IMDB Dataset Analysis

The IMDB Dataset consists of Movie Information from IMDB.

Number of Movies: 3475
Total Number of People associated with the Movies in the Database : 38284
Timespan of Movies: 1931-2018

Attribute Information:

1. MID :- Movie ID ie a Unique Identification for each movie in our Dataset.
2. PID :- Person ID ie a Unique Identification for each Person associated with a Movie. This Person could be an Actor, Director or Producer for the Movie.
3. GID :- Genre ID : This is, again a unique identification for each Movie Genre. Note that a Single Movie in the Database can have Multiple Genres associated with it. Eg: Comedy,Drama,Action,Horror etc.
4. Year :- Year denotes the Year in which the Movie was Released.
5. Title :- Title denotes the Name of the Movie.
6. Rating :- Rating indicates a Score between 1 to 10. A Higher Rating denotes the extent to which the Movie has been liked by the Audience.
7. Num_Votes :- This basically denotes the Number of Votes casted on the IMDB Platform that have contributed to the Movie Rating.

Objective:

Given the Movies Information shared with us across Multiple Tables in the Normalized Form, find Answers to Relevant Questions that could be required to be obtained in the Real World.

Loading the Data

Firstly we will import the Required Packages that will help us in solving the problems at Hand as well as the Database in consideration using sqlite3. This is carried out as follows :-

In [1]:
```
import sqlite3
import pandas as pd

# using the SQLite Table to read data.
con = sqlite3.connect('Db-IMDB.db')
```

Solving Questions Using MySQL :-

Firstly we realise that our Data is not in a Perfect Form to carry out all the analysis (Eg:- Year is not in a Proper Format :- We have both years 'I 2018' as well as '2018',excesses whitespaces at the beginning & at the end etc). Therefore, in order to make our Nested Sub-Queries as well as our Joins Efficient, instead of using the SQL TRIM() function with the joins repeatedly, we will update all the Relevant Tables in our IMDB Dataset as follows :-

In [2]:
```
#Updating the MOVIE Table : Taking the Required Substring
movie_update = """UPDATE MOVIE SET YEAR = TRIM(SUBSTR(YEAR,-4,4)),MID =
 TRIM(MID)"""
mov = con.cursor()
mov.execute(movie_update)

#Updating the M_CAST Table
cast_update = """UPDATE M_CAST SET MID=TRIM(MID),PID=TRIM(PID)"""
cst = con.cursor()
cst.execute(cast_update)

#Updating the PERSON Table
person_update = """UPDATE PERSON SET PID=TRIM(PID),NAME=TRIM(NAME),GEND
ER=TRIM(GENDER)"""
person = con.cursor()
person.execute(person_update)
```

```
#Updating the GENRE Table
genre_update = """UPDATE GENRE SET GID=TRIM(GID),NAME=TRIM(NAME)"""
genre = con.cursor()
genre.execute(genre_update)

#Updating the M_PRODUCER Table
producer_update = """UPDATE M_PRODUCER SET MID=TRIM(MID),PID=TRIM(PI
D)"""
genre = con.cursor()
genre.execute(genre_update)

#Updating the M_DIRECTOR Table
director_update = """UPDATE M_DIRECTOR SET MID=TRIM(MID),PID=TRIM(PI
D)"""
genre = con.cursor()
genre.execute(genre_update)

#Updating the M_GENRE Table
mov_genre_update = """UPDATE M_GENRE SET MID=TRIM(MID),GID=TRIM(GID)"""
genre = con.cursor()
genre.execute(genre_update)

con.commit()
```

Question 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 [3]:
```
pd.read_sql_query("""SELECT DISTINCT NAME AS DIRECTOR_NAME,TITLE AS MOV
IE_NAME,YEAR AS RELEASE_YEAR
                    FROM
                    (SELECT M.MID,TITLE,YEAR,D.PID,P.NAME,GENDER,G.GID
 FROM MOVIE AS M
                    INNER JOIN M_DIRECTOR AS D ON M.MID=D.MID
                    INNER JOIN PERSON AS P ON D.PID = P.PID
                    INNER JOIN M_GENRE AS G ON M.MID = G.MID
                    INNER JOIN GENRE AS MG ON MG.GID = G.GID
                    WHERE MG.NAME LIKE '%COMEDY%' AND YEAR%4=0 )
                    ORDER BY RELEASE_YEAR DESC
                    """, con).head(10)
```

