In [1]:

```
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 [2]:

```
conn = sqlite3.connect(r'D:\AppliedAI\Homework-n-Assignments\# 22 SQL Assignment on IMDB data\Db-I
MDB.db')
movie_table = pd.read_sql_query('SELECT * FROM Movie ', conn)
```

In [3]:

```
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 'Movies 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 m
g.MID=m.MID JOIN Genre g on g.GID=mg.GID WHERE m.year %4==0 and trim(g.name) LIKE '%Comedy%'", con
n)
```

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	2004

	Director Name	Movies Name	Year
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 Name' 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 [7]:

```
#cmd3=pd.read_sql_query("SELECT p.Name as Actor FROM Person p JOIN M_Cast mc ON
trim(p.PID)=trim(mc.PID) JOIN Movie m ON trim(m.MID)=trim(mc.MID) IN (SELECT p.Name FROM Person p
JOIN M_Cast mc ON trim(p.PID)=trim(mc.PID) JOIN Movie m ON trim(m.MID)=trim(mc.MID) WHERE m.Year >
1990) INNER JOIN (SELECT p.Name FROM Person p JOIN M_Cast mc ON trim(p.PID)=trim(mc.PID) JOIN Movi
e m ON trim(m.MID)=trim(mc.MID) WHERE m.Year < 1970)",conn)

cmd3=pd.read_sql_query("SELECT p.Name as 'Actor' FROM Person p JOIN M_Cast mc ON
(p.PID)=trim(mc.PID) JOIN Movie m ON (m.MID)=(mc.MID) WHERE m.Year <1970 AND m.Year>1990",conn)
cmd3
```

Out[7]:

Actor

In [5]:

```
cmd3 = pd.read_sql_query ("SELECT p.Name from Person p WHERE p.Name IN\
(SELECT p.Name FROM Person p JOIN M_Cast mc ON (p.PID)=(mc.PID)\
JOIN Movie m1 ON (m1.MID)=(mc.MID)\
WHERE m1.Year > 1990)\
and p.Name IN \
(SELECT p.Name FROM Person p JOIN M_Cast mc ON (p.PID)=(mc.PID)\
JOIN Movie m2 ON (m2.MID)=(mc.MID)\
WHERE m2.Year < 1970)",conn)
cmd3
```

Out[5]:

Name

In [3]:

```
cmd3 = pd.read_sql_query("SELECT p.Name FROM Person p JOIN M_Cast mc ON (p.PID)=(mc.PID)\
JOIN Movie m1 ON (m1.MID)=(mc.MID)\
WHERE m1.Year NOT BETWEEN 1970 and 1990",conn)
cmd3
```

Out[3]:

Name

In [ ]:

```
cmd3 = pd.read_sql_query("SELECT p.Name as 'Actor' FROM Person p JOIN M_Cast mc ON
trim(p.PID)=trim(mc.PID) JOIN Movie m ON trim(m.MID)=trim(mc.MID) WHERE m.year >1990",conn)
cmd3
```

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

	Actor	Movie	Year

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 WHE
RE 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 WH
ERE 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

	Actor

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-after-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 [11]:

```
cmd4 = pd.read_sql_query("SELECT p.Name,count(*) as Count from Person p, M_Director md on
trim(md.PID)=trim(p.PID) GROUP BY p.PID,p.Name having Count(*) > 9 ORDER BY Count DESC", conn)
cmd4
```

Out[11]:

	Name	Count
0	David Dhawan	39
1	David Dhawan	39
2	Mahesh Bhatt	35
3	Mahesh Bhatt	35
4	Priyadarshan	30
...	...	...
106	Pankaj Parashar	10
107	J. Om Prakash	10
108	J. Om Prakash	10
109	Bimal Roy	10
110	Bimal Roy	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 [14]:

```
cmd5a= pd.read_sql_query("SELECT m.year,count(*) Movies_Count from Movie m where m.MID NOT IN
(SELECT mc.MID from M_Cast mc INNER JOIN  Person p on trim(p.PID) = trim(mc.PID) where p.Gender =
'Male') GROUP BY m.year ORDER BY m.year DESC", conn)
cmd5a
```

Out[14]:

	year	Movies_Count
0	I 2018	1
1	2018	1
2	2012	1
3	2009	1
4	2000	1
5	1999	1
6	1939	1

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

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 [15]:

```
cmd6=pd.read_sql_query("SELECT m.title,count(distinct(mc.PID)) as Cast_Size from Movie m JOIN M_Ca
st mc on trim(mc.MID) = trim(m.MID) GROUP BY m.MID ORDER BY Cast_Size DESC limit 1",conn)
cmd6
```

Out[15]:

	title	Cast_Size
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.

In [17]:

```
cmd7 = pd.read_sql_query("SELECT d.year as Decade_Start_year, d.year+10 as Decade_End_year,
count(*) as num_movies from (SELECT DISTINCT year from Movie) d JOIN Movie m on
m.year>=Decade_Start_year and m.year<= Decade_End_year GROUP BY Decade_End_year ORDER BY
num_movies DESC limit 1",conn)
cmd7
```

Out[17]:

	Decade_Start_year	Decade_End_year	num_movies
0	2007	2017	1232

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 '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", 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 MovieN
ame1 ,t2.MovieName as \
                           MovieName2,t1.Year as Year1,t2.Year as Year2 FROM 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	GAP
0	zzdummy123	zzdummyname	Zeishan Quadri	zzdummymovie	Gangs of Wasseypur	0	2012	NULL
1	nm5113220	Zeishan Quadri	Zeishan Quadri	Gangs of Wasseypur	Revolver Rani	2012	2014	2
2	nm5113220	Zeishan Quadri	Yograj Bhat	Revolver Rani	Maanikya	2014	2014	NULL
3	nm1680229	Yograj Bhat	Yash Chopra	Maanikya	Dil To Pagal Hai	2014	1997	NULL
4	nm0007181	Yash Chopra	Yash Chopra	Dil To Pagal Hai	Veer-Zaara	1997	2004	7
...	...	...	...	...	...	...	...	...
86846	nm5163714	'Nandha' Saravanan	'Musafir' Radio Performing	Nandha	Rock On!!	2001	2008	NULL
86847	nm8644387	'Musafir' Radio Performing	'Lee' George Quinones	Rock On!!	Bomb the System	2008	2002	NULL
86848	nm0704042	'Lee' George Quinones	'Ganja' Karuppu	Bomb the System	Sandai Kozhi	2002	2005	NULL
86849	nm2128968	'Ganja' Karuppu	'Ganja' Karuppu	Sandai Kozhi	Pazhani	2005	2008	3
86850	nm2128968	'Ganja' Karuppu	zzdummyname	Pazhani	zzdummymovie	2008	0	NULL

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,MovieName2 ,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 Tab
le5 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 Movie',MovieName2 AS
'Next Movie',Year1 \
                           AS 'Previous Movie Year',Year2 AS 'Next Movie Year',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

	Name	Previous Movie	Next Movie	Previous Movie Year	Next Movie Year	GAP
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 [19]:

```
cmd9 = pd.read_sql_query("SELECT DISTINCT Actor, Count(*) as Movies_with_YashChopra FROM(SELECT p1
.Name as Director, m1.title as Movie FROM Person p1 INNER JOIN M_Director md on
TRIM(md.PID)=p1.PID INNER JOIN Movie m1 on TRIM(md.MID)=m1.MID and p1.Name LIKE 'Yash%' GROUP BY p
1.Name, m1.title) t1 INNER JOIN (SELECT p2.Name as Actor,m2.title as Movie FROM Person p2 INNER JO
IN M_Cast mc on trim(mc.PID)=p2.PID INNER JOIN Movie m2 on trim(mc.MID)=m2.MID Group By p2.Name, m
2.title) t2 on t1.Movie=t2.Movie Group By t1.Director, t2.Actor ORDER BY Movies_with_YashChopra DE
SC",conn)
cmd9
```

Out[19]:

	Actor	Movies_with_YashChopra
0	Jagdish Raj	11
1	Manmohan Krishna	10
2	Manmohan Krishna	10
3	Iftekhar	9
4	Madan Puri	8
...	...	...
509	Romesh Sharma	1
510	Sachin	1
511	Sajid Khan	1
512	Sunny Deol	1
513	Tinnu Verma	1

514 rows × 2 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 Movie m ON
trim(mc.MID)=trim(m.MID)  WHERE trim(mc.PID) LIKE 'nm0451321' ",conn )
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 SRK_Movies s JOIN M_Cast
mc ON trim(mc.MID)=trim(s.MID) JOIN Person p ON trim(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 FROM 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 [ ]: