

ASSIGNMENT 5

- Anirudh Narayanan(an9425)

README:

RUN THE QUERIES BELOW FOR EACH QUESTION.

OUTPUT IS PROVIDED IN THE END FOR QUESTION 3,4,5.

```
-----Q1-----
---- CREATE VIEWS-----
-- CREATE VIEW WITH COMEDY MOVIES
CREATE VIEW ComedyMovies
AS
SELECT Distinct tr.tid, tr.original_title, tr.startYear
FROM title tr
      JOIN title_genre tg on tr.tid = tg.title
      JOIN genre g on tg.genre = 5
WHERE (cast(tr.runtimeminutes as int) > 75);

-- CREATE VIEW WITH EVERYTHING EXCEPT COMEDY MOVIES
CREATE VIEW NonComedyMovies
AS
SELECT Distinct tr.tid, tr.original_title, tr.startYear
FROM title tr
      JOIN title_genre tg on tr.tid = tg.title
      JOIN genre g on tg.genre != 5
WHERE (cast(tr.runtimeminutes as int) > 75);

-- CREATE VIEW WITH ACTOR WHO HAVE ACTED IN AT LEAST 1 COMEDY MOVIE
CREATE VIEW ComedyActor
AS
SELECT Distinct nm.nid, nm.primaryName, nm.birthYear, nm.deathYear
FROM name_basics nm
      JOIN actor_title_character atc on nm.nid = atc.actor
      JOIN title_genre tg on atc.title = tg.title
      JOIN ComedyMovies cm on cm.tid = atc.title
      JOIN genre g on tg.genre = 5;

-- CREATE VIEW WITH ACTOR WHO HAVE NOT ACTED IN COMEDY MOVIES
CREATE VIEW NonComedyActor
AS
SELECT Distinct nm.nid, nm.primaryName, nm.birthYear, nm.deathYear
FROM name_basics nm
      JOIN actor_title_character atc on nm.nid = atc.actor
      JOIN title_genre tg on atc.title = tg.title
      JOIN NonComedyMovies ncm on ncm.tid = atc.title
```

```

JOIN genre g on tg.genre != 5;

-- CREATE VIEW WITH LIST OF ACTORS FOR EACH MOVIE
CREATE VIEW ActedIn AS
SELECT DISTINCT atc.title AS movie, nb.nid AS actor
FROM actor_title_character atc
    JOIN name_basics nb ON atc.actor = nb.nid
    JOIN title t ON t.tid = atc.title
where (cast(t.runtimeminutes as int) > 75);

-----CREATE MATERIALIZED VIEWS-----

CREATE MATERIALIZED VIEW ComedyMoviesm
AS
SELECT Distinct tr.tid, tr.original_title, tr.startYear
FROM title tr
    JOIN title_genre tg on tr.tid = tg.title
    JOIN genre g on tg.genre = 5
WHERE (cast(tr.runtimeminutes as int) > 75);

CREATE MATERIALIZED VIEW NonComedyMoviesm
AS
SELECT Distinct tr.tid, tr.original_title, tr.startYear
FROM title tr
    JOIN title_genre tg on tr.tid = tg.title
    JOIN genre g on tg.genre != 5
WHERE (cast(tr.runtimeminutes as int) > 75);

CREATE MATERIALIZED VIEW ComedyActorm
AS
SELECT Distinct nm.nid, nm.primaryName, nm.birthYear, nm.deathYear
FROM name_basics nm
    JOIN actor_title_character atc on nm.nid = atc.actor
    JOIN title_genre tg on atc.title = tg.title
    JOIN ComedyMovies cm on cm.tid = atc.title
    JOIN genre g on tg.genre = 5;

CREATE MATERIALIZED VIEW NonComedyActorm
AS
SELECT Distinct nm.nid, nm.primaryName, nm.birthYear, nm.deathYear
FROM name_basics nm
    JOIN actor_title_character atc on nm.nid = atc.actor
    JOIN title_genre tg on atc.title = tg.title
    JOIN NonComedyMovies ncm on ncm.tid = atc.title
    JOIN genre g on tg.genre != 5;

CREATE MATERIALIZED VIEW ActedInm AS
SELECT DISTINCT atc.title AS movie, nb.nid AS actor

```

```
FROM actor_title_character atc
  JOIN name_basics nb ON atc.actor = nb.nid
  JOIN title t ON t.tid = atc.title
where (cast(t.runtimeminutes as int) > 75);
```

-----Q2-----

-----CREATE GLOBAL VIEWS-----

-----CREATE VIEWS-----

```
CREATE VIEW All_Movie (tid, title, year, genres) AS
SELECT cm.tid, cm.original_title, cm.startYear, 'Comedy' AS genres
FROM ComedyMovies cm
UNION ALL
SELECT ncm.tid, ncm.original_title, ncm.startYear, 'Non-Comedy' AS genres
FROM NonComedyMovies ncm;
```

```
CREATE VIEW All_Actor (tid, name, birthyear, deathyear) AS
SELECT ca.nid, ca.primaryname, ca.birthyear, ca.deathYear
FROM Comedyactor ca
UNION ALL
SELECT nca.nid, nca.primaryName, nca.birthYear, nca.deathYear
FROM NonComedyactor nca;
```

```
CREATE VIEW All_movie_actor (movie, actor) AS
SELECT *
from actedin;
```

-----CREATE MATERIALIZED GLOBAL VIEWS-----

```
CREATE VIEW All_Moviem (tid, title, year, genres) AS
SELECT cm.tid, cm.original_title, cm.startYear, 'Comedy' AS genres
FROM ComedyMoviesm cm
UNION ALL
SELECT ncm.tid, ncm.original_title, ncm.startYear, 'Non-Comedy' AS genres
FROM NonComedyMoviesm ncm;
```

```
CREATE VIEW All_Actorm (tid, name, birthyear, deathyear) AS
SELECT ca.nid, ca.primaryname, ca.birthyear, ca.deathYear
FROM Comedyactorm ca
UNION ALL
SELECT nca.nid, nca.primaryName, nca.birthYear, nca.deathYear
FROM NonComedyactorm nca;
```

```
CREATE VIEW All_movie_actorm (movie, actor) AS
SELECT *
from actedinm;
```

-----Q3-----

--3.1. Alive actors who have participated
-- in more than 10 movies between 2000 and 2005.

--USING VIEW--

```
SELECT aa2.name, COUNT(DISTINCT ama.movie) as mov_count
FROM All_Movie_Actor ama
    JOIN All_Movie am ON ama.movie = am.tid
    JOIN all_actor aa2 ON aa2.tid = ama.actor
WHERE cast(am.year as int) BETWEEN 2000 and 2005 and aa2.deathyear is NULL
GROUP BY (aa2.name)
HAVING COUNT(DISTINCT ama.movie) > 10; -- TIME:25 s 598 ms
```

--USING MATERIALIZED VIEW--

```
SELECT aa2.name, COUNT(DISTINCT ama.movie) as mov_count
FROM All_Movie_Actorm ama
    JOIN All_Moviem am ON ama.movie = am.tid
    JOIN all_actorm aa2 ON aa2.tid = ama.actor
WHERE cast(am.year as int) BETWEEN 2000 and 2005 and aa2.deathyear is NULL
GROUP BY (aa2.name)
HAVING COUNT(DISTINCT ama.movie) > 10;-- TIME: 320 ms
```

--3.2

--Actors whose name starts with "Ja" and who have never participated in any comedy movie.