Out[3]:

	DIRECTOR_NAME	MOVIE_NAME	RELEASE_YEAR
0	Milap Zaveri	Mastizaade	2016
1	Shakun Batra	Kapoor & Sons	2016
2	Rohit Dhawan	Dishoom	2016
3	Nitya Mehra	Baar Baar Dekho	2016
4	Aditya Chopra	Befikre	2016
5	Shubhashish Bhutiani	Hotel Salvation	2016
6	Mudassar Aziz	Happy Bhag Jayegi	2016
7	Remo D'Souza	A Flying Jatt	2016
8	Sohail Khan	Freaky Ali	2016
9	Umesh Ghadge	Kyaa Kool Hain Hum 3	2016

So, as required our Query Returns the Director Name, Movie Name as well as the Year of Release. We obtain a total of 380 Rows in our Resultant obtained above. A Sample of 10 Records is shown.

Approach :-

1. First I used MID to obtain the PID (for the Directors associated with each MID) as well as the GID for each movie.
2. The M_Genre Table has the NAME field which corresponds to the Genre Name as is asked for, and is filtered as having 'Comedy' as one of the Genres.

3. To filter the Leap Year, the year must be divisible by 4. Also a year is Leap Year if it is divisible by 4 and not divisible by 100 but is divisible by 400. Here, 1931 is the oldest Release_Year for a Movie in a Database and hence we need not consider about Year<1931, and 2000%400=0. So 2000 also is a Leap Year.

Question 2 :-

List the names of all the actors who played in the movie 'Anand' (1971) :-

In [4]:
```
pd.read_sql_query("""SELECT NAME AS ACTOR_NAME FROM
                    (SELECT A.MID,TITLE,YEAR,PID FROM (SELECT MID,TITL
E,YEAR FROM MOVIE
                     WHERE TITLE LIKE '%ANAND' AND YEAR='1971') AS A
                     INNER JOIN M_CAST AS M ON A.MID=M.MID) AS B
                     INNER JOIN PERSON AS P ON B.PID=P.PID""",con)
```

Out[4]:

	ACTOR_NAME
0	Amitabh Bachchan
1	Rajesh Khanna
2	Brahm Bhardwaj
3	Ramesh Deo
4	Seema Deo
5	Dev Kishan
6	Durga Khote
7	Lalita Kumari
8	Lalita Pawar
9	Atam Prakash
10	Sumita Sanyal
11	Asit Kumar Sen

11	Asit Kumar Sen

	ACTOR_NAME
12	Dara Singh
13	Johnny Walker
14	Moolchand
15	Gurnam Singh
16	Savita

Thus, as seen above we obtain all 17 Actors Credited with Working in 'Anand'(1971).

Approach :-

1. Firstly we will obtain the Records in consideration for the Movie 'Anand' so as to obtain the Movie ID for 'Anand'. (T1)
2. Now to obtain the Cast for the Movie Anand, we can join the Table T1 with the M_Cast Table basis the MID Column so as to obtain another Intermediate Table. (T2)
3. Now we have all the PIDs for Actors who acted in 'Anand'. However in order to obtain the Actor Names we join this Table with PERSON Table basis the PID Column, and obtain all the Actor names.

Question 3 :-

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

```
In [5]: pd.read_sql_query("""SELECT DISTINCT NAME AS ACTOR_NAME
                             FROM PERSON AS P
                             WHERE P.PID IN
                             (SELECT PID FROM MOVIE AS M
                             INNER JOIN M_CAST AS MC ON M.MID = MC.MID
                             WHERE YEAR<1970)
                             AND P.PID IN
                             (SELECT PID FROM MOVIE AS M INNER JOIN M_CAST AS MC

                             ON M.MID=MC.MID WHERE YEAR>1990)
                             """,con).head(10)
```

Out[5]:

	ACTOR_NAME
0	Rishi Kapoor
1	Amitabh Bachchan
2	Asrani
3	Zohra Sehgal
4	Parikshat Sahni
5	Rakesh Sharma
6	Sanjay Dutt
7	Ric Young
8	Yusuf
9	Suhasini Mulay

As seen above, We obtain a total of 298 Actors who have acted in Movies which were not released in the Timeline specified.

Approach :-

1. PIDs are taken for all the Movies that were released before 1970 as well as after 1990. In order to do that, we join Movie Table as well as M_Cast Table.
2. We choose those Actor Names from Person Table whose PID is falling in both of the cases above.

Question 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 [6]:
```
pd.read_sql_query("""SELECT DISTINCT NAME AS DIRECTOR_NAME,COUNT(MID) A
S MOVIE_COUNT FROM
                    (SELECT P.PID,NAME,MID,TITLE FROM
                    (SELECT M.MID AS MID,TITLE,PID FROM MOVIE AS M
                    INNER JOIN M_DIRECTOR AS D
                    ON M.MID=D.MID) AS A
                    INNER JOIN PERSON AS P ON A.PID=P.PID) AS B
                    GROUP BY PID
                    HAVING MOVIE_COUNT >=10
                    ORDER BY MOVIE_COUNT DESC
                    """, con).head(10)
```

Out[6]:

	DIRECTOR_NAME	MOVIE_COUNT
0	David Dhawan	78
1	Mahesh Bhatt	70
2	Ram Gopal Varma	60
3	Vikram Bhatt	58
4	Hrishikesh Mukherjee	54
5	Yash Chopra	42
6	Basu Chatterjee	38
7	Shakti Samanta	38
8	Subhash Ghai	36
9	Shyam Benegal	34

Thus we basically obtain a Total of 156 Directors in Total who have directed a Minimum of 10 Movies. A sample of the same are shown above.

Approach :-

1. Movie Director's PID Information is obtained by Joining Movie, M_Director as well as Person Tables.
2. Grouping is carried out basis the PID and Count of Distinct Movie IDs per Group are computed. Then, filtering is carried out basis the MOVIE_COUNT (Minimum Value being equal to 10).

Question 5 :-

a. For each year, count the number of movies in that year that had only Female Actors.:-

In [7]:

```
pd.read_sql_query("""SELECT DISTINCT YEAR AS RELEASE_YEAR,
                    COUNT(DISTINCT MID) AS ALL_FEMALE_MOV_COUNT FROM MO
VIE
                    WHERE MID NOT IN
                    (SELECT DISTINCT M.MID FROM MOVIE AS M WHERE M.MID
 IN
                    (SELECT DISTINCT C.MID FROM M_CAST AS C WHERE C.PID
 IN
                    (SELECT P.PID FROM PERSON AS P WHERE GENDER NOT LIK
E '%FEMALE%')))
                    GROUP BY YEAR
                    ORDER BY YEAR
                    """, con)
```

Out[7]:

	RELEASE_YEAR	ALL_FEMALE_MOV_COUNT
0	1939	1
1	1999	1

2	2000	1
3	2009	1

	RELEASE_YEAR	ALL_FEMALE_MOV_COUNT
4	2012	1
5	2018	2

Approach :-

1. Those Movies are Selected where the Corresponding Cast does not have any Females ie. the Subquery Results in MIDs with At least 1 Male Cast. These MIDs are then Excluded.
2. Note that MIDs where the Cast Gender is 'None' are also included and considered as an 'All Female Movie'.

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 [8]:
```
pd.read_sql_query("""SELECT YEAR, (CAST(ALL_FEMALE_MOV_COUNT AS FLOAT)/
CAST(TOTAL AS FLOAT) *100)
                    AS ALL_FEMALE_MOVIE_PERCENTAGE,
                    TOTAL AS TOTAL_MOVIES FROM
                    (SELECT M.YEAR, ALL_FEMALE_MOV_COUNT,
                    COUNT(M.MID) AS TOTAL FROM MOVIE AS M
                    INNER JOIN
                    (SELECT DISTINCT M.YEAR,
                    COUNT(DISTINCT MID) AS ALL_FEMALE_MOV_COUNT FROM MO
VIE AS M
                    WHERE M.MID NOT IN
                    (SELECT DISTINCT M.MID FROM MOVIE AS M WHERE M.MID
 IN
                    (SELECT DISTINCT C.MID FROM M_CAST AS C WHERE C.PID
 IN (SELECT P.PID FROM
                    PERSON AS P WHERE GENDER NOT LIKE '%FEMALE%')))
                    GROUP BY YEAR) AS A
                    ON M.YEAR = A.YEAR
                    GROUP BY M.YEAR)
                    """, con)
```

Out[8]:

	YEAR	ALL_FEMALE_MOVIE_PERCENTAGE	TOTAL_MOVIES
0	1939	50.000000	2
1	1999	1.515152	66
2	2000	1.562500	64
3	2009	0.909091	110
4	2012	0.900901	111
5	2018	1.923077	104

Approach :-

1. Count of Total Movies is Computed for Each Year and then it is joined with the Result obtained in (a) basis the Year Column.
2. Now the Percentage Ratio of 'All Female Movies' and 'Total Movies Released for the Year' is computed for each year in our Final Result.

Question 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 [9]:
```
pd.read_sql_query("""SELECT TITLE AS MOVIE_TITLE,COUNT(DISTINCT PID) AS
 CAST_SIZE
                    FROM MOVIE AS M INNER JOIN M_CAST AS C
                    ON M.MID=C.MID
                    GROUP BY M.MID,TITLE,YEAR
                    ORDER BY CAST_SIZE DESC LIMIT 1""", con)
```

Out[9]:

	MOVIE_TITLE	CAST_SIZE
0	Ocean's Eight	238

Therefore as we see above, the Movie "Ocean's Eight" has the Largest Cast Size with 238 Unique Actors Acting in the same.

Approach :-

1. This is simply obtained by Joining the Movie Table with M_CAST Table basis the MID Columns, and then finding the Count of Distinct PIDs for each Movie after Grouping.
2. After this, Maximum of the Cast Size is Obtained.

Question 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 [10]:
```
pd.read_sql_query("""SELECT Y.YEAR AS DECADE_START, (Y.YEAR+9) AS DECAD
E_END,COUNT(*) AS NO_OF_MOVIES FROM
                    (SELECT DISTINCT YEAR FROM MOVIE) AS Y INNER JOIN M
OVIE AS M
                    ON M.YEAR >= Y.YEAR AND M.YEAR <= (Y.YEAR + 9)
                    GROUP BY Y.YEAR
                    ORDER BY NO_OF_MOVIES DESC LIMIT 1;""", con)
# Source Code:-
# https://stackoverflow.com/questions/51609285/sql-query-for-find-the-d
ecade-with-the-largest-number-of-records
```

Out[10]:

	DECADE_START	DECADE_END	NO_OF_MOVIES
0	2008	2017	1205

Therefore, as seen above, the Time Period Between (2008-2017) is where the Largest Number of Films are Released -> Total of 1128 Films.

Approach :-

1. Here we can note that the Years in a Decade are not exclusive to a single Decade. Say, for the above example, the Years 1967,1968 etc are a part of >1 Decade.
2. SELECT DISTINCT Years from the Movie Table. The Year Selected will be the Decade_Start and Decade_End will be the (Decade_Start+9).
3. Now Join this result set with the Year from 'Movie' Table basis the condition that the M.Year is in the range specified for our Decade Start and End. Then Grouping is carried out and Movie Count is calculated for that particular Decade.

Question 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). :-

In [11]:
```
pd.read_sql_query("""SELECT DISTINCT NAME AS ACTOR_NAME FROM PERSON AS
 P WHERE TRIM(P.PID) IN
                    (SELECT DISTINCT A.PID FROM
                    ((SELECT C1.PID AS PID,M1.MID AS MID, CAST(YEAR AS
 INT) AS YEAR
                    FROM MOVIE AS M1 INNER JOIN M_CAST AS C1 ON M1.MID=
C1.MID) AS A
                    INNER JOIN
                    (SELECT C2.PID AS PID,CAST(YEAR AS INT) AS NEXT_YEA
R FROM MOVIE AS M2
                    INNER JOIN M_CAST AS C2 ON M2.MID=C2.MID) AS B
                    ON A.PID=B.PID)
                    WHERE NEXT_YEAR > YEAR AND (NEXT_YEAR-YEAR)<=3
                    ORDER BY YEAR,NEXT_YEAR)
                    ORDER BY NAME""",con).head(10)
```

Out[11]:

	ACTOR_NAME
0	'Ganja' Karuppu
1	A. Abdul Hameed
2	A.K. Agnihotri
3	A.K. Hangal
4	A.R. Manikandan
5	A.R. Murugadoss
6	A.R. Rama
7	A.V.S. Subramanyam
8	Aachal Munjal
9	Aachi Manorama

Therefore there are a total of 5464 Actors who were Never Unemployed for More than 3 Years at a Stretch.

Approach :-

1. First the Movie Table is joined with the M_Cast Table to obtain the PID and MID.

2. Now this is repeated because we want to obtain the mapping of all the Years where a particular Person has acted in Final Join is basis the PID for both the intermediate Tables obtained.
3. Filtering is carried out so that the Next_Year Column is always larger than the current year and difference is less than 3 Years.

Question 9 :-

Find all the actors that made more movies with Yash Chopra than any other director :-

In [12]:
```
pd.read_sql_query("""SELECT DISTINCT E.ACTOR_NAME AS ACTORS FROM
                    (SELECT DISTINCT ACTOR_ID,ACTOR_NAME,DIRECTOR_NAME,
                    COUNT(DISTINCT MID) AS NO_MOVIES_TOTAL FROM
                    (SELECT MID,TITLE,YEAR,ACTOR_ID,ACTOR_NAME,DIRECTOR
_ID,NAME AS DIRECTOR_NAME FROM
                    (SELECT MID,TITLE,YEAR,ACTOR_ID,NAME AS ACTOR_NAME,
DIRECTOR_ID FROM
                     (SELECT A.MID,TITLE,YEAR,A.PID AS ACTOR_ID,MD.PID
 AS DIRECTOR_ID FROM
                    (SELECT M.MID,TITLE,YEAR,PID FROM MOVIE AS M INNER
 JOIN M_CAST AS MC
                     ON M.MID=MC.MID) AS A
                     INNER JOIN M_DIRECTOR AS MD ON A.MID = MD.MID) AS
 B
                     INNER JOIN PERSON AS P ON B.ACTOR_ID = P.PID) AS C
                     INNER JOIN PERSON AS P ON C.DIRECTOR_ID = P.PID) A
S D
                     GROUP BY ACTOR_ID,ACTOR_NAME,DIRECTOR_NAME
                     ORDER BY NO_MOVIES_TOTAL DESC) AS E
                     LEFT JOIN
                     ((SELECT DISTINCT ACTOR_ID,ACTOR_NAME,
                                       DIRECTOR_NAME,
                                       COUNT(DISTINCT MID) AS NO_MOVIE
S_YASH
                                       FROM
                    (SELECT MID,TITLE,YEAR,ACTOR_ID,ACTOR_NAME,DIRECTO
R_ID,NAME AS DIRECTOR_NAME FROM
                   (SELECT M.MID,TITLE,YEAR,C.PID AS ACTOR_ID,NAME AS
 ACTOR_NAME,D.PID AS DIRECTOR_ID
                        FROM MOVIE AS M INNER JOIN M_CAST AS C
                        ON M.MID=C.MID INNER JOIN M_DIRECTOR AS D ON M.
MID=D.MID
                        INNER JOIN PERSON AS P ON P.PID=C.PID) AS Q
                        INNER JOIN PERSON AS P ON P.PID = Q.DIRECTOR_ID
 )
                        WHERE DIRECTOR_NAME LIKE '%YASH%'
                                        GROUP BY ACTOR_ID,ACTOR_NAME,DI
RECTOR_NAME
                                        ORDER BY NO_MOVIES_YASH DESC))
 AS Y
                    ON E.ACTOR_ID = Y.ACTOR_ID
                    WHERE NO_MOVIES_YASH > NO_MOVIES_TOTAL
                    """,con).head(10)
```

