TV Show / Movie Recommendation Database

Venkata Rohil Wardhan Kancharla[1], Preetam Sanjay Ozarde[2], Vyuha Kurapati[3]
Team VR Cypher
University at Buffalo
Id:vkanchar[1], Id:preetams[2], Id:vyuhakur [3]

***Abstract*—In the area of information filtering systems, delving into research on recommender systems is a critical topic and can be applied to various fields including media platforms. In this paper, we focus on a recommendation system for tv shows and movies. A database maintains all the data of the movies and tv series containing various details such as ratings, details and cast. Using sql queries, any user can look up a movie//tvshow based on their preference. A ranking administrator can also insert new entries into the database.**

***Index Terms*—Information filtering, recommender systems, database, sql**

I. Problem Statement

The database maintains the data of all the movies and tv series along with the ratings and details of those movies and tv shows. This gives us recommendations for the movies or shows according to our preferences.
The processing time using excel spreadsheets is higher when the volume of data is high in comparison to databases.
Excel allows us to make rows and columns to understand data while Databases stores this information in tables which makes it more well organised. We also have keys to form links between the tables.
Data can be accessed and modified by multiple users in databases while the user access can be restricted to specific users for security purposes.

II. Users

A. Target User - Audience

The target user is the tv show, movie audience which want to look up various tv series and movies.

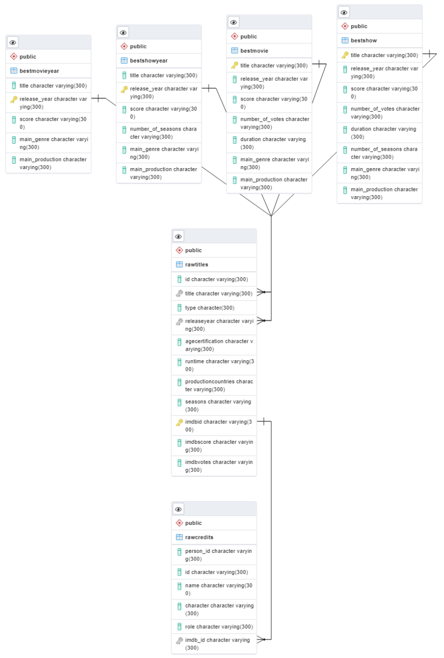
B. Administrator - Ranking Administrator

The ranking administrator will be allowed to make changes and modify the database. These changes include making new entries, deleting or updating existing ones etc.

C. Real-time scenario description

When a user is looking for a recently released horror or comedy movie with a rating above 8 on 10 starring Seth Rogan.

III. ER diagram


public
bestmovieyear
title character varying(300)
release_year character varying(300)
score character varying(300)
main_genre character varying(300)
main_production character varying(300)
public
bestshowyear
title character varying(300)
release_year character varying(300)
score character varying(300)
number_of_seasons character varying(300)
main_genre character varying(300)
main_production character varying(300)
public
bestmovie
title character varying(300)
release_year character varying(300)
score character varying(300)
number_of_votes character varying(300)
duration character varying (300)
main_genre character varying(300)
main_production character varying(300)
public
bestshow
title character varying(300)
release_year character varying(300)
score character varying(300)
number_of_votes character varying(300)
duration character varying (300)
number_of_seasons character varying(300)
main_genre character varying(300)
main_production character varying(300)
public
rawtitles
id character varying(300)
title character varying(300)
type character(300)
releaseyear character varying(300)
agecertification character varying(300)
runtime character varying(300)
productioncountries character varying(300)
seasons character varying (300)
imdbid character varying(300)
imdbscore character varying(300)
imdbvotes character varying(300)
public
rawcredits
person_id character varying(300)
id character varying(300)
name character varying(300)
character character varying (300)
role character varying(300)
imdb_id character varying (300)

Th ER diagram describes the structure of the database with the help of a diagram. It has been changed since Milestone 1 to make the database into BCNF form.

IV. Database implementation

A. Data schema

1) Relation 1: rawtitles (id (varchar), title (varchar), type (char), release_year(numeric), age_certificate(varchar), runtime (int), genres(varchar), production_countries(varchar), seasons(int), imdb_id(varchar), imdb_score(varchar), imdb_votes(numeric)) Contains data about all the movies and TV shows with many variables. It has 5,000 entities of data.

2) Relation 2: rawcredits(person_id (varchar), id (varchar), name(varchar), role (char)) This relation contains data on the cast and director involved in the TV series and Movies.

3) Relation 3: bestmovieyear (title (varchar), release_year(numeric), score(varchar), main_genre(char), main_production(char)) Table contains names of the highest ranked movie of that year from years 1954-2022

4) Relation 4: bestmovie (title(varchar), release_year(numeric), score(varchar), number_of_votes(numeric), duration(numeric), main_genre(char), main_production(char)) The table contains top rated movies by score achieved irrespective of the year.

5) Relation 5: bestshowyear (title (varchar), release_year(numeric), score(varchar), number_of_seasons(numeric), main_genre(char), main_production(char)) Table contains names of the highest ranked Tv series of that year from years 1954-2022.

6) Relation 6: Bestshow (title(varchar), release_year(numeric), score(varchar), number_of_votes(numeric), duration(numeric), number_of_seasons(numeric), main_genre(char), main_production(char)) The table contains top rated TV shows by score achieved irrespective of the year.

B. Attributes

1) Id: The id is a varchar datatype which maps all the movie and tv show titles. The id is a unique for every movie title.

2) Title: Title is the name of the movie. It is also unique for every entry.

3) Type: Type is whether the title associated with it is a movie or tv show

4) Release year: Release year is the year in which the movie or tv show was released. This is further used in the bestmovieyear and bestshowyear tables to show the best movie and tv show of that particular year.

5) Age certificate: Age certificate shows the titles censorship like R rated, PG-13 etc.

6) Genres: A particular movie or tv show is classified into different genres based on the reviewer.

7) Production Countries: The list of countries where the title is available.

8) Seasons: The number of seasons a tv show has run for. Seasons is only applicable to tv shows.

9) Imdb Id: Imdb id is unique for every title and it correlates with imdb score and imdb votes.

10) Imdb Score: Imdb score is the score obtained for every movie and tv show

11) Imdb votes: Imdb votes is the number of votes given by people for that particular title

12) Person Id: Person id is unique for every name since id is repeated values as there are many cast for one title.

13) Name: Name is the name of the cast or director for that title. It is mapped by id.

14) Role: Role is the name of the role they acted in for that particular title.

15) Score: Score is the imdb score for the title. It is ranked in descending order for the bestmovie and bestshow table.

16) Main Genre: The main genre is only one genre that matches the depiction of the movie or tv show rather that a cluster of genres for a single title.

17) Main production: The main production is same as main genre but instead of genre, the main country the title was produced is is displayed.

18) Number of votes: The number of votes is the total number of votes obtained for that title.

19) Duration: Duration is the max time of the movie or a single episode of a tv show.

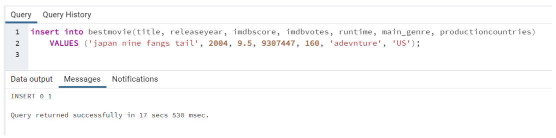
20) Number of seasons: Number of seasons are the total seasons for which the show was broadcasted.

V. Primary and Foreign Keys

1) *bestmovie:* Primary Key - title
2) *bestshow:* Primary Key - title
3) *bestmovieyear:* Primary Key - year
4) *bestshowyear:* Primary Key - year
5) *rawtitles:* Primary Key - imdbid Foreign Key - title,year
6) *Relation 6:* Primary Key - title Foreign Key - title is the foreign key for relation bestshow that references relation rawtitles

VI. Queries

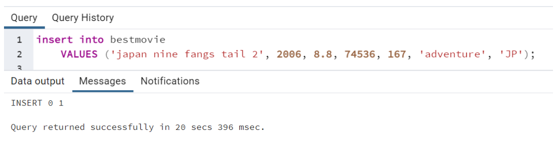
A. Insert Query 1



```
insert into bestmovie(title, releaseyear, imdbscore, imdbvotes, runtime, main_genre, productioncountries)
    VALUES ('japan nine fangs tail', 2004, 9.5, 9307447, 160, 'adevnture', 'US');
```

```
INSERT 0 1

Query returned successfully in 17 secs 530 msec.
```

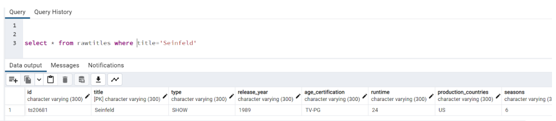
B. Insert Query 2



```
insert into bestmovie
    VALUES ('japan nine fangs tail 2', 2006, 8.8, 74536, 167, 'adventure', 'JP');
```

