies m
LEFT JOIN MoviesSupplement_link ms ON m.movieID= ms.movieID
LEFT JOIN MoviesGenres mg ON m.movieID = mg.movieID
LEFT JOIN GenomeScores gs ON m.movieID = gs.movieID
LEFT JOIN GenomeTags gt ON gs.tagID = gt.tagID
LEFT JOIN movieratings mr ON m.movieID = mr.movieID
WHERE (m.title ILIKE '%search_term%' OR
    gt.tag ILIKE '%search_term%' OR
    mg.genre ILIKE '%Comedy%' OR
    ms.director ILIKE '%search_term%' OR
    ms.language ILIKE '%search_term%' OR
    ms.release_year = 'year')
ORDER BY average_rating DESC, ms.release_year DESC;
*/
```