DBMS Lab Assignment 5

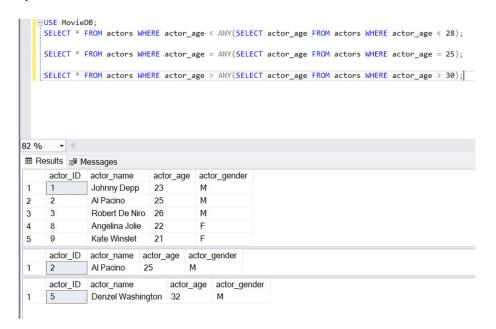
Name: B S Sathwik Goud

Reg.No: 19BCS015

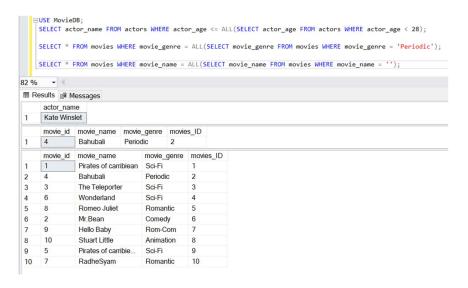
Q1) Illustrate logical ANY, ALL and LIKE operator- the queries should be relevant to your

respective databases 3 queries for each operator. One query explaining the difference between ANY and ALL.

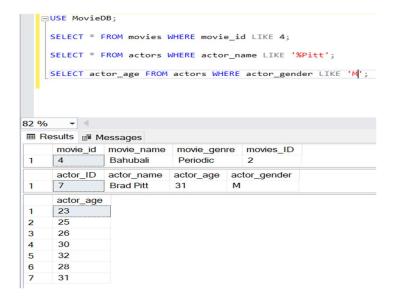
QUERIES FOR "ANY"



QUERIES FOR "ALL"



QUERIES FOR "LIKE"



Q2) One query for each Aggregate function.

The aggregate functions are MIN(), MAX(), COUNT(), AVG(), SUM()

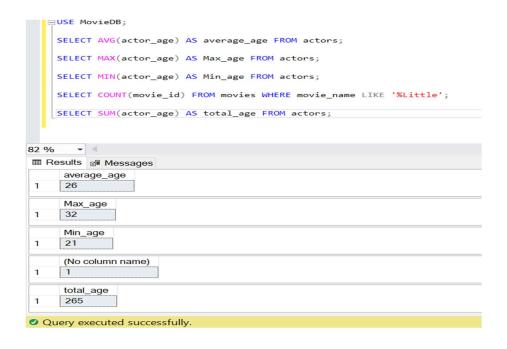
AVG() – return the average of the set

MIN() – returns the minimum value in a set

MAX() – returns the maximum value in set

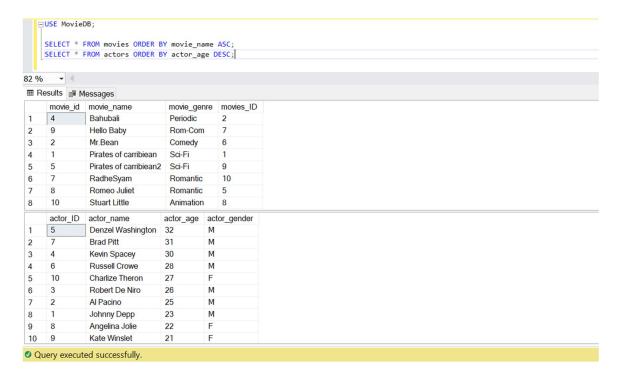
SUM() – returns the sum of all distinct values of a set

COUNT() – returns the number of items in a set

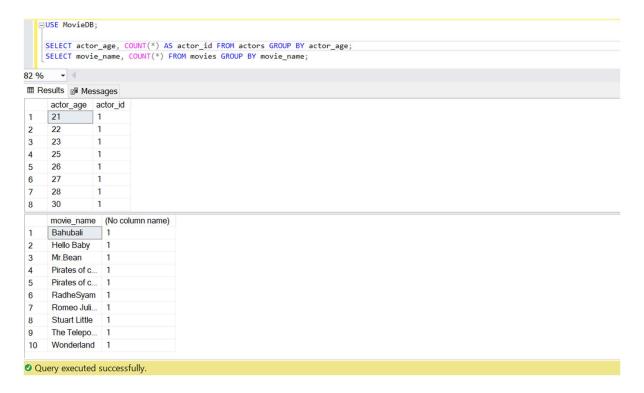


Q3) Illustrate the usage of order by, group by and having clause (2 queries for each case)