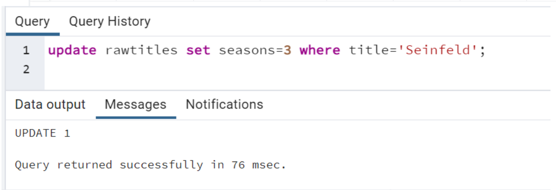
```
INSERT 0 1

Query returned successfully in 20 secs 396 msec.
```

C. Update Query 1



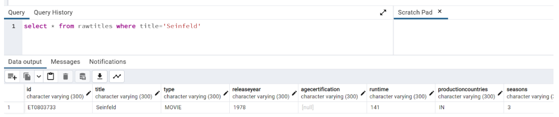
```
select * from rawtitles where title='Seinfeld'
```

	id character varying (300)	title [PK] character varying (300)	type character varying (300)	release_year character varying (300)	age_certification character varying (300)	runtime character varying (300)	production_countries character varying (300)	seasons character varying (3
1	ts20681	Seinfeld	SHOW	1989	TV-PG	24	US	6



```
update rawtitles set seasons=3 where title='Seinfeld';
```

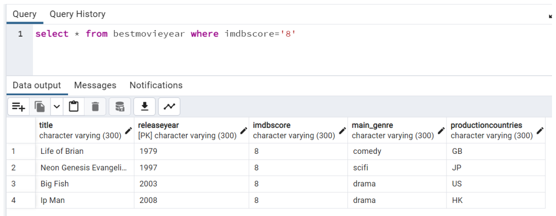
```
UPDATE 1

Query returned successfully in 76 msec.
```



```
select * from rawtitles where title='Seinfeld'
```

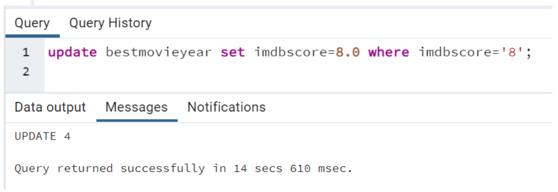
	id character varying (300)	title character varying (300)	type character varying (300)	releaseyear character varying (300)	agecertification character varying (300)	runtime character varying (300)	productioncountries character varying (300)	seasons character varying (
1	ET0803733	Seinfeld	MOVIE	1978	[null]	141	IN	3

D. Update Query 2



```
select * from bestmovieyear where imdbscore='8'
```

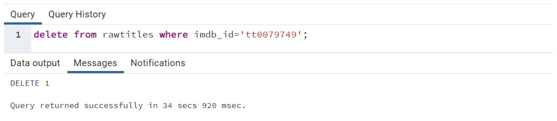
	title character varying (300)	releaseyear [PK] character varying (300)	imdbscore character varying (300)	main_genre character varying (300)	productioncountries character varying (300)
1	Life of Brian	1979	8	comedy	GB
2	Neon Genesis Evangeli...	1997	8	scifi	JP
3	Big Fish	2003	8	drama	US
4	Ip Man	2008	8	drama	HK



```
update bestmovieyear set imdbscore=8.0 where imdbscore='8';
```

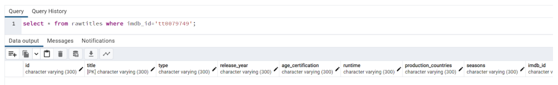
```
UPDATE 4

Query returned successfully in 14 secs 610 msec.
```

E. Delete Query 1



```sql
delete from rawtitles where imdb_id='tt0079749';
```

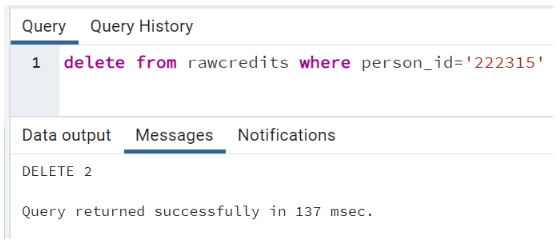
```
DELETE 1

Query returned successfully in 34 secs 920 msec.
```



```sql
select * from rawtitles where imdb_id='tt0079749';
```

	id character varying (300)	title [PK] character varying (300)	type character varying (300)	release_year character varying (300)	age_certification character varying (300)	runtime character varying (300)	production_countries character varying (300)	seasons character varying (300)	imdb_id character v