--USING VIEW--

```
SELECT Distinct aa.name, am.title
FROM All_Actor aa
    JOIN all_movie_actor ama on aa.tid = ama.actor
    JOIN all_movie am on ama.movie = am.tid
WHERE aa.name LIKE 'Ja%'
and am.tid not in
(SELECT DISTINCT ama.movie
FROM All_Movie_Actor ama
    JOIN All_Movie am ON ama.movie = am.tid
WHERE am.genres = 'Comedy'); -- TIME: 34 s 684 ms
```

```
--USING MATERIALIZED VIEW--
SELECT Distinct aa.name, am.title
FROM All_Actorm aa
    JOIN all_movie_actorm ama on aa.tid = ama.actor
    JOIN all_moviem am on ama.movie = am.tid
WHERE aa.name LIKE 'Ja%'
and am.tid not in
    (SELECT DISTINCT ama.movie
    FROM All_Movie_Actorm ama
    JOIN All_Moviem am ON ama.movie = am.tid
    WHERE am.genres = 'Comedy');
-- TIME: 578 ms
```

\

-----Q4-----

--4.1

--USING VIEW--

```
SELECT aa2.name, COUNT(DISTINCT ama.movie) as mov_count
FROM (SELECT * from actedin) as ama
    JOIN (SELECT cm.tid, cm.original_title, cm.startYear as year, 'Comedy' AS genres
FROM ComedyMovies cm
UNION ALL
SELECT ncm.tid, ncm.original_title, ncm.startYear as year, 'Non-Comedy' AS genres
FROM NonComedyMovies ncm) as am ON ama.movie = am.tid
    JOIN (SELECT ca.nid, ca.primaryname as name, ca.birthyear, ca.deathYear
FROM Comedyactor ca
UNION ALL
SELECT nca.nid, nca.primaryName as name, nca.birthYear, nca.deathYear
FROM NonComedyactor nca) as aa2 ON aa2.nid = ama.actor
WHERE cast(am.year as int) BETWEEN 2000 and 2005 and aa2.deathyear is NULL
GROUP BY (aa2.name)
HAVING COUNT(DISTINCT ama.movie) > 10; -- TIME:28 s 471 ms
```

```

--USING MATERIALIZED VIEW--
SELECT aa2.name, COUNT(DISTINCT ama.movie) as mov_count
FROM (SELECT * from actedinm) as ama
      JOIN (SELECT cm.tid, cm.original_title, cm.startYear as year, 'Comedy' AS genres
FROM ComedyMoviesm cm
UNION ALL
SELECT ncm.tid, ncm.original_title, ncm.startYear as year, 'Non-Comedy' AS genres
FROM NonComedyMoviesm ncm) as am ON ama.movie = am.tid
      JOIN (SELECT ca.nid, ca.primaryname as name, ca.birthyear, ca.deathYear
FROM Comedyactor ca
UNION ALL
SELECT nca.nid, nca.primaryName as name, nca.birthYear, nca.deathYear
FROM NonComedyactor nca) as aa2 ON aa2.nid = ama.actor
WHERE cast(am.year as int) BETWEEN 2000 and 2005 and aa2.deathyear is NULL
GROUP BY (aa2.name)
HAVING COUNT(DISTINCT ama.movie) > 10; -- TIME:459 ms

```

```

--4.2
--USING VIEW--
SELECT Distinct aa.name, am.title
FROM (SELECT ca.nid, ca.primaryname as name, ca.birthyear, ca.deathYear
FROM Comedyactor ca
UNION ALL
SELECT nca.nid, nca.primaryName as name, nca.birthYear, nca.deathYear
FROM NonComedyactor nca) as aa
      JOIN (SELECT * from actedin) as ama on aa.nid = ama.actor
      JOIN (SELECT cm.tid, cm.original_title as title, cm.startYear, 'Comedy' AS genres
FROM ComedyMovies cm
UNION ALL
SELECT ncm.tid, ncm.original_title as title, ncm.startYear, 'Non-Comedy' AS genres
FROM NonComedyMovies ncm) as am on ama.movie = am.tid
WHERE aa.name LIKE 'Ja%'
and am.tid not in
  (SELECT DISTINCT ama.movie
FROM (SELECT * from actedin) as ama
      JOIN (SELECT cm.tid, cm.original_title, cm.startYear, 'Comedy' AS genres
FROM ComedyMovies cm
UNION ALL
SELECT ncm.tid, ncm.original_title, ncm.startYear, 'Non-Comedy' AS genres
FROM NonComedyMovies ncm) as am ON ama.movie = am.tid
WHERE am.genres = 'Comedy');
-- TIME :34 s 362 ms

