In [4]:
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import sqlite3
import csv
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from wordcloud import WordCloud
import re
import os
from sqlalchemy import create_engine # database connection
import datetime as dt
from sklearn import metrics
from sklearn.metrics import f1_score,precision_score,recall_score
from datetime import datetime
```

In [5]:
```
conn = sqlite3.connect(r'D:\AppliedAI\Homework-n-Assignments\# 22 SQL Assig
nment on IMDB data\Db-IMDB.db')
movie_table = pd.read_sql_query('SELECT * FROM Movie ', conn)
```

In [6]:
```
print (movie_table)
```

```
      index        MID           title  year  rating  num_votes
0         0  tt2388771          Mowgli  2018     6.6      21967
1         1  tt5164214  Ocean's Eight  2018     6.2     110861
2         2  tt1365519    Tomb Raider  2018     6.4     142585
3         3  tt0848228   The Avengers  2012     8.1    1137529
4         4  tt8239946         Tumbbad  2018     8.5       7483
...     ...        ...             ...   ...     ...        ...
3470   3470  tt0090611    Allah-Rakha  1986     6.2         96
3471   3471  tt0106270          Anari  1993     4.7        301
3472   3472  tt0852989  Come December  2006     5.7         57
3473   3473  tt0375882     Kala Jigar  1939     3.3        174
3474   3474  tt0375890         Kanoon  1994     3.2        103

[3475 rows x 6 columns]
```

1. List all the directors who directed a 'Comedy' movie in a leap year. (You need to check that the genre is 'Comedy' and year is a leap year) Your query should return director name, the movie name, and the year.

In [4]:
```
cmd1=pd.read_sql_query("SELECT p.Name as 'Director Name', m.title as 'Movie
s Name', m.year as Year from Person p JOIN M_Director d on p.PID=d.PID JOIN
Movie m on d.MID=m.MID JOIN M_Genre as mg on mg.MID=m.MID JOIN Genre g on
 g.GID=mg.GID WHERE m.year %4==0 and trim(g.name) LIKE '%Comedy%'", conn)
```

In [5]:
```
cmd1
```

Out[5]:

	Director Name	Movies Name	Year
0	Milap Zaveri	Mastizaade	2016
1	Milap Zaveri	Mastizaade	2016
2	Danny Leiner	Harold & Kumar Go to White Castle	2004
3	Danny Leiner	Harold & Kumar Go to White Castle	2004
4	Anurag Kashyap	Gangs of Wasseypur	2012
...	...	...	...
410	Siddharth Anand Kumar	Let's Enjoy	2004
411	Amma Rajasekhar	Sathyam	2008
412	Oliver Paulus	Tandoori Love	2008
413	Raja Chanda	Le Halua Le	2012
414	K.S. Prakash Rao	Raja Aur Rangeeli	1996

415 rows × 3 columns

2. List the names of all the actors who played in the movie 'Anand' (1971)

In [6]:
```
cmd2 = pd.read_sql_query("SELECT p.Name as 'Actor Name',m.title as 'Movie N
ame' FROM Person p JOIN M_Cast mc ON trim(p.PID)=trim(mc.PID) JOIN Movie M
 ON trim(mc.MID)=trim(m.MID) WHERE trim(m.title) = 'Anand' ", conn)
```

In [7]:
```
cmd2
```

Out[7]:

	Actor Name	Movie Name
0	Amitabh Bachchan	Anand
1	Rajesh Khanna	Anand
2	Sumita Sanyal	Anand
3	Ramesh Deo	Anand
4	Seema Deo	Anand
5	Asit Kumar Sen	Anand
6	Dev Kishan	Anand
7	Atam Prakash	Anand
8	Lalita Kumari	Anand
9	Savita	Anand
10	Brahm Bhardwaj	Anand
11	Gurnam Singh	Anand
12	Lalita Pawar	Anand
13	Durga Khote	Anand
14	Dara Singh	Anand
15	Johnny Walker	Anand
16	Moolchand	Anand

3. List all the actors who acted in a film before 1970 and in a film after 1990. (That is: < 1970 and > 1990.)

In [40]:
```
cmd3 = pd.read_sql_query("SELECT p.Name as 'Actor',m.title as 'Movie',m.yea
r AS Year FROM Person p JOIN M_Cast mc ON trim(p.PID)=trim(mc.PID) JOIN Mov
ie m ON trim(m.MID)=trim(mc.MID)",conn)
```

In [41]:
```
cmd3
```

Out[41]:

	Actor	Movie	Year
0	Christian Bale	Mowgli	2018
1	Cate Blanchett	Mowgli	2018
2	Cate Blanchett	Ocean's Eight	2018
3	Benedict Cumberbatch	Mowgli	2018
4	Naomie Harris	Mowgli	2018
...	...	...	...
86845	Abbas	Man on Mission Taqatwar	2005
86846	Gulshan Kumar	Dance Dance	1987
86847	Gulshan Kumar	Naseeb Apna Apna	1986
86848	Iqbal	Kala Jigar	1939
86849	Sushma Shiromani	Talash	1969

86850 rows × 3 columns

In [44]:
```
cmd3.to_sql("Actor-Movie-Year", conn, if_exists="replace")
```

In [45]:
```
cmd3_after_1990 = pd.read_sql_query("SELECT a.Actor ,a.Movie ,a.Year FROM
 'Actor-Movie-Year' a WHERE a.Year > 1990  ",conn)
```

In [47]:
```
cmd3_before_1970 = pd.read_sql_query("SELECT a.Actor ,a.Movie ,a.Year FROM
 'Actor-Movie-Year' a WHERE a.Year < 1970  ",conn)
```

In [48]:
```
cmd3_after_1990
```

Out[48]:

	Actor	Movie	Year
0	Christian Bale	Mowgli	2018
1	Cate Blanchett	Mowgli	2018
2	Cate Blanchett	Ocean's Eight	2018
3	Benedict Cumberbatch	Mowgli	2018
4	Naomie Harris	Mowgli	2018
...	...	...	...
65640	Srinivas Sunderrajan	The Untitled Kartik Krishnan Project	2010
65641	Abbas	Hey Ram	2000
65642	Abbas	Kadhal Desam	1996
65643	Abbas	Woh Lamhe	2006
65644	Abbas	Man on Mission Taqatwar	2005

65645 rows × 3 columns

In [55]:
```
s1=cmd3_after_1990[['Actor']]
s1
```

Out[55]:

	Actor
0	Christian Bale
1	Cate Blanchett
2	Cate Blanchett
3	Benedict Cumberbatch
4	Naomie Harris
...	...
65640	Srinivas Sunderrajan
65641	Abbas
65642	Abbas
65643	Abbas
65644	Abbas

65645 rows × 1 columns

In [49]:
```
cmd3_before_1970
```

Out[49]:

	Actor	Movie	Year
0	Rishi Kapoor	Shree 420	1955
1	Amitabh Bachchan	Saat Hindustani	1969
2	Asrani	Satyakam	1969
3	Zohra Sehgal	The Long Duel	1967
4	Zohra Sehgal	Neecha Nagar	1946
...	...	...	...
5123	Manmohan Krishna	Pardesi	1957
5124	Manmohan Krishna	Hamraaz	1967
5125	Manmohan Krishna	Izzat	1968
5126	Iqbal	Kala Jigar	1939
5127	Sushma Shiromani	Talash	1969

5128 rows × 3 columns

In [52]:
```
s2=cmd3_before_1970[['Actor']]
s2
```

Out[52]:

	Actor
0	Rishi Kapoor
1	Amitabh Bachchan
2	Asrani
3	Zohra Sehgal
4	Zohra Sehgal
...	...
5123	Manmohan Krishna
5124	Manmohan Krishna
5125	Manmohan Krishna
5126	Iqbal
5127	Sushma Shiromani

5128 rows × 1 columns

In [56]:
```
print (type(s1))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [57]:
```
s3 = pd.merge(s1, s2, how='inner', on=['Actor'])
```

In [58]:
```
s3
```

Out[58]:

	Actor
0	Rishi Kapoor
1	Rishi Kapoor
2	Rishi Kapoor
3	Rishi Kapoor
4	Rishi Kapoor
...	...
14806	Asrani
14807	Asrani
14808	Asrani
14809	Asrani
14810	Asrani

14811 rows × 1 columns

In [59]:
```
s3.to_sql("Actor-before-1970-after-1990", conn, if_exists="replace")
```

In [61]:
```
cmd3 = pd.read_sql_query("SELECT DISTINCT Actor FROM 'Actor-before-1970-aft
er-1990'",conn)
```

Below is table for all actors who acted before 1970 and after 1990

In [62]:
```
cmd3
```

Out[62]:

	Actor
0	Rishi Kapoor
1	Rajesh Kumar
2	Anand Tiwari
3	Amitabh Bachchan
4	Asrani
...	...
468	Vinod Mehra
469	Deven Verma
470	Master Bhagwan
471	Rishi Kapoor
472	Asrani

473 rows × 1 columns

4. List all directors who directed 10 movies or more, in descending order of the number of movies they directed. Return the directors' names and the number of movies each of them directed.

In [ ]:
```
#cmd4 = pd.read_sql_query("SELECT p.Name as 'Director Name',COUNT(md.MID)
 as 'Number' FROM Person p JOIN M_Director md ON trim(p.PID)=trim(md.PID) W
HERE Number > 10  ORDER BY Number DESC", conn)

#cmd5 = pd.read_sql_query("SELECT p.Name as 'Director Name',COUNT(md.MID)
 as 'Number' FROM Person p JOIN M_Director md ON trim(p.PID)=trim(md.PID)
 HAVING Number > 10  GROUP BY Number ORDER BY Number DESC", conn)
```

In [72]:
```
cmd5_1 = pd.read_sql_query("SELECT p.Name as 'Director',(md.MID) as 'Movie'
FROM Person p JOIN M_Director md ON trim(p.PID)=trim(md.PID) ORDER BY 'Dire
ctor' ASC" ,conn)
```

In [73]:
```
cmd5_1
```

Out[73]:

	Director	Movie
0	Andy Serkis	tt2388771
1	Griffin Dunne	tt0809504
2	Rishi Kapoor	tt0149568
3	Saurabh Shukla	tt1340778
4	Saurabh Shukla	tt1772332
...	...	...
5778	Kannan	tt0318607
5779	Adrian Fulle	tt0432288
5780	Gulshan Kumar	tt0439464
5781	Iqbal	tt0375882
5782	Sushma Shiromani	tt0375890

5783 rows × 2 columns

In [74]:
```
cmd5_1.to_sql("Director_Movie", conn, if_exists="replace")
```

In [86]:
```
cmd5_2 = pd.read_sql_query("SELECT DISTINCT d.Director ,COUNT(d.Director) A
S MovieCount FROM Director_Movie d GROUP BY d.Director" ,conn)
```

In [87]:
```
cmd5_2
```

Out[87]:

	Director	MovieCount
0	A. Bhimsingh	7
1	A. Muthu	1
2	A.M.R. Ramesh	1
3	A.R. Murugadoss	6
4	Aamir Bashir	1
...	...	...
2167	Zaigham Imam	1
2168	Zeishan Quadri	1
2169	Zoya Akhtar	5
2170	Zubair Khan	1
2171	Zunaid Memon	1

2172 rows × 2 columns

In [88]:
```
cmd5_2.to_sql("Director_MovieCount", conn, if_exists="replace")
```

In [94]:
```
cmd5_3 = pd.read_sql_query("SELECT DISTINCT d.Director,d.MovieCount FROM Di
rector_MovieCount d WHERE MovieCount > 9 ORDER BY MovieCount DESC" ,conn)
```

In [95]:
```
cmd5_3
```

Out[95]:

	Director	MovieCount
0	David Dhawan	39
1	David Dhawan	39
2	Mahesh Bhatt	36
3	Mahesh Bhatt	36
4	Ram Gopal Varma	30
...	...	...
106	Pankaj Parashar	10
107	Raj Kapoor	10
108	Sudhir Mishra	10
109	Tigmanshu Dhulia	10
110	Vishal Bhardwaj	10

111 rows × 2 columns

5.a. For each year, count the number of movies in that year that had only female actors.

b. Now include a small change: report for each year the percentage of movies in that year with only female actors, and the total number of movies made that year. For example, one answer will be: 1990 31.81 13522 meaning that in 1990 there were 13,522 movies, and 31.81% had only female actors. You do not need to round your answer

In [10]:
```
cmd5 = pd.read_sql_query("SELECT DISTINCT p.Gender FROM Person p ", conn)
```

In [11]:
```
cmd5
```

Out[11]:

	Gender
0	Male
1	Female
2	None

In [110]:
```
cmd5a_female = pd.read_sql_query("SELECT DISTINCT m.Year as Year,(mc.MID),
m.title FROM Movie m JOIN M_Cast mc \
                            ON trim(m.MID)=trim(mc.MID) JOIN Person p ON trim
(mc.PID)=trim(p.PID) WHERE p.Gender == 'Female' \
                             ORDER BY Year DESC", conn)
```

Table of movies which had either ONLY female actors or female and male actors both

In [111]:
```
cmd5a_female
```

Out[111]:

	Year	MID	title
0	XVII 2016	tt6206564	Trapped
1	VI 2015	tt4271730	Alone
2	V 2015	tt4467202	Hero
3	IV 2017	tt7399620	Game Over
4	IV 2011	tt1702543	Lucky
...	...	...	...
3445	1939	tt0375882	Kala Jigar
3446	1936	tt0028217	Sant Tukaram
3447	1936	tt0027256	Achhut Kanya
3448	1936	tt0026274	Devdas
3449	1931	tt0021594	Alam Ara

3450 rows × 3 columns

In [112]:
```
cmd5a_male = pd.read_sql_query("SELECT DISTINCT m.Year as Year,(mc.MID),m.t
itle FROM Movie m JOIN M_Cast mc \
                            ON trim(m.MID)=trim(mc.MID) JOIN Person p ON trim
(mc.PID)=trim(p.PID) WHERE p.Gender == 'Male' \
                             ORDER BY Year DESC", conn)
```

Table of movies which had either ONLY male actors or female and male actors both

In [113]:
```
cmd5a_male
```

Out[113]:

	Year	MID	title
0	XVII 2016	tt6206564	Trapped
1	VI 2015	tt4271730	Alone
2	V 2015	tt4467202	Hero
3	IV 2017	tt7399620	Game Over
4	IV 2011	tt1702543	Lucky
...	...	...	...
3463	1939	tt0031580	The Little Princess
3464	1936	tt0028217	Sant Tukaram
3465	1936	tt0027256	Achhut Kanya
3466	1936	tt0026274	Devdas
3467	1931	tt0021594	Alam Ara

3468 rows × 3 columns

In [116]:
```
cmd5a_male.to_sql("Male", conn, if_exists="replace")
```

In [117]:
```
cmd5a_female.to_sql("Female", conn, if_exists="replace")
```

In [119]:
```
cmd5a_Female_WithoutMale = pd.read_sql_query("SELECT f.year,f.MID,f.title F
ROM Female f EXCEPT SELECT m.year,m.MID,m.title FROM MALE m",conn)
```

Table of movies which had ONLY female actors

In [120]:
```
cmd5a_Female_WithoutMale
```

Out[120]:

	Year	MID	title
0	1939	tt0375882	Kala Jigar
1	1999	tt0272001	Bindhaast
2	2000	tt0354922	Snegithiye
3	I 2018	tt8458202	Pihu

In [ ]:
```
cmd5b = pd.read_sql_query("  ", conn)
```

6. Find the film(s) with the largest cast. Return the movie title and the size of the cast. By "cast size" we mean the number of distinct actors that played in that movie: if an actor played multiple roles, or if it simply occurs multiple times in casts, we still count her/him only once.

In [133]:
```
cmd6 = pd.read_sql_query("SELECT m.title AS 'Title', (mc.PID) FROM Movie m
 JOIN M_Cast mc ON trim(mc.MID)=trim(m.MID) ORDER BY 'Title' DESC", conn)


#cmd6_1 = pd.read_sql_query("SELECT m.title AS 'Movie Title', MAX(c) as 'Ca
st Count' FROM  Movie m JOIN M_Cast mc ON trim(mc.MID)=trim(m.MID) HAVING C
OUNT(mc.PID) AS c)", conn)
```

In [134]:
```
cmd6
```

Out[134]:

	Title	PID
0	Mowgli	nm0000288
1	Mowgli	nm0000949
2	Mowgli	nm1212722
3	Mowgli	nm0365140
4	Mowgli	nm0785227
...	...	...
82832	Kanoon	nm0664109
82833	Kanoon	nm0505323
82834	Kanoon	nm0019427
82835	Kanoon	nm0197582
82836	Kanoon	nm0438467

82837 rows × 2 columns

In [135]:
```
cmd6.to_sql("Title_PID", conn, if_exists="replace")
```

In [148]:
```
cmd6_1 = pd.read_sql_query("SELECT Title , COUNT(Title) as 'CastSize' FROM
 Title_PID  GROUP BY Title ", conn)
```

In [149]:
```
cmd6_1
```

Out[149]:

	Title	CastSize
0	'D'	32
1	'Kaash'	22
2	...Aur Pyaar Ho Gaya	21
3	...Yahaan	14
4	100 Days	19
...	...	...
3339	Zor: Never Underestimate the Force	28
3340	Zubaan	17
3341	Zubeidaa	24
3342	Zulm Ki Hukumat	24
3343	Zulmi	20

3344 rows × 2 columns

In [150]:
```
cmd6_1.to_sql("Title_CastSize", conn, if_exists="replace")
```

In [151]:
```
cmd6_2 = pd.read_sql_query("SELECT Title , CastSize FROM Title_CastSize  ORDER BY CastSize DESC", conn)
```

Movies with Largest crew is descending order

In [152]:
```
cmd6_2
```

Out[152]:

	Title	CastSize
0	Ocean's Eight	238
1	Apaharan	233
2	Gold	215
3	My Name Is Khan	213
4	Captain America: Civil War	191
...	...	...
3339	Return of Hanuman	1
3340	Subah Subah	1
3341	The Wish Fish	1
3342	Vaibhav Sethia: Don't	1
3343	Yeh Hai Malegaon Ka Superman	1

3344 rows × 2 columns

Movies with Largest crew

In [3]:
```
cmd6_3 = pd.read_sql_query("SELECT Title , CastSize FROM Title_CastSize  ORDER BY CastSize DESC LIMIT 1", conn)
cmd6_3
```

Out[3]:

	Title	CastSize
0	Ocean's Eight	238

7. A decade is a sequence of 10 consecutive years. For example, say in your database you have movie information starting from 1965. Then the first decade is 1965, 1966, ..., 1974; the second one is 1967, 1968, ..., 1976 and so on. Find the decade D with the largest number of films and the total number of films in D.

Yearwise Movie Table

In [17]:
```
cmd7_1 = pd.read_sql_query("SELECT Year , title, MID  FROM Movie  ORDER BY
 Year", conn)
cmd7_1.to_sql("Year_title", conn, if_exists="replace")
cmd7_1
```

Out[17]:

	year	title	MID
0	1931	Alam Ara	tt0021594
1	1936	Sant Tukaram	tt0028217
2	1936	Achhut Kanya	tt0027256
3	1936	Devdas	tt0026274
4	1939	The Little Princess	tt0031580
...	...	...	...
3470	IV 2011	Lucky	tt1702543
3471	IV 2017	Game Over	tt7399620
3472	V 2015	Hero	tt4467202
3473	VI 2015	Alone	tt4271730
3474	XVII 2016	Trapped	tt6206564

3475 rows × 3 columns

Yearwise Movie Counts Table

In [18]:
```
cmd7_2 = pd.read_sql_query("SELECT Year, COUNT(*) AS MovieCount FROM Year_t
itle GROUP BY year", conn)
cmd7_2
```

Out[18]:

	year	MovieCount
0	1931	1
1	1936	3
2	1939	2
3	1941	1
4	1943	1
...	...	...
120	IV 2011	1
121	IV 2017	1
122	V 2015	1
123	VI 2015	1
124	XVII 2016	1

125 rows × 2 columns

Deacdewise Movie Table

In [19]:
```
cmd7_2.to_sql("Year_MovieCount", conn, if_exists="replace")
cmd7_3 = pd.read_sql_query("SELECT Year/10 AS Deacde, MovieCount FROM Year_
MovieCount", conn)
cmd7_3
```

Out[19]:

	Deacde	MovieCount
0	193	1
1	193	3
2	193	2
3	194	1
4	194	1
...	...	...
120	0	1
121	0	1
122	0	1
123	0	1
124	0	1

125 rows × 2 columns

Deacdewise Movie Count Table

In [22]:
```
cmd7_3.to_sql("Decade_MovieCount", conn, if_exists="replace")
cmd7_4 = pd.read_sql_query("SELECT Deacde, SUM(MovieCount) as MovieCount FR
OM Decade_MovieCount GROUP BY Deacde ORDER BY MovieCount DESC", conn)
cmd7_4
```

Out[22]:

	Deacde	MovieCount
0	201	1018
1	200	959
2	199	551
3	198	342
4	197	254
5	196	145
6	0	117
7	195	71
8	194	12
9	193	6

Max movies in 2010-2016 Deacde

8. Find the actors that were never unemployed for more than 3 years at a stretch. (Assume that the actors remain unemployed between two consecutive movies).

The way we plan to do this (Approach)

1. For all actor (groupwise) we will createtable yearwise od their movies. SO we will have their back-toback movies placed at consecutive cell
2. Now if we can subtract for each actor their consecutive years of acting we can figure out if they were unemployed for more than 'n' years or not
   ## #### Solution
3. For all actor (groupwise) we will createtable yearwise od their movies
4. Now create same table as above only such that first row is some dummy value and table 2 is same as table 1 from row2 onwards
5. Now we subtract rowid wise table to find GAP of unemployment

In [153]:
```
cmd8 = pd.read_sql_query("SELECT p.pid as ID ,p.Name as Name,m.title AS 'Mo
vieName',m.Year as Year FROM Person p JOIN M_Cast mc ON trim(mc.PID)=trim
(p.PID) JOIN Movie m ON trim(m.MID)=trim(mc.MID) ORDER BY Name DESC", conn)
```

In [154]:
```
cmd8
```

Out[154]:

	ID	Name	MovieName	Year
0	nm5113220	Zeishan Quadri	Gangs of Wasseypur	2012
1	nm5113220	Zeishan Quadri	Revolver Rani	2014
2	nm1680229	Yograj Bhat	Maanikya	2014
3	nm0007181	Yash Chopra	Om Shanti Om	2007
4	nm0007181	Yash Chopra	Veer-Zaara	2004
...	...	...	...	...
86845	nm5163714	'Nandha' Saravanan	Nandha	2001
86846	nm8644387	'Musafir' Radio Performing	Rock On!!	2008
86847	nm0704042	'Lee' George Quinones	Bomb the System	2002
86848	nm2128968	'Ganja' Karuppu	Sandai Kozhi	2005
86849	nm2128968	'Ganja' Karuppu	Pazhani	2008

86850 rows × 4 columns

In [12]:
```
cmd8_1 = pd.read_sql_query("SELECT p.pid as ID ,p.Name as Name,m.title AS
 'MovieName',m.Year as Year FROM Person p \
                           JOIN M_Cast mc ON trim(mc.PID)=trim(p.PID) JOIN
 Movie m ON trim(m.MID)=trim(mc.MID) \
                           ORDER BY Name DESC,Year ASC", conn)
print (cmd8_1)
cmd8_1.to_sql("Actor-Movie-Year", conn, if_exists="replace")
```

```
             ID                         Name           MovieName  Year
0      nm5113220               Zeishan Quadri  Gangs of Wasseypur  2012
1      nm5113220               Zeishan Quadri       Revolver Rani  2014
2      nm1680229                  Yograj Bhat            Maanikya  2014
3      nm0007181                  Yash Chopra    Dil To Pagal Hai  1997
4      nm0007181                  Yash Chopra          Veer-Zaara  2004
...          ...                          ...                 ...   ...
86845  nm5163714          'Nandha' Saravanan              Nandha  2001
86846  nm8644387   'Musafir' Radio Performing           Rock On!!  2008
86847  nm0704042        'Lee' George Quinones     Bomb the System  2002
86848  nm2128968             'Ganja' Karuppu        Sandai Kozhi  2005
86849  nm2128968             'Ganja' Karuppu             Pazhani  2008

[86850 rows x 4 columns]
```

Now what we can do is subtract Table 2 from Table 1 (Year column) when Name column matches. However for subtracting number of row should be same. We can insert one null value on the top of Table 1 and start Table 2 from

In [25]:
```
cursorObject= conn.cursor()
insertStatement = "INSERT INTO 'Actor-Movie-Year' (ID,Name,MovieName,Year)
 VALUES('zzdummy123','zzdummyname','zzdummymovie',0)"
cursorObject.execute(insertStatement)
cursorObject.execute("COMMIT")
#print (cmd8_2)
"""
sql_delete_query = "DELETE from 'Actor-Movie-Year' where ID = 'dummy123' "
cursorObject.execute(sql_delete_query)
cursorObject.execute("COMMIT")
print (cmd8_2)
"""
cmd8_2 = pd.read_sql_query("SELECT  a.ID,a.Name ,a.MovieName ,a.Year FROM
 'Actor-Movie-Year' a ORDER BY Name DESC,Year ASC", conn)
cmd8_2
```

Out[25]:

	ID	Name	MovieName	Year
0	zzdummy123	zzdummyname	zzdummymovie	0
1	nm5113220	Zeishan Quadri	Gangs of Wasseypur	2012
2	nm5113220	Zeishan Quadri	Revolver Rani	2014
3	nm1680229	Yograj Bhat	Maanikya	2014
4	nm0007181	Yash Chopra	Dil To Pagal Hai	1997
...	...	...	...	...
86846	nm5163714	'Nandha' Saravanan	Nandha	2001
86847	nm8644387	'Musafir' Radio Performing	Rock On!!	2008
86848	nm0704042	'Lee' George Quinones	Bomb the System	2002
86849	nm2128968	'Ganja' Karuppu	Sandai Kozhi	2005
86850	nm2128968	'Ganja' Karuppu	Pazhani	2008

86851 rows × 4 columns

In [27]:
```
cmd8_2.to_sql("Table1", conn, if_exists="replace")
```

In [26]:
```
cmd8_3 = pd.read_sql_query("SELECT  a.ID,a.Name ,a.MovieName ,a.Year FROM
 'Actor-Movie-Year' a ", conn)
cmd8_3
```

Out[26]:

	ID	Name	MovieName	Year
0	nm5113220	Zeishan Quadri	Gangs of Wasseypur	2012
1	nm5113220	Zeishan Quadri	Revolver Rani	2014
2	nm1680229	Yograj Bhat	Maanikya	2014
3	nm0007181	Yash Chopra	Dil To Pagal Hai	1997
4	nm0007181	Yash Chopra	Veer-Zaara	2004
...	...	...	...	...
86846	nm8644387	'Musafir' Radio Performing	Rock On!!	2008
86847	nm0704042	'Lee' George Quinones	Bomb the System	2002
86848	nm2128968	'Ganja' Karuppu	Sandai Kozhi	2005
86849	nm2128968	'Ganja' Karuppu	Pazhani	2008
86850	zzdummy123	zzdummyname	zzdummymovie	0

86851 rows × 4 columns

In [28]:
```
cmd8_3.to_sql("Table2", conn, if_exists="replace")
```

In [55]:
```
cmd8_4 = pd.read_sql_query("SELECT  t1.ID,t1.Name as Name1,t2.Name as Name2
,t1.MovieName as MovieName1 ,t2.MovieName as \
                           MovieName2,t1.Year as Year1,t2.Year as Year2 FRO
M Table1 t1 JOIN Table2 t2 ON t1.rowid=t2.rowid ", conn)
```

In [56]:
```
cmd8_4
```

Out[56]:

	ID	Name1	Name2	MovieName1	MovieName2	Year1	Year2
0	zzdummy123	zzdummyname	Zeishan Quadri	zzdummymovie	Gangs of Wasseypur	0	2012
1	nm5113220	Zeishan Quadri	Zeishan Quadri	Gangs of Wasseypur	Revolver Rani	2012	2014
2	nm5113220	Zeishan Quadri	Yograj Bhat	Revolver Rani	Maanikya	2014	2014
3	nm1680229	Yograj Bhat	Yash Chopra	Maanikya	Dil To Pagal Hai	2014	1997
4	nm0007181	Yash Chopra	Yash Chopra	Dil To Pagal Hai	Veer-Zaara	1997	2004
...	...	...	...	...	...	...	...
86846	nm5163714	'Nandha' Saravanan	'Musafir' Radio Performing	Nandha	Rock On!!	2001	2008
86847	nm8644387	'Musafir' Radio Performing	'Lee' George Quinones	Rock On!!	Bomb the System	2008	2002
86848	nm0704042	'Lee' George Quinones	'Ganja' Karuppu	Bomb the System	Sandai Kozhi	2002	2005
86849	nm2128968	'Ganja' Karuppu	'Ganja' Karuppu	Sandai Kozhi	Pazhani	2005	2008
86850	nm2128968	'Ganja' Karuppu	zzdummyname	Pazhani	zzdummymovie	2008	0