ORDER BY



GROUP BY



HAVING CLAUSE

```
SELECT COUNT(movies_ID), movie_name FROM movies GROUP BY movie_name HAVING COUNT(movies_ID) = 2;

82 %

Results Messages

(No column name) movie_name
```

Q4) Use Aggregate function with group by and having

AVG():



COUNT():



MIN():

```
SELECT MIN(actor_age) FROM actors GROUP BY actor_gender HAVING actor_gender = 'F';

82 % 

■ Results M Messages

(No column name)

1 21
```

MAX():

```
SELECT MAX(actor_age) FROM actors GROUP BY actor_gender HAVING actor_gender = 'M';

82 % 

Results M Messages

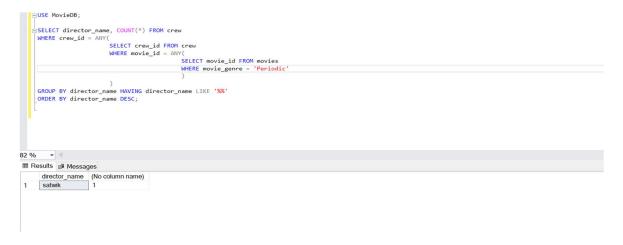
(No column name)

1 32
```

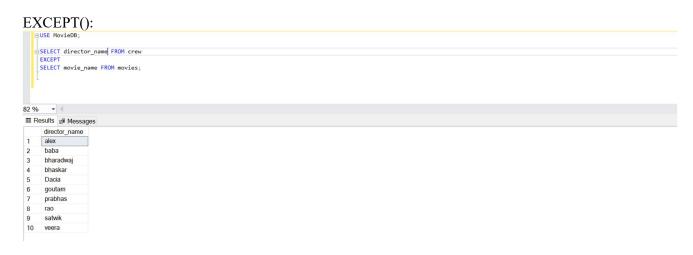
SUM():

Q5) Write at least 3 nested queries using order by, group by and having clause.

QUERY:



Q6) Illustrate the Usage of Except, Exists, Not Exists, Union, Intersection



EXISTS():



NOT EXISTS():

```
BUSE MovieDB;

SELECT * FROM crew | WHERE NOT EXISTS | (SELECT movie_id FROM movies);

B2 % 

Results @ Messages | crew_id | movie_id | director_name | producer_name | musicdirector_name |
```

UNION():

INTERSECT():

```
SELECT movie_id FROM crew
INTERSECT
SELECT movie_id FROM movies;

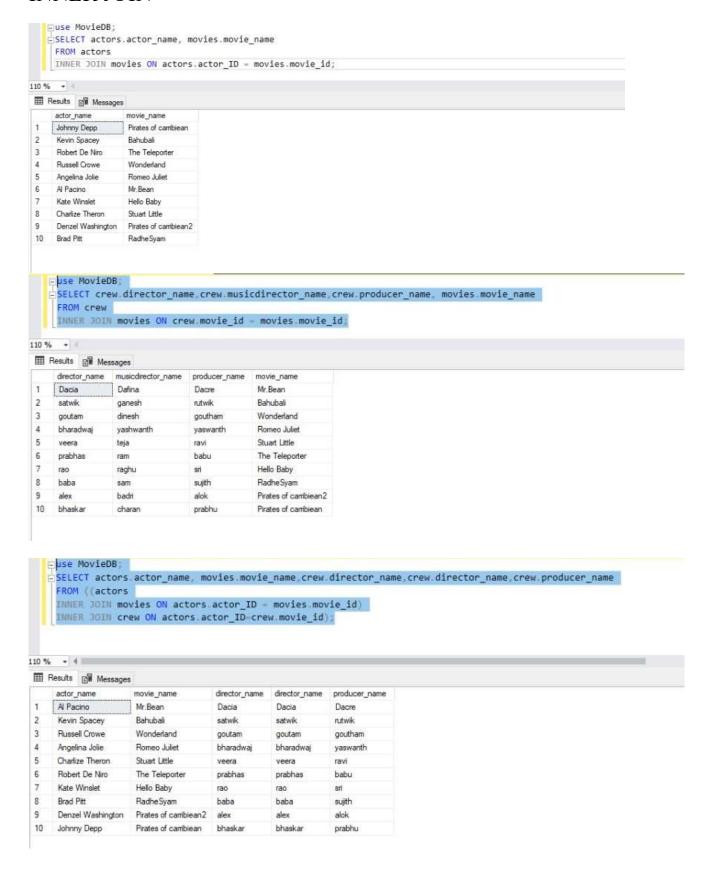
82 %

Results ps Messages

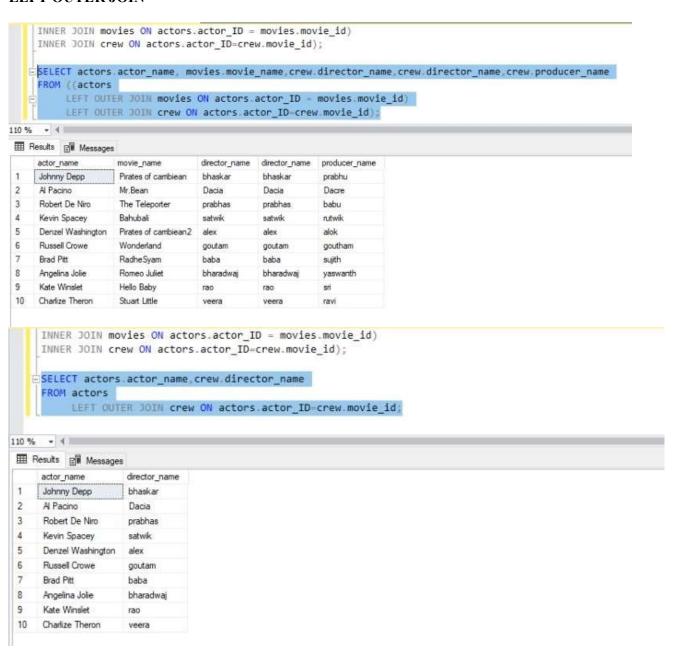
move_id
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 8
9 9
9 10
```

Q7) INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN- 3 queries for each instance

INNER JOIN



LEFT OUTER JOIN



RIGHT OUTER JOIN

INNER JOIN movies ON actors.actor_ID = movies.movie_id)
INNER JOIN crew ON actors.actor_ID=crew.movie_id);

DESELECT *
FROM actors
RIGHT OUTER JOIN movies ON actors.actor_ID = movies.movie_id;

actor_ID actor_name actor age actor gender movie id movie name movie genre movies ID Johnny Depp 23 M Pirates of cambiean Sci-Fi 2 Kevin Spacey 30 Bahubali Periodic 3 3 Robert De Niro 26 M 3 The Teleporter Sci-Fi 3 Russell Crowe 28 M Wonderland Sci-Fi 5 22 F 8 Romeo Juliet Romantic 8 Angelina Jolie 6 2 Al Pacino 25 M 2 Mr Rean Comedy 6 Kate Winslet 21 Helio Baby Rom-Com 8 F 10 Charlize Theron 27 10 Stuart Little Animation 2 9 5 Denzel Washington 32 M 5 Pirates of carribiean2 Sci-Fi 9 10 10 Brad Pitt 31 Radhe Syam Romantic

Results Messages actor_name movie_name director_name director_name producer_name Al Pacino Mr. Bean Dacia Dacia Dacre Kevin Spacey Bahubali satwik satwik rutwik 3 Russell Crowe Wonderland goutham goutam goutam Romeo Juliet 4 Angelina Jolie bharadwaj bharadwaj yaswanth 5 Charlize Theron Stuart Little veera veera ravi 6 Robert De Niro The Teleporter prabhas prabhas babu 7 Kate Winslet Hello Baby rao rao 8 Brad Pitt Radhe Syam baba baba sujith Denzel Washington Pirates of carribiean 2 alex alex 9 alok Pirates of cambiean bhaskar bhaskar prabhu Johnny Depp

INNER JOIN movies ON actors.actor_ID = movies.movie_id)
INNER JOIN crew ON actors.actor_ID=crew.movie_id);

SELECT actors actor_name,crew.director_name FROM (actors

RIGHT OUTER JOIN crew ON actors.actor_ID=crew.movie_id);