Out[12]:

	ACTORS
0	Vikas Anand
1	Anupam Kher
2	Jagdish Raj
3	Amitabh Bachchan
4	Madan Puri
5	Iftekhar
6	Manmohan Krishna
7	Shashi Kapoor
8	Hema Malini
9	Rakhee Gulzar

Therefore we obtain a Total of 75 Actors who have Worked the Most with Yash Chopra as

Compared to any other Director.

Approach :-

1. Again, first we obtain all the Cast that Acted in a Particular Movie. In order to do so we Join the MOVIE Table with M_CAST Table basis the MID Column. (T1)
2. Now, the PID obtained in the Table T1 is basically obtained from the M_CAST Table and hence corresponds to the ACTOR_ID. Now basis the MID in the ALL_MOVIE_CAST Table we can join the M_DIRECTOR Table so as to obtain the DIRECTOR_ID for the Corresponding Movie. (T2)
3. Now we can Join the T2 Table with the PERSON Table so as to obtain the Corresponding Actor Name.
4. Similarly, from the same PERSON Table and joining basis the PID and the DIRECTOR_ID we can obtain the Director Name as well.
5. Now our Task is to obtain for each actor the number of Movies that he has Worked with Various Directors, and we obtain the Count of these Movies, which is stored in the column 'NO_MOVIES_TOTAL'
6. Similarly, for each Actor we also calculate the Number of Movies that he has worked with Yash Chopra and Count this Number as No_Movies_Yash in the Table T3. (Subquery Q)
7. Now we can Join the 2 Tables. Note that it is not necessary that every Actor in the first subquery Table has worked in a Movie with Yash Chopra. Therefore we carry out a Left Join.

Question 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 [13]: pd.read_sql_query("""SELECT DISTINCT NAME AS ACTOR_NAME FROM PERSON WHE
RE PID IN
                    (SELECT PID FROM M_CAST WHERE MID IN
                    (SELECT MID FROM M_CAST WHERE PID IN
                    (SELECT PID FROM PERSON WHERE PID IN
                    (SELECT PID FROM M_CAST WHERE MID IN
                    (SELECT MID FROM M_CAST WHERE PID IN
                    (SELECT PID FROM PERSON WHERE NAME LIKE '%SHAH_RUK
H%'))))))
                    AND NAME NOT LIKE '%SHAH_RUKH%'
                    AND PID NOT IN (SELECT PID FROM M_CAST WHERE MID IN

                                    (SELECT MID FROM M_CAST WHERE PID I
N
                                    (SELECT PID FROM PERSON WHERE NAME
 LIKE '%SHAH_RUKH%')))""",con).head(10)
```

Out[13]:

	ACTOR_NAME
0	Freida Pinto
1	Rohan Chand
2	Damian Young
3	Waris Ahluwalia
4	Caroline Christl Long
5	Rajeev Pahuja
6	Michelle Santiago
7	Alicia Vikander
8	Dominic West
9	Walton Goggins

We see that there are a total of 24308 Actors who have Not Worked Directly, but have worked with a Costar of Shah Rukh Khan.

Approach :-

1. First we find Shah Rukh Khan's PID and obtain all the Movies that Shah Rukh Khan has starred in.
2. After this, we find the Cast ie. PIDs of all the Movies that Shah Rukh Khan has starred in.
3. Now we find movies of all of these Co-Actors and their Corresponding Cast :- Actors who have not Worked Directly, but have only worked with a Costar of Shah Rukh Khan.