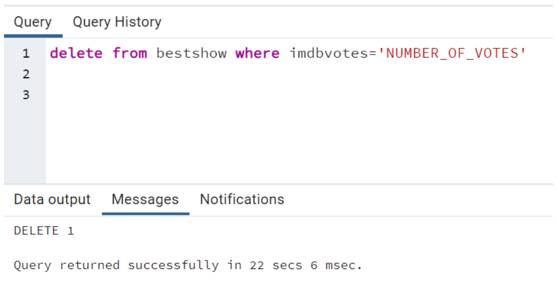
F. Delete Query 2



```sql
delete from rawcredits where person_id='222315'
```

```
DELETE 2

Query returned successfully in 137 msec.
```

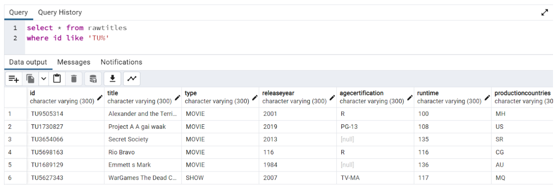
G. Delete Query 3



```sql
delete from bestshow where imdbvotes='NUMBER_OF_VOTES'
```

```
DELETE 1

Query returned successfully in 22 secs 6 msec.
```

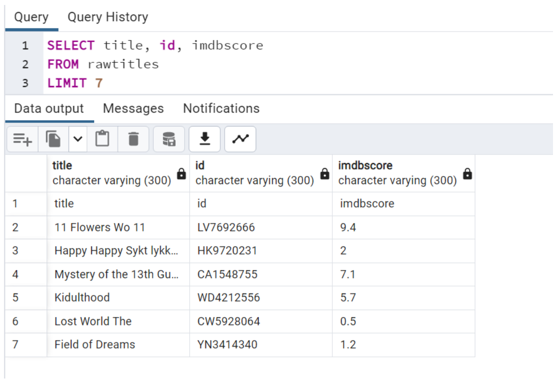
H. Like Query



```sql
select * from rawtitles
where id like 'TU%'
```

	id character varying (300)	title character varying (300)	type character varying (300)	releaseyear character varying (300)	agecertification character varying (300)	runtime character varying (300)	productioncountries character varying (30
1	TU9505314	Alexander and the Terri...	MOVIE	2001	R	100	MH
2	TU1730827	Project A A gai waak	MOVIE	2019	PG-13	108	US
3	TU3654066	Secret Society	MOVIE	2013	[null]	135	SR
4	TU5698163	Rio Bravo	MOVIE	116	R	116	CG
5	TU1689129	Emmett s Mark	MOVIE	1984	[null]	136	AU
6	TU5627343	WarGames The Dead C...	SHOW	2007	TV-MA	117	MQ

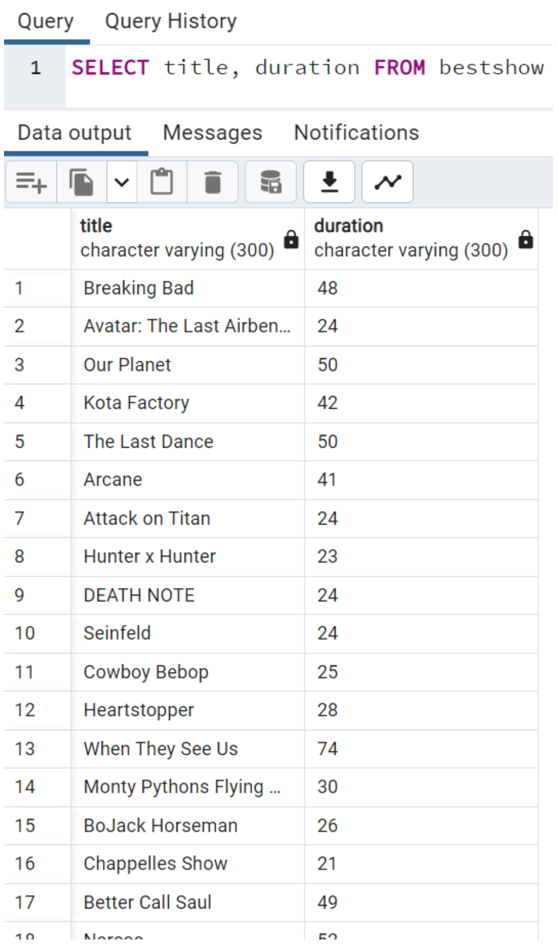
I. Limit Query



```sql
SELECT title, id, imdbscore
FROM rawtitles
LIMIT 7
```

	title character varying (300)	id character varying (300)	imdbscore character varying (300)
1	title	id	imdbscore
2	11 Flowers Wo 11	LV7692666	9.4
3	Happy Happy Sykt lykk...	HK9720231	2
4	Mystery of the 13th Gu...	CA1548755	7.1
5	Kidulthood	WD4212556	5.7
6	Lost World The	CW5928064	0.5
7	Field of Dreams	YN3414340	1.2