86851 rows × 7 columns

In [57]:
```
cmd8_4.to_sql("Table3", conn, if_exists="replace")
cmd8_5 = pd.read_sql_query("SELECT  ID,Name1,Name2,MovieName1,MovieName2 ,Year1,Year2, CASE WHEN Name1=Name2 THEN \
                           (Year2-Year1) ELSE 'NULL' END GAP FROM Table3 ", conn)
cmd8_5
```

Out[57]:

	ID	Name1	Name2	MovieName1	MovieName2	Year1	Year2	
0	zzdummy123	zzdummyname	Zeishan Quadri	zzdummymovie	Gangs of Wasseypur	0	2012	I
1	nm5113220	Zeishan Quadri	Zeishan Quadri	Gangs of Wasseypur	Revolver Rani	2012	2014	
2	nm5113220	Zeishan Quadri	Yograj Bhat	Revolver Rani	Maanikya	2014	2014	I
3	nm1680229	Yograj Bhat	Yash Chopra	Maanikya	Dil To Pagal Hai	2014	1997	I
4	nm0007181	Yash Chopra	Yash Chopra	Dil To Pagal Hai	Veer-Zaara	1997	2004	
...	...	...	...	...	...	...	...	
86846	nm5163714	'Nandha' Saravanan	'Musafir' Radio Performing	Nandha	Rock On!!	2001	2008	I
86847	nm8644387	'Musafir' Radio Performing	'Lee' George Quinones	Rock On!!	Bomb the System	2008	2002	I
86848	nm0704042	'Lee' George Quinones	'Ganja' Karuppu	Bomb the System	Sandai Kozhi	2002	2005	I
86849	nm2128968	'Ganja' Karuppu	'Ganja' Karuppu	Sandai Kozhi	Pazhani	2005	2008	
86850	nm2128968	'Ganja' Karuppu	zzdummyname	Pazhani	zzdummymovie	2008	0	I

86851 rows × 8 columns

In [58]:
```
cmd8_5.to_sql("Table4", conn, if_exists="replace")
cmd8_6 = pd.read_sql_query("SELECT DISTINCT ID,Name1 ,Name2,MovieName1,Movi
eName2 ,Year1,Year2,GAP FROM Table4 \
                           WHERE GAP!='NULL'", conn)
cmd8_6
```

Out[58]:

	ID	Name1	Name2	MovieName1	MovieName2	Year1	Year2	GAP
0	nm5113220	Zeishan Quadri	Zeishan Quadri	Gangs of Wasseypur	Revolver Rani	2012	2014	2
1	nm0007181	Yash Chopra	Yash Chopra	Dil To Pagal Hai	Veer-Zaara	1997	2004	7
2	nm0007181	Yash Chopra	Yash Chopra	Veer-Zaara	Om Shanti Om	2004	2007	3
3	nm1318999	Yana Gupta	Yana Gupta	Dum	Rakht	2003	2004	1
4	nm1318999	Yana Gupta	Yana Gupta	Rakht	Anniyan	2004	2005	1
...	...	...	...	...	...	...	...	...
56024	nm0359845	A.K. Hangal	A.K. Hangal	Mr Prime Minister	Krishna Aur Kans	2005	2012	7
56025	nm1693065	A.K. Agnihotri	A.K. Agnihotri	Main Tulsi Tere Aangan Ki	Qatl	1978	1986	8
56026	nm1693065	A.K. Agnihotri	A.K. Agnihotri	Qatl	Purani Haveli	1986	1989	3
56027	nm1869655	A. Abdul Hameed	A. Abdul Hameed	Prem Nagar	Julie	1974	1975	1
56028	nm2128968	'Ganja' Karuppu	'Ganja' Karuppu	Sandai Kozhi	Pazhani	2005	2008	3

56029 rows × 8 columns

In [59]:
```
cmd8_6.to_sql("Table5", conn, if_exists="replace")
cmd8_7 = pd.read_sql_query("SELECT ID,Name1 ,Name2,MovieName1,MovieName2 ,Year1,Year2,GAP FROM Table5 GROUP BY Name1 \
                           HAVING GAP<3", conn)
cmd8_7
```

Out[59]:

	ID	Name1	Name2	MovieName1	MovieName2	Year1	Year2	GAP
0	nm1869655	A. Abdul Hameed	A. Abdul Hameed	Prem Nagar	Julie	1974	1975	1
1	nm1436693	A.R. Murugadoss	A.R. Murugadoss	7 Aum Arivu	Thuppakki	2011	2012	1
2	nm4563111	A.R. Rama	A.R. Rama	Padman	Bioscopewala	2018	2018	0
3	nm3022788	Aabhas Yadav	Aabhas Yadav	Bunty Aur Babli	The Wishing Tree	2005	2017	12
4	nm7390393	Aachi Manorama	Aachi Manorama	Singam 2	Vikram	2013	I 1986	-2013
...	...	...	...	...	...	...	...	...
5629	nm2134474	Vikas Bahl	Vikas Bahl	Hasee Toh Phasee	Bombay Velvet	2014	2015	1
5630	nm0220849	Vikas Desai	Vikas Desai	Arvind Desai Ki Ajeeb Dastaan	Aar Ya Paar	1978	1997	19
5631	nm0576495	Vinod Mehra	Vinod Mehra	Insaniyat	Aatank	1994	1996	2
5632	nm1318999	Yana Gupta	Yana Gupta	Murder 2	Chalo Dilli	2011	2011	0
5633	nm5113220	Zeishan Quadri	Zeishan Quadri	Gangs of Wasseypur	Revolver Rani	2012	2014	2

5634 rows × 8 columns

In [61]:
```
cmd8_7.to_sql("Table6", conn, if_exists="replace")
cmd8_8 = pd.read_sql_query("SELECT Name1 as Name ,MovieName1 AS 'Previous M
ovie',MovieName2 AS 'Next Movie',Year1 \
                           AS 'Previous Movie Year',Year2 AS 'Next Movie Ye
ar',GAP FROM Table6", conn)
cmd8_8
```

Out[61]:

	Name	Previous Movie	Next Movie	Previous Movie Year	Next Movie Year	GAP
0	A. Abdul Hameed	Prem Nagar	Julie	1974	1975	1
1	A.R. Murugadoss	7 Aum Arivu	Thuppakki	2011	2012	1
2	A.R. Rama	Padman	Bioscopewala	2018	2018	0
3	Aabhas Yadav	Bunty Aur Babli	The Wishing Tree	2005	2017	12
4	Aachi Manorama	Singam 2	Vikram	2013	I 1986	-2013
...	...	...	...	...	...	...
5629	Vikas Bahl	Hasee Toh Phasee	Bombay Velvet	2014	2015	1
5630	Vikas Desai	Arvind Desai Ki Ajeeb Dastaan	Aar Ya Paar	1978	1997	19
5631	Vinod Mehra	Insaniyat	Aatank	1994	1996	2
5632	Yana Gupta	Murder 2	Chalo Dilli	2011	2011	0
5633	Zeishan Quadri	Gangs of Wasseypur	Revolver Rani	2012	2014	2

5634 rows × 6 columns

9. Find all the actors that made more movies with Yash Chopra than any other director.

In [83]:
```
cmd9 = pd.read_sql_query("SELECT  pa.name as Actor, pd.name as Director,COU
NT(*) as MovieCount FROM M_director d JOIN Person pd ON trim(pd.PID) = trim
(d.PID) JOIN M_Cast c ON trim(c.MID) = trim(d.MID) JOIN Person pa ON trim(p
a.PID) = trim(c.PID) GROUP BY pa.name, pd.name", conn)
```

In [84]:
```
cmd9
```

Out[84]:

	Director	Actor	MovieCount
0	A. Bhimsingh	Abhimanyyu Abhimanyu	1
1	A. Bhimsingh	Achala Sachdev	1
2	A. Bhimsingh	Agha	1
3	A. Bhimsingh	Alka	1
4	A. Bhimsingh	Anand	1
...	...	...	...
128398	Zunaid Memon	Satyendra Kapoor	1
128399	Zunaid Memon	Sergio Kato	1
128400	Zunaid Memon	Sulabha Deshpande	1
128401	Zunaid Memon	Vaibhav Jhalani	1
128402	Zunaid Memon	Vivek Madan	1

128403 rows × 3 columns

In [5]:
```
cmd9 = pd.read_sql_query("SELECT  pa.name as Actor, pd.name as Director,COUNT(*) as MovieCount FROM M_director d JOIN Person pd ON trim(pd.PID) = trim(d.PID) JOIN M_Cast c ON trim(c.MID) = trim(d.MID) JOIN Person pa ON trim(pa.PID) = trim(c.PID) GROUP BY pa.name, pd.name", conn)
```

In [7]:
```
cmd9
```

Out[7]:

	Actor	Director	MovieCount
0	'Ganja' Karuppu	N. Linguswamy	1
1	'Ganja' Karuppu	Perarasu	1
2	'Lee' George Quinones	Adam Bhala Lough	1
3	'Musafir' Radio Performing	Abhishek Kapoor	1
4	'Musafir' Radio Performing	Abhishek Kapoor	1
...	...	...	...
128398	Yograj Bhat	Sudeep	1
128399	Yograj Bhat	Sudeep	1
128400	Zeishan Quadri	Anurag Kashyap	1
128401	Zeishan Quadri	Anurag Kashyap	1
128402	Zeishan Quadri	Sai Kabir	1

128403 rows × 3 columns

In [85]:
```
cmd9.to_sql("Dir_Act_Count", conn, if_exists="replace")
```

In [8]:
```
cmd9_1 = pd.read_sql_query("SELECT Actor,Director,MovieCount FROM 'Dir_Act_
Count' GROUP BY Actor ORDER BY MovieCount DESC",conn )
```

In [9]:
```
cmd9_1
```

Out[9]:

	Actor	Director	MovieCount
0	Jagdish Raj	Yash Chopra	11
1	Manmohan Krishna	Yash Chopra	10
2	Subhash Ghai	Subhash Ghai	10
3	Manmohan Krishna	Yash Chopra	10
4	Subhash Ghai	Subhash Ghai	10
...	...	...	...
30816	Vinod Pande	Vinod Pande	1
30817	Vishal Bhardwaj	Kamal Haasan	1
30818	Yana Gupta	Yana Gupta	1
30819	Yograj Bhat	Sudeep	1
30820	Zeishan Quadri	Sai Kabir	1

30821 rows × 3 columns

In [10]:
```
cmd9_1.to_sql("Act_Dir_Count", conn, if_exists="replace")
cmd9_2 = pd.read_sql_query("SELECT Actor,Director,MovieCount FROM 'Act_Dir_
Count' WHERE Director LIKE 'Yash Chopra'",conn )
cmd9_2
```

Out[10]:

	Actor	Director	MovieCount
0	Jagdish Raj	Yash Chopra	11
1	Manmohan Krishna	Yash Chopra	10
2	Manmohan Krishna	Yash Chopra	10
3	Iftekhar	Yash Chopra	9
4	Madan Puri	Yash Chopra	8
...	...	...	...
415	Raman Kumar	Yash Chopra	1
416	Romesh Sharma	Yash Chopra	1
417	Sachin	Yash Chopra	1
418	Sajid Khan	Yash Chopra	1
419	Tinnu Verma	Yash Chopra	1

420 rows × 3 columns

10. The Shahrukh number of an actor is the length of the shortest path between the actor and Shahrukh Khan in the "co-acting" graph. That is, Shahrukh Khan has Shahrukh number 0; all actors who acted in the same film as Shahrukh have Shahrukh number 1; all actors who acted in the same film as some actor with Shahrukh number 1 have Shahrukh number 2, etc. Return all actors whose Shahrukh number is 2.