```

```
--USING MATERIALIZED VIEW--
SELECT Distinct aa.name, am.title
FROM (SELECT ca.nid, ca.primaryname as name, ca.birthyear, ca.deathYear
      FROM Comedyactorm ca
      UNION ALL
      SELECT nca.nid, nca.primaryName as name, nca.birthYear, nca.deathYear
      FROM NonComedyactorm nca) as aa
      JOIN (SELECT * from actedinm) as ama on aa.nid = ama.actor
      JOIN (SELECT cm.tid, cm.original_title as title, cm.startYear, 'Comedy' AS genres
            FROM ComedyMoviesm cm
            UNION ALL
            SELECT ncm.tid, ncm.original_title as title, ncm.startYear, 'Non-Comedy' AS genres
            FROM NonComedyMoviesm ncm) as am on ama.movie = am.tid
WHERE aa.name LIKE 'Ja%'
and am.tid not in
  (SELECT DISTINCT ama.movie
   FROM (SELECT * from actedinm) as ama
        JOIN (SELECT cm.tid, cm.original_title, cm.startYear, 'Comedy' AS genres
              FROM ComedyMoviesm cm
              UNION ALL
              SELECT ncm.tid, ncm.original_title, ncm.startYear, 'Non-Comedy' AS genres
              FROM NonComedyMoviesm ncm) as am ON ama.movie = am.tid
        WHERE am.genres = 'Comedy'); -- TIME: 593ms
```

-----Q5-----

```
--5.1
--USING VIEW--

SELECT aa2.name, COUNT(DISTINCT ama.movie) as mov_count
FROM (SELECT * from actedin) as ama
      JOIN (SELECT cm.tid, cm.startYear as year
            FROM ComedyMovies cm
            UNION ALL
            SELECT ncm.tid, ncm.startYear as year
            FROM NonComedyMovies ncm) as am ON ama.movie = am.tid
      JOIN (SELECT ca.nid, ca.primaryname as name, ca.deathYear
            FROM Comedyactor ca
            UNION ALL
            SELECT nca.nid, nca.primaryName as name, nca.deathYear
            FROM NonComedyactor nca) as aa2 ON aa2.nid = ama.actor
WHERE cast(am.year as int) BETWEEN 2000 and 2005 and aa2.deathyear is NULL
GROUP BY (aa2.name)
HAVING COUNT(DISTINCT ama.movie) > 10; -- TIME: 25 s 358 ms
```

TIME IMPROVEMENT: ~ 3s

```
--USING MATERIALIZED VIEW--
SELECT aa2.name, COUNT(DISTINCT ama.movie) as mov_count
FROM (SELECT * from actedinm) as ama
      JOIN (SELECT cm.tid, cm.startYear as year
            FROM ComedyMoviesm cm
            UNION ALL
            SELECT ncm.tid, ncm.startYear as year
            FROM NonComedyMoviesm ncm) as am ON ama.movie = am.tid
      JOIN (SELECT ca.nid, ca.primaryname as name, ca.deathYear
            FROM Comedyactor ca
            UNION ALL
            SELECT nca.nid, nca.primaryName as name, nca.deathYear
            FROM NonComedyactor nca) as aa2 ON aa2.nid = ama.actor
WHERE cast(am.year as int) BETWEEN 2000 and 2005 and aa2.deathyear is NULL
GROUP BY (aa2.name)
HAVING COUNT(DISTINCT ama.movie) > 10; -- TIME:200 ms
TIME IMPROVEMENT ~ 200ms
```

---5.2

```
--USING VIEW--
SELECT Distinct aa.name, am.title
FROM (SELECT ca.nid, ca.primaryname as name
      FROM Comedyactor ca
      UNION ALL
      SELECT nca.nid, nca.primaryName as name
      FROM NonComedyactor nca) as aa
      JOIN (SELECT * from actedin) as ama on aa.nid = ama.actor
      JOIN (SELECT ncm.tid, ncm.original_title as title
            FROM NonComedyMovies ncm) as am on ama.movie = am.tid
WHERE aa.name LIKE 'Ja%'
and am.tid not in
      (SELECT DISTINCT tid from ComedyMovies);-- TIME:23 s 979 ms
TIME IMPROVEMENT ~ 11s
```

```
--USING MATERIALIZED VIEW--
SELECT Distinct aa.name, am.title
FROM (SELECT ca.nid, ca.primaryname as name
      FROM Comedyactorm ca
      UNION ALL
      SELECT nca.nid, nca.primaryName as name
      FROM NonComedyactorm nca) as aa
      JOIN (SELECT * from actedinm) as ama on aa.nid = ama.actor
      JOIN (SELECT ncm.tid, ncm.original_title as title
            FROM NonComedyMoviesm ncm) as am on ama.movie = am.tid
WHERE aa.name LIKE 'Ja%'
and am.tid not in
```



```
(SELECT DISTINCT tid FROM ComedyMoviesm); -- TIME: 629 ms  
TIME IMPROVEMENT ~ 100ms
```

OUTPUT:

Q3,4,5: Output was the same for all the queries of Q3,4,5.

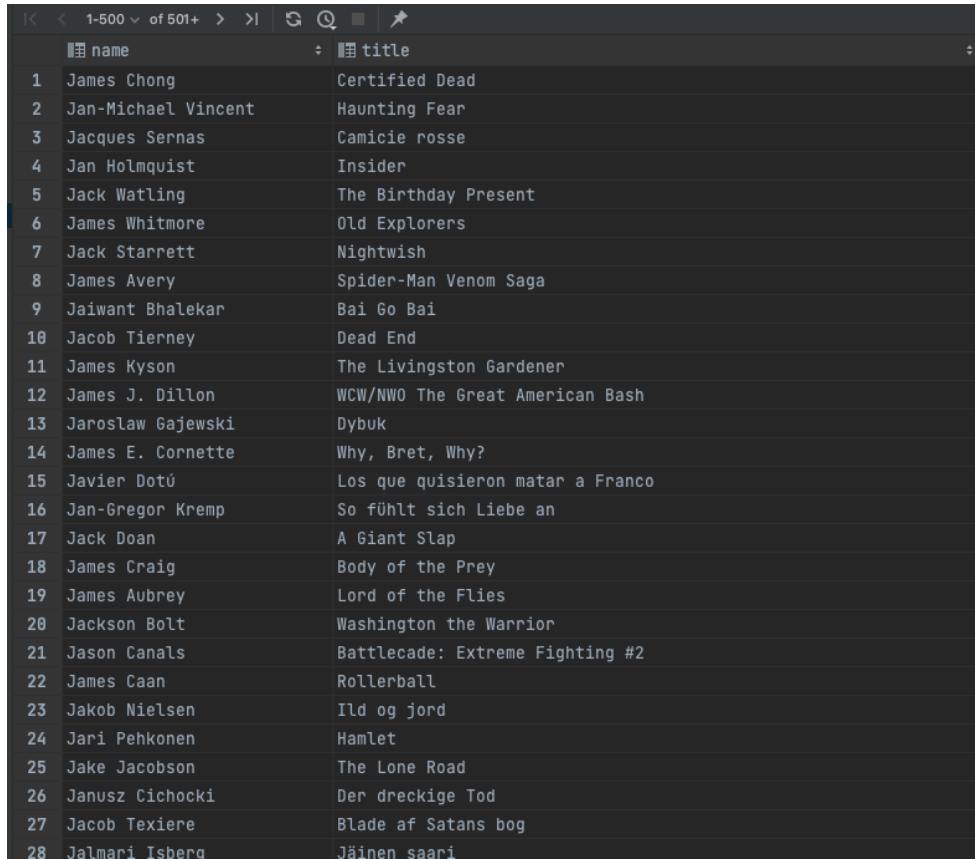
1.



The screenshot shows a database query result in a dark-themed interface. At the top, there is a navigation bar with a back arrow, a search bar containing '1-500 of 501+', and several icons. Below the navigation bar is a table with two columns: 'name' and 'mov_count'. The table contains 28 rows of data, each with a row number, an actor's name, and their corresponding movie count.

	name	mov_count
1	A.C. Connor	47
2	Abbas	11
3	Abhishek Bachchan	22
4	Adam Baldwin	13
5	Adam Beach	12
6	Adam Birch	30
7	Adam Copeland	254
8	Aftab Shivdasani	15
9	Ahmed Helmy	11
10	Aidan Quinn	14
11	Ajay Devgn	29
12	Ajith Kumar	13
13	Akira Emoto	13
14	Akshay Kumar	25
15	Aladin Reibel	11
16	Alberto Estrella	13
17	Alec Baldwin	12
18	Alec Newman	12
19	Aleksandr Baluev	16
20	Aleksandr Domogarov	13
21	Alex Fong	14
22	Alexandre Brasseur	11
23	Amin Hayayee	12
24	Amit Pachori	13
25	Amitabh Bachchan	26
26	André Dussollier	16
27	Andreas Hoppe	20
28	Andreas Kern	16

2.



The image shows a web browser window displaying a list of 28 items. The browser's address bar shows '1-500 of 501+'. The list has two columns: 'name' and 'title'. The items are numbered 1 through 28. The list includes names like James Chong, Jan-Michael Vincent, Jacques Sernas, Jan Holmquist, Jack Watling, James Whitmore, Jack Starrett, James Avery, Jaiwant Bhalekar, Jacob Tierney, James Kyson, James J. Dillon, Jaroslaw Gajewski, James E. Cornette, Javier Dotú, Jan-Gregor Kremp, Jack Doan, James Craig, James Aubrey, Jackson Bolt, Jason Canals, James Caan, Jakob Nielsen, Jari Pehkonen, Jake Jacobson, Janusz Cichocki, Jacob Texiere, and Jalmari Isberg. The titles include 'Certified Dead', 'Haunting Fear', 'Camicie rosse', 'Insider', 'The Birthday Present', 'Old Explorers', 'Nightwish', 'Spider-Man Venom Saga', 'Bai Go Bai', 'Dead End', 'The Livingston Gardener', 'WCW/NWO The Great American Bash', 'Dybuk', 'Why, Bret, Why?', 'Los que quisieron matar a Franco', 'So fühlt sich Liebe an', 'A Giant Slap', 'Body of the Prey', 'Lord of the Flies', 'Washington the Warrior', 'Battlecade: Extreme Fighting #2', 'Rollerball', 'Ild og jord', 'Hamlet', 'The Lone Road', 'Der dreckige Tod', 'Blade af Satans bog', and 'Jäinen saari'.

	name	title
1	James Chong	Certified Dead
2	Jan-Michael Vincent	Haunting Fear
3	Jacques Sernas	Camicie rosse
4	Jan Holmquist	Insider
5	Jack Watling	The Birthday Present
6	James Whitmore	Old Explorers
7	Jack Starrett	Nightwish
8	James Avery	Spider-Man Venom Saga
9	Jaiwant Bhalekar	Bai Go Bai
10	Jacob Tierney	Dead End
11	James Kyson	The Livingston Gardener
12	James J. Dillon	WCW/NWO The Great American Bash
13	Jaroslaw Gajewski	Dybuk
14	James E. Cornette	Why, Bret, Why?
15	Javier Dotú	Los que quisieron matar a Franco
16	Jan-Gregor Kremp	So fühlt sich Liebe an
17	Jack Doan	A Giant Slap
18	James Craig	Body of the Prey
19	James Aubrey	Lord of the Flies
20	Jackson Bolt	Washington the Warrior
21	Jason Canals	Battlecade: Extreme Fighting #2
22	James Caan	Rollerball
23	Jakob Nielsen	Ild og jord
24	Jari Pehkonen	Hamlet
25	Jake Jacobson	The Lone Road
26	Janusz Cichocki	Der dreckige Tod
27	Jacob Texiere	Blade af Satans bog
28	Jalmari Isberg	Jäinen saari