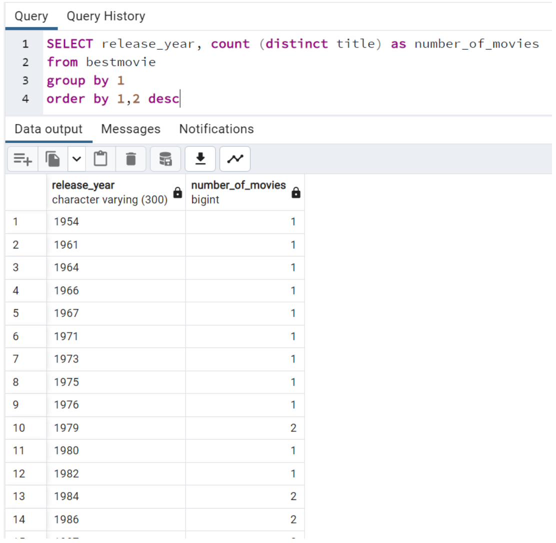
J. Select Clause



```sql
SELECT title, duration FROM bestshow
```

	title character varying (300)	duration character varying (300)
1	Breaking Bad	48
2	Avatar: The Last Airben...	24
3	Our Planet	50
4	Kota Factory	42
5	The Last Dance	50
6	Arcane	41
7	Attack on Titan	24
8	Hunter x Hunter	23
9	DEATH NOTE	24
10	Seinfeld	24
11	Cowboy Bebop	25
12	Heartstopper	28
13	When They See Us	74
14	Monty Pythons Flying ...	30
15	BoJack Horseman	26
16	Chappelles Show	21
17	Better Call Saul	49

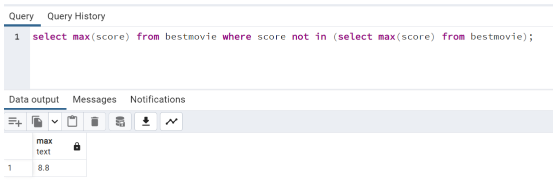
K. Group By



```sql
SELECT release_year, count (distinct title) as number_of_movies
from bestmovie
group by 1
order by 1,2 desc
```

	release_year character varying (300)	number_of_movies bigint
1	1954	1
2	1961	1
3	1964	1
4	1966	1
5	1967	1
6	1971	1
7	1973	1
8	1975	1
9	1976	1
10	1979	2
11	1980	1
12	1982	1
13	1984	2
14	1986	2

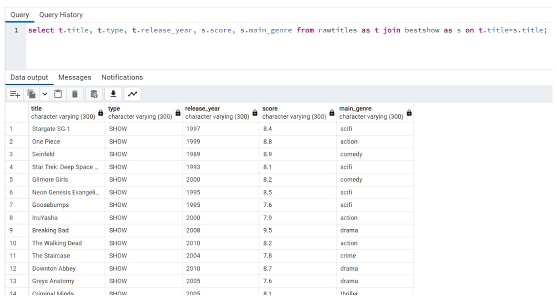
L. Sub Query



```sql
select max(score) from bestmovie where score not in (select max(score) from bestmovie);
```

	max text
1	8.8

M. Join Query



Query Query History

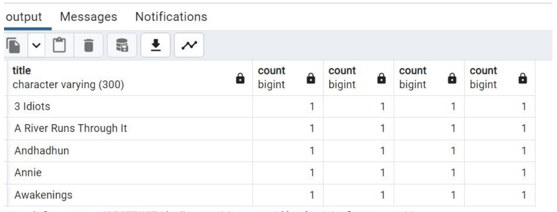
```sql
select t.title, t.type, t.release_year, s.score, s.main_genre from rawtitles as t join bestshow as s on t.title=s.title;
```