In [11]:
```
cmd10_1 = pd.read_sql_query("SELECT p.PID,p.Name FROM person p WHERE p.Name
LIKE '%Shah % Khan%' ",conn )
cmd10_1
```

Out[11]:

	PID	Name
0	nm0451321	Shah Rukh Khan

Step 1: List of all SRK Movies

In [12]:
```
SRK_Movies = pd.read_sql_query("SELECT mc.MID,m.title FROM M_Cast mc JOIN M
ovie m ON trim(mc.MID)=trim(m.MID)  WHERE trim(mc.PID) LIKE 'nm0451321' ",c
onn )
SRK_Movies.to_sql("SRK_Movies", conn, if_exists="replace")
SRK_Movies
```

Out[12]:

	MID	title
0	tt1188996	My Name Is Khan
1	tt1285241	Don 2
2	tt0248126	Kabhi Khushi Kabhie Gham...
3	tt5946128	Dear Zindagi
4	tt3405236	Raees
...	...	...
85	tt1538210	Aao Wish Karein
86	tt0250415	Har Dil Jo Pyar Karega...
87	tt0453748	Kuchh Meetha Ho Jaye
88	tt0286664	Gudgudee
89	tt1773042	Shahrukh Bola 'Khoobsurat Hai Tu'... And She B...

90 rows × 2 columns

Step 2: Find all co-actors in the above MIDs

In [10]:
```
SRK_coactors = pd.read_sql_query("SELECT s.MID,s.title,mc.PID,p.Name FROM S
RK_Movies s JOIN M_Cast mc ON trim(mc.MID)=trim(s.MID) JOIN Person p ON tri
m(p.PID)=trim(mc.PID)",conn )
SRK_coactors
```

Out[10]:

	MID	title	PID	Name
0	tt1188996	My Name Is Khan	nm0451321	Shah Rukh Khan
1	tt1188996	My Name Is Khan	nm0004418	Kajol
2	tt1188996	My Name Is Khan	nm1995953	Katie A. Keane
3	tt1188996	My Name Is Khan	nm2778261	Kenton Duty
4	tt1188996	My Name Is Khan	nm0631373	Benny Nieves
...	...	...	...	...
3487	tt1773042	Shahrukh Bola 'Khoobsurat Hai Tu'... And She B...	nm3093045	Choiti Ghosh
3488	tt1773042	Shahrukh Bola 'Khoobsurat Hai Tu'... And She B...	nm0451154	Afzal Khan
3489	tt1773042	Shahrukh Bola 'Khoobsurat Hai Tu'... And She B...	nm0451321	Shah Rukh Khan
3490	tt1773042	Shahrukh Bola 'Khoobsurat Hai Tu'... And She B...	nm1946407	Kay Kay Menon
3491	tt1773042	Shahrukh Bola 'Khoobsurat Hai Tu'... And She B...	nm3385526	Gopal K. Singh

3492 rows × 4 columns

In [11]:
```
SRK_coactors.to_sql("SRK_coactors", conn, if_exists="replace")
SRK_coactors_distinct = pd.read_sql_query("SELECT DISTINCT s.PID,s.Name FRO
M SRK_coactors s ",conn )
SRK_coactors_distinct
```

Out[11]:

	PID	Name
0	nm0451321	Shah Rukh Khan
1	nm0004418	Kajol
2	nm1995953	Katie A. Keane
3	nm2778261	Kenton Duty
4	nm0631373	Benny Nieves
...	...	...
2456	nm4173451	Sanjay Dadheech
2457	nm7620177	Dhananjay Galani
2458	nm3093045	Choiti Ghosh
2459	nm0451154	Afzal Khan
2460	nm3385526	Gopal K. Singh

2461 rows × 2 columns

Step 3: Find list of all movies of all above PIDs (i.e SRK Coactors)

In [13]:
```
SRK_coactors.to_sql("SRK_coactors_distinct", conn, if_exists="replace")
SRK_co_coactors_Movies = pd.read_sql_query("SELECT s.PID,mc.MID,s.Name,m.title FROM SRK_coactors_distinct s JOIN M_Cast mc ON trim(s.PID) = trim(mc.PID) JOIN Movie m ON trim(mc.MID)=trim(m.MID) WHERE trim(s.PID) NOT LIKE 'nm0451321'",conn )
SRK_co_coactors_Movies
```

Out[13]:

	PID	MID	Name	title
0	nm0004418	tt1188996	Kajol	My Name Is Khan
1	nm0004418	tt0248126	Kajol	Kabhi Khushi Kabhie Gham...
2	nm0004418	tt0112870	Kajol	Dilwale Dulhania Le Jayenge
3	nm0004418	tt0347304	Kajol	Kal Ho Naa Ho
4	nm0004418	tt4535650	Kajol	Dilwale
...	...	...	...	...
86342	nm3385526	tt0439714	Gopal K. Singh	Mumbai Express
86343	nm3385526	tt0409724	Gopal K. Singh	Bardaasht
86344	nm3385526	tt0366276	Gopal K. Singh	Calcutta Mail
86345	nm3385526	tt1948640	Gopal K. Singh	The Waiting Room
86346	nm3385526	tt1773042	Gopal K. Singh	Shahrukh Bola 'Khoobsurat Hai Tu'... And She B...

86347 rows × 4 columns

So above is the list we are looking for which are SRK's co co actor. But in case we dont want what movies they acted in and just the SRK co co actors the we could just find distinct people from above table

In [16]:
```
SRK_co_coactors_Movies.to_sql("SRK_co_coactors_Movies", conn, if_exists="replace")
SRK_co_coactors_distinct = pd.read_sql_query("SELECT DISTINCT s.PID,s.Name
 FROM SRK_co_coactors_Movies s ",conn )
SRK_co_coactors_distinct
```

Out[16]:

	PID	Name
0	nm0004418	Kajol
1	nm1995953	Katie A. Keane
2	nm2778261	Kenton Duty
3	nm0631373	Benny Nieves
4	nm0241935	Christopher B. Duncan
...	...	...
2455	nm4173451	Sanjay Dadheech
2456	nm7620177	Dhananjay Galani
2457	nm3093045	Choiti Ghosh
2458	nm0451154	Afzal Khan
2459	nm3385526	Gopal K. Singh

2460 rows × 2 columns

In [ ]: