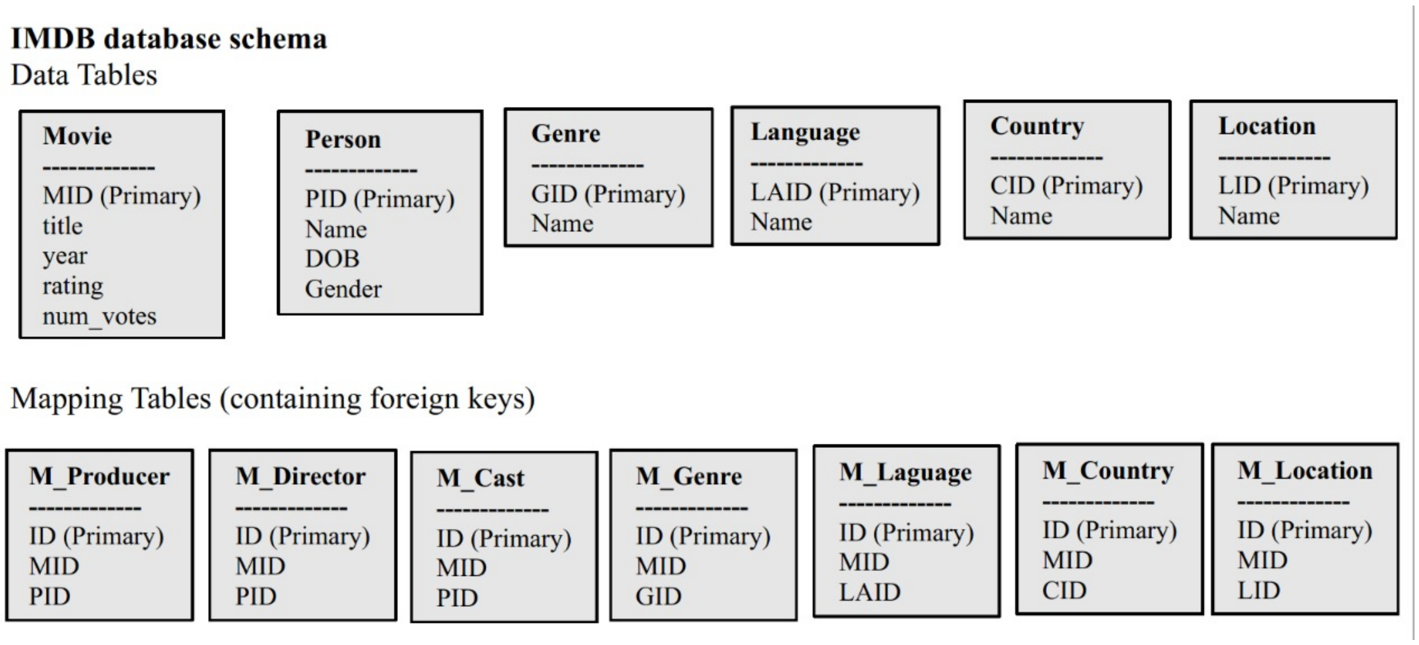
In [1]:

```
import pandas as pd
import sqlite3

from IPython.display import display,Image

Image(r'/content/db_schema.jpeg')
#conn = sqlite3.connect("/content/Db-IMDB-Assignment.db")
```

Out[1]:



IMDB database schema

Data Tables

Movie

MID (Primary)
title
year
rating
num_votes

Person

PID (Primary)
Name
DOB
Gender

Genre

GID (Primary)
Name

Language

LAID (Primary)
Name

Country

CID (Primary)
Name

Location

LID (Primary)
Name

Mapping Tables (containing foreign keys)

M_Producer

ID (Primary)
MID
PID

M_Director

ID (Primary)
MID
PID

M_Cast

ID (Primary)
MID
PID

M_Genre

ID (Primary)
MID
GID

M_Laguage

ID (Primary)
MID
LAID

M_Country

ID (Primary)
MID
CID

M_Location

ID (Primary)
MID
LID

In [2]:

```
conn = sqlite3.connect(r"/content/Db-IMDB-Assignment.db")
```

In [ ]:

```
tables = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM sqlite_master WHERE type='ta
ble'",conn)
tables = tables["Table_Name"].values.tolist()
```

In [ ]:

```
for table in tables:
    query = "PRAGMA TABLE_INFO({})".format(table)
    schema = pd.read_sql_query(query,conn)
    print("Schema of",table)
    display(schema)
    print("-"*100)
    print("\n")
```

```
Schema of Movie
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	title	TEXT	0	None	0
3	3	year	TEXT	0	None	0
4	4	rating	REAL	0	None	0
5	5	num_votes	INTEGER	0	None	0

```
----------------------------------------------------------------------------------------------------
```

```
-------------
```

```
Schema of Genre
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	Name	TEXT	0	None	0
2	2	GID	INTEGER	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of Language
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	Name	TEXT	0	None	0
2	2	LAID	INTEGER	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of Country
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	Name	TEXT	0	None	0
2	2	CID	INTEGER	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of Location
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	Name	TEXT	0	None	0
2	2	LID	INTEGER	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of M_Location
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	LID	REAL	0	None	0
3	3	ID	INTEGER	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of M_Country
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	CID	REAL	0	None	0
3	3	ID	INTEGER	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of M_Language
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	LAID	INTEGER	0	None	0
3	3	ID	INTEGER	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of M_Genre
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	GID	INTEGER	0	None	0
3	3	ID	INTEGER	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of Person
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	PID	TEXT	0	None	0
2	2	Name	TEXT	0	None	0
3	3	Gender	TEXT	0	None	0

```
---------------------------------------------------------------------------------------
-------------
```

```
Schema of M_Producer
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	PID	TEXT	0	None	0
3	3	ID	INTEGER	0	None	0

```
--------------------------------------------------------------------------------------------
-------------
```

```
Schema of M_Director
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	PID	TEXT	0	None	0
3	3	ID	INTEGER	0	None	0

```
--------------------------------------------------------------------------------------------
-------------
```

```
Schema of M_Cast
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	PID	TEXT	0	None	0
3	3	ID	INTEGER	0	None	0

```
--------------------------------------------------------------------------------------------
-------------
```

Q1 --- 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.

To determine whether a year is a leap year, follow these steps:

- **STEP-1:** If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
- **STEP-2:** If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
- **STEP-3:** If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
- **STEP-4:** The year is a leap year (it has 366 days).
- **STEP-5:** The year is not a leap year (it has 365 days).

Year 1900 is divisible by 4 and 100 but it is not divisible by 400, so it is not a leap year.

In [ ]:

```
Query1 = pd.read_sql_query("""SELECT Distinct p.Name Director_Name,a.title Movie_Title,a.
year Year
                          FROM Movie a , M_Director b,Genre c,M_Genre d,Person p
                          ON a.MID = d.MID AND a.MID = b.MID AND c.Name LIKE "%Comedy%"
AND b.PID=p.PID
                          AND a.year%4==0  group by p.Name,a.title""",conn)

print(Query1)
```

```
              Director_Name              Movie_Title    Year
0               A. Bhimsingh                    Aadmi    1968
1               A. Bhimsingh         Joroo Ka Ghulam    1972
2               A. Bhimsingh      Sadhu Aur Shaitaan    1968
3                   A. Muthu  Tera Jadoo Chal Gayaa    2000
4            A.R. Murugadoss                    Akira  I 2016
```

```
..                      ...                      ...   ...
940            Vishal Pandya            Wajah Tum Ho   2016
941  Vishnupant Govind Damle            Sant Tukaram   1936
942             Vivek Sharma              Bhoothnath   2008
943             Xavier Agudo         Train Station  I 2015
944          Y.V.S. Chowdary                Yuvaraju   2000

[945 rows x 3 columns]
```

Q2 --- List the names of all the actors who played in the movie 'Anand' (1971)

In [ ]:

```
Query2 = pd.read_sql_query("SELECT Distinct p.Name Actor_Name FROM Person p, M_Cast c \
                            ON TRIM(p.PID) = TRIM(c.PID) WHERE MID IN \
                           (SELECT MID from Movie m WHERE m.title = 'Anand' AND year=1971
)",conn)

print(Query2)
```

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

Q3 --- List all the actors who acted in a film before 1970 and in a film after 1990. (That is: < 1970 and > 1990.)

In [ ]:

```
Query3 = pd.read_sql_query('''SELECT p.name from person p where pid IN
                           (SELECT TRIM(pid) FROM m_cast where mid IN
                           (SELECT mid FROM movie WHERE year NOT BETWEEN 1970 AND 1990))'
'', conn)
Query3
```

Out[ ]:

	Name
0	Christian Bale
1	Cate Blanchett
2	Benedict Cumberbatch
3	Naomie Harris
4	Andy Serkis
...	...
29656	Hayley Cleghorn
29657	Nirvasha Jithoo

	Name
29658	Kamal Maharshi
29659	Mohini Manik
29660	Iqbal

29661 rows × 1 columns

Q4 --- 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 [ ]:

```
Query4 = pd.read_sql_query('Select distinct name Director_Name, Count(m.MID) Movie_Count 
\
                            from Person p JOIN M_director d On TRIM(p.PID) = TRIM(d.PID) 
\
                            JOIN Movie m ON d.MID=m.MID Group by p.Name Having COUNT(d.M
ID)>=10 order by Movie_Count desc',conn)

print(Query4)
```

```
                Director_Name  Movie_Count
0                David Dhawan           39
1                Mahesh Bhatt           36
2                Priyadarshan           30
3             Ram Gopal Varma           30
4                Vikram Bhatt           29
5         Hrishikesh Mukherjee          27
6                 Yash Chopra           21
7              Shakti Samanta           19
8             Basu Chatterjee           19
9                Subhash Ghai           18
10         Rama Rao Tatineni           17
11              Shyam Benegal           17
12    Abbas Alibhai Burmawalla          17
13               Raj N. Sippy           16
14             Manmohan Desai           16
15                     Gulzar           16
16                 Raj Kanwar           15
17           Mahesh Manjrekar           15
18          Rajkumar Santoshi           14
19                 Raj Khosla           14
20               Rahul Rawail           14
21                Indra Kumar           14
22                Vijay Anand           13
23              Rakesh Roshan           13
24        K. Raghavendra Rao            13
25               Harry Baweja           13
26                  Dev Anand           13
27             Anurag Kashyap           13
28    Ananth Narayan Mahadevan          13
29                Umesh Mehra           12
30             Satish Kaushik           12
31               Rohit Shetty           12
32              Prakash Mehra           12
33                Prakash Jha           12
34            Nagesh Kukunoor           12
35          Madhur Bhandarkar           12
36               Guddu Dhanoa           12
37                Anil Sharma           12
38               Anees Bazmee           12
39               Sanjay Gupta           11
40         Pramod Chakravorty           11
41              Nasir Hussain           11
42                 Mohit Suri           11
43                Ketan Mehta           11
44            Govind Nihalani           11
45            Pankaj Parashar           10
```

```
46       K. Muralimohana Rao          10
47                K. Bapaiah          10
48           Vishal Bhardwaj          10
49          Tigmanshu Dhulia          10
50             Sudhir Mishra          10
51                Raj Kapoor          10
52                N. Chandra          10
53               Mehul Kumar          10
54                J.P. Dutta          10
55             J. Om Prakash          10
56              Hansal Mehta          10
57                 Bimal Roy          10
```

Q5.a --- For each year, count the number of movies in that year that had only female actors.

In [ ]:

```
Query5a = pd.read_sql_query("SELECT DISTINCT year,COUNT(*) COUNT FROM Movie WHERE TRIM(MI
D) NOT IN ( \
                                SELECT DISTINCT TRIM(C.MID) FROM M_Cast C JOIN Person P
WHERE C.PID = P.PID \
                                AND P.Gender = 'Male') GROUP BY year; ",conn)
Query5a
```

Out[ ]:

	year	COUNT
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

Q5.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 [ ]:

```
Query5b = pd.read_sql_query("""
SELECT Fem_Mov.year Year, Fem_Mov.Count Movie_count,(Fem_Mov.Count*100.0)/Fem_Mov.Count P
ercentage FROM Movie M JOIN (
SELECT year,COUNT(*) as Count FROM Movie m WHERE m.MID NOT IN (
SELECT DISTINCT TRIM(C.MID) FROM M_Cast C JOIN Person P ON TRIM(C.PID)=TRIM(P.PID)
WHERE TRIM(P.Gender) = 'Male') GROUP BY year) AS Fem_Mov, (SELECT count(*) Count, m.year
FROM Movie m GROUP BY m.year) Total on Fem_Mov.year = Total.year
""",conn)
Query5b
```

Out[ ]:

	Year	Movie_count	Percentage
0	1939	1	100.0
1	1939	1	100.0
2	1939	1	100.0
3	1939	1	100.0
4	1939	1	100.0
...	...	...	...
13887	I 2018	1	100.0
13888	I 2018	1	100.0
13889	I 2018	1	100.0
13890	I 2018	1	100.0
13891	I 2018	1	100.0

13892 rows × 3 columns

Q6 --- 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 [ ]:

```
Query6 = pd.read_sql_query(" Select m.title, count(Distinct mc. PID) Cast_Size From M_Cas
t mc \
    INNER JOIN Movie m ON mc.MID = m.MID GROUP BY m.MID order by Cast_Size DESC",conn)
Query6
```

Out[ ]:

	title	Cast_Size
0	Ocean's Eight	238
1	Apaharan	233
2	Gold	215
3	My Name Is Khan	213
4	Captain America: Civil War	191
...	...	...
3468	Vaibhav Sethia: Don't	1
3469	Chaar Sahibzaade 2: Rise of Banda Singh Bahadur	1
3470	Subah Subah	1
3471	Return of Hanuman	1
3472	Kala Jigar	1

3473 rows × 2 columns

Q7 --- A decade is a sequence of 10 consecutive years.

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 [ ]:

```
Query7 = pd.read_sql_query("SELECT d.year Start, d.year+9 End, count(1) Movie_Count FROM
(SELECT DISTINCT(year) from Movie) \
                             d JOIN Movie m Where m.year >= Start and m.year<= End \
                            GROUP BY End HAVING End <= 2020 ORDER BY Movie_Count desc LI
MIT 1",conn)
Query7
```

Out[ ]:

	Start	End	Movie_Count
0	2008	2017	1126

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).

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

In [19]:

```
Query9 = pd.read_sql_query("""WITH Casting_Dir As (
SELECT DID, CID,num_of_Movies,
ROW_NUMBER() OVER( Partition BY CID Order By num_of_Movies DESC) Row_Num FROM (
SELECT TRIM(C.PID) As CID,TRIM(D.PID) As DID,COUNT(DISTINCT TRIM(C.MID)) As num_of_Movies

FROM M_Cast C JOIN M_Director D ON TRIM(C.MID)=TRIM(D.MID)
GROUP BY TRIM(C.PID),TRIM(D.PID)) As TEMP)

SELECT DISTINCT TRIM(Name) Actor_Name FROM Person p  WHERE PID IN (
SELECT DISTINCT CID FROM Casting_Dir As FD WHERE Row_Num = 1
AND DID IN (SELECT DISTINCT TRIM(PID) FROM Person WHERE NAME LIKE '%YASH%'))""",conn)
Query9
```

Out[19]:

	Actor_Name
0	Kulbir Badesron
1	Gurdas Maan
2	Parikshat Sahni
3	Waheeda Rehman
4	Taj Gill
...	...
221	Ramchandra
222	Sandow S. Sethi
223	Naval
224	Prem Sood
225	Ramlal Shyamlal

226 rows × 1 columns

Q10---- The Shahrukh number of an actor is the length of the shortest

Q10 --- 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 [10]:

```
Query10 = pd.read_sql_query("""WITH M_Cast_SRK AS (
  SELECT TRIM(MID) MID, TRIM(PID) PID
  FROM M_Cast
  WHERE TRIM(PID) = (
    SELECT TRIM(PID)
    FROM Person
    WHERE Name LIKE '%Shah Rukh Khan%'
  )
),
M_Cast_Others AS (
  SELECT TRIM(MID) MID, TRIM(PID) PID
  FROM M_Cast
  WHERE TRIM(MID) IN (
    SELECT MID
    FROM M_Cast_SRK
  ) AND TRIM(PID) NOT IN (
    SELECT PID
    FROM M_Cast_SRK
  )
)
SELECT TRIM(p.Name) Actor_Name
FROM Person p
WHERE PID IN (
  SELECT PID
  FROM M_Cast_Others
)""",conn)

Query10
```

Out[10]:

	Actor_Name
0	Raj Awasti
1	Alex Jaep
2	Celina Nessa
3	Elena Valdameri
4	Martavious Gayles
...	...
2377	Sheetal
2378	Pratibha Lonkar
2379	Dolon Roy
2380	Indira Mukherjee
2381	Choiti Ghosh

2382 rows × 1 columns