Data output Messages Notifications

	title character varying (300)	type character varying (300)	release_year character varying (300)	score character varying (300)	main_genre character varying (300)
1	Stargate SG-1	SHOW	1997	8.4	scifi
2	One Piece	SHOW	1999	8.8	action
3	Seinfeld	SHOW	1989	8.9	comedy
4	Star Trek: Deep Space ...	SHOW	1993	8.1	scifi
5	Gilmore Girls	SHOW	2000	8.2	comedy
6	Neon Genesis Evangeli...	SHOW	1995	8.5	scifi
7	Goosebumps	SHOW	1995	7.6	scifi
8	InuYasha	SHOW	2000	7.9	action
9	Breaking Bad	SHOW	2008	9.5	drama
10	The Walking Dead	SHOW	2010	8.2	action
11	The Staircase	SHOW	2004	7.8	crime
12	Downton Abbey	SHOW	2010	8.7	drama
13	Greys Anatomy	SHOW	2005	7.6	drama
14	Criminal Minds	SHOW	2005	8.1	thriller

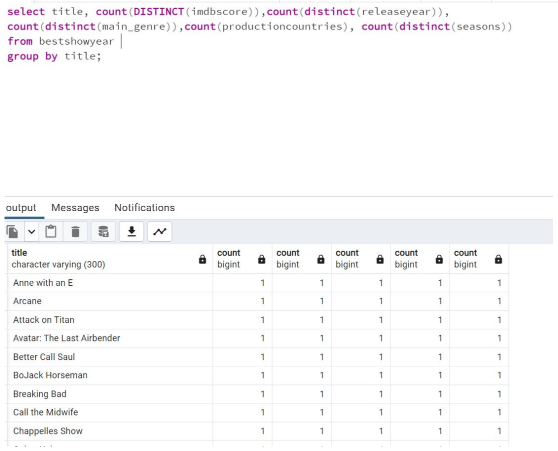
N. Complex Queries

```sql
select title, count(DISTINCT(imdbscore)),count(distinct(releaseyear)),
count(distinct(main_genre)),count(productioncountries)
from bestmovieyear
group by title;
```



output Messages Notifications

title character varying (300)	count bigint	count bigint	count bigint	count bigint
3 Idiots	1	1	1	1
A River Runs Through It	1	1	1	1
Andhadhun	1	1	1	1
Annie	1	1	1	1
Awakenings	1	1	1	1

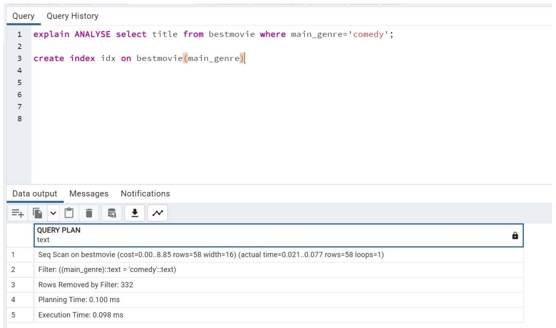


```sql
select title, count(DISTINCT(imdbscore)),count(distinct(releaseyear)),
count(distinct(main_genre)),count(productioncountries), count(distinct(seasons))
from bestshowyear
group by title;
```

output Messages Notifications

title character varying (300)	count bigint	count bigint	count bigint	count bigint	count bigint
Anne with an E	1	1	1	1	1
Arcane	1	1	1	1	1
Attack on Titan	1	1	1	1	1
Avatar: The Last Airbender	1	1	1	1	1
Better Call Saul	1	1	1	1	1
BoJack Horseman	1	1	1	1	1
Breaking Bad	1	1	1	1	1
Call the Midwife	1	1	1	1	1
Chappelles Show	1	1	1	1	1

VII. Query Optimization

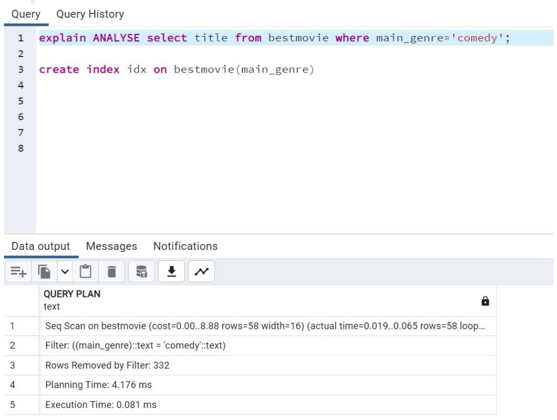


Query Query History

```sql
explain ANALYSE select title from bestmovie where main_genre='comedy';

create index idx on bestmovie(main_genre)
```

Data output Messages Notifications

```
QUERY PLAN
text
1  Seq Scan on bestmovie (cost=0.00..8.85 rows=58 width=16) (actual time=0.021..0.077 rows=58 loops=1)
2  Filter: ((main_genre)::text = 'comedy'::text)
3  Rows Removed by Filter: 332
4  Planning Time: 0.100 ms
5  Execution Time: 0.098 ms
```



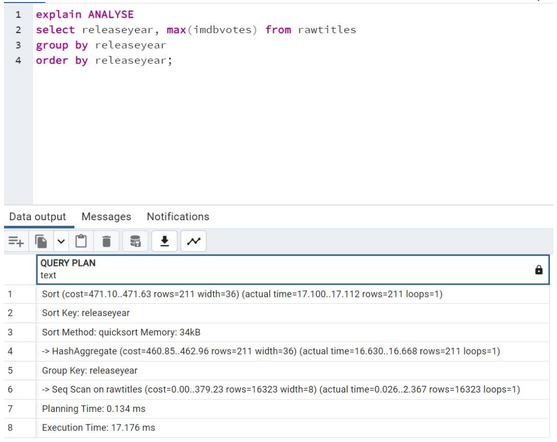
Query Query History

```sql
explain ANALYSE select title from bestmovie where main_genre='comedy';

create index idx on bestmovie(main_genre)
```

Data output Messages Notifications

```
QUERY PLAN
text
1  Seq Scan on bestmovie (cost=0.00..8.88 rows=58 width=16) (actual time=0.019..0.065 rows=58 loop...
2  Filter: ((main_genre)::text = 'comedy'::text)
3  Rows Removed by Filter: 332
4  Planning Time: 4.176 ms
5  Execution Time: 0.081 ms
```

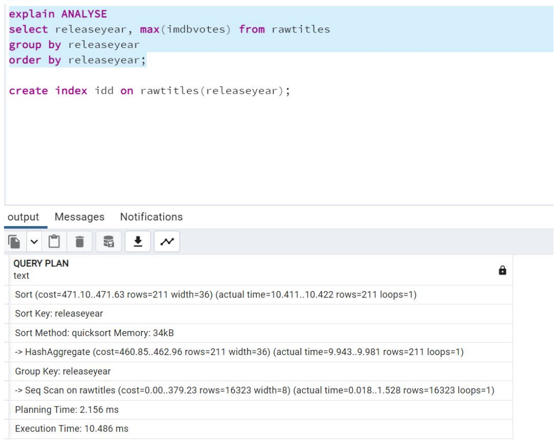
Index created on bestmovie(main_genre)



```sql
explain ANALYSE
select releaseyear, max(imdbvotes) from rawtitles
group by releaseyear
order by releaseyear;
```

Data output Messages Notifications

```
QUERY PLAN
text
1  Sort (cost=471.10..471.63 rows=211 width=36) (actual time=17.100..17.112 rows=211 loops=1)
2  Sort Key: releaseyear
3  Sort Method: quicksort Memory: 34kB
4  -> HashAggregate (cost=460.85..462.96 rows=211 width=36) (actual time=16.630..16.668 rows=211 loops=1)
5  Group Key: releaseyear
6  -> Seq Scan on rawtitles (cost=0.00..379.23 rows=16323 width=8) (actual time=0.026..2.367 rows=16323 loops=1)
7  Planning Time: 0.134 ms
8  Execution Time: 17.176 ms
```

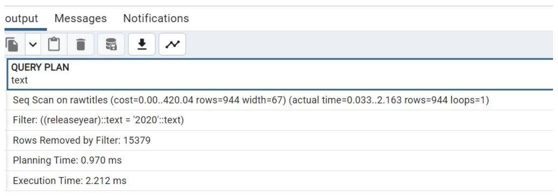


```sql
explain ANALYSE
select releaseyear, max(imdbvotes) from rawtitles
group by releaseyear
order by releaseyear;

create index idd on rawtitles(releaseyear);
```

output Messages Notifications

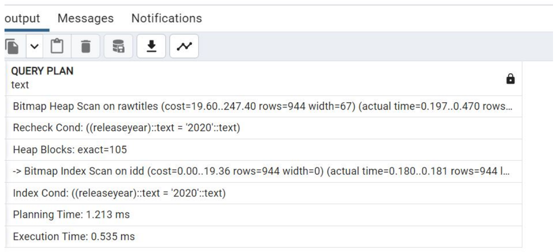
```
QUERY PLAN
text
Sort (cost=471.10..471.63 rows=211 width=36) (actual time=10.411..10.422 rows=211 loops=1)
Sort Key: releaseyear
Sort Method: quicksort Memory: 34kB
-> HashAggregate (cost=460.85..462.96 rows=211 width=36) (actual time=9.943..9.981 rows=211 loops=1)
Group Key: releaseyear
-> Seq Scan on rawtitles (cost=0.00..379.23 rows=16323 width=8) (actual time=0.018..1.528 rows=16323 loops=1)
Planning Time: 2.156 ms
Execution Time: 10.486 ms
```

```
explain ANALYSE select * from rawtitles where releaseyear='2020';
```


output Messages Notifications

```
QUERY PLAN
text
Seq Scan on rawtitles (cost=0.00..420.04 rows=944 width=67) (actual time=0.033..2.163 rows=944 loops=1)
Filter: ((releaseyear)::text = '2020'::text)
Rows Removed by Filter: 15379
Planning Time: 0.970 ms
Execution Time: 2.212 ms
```

```
create index idd on rawtitles(releaseyear);

explain ANALYSE select * from rawtitles where releaseyear='2020';
```

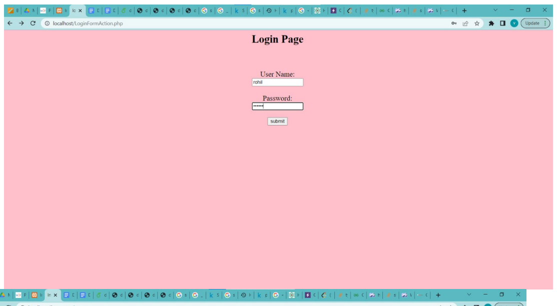

output Messages Notifications

```
QUERY PLAN
text
Bitmap Heap Scan on rawtitles (cost=19.60..247.40 rows=944 width=67) (actual time=0.197..0.470 rows...
Recheck Cond: ((releaseyear)::text = '2020'::text)
Heap Blocks: exact=105
-> Bitmap Index Scan on idd (cost=0.00..19.36 rows=944 width=0) (actual time=0.180..0.181 rows=944 l...
Index Cond: ((releaseyear)::text = '2020'::text)
Planning Time: 1.213 ms
Execution Time: 0.535 ms
```

Index created on rawtitles(releaseyear)

We have created indexes to better handle complex queries and reduce the runtime as shown in the figures above.

VIII. Running Website

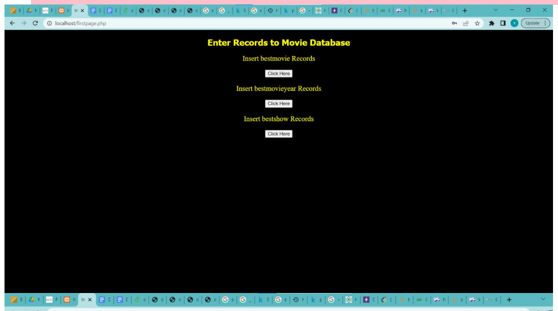

localhost/LoginFormAction.php

Login Page

User Name: rohit

Password:

submit


localhost/firstpage.php

Enter Records to Movie Database

Insert bestmovie Records

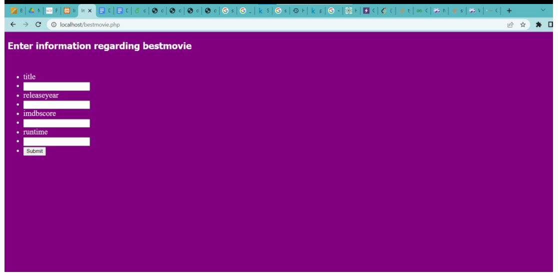
Click Here

Insert bestmovieyear Records

Click Here

Insert bestshow Records

Click Here


localhost/bestmovie.php

Enter information regarding bestmovie

- title
- releaseyear
- imdbscore
- runtime
- Submit

IX. Contribution

TABLE I
Contribution of Team Members

Team Members	Contribution
Venkata Rohil Wardhan Kancharla	Writing of simple and complex queries, designing of UI, contributed to the report.
Preetam Sanjay Ozarde	Designing the ER Diagram based on the dataset, created the final report
Vyuha Kurapati	Finding the dataset, loading the dataset into PG admin and minimal preprocessing of the data, contributed to the report.