SQL Assignment

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

from IPython.display import display, HTML
```

In [2]:
```python
# Note that this is not the same db we have used in course videos, please download from this link
# https://drive.google.com/file/d/1O-1-L1DdNxEK6O6nG2jS31MbrMh-OnXM/view?usp=sharing
```

In [3]:
```python
conn = sqlite3.connect("Db-IMDB-Assignment.db")
```

Overview of all tables

In [4]:
```python
tables = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM sqlite_master WHERE type='table'",conn)
tables = tables["Table_Name"].values.tolist()
```

In [5]:
```python
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

Useful tips:

1. the year column in 'Movie' table, will have few chracters other than numbers which you need to be preprocessed, you need to get a substring of last 4 characters, its better if you convert it as int type, ex: CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)
2. For almost all the TEXT columns we have show, please try to remove trailing spaces, you need to use TRIM() function
3. When you are doing count(coulmn) it won't consider the "NULL" values, you might need to explore other alternatives like Count(*)

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 [6]:
```python
%%time
def grader_1(q1):
    q1_results  = pd.read_sql_query(q1,conn)
    print(q1_results.head(10))

query1 = '''select b.director , a.movies , a.y year from (select title movies ,year y, mid mi \
            from (select title , year, mid from movie where mid in (select mid from m_genre \
            where gid in(select gid from genre where trim(name) like '%Comedy%' )) and year%4==0)) \
            a, (select m_d.mid mid, b.name director from m_director m_d, (select * from \
            person where rowid in(select min(rowid) from person group by pid)) b where m_d.pid=b.pid) \
            b where a.mi=b.mid'''

grader_1(query1)
```

```
           director                           movies    year
0       Milap Zaveri                       Mastizaade    2016
1       Danny Leiner  Harold & Kumar Go to White Castle    2004
2     Anurag Kashyap                 Gangs of Wasseypur    2012
3       Frank Coraci        Around the World in 80 Days    2004
4      Griffin Dunne             The Accidental Husband    2008
5        Anurag Basu                             Barfi!    2012
6    Gurinder Chadha                  Bride & Prejudice    2004
7         Mike Judge    Beavis and Butt-Head Do America    1996
8       Abhinay Deo                          Blackmail  I 2018
9   Tarun Mansukhani                            Dostana    2008
Wall time: 251 ms
```

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

In [7]:
```python
%%time
def grader_2(q2):
    q2_results  = pd.read_sql_query(q2,conn)
    print(q2_results.head(10))
    assert (q2_results.shape == (17,1))

query2 ='''select name from person where trim(pid) in(select trim(pid) from m_cast \
           where mid in(select mid from movie where title='Anand'))'''


grader_2(query2)
```

```
                 Name
0   Amitabh Bachchan
1      Rajesh Khanna
2      Sumita Sanyal
3         Ramesh Deo
4          Seema Deo
5     Asit Kumar Sen
6         Dev Kishan
7       Atam Prakash
8      Lalita Kumari
9             Savita
Wall time: 53 ms
```

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

In [6]:
```python
%%time

def grader_3a(query_less_1970, query_more_1990):
    q3_a = pd.read_sql_query(query_less_1970,conn)
    q3_b = pd.read_sql_query(query_more_1990,conn)
    return (q3_a.shape == (4942,1)) and (q3_b.shape == (62570,1))

query_less_1970 = '''select distinct name , pid from person where trim(pid) in(select trim(pid) \
                     from m_cast where mid in(select mid from movie where year<1970)) '''

query_more_1990 = '''select distinct name , pid from person where trim(pid) in(select trim(pid) \
                     from m_cast where mid in(select mid from movie where year>1990)) '''
print(grader_3a(query_less_1970, query_more_1990))

# using the above two queries, you can find the answer to the given question
```

```
False
Wall time: 386 ms
```

In [7]:
```python
%%time
def grader_3(q3):
    q3_results  = pd.read_sql_query(q3,conn)
    print(q3_results.head(10))
    #assert (q3_results.shape == (300,1))

query3 = '''select distinct a.name from (select name , pid from person where trim(pid) in(select \
            trim(pid) from m_cast where mid in(select mid from movie where year<1970))) a , \
            (select name , pid from person where trim(pid) in(select trim(pid) from m_cast \
            where mid in(select mid from movie where year>1990))) b where a.pid = b.pid '''

grader_3(query3)
```

```
                 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
Wall time: 304 ms
```

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 [10]:
```python
%%time
def grader_4(q4):
    q4_results  = pd.read_sql_query(q4,conn)
    print(q4_results.head(10))
    assert (q4_results.shape == (58,2))

query_4a = '''select  p.name director , count(*) mov_count from (select * from person where \
              rowid in(select min(rowid) from person group by pid)) p, m_director m_d where \
              p.pid=m_d.pid group by p.name having count(*)>=10 order by count(*) desc'''

grader_4(query_4a)
```

```
                 director  mov_count
0           David Dhawan         39
1           Mahesh Bhatt         36
2         Ram Gopal Varma         30
3           Priyadarshan         30
4           Vikram Bhatt         29
5   Hrishikesh Mukherjee         27
6             Yash Chopra         21
7        Basu Chatterjee         19
8         Shakti Samanta         19
9            Subhash Ghai         18
Wall time: 167 ms
```

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

In [11]:
```python
%%time

# note that you don't need TRIM for person table

def grader_5aa(query_5aa):
    query_5aa = pd.read_sql_query(query_5aa,conn)
    print(query_5aa.head(10))
    return (query_5aa.shape == (8846,3))

query_5aa = '''select m.mid Mid, b.gends gend ,count(m.mid) count from movie m ,(select m_c.mid \
               mids, p.gender gends from m_cast m_c, (select * from person where rowid in(select \
               min(rowid) from person group by pid)) p where p.pid = trim(m_c.pid)) b \
               where b.mids = m.mid group by m.mid,b.gends  '''

print(grader_5aa(query_5aa))

def grader_5ab(query_5ab):
    query_5ab = pd.read_sql_query(query_5ab,conn)
    print(query_5ab.head(10))
    return (query_5ab.shape == (3469, 3))

query_5ab = '''select m.mid Mid, b.gends gend ,count(m.mid) count from movie m ,(select m_c.mid \
               mids, p.gender gends from m_cast m_c, (select * from person where rowid in(select \
               min(rowid) from person group by pid)) p where p.pid = trim(m_c.pid)) b \
               where b.mids = m.mid group by m.mid,b.gends having b.gends ='Male' '''

print(grader_5ab(query_5ab))


# using the above queries, you can write the answer to the given question
```

```
         Mid    gend  count
0  tt0021594    None      1
1  tt0021594  Female      3
2  tt0021594    Male      5
3  tt0026274    None      2
4  tt0026274  Female     11
5  tt0026274    Male      9
6  tt0027256    None      2
7  tt0027256  Female      5
8  tt0027256    Male      8
9  tt0028217  Female      3
True
         Mid  gend  count
0  tt0021594  Male      5
1  tt0026274  Male      9
2  tt0027256  Male      8
3  tt0028217  Male      7
4  tt0031580  Male     27
5  tt0033616  Male     46
6  tt0036077  Male     11
7  tt0038491  Male      7
8  tt0039654  Male      6
9  tt0040067  Male     10
True
Wall time: 1.34 s
```

In [12]:
```python
%%time
def grader_5a(q5a):
    q5a_results  = pd.read_sql_query(q5a,conn)
    print(q5a_results.head(10))
    assert (q5a_results.shape == (4,2))

query5a = '''select year, count(*) Female_Cast_Only_Movies from (select title, year from movie \
             where mid not in(select trim(mid) from m_cast where trim(pid) in(select trim(pid) \
             from person where trim(gender)='Male')) and mid in(select trim(mid) from m_cast \
             where trim(pid) in(select trim(pid) from person where trim(gender)='Female'))) \
             group by year'''

grader_5a(query5a)
```

```
     year  Female_Cast_Only_Movies
0    1939                        1
1    1999                        1
2    2000                        1
3  I 2018                        1
Wall time: 267 ms
```

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 [13]:
```python
%%time
def grader_5b(q5b):
    q5b_results  = pd.read_sql_query(q5b,conn)
    print(q5b_results.head(10))
    assert (q5b_results.shape == (4,3))

query5b = '''select b.year ,(100/a.tot_movie)*0.01 Percentage_Female_Only_Movie,a.tot_movie \
             from (select year, count(*) tot_movie from movie group by year order by year desc) a \
             inner join  (select  year, count(*) only_female from movie where mid not \
             in(select trim(mid) from m_cast where trim(pid) in(select trim(pid) from \
             (select * from person where rowid in(select min(rowid) from person group by pid)) \
             where trim(gender)='Male')) and mid in(select trim(mid) from m_cast where \
             trim(pid) in(select trim(pid) from (select * from person where rowid \
             in(select min(rowid) from person group by pid)) where trim(gender)='Female')) \
             group by year order by year desc) b on a.year=b.year order by a.tot_movie'''

grader_5b(query5b)
```

```
     year  Percentage_Female_Only_Movie  tot_movie
0    1939                          0.50          2
1  I 2018                          0.10         10
2    2000                          0.01         64
3    1999                          0.01         66
Wall time: 485 ms
```

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 [14]:
```python
%%time
def grader_6(q6):
    q6_results  = pd.read_sql_query(q6,conn)
    print(q6_results.head(10))
    assert (q6_results.shape == (3473, 2))

query6 ='''select m.title , a.no_of_cast from movie m, (select mid m_id ,count(*) no_of_cast  \
           from m_cast where mid in(select mid from movie) group by mid order by count(*) \
           desc) a where m.mid=a.m_id order by a.no_of_cast desc '''


grader_6(query6)
```

```
                         title  no_of_cast
0               Ocean's Eight         238
1                    Apaharan         233
2                        Gold         215
3              My Name Is Khan         213
4  Captain America: Civil War         191
5                    Geostorm         170
6                     Striker         165
7                        2012         154
8                      Pixels         144
9       Yamla Pagla Deewana 2         140
Wall time: 166 ms
```

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

For example, say in your database you have movie information starting from 1931.

the first decade is 1931, 1932, ..., 1940,

the second decade is 1932, 1933, ..., 1941 and so on.

Find the decade D with the largest number of films and the total number of films in D.

In [15]:
```python
%%time
def grader_7a(q7a):
    q7a_results  = pd.read_sql_query(q7a,conn)
    print(q7a_results.head(10))
    #assert (q7a_results.shape == (78, 2))

query7a = '''select year Movie_Year, count(*) Total_Movies from movie group by year'''

grader_7a(query7a)

# using the above query, you can write the answer to the given question
```

```
  Movie_Year  Total_Movies
0       1931             1
1       1936             3
2       1939             2
3       1941             1
4       1943             1
5       1946             2
6       1947             2
7       1948             3
8       1949             3
9       1950             2
Wall time: 10 ms
```

In [16]:
```python
%%time
def grader_7b(q7b):
    q7b_results  = pd.read_sql_query(q7b,conn)
    print(q7b_results.head(10))
    #assert (q7b_results.shape == (713, 4))

query7b ='''select a.Movie_Year ,a.Total_Movies, b.Movie_Year, b.Total_Movies from (select year \
            Movie_Year, count(*) Total_Movies from movie group by year) a, (select year Movie_Year, \
            count(*) Total_Movies from movie group by year) b where b.Movie_Year >= a.Movie_Year \
            and b.Movie_Year <a.Movie_Year+9 order by a.Movie_Year'''

grader_7b(query7b)
```

```
  Movie_Year  Total_Movies Movie_Year  Total_Movies
0       1931             1       1931             1
1       1931             1       1936             3
2       1931             1       1939             2
3       1936             3       1936             3
4       1936             3       1939             2
5       1936             3       1941             1
6       1936             3       1943             1
7       1939             2       1939             2
8       1939             2       1941             1
9       1939             2       1943             1
Wall time: 26 ms
```

In [5]:
```python
%%time
def grader_7(q7):
    q7_results  = pd.read_sql_query(q7,conn)
    print(q7_results.head(10))
    #assert (q7_results.shape == (1, 2))

query7 ='''select y.year start_decade, y.year+9 end_decade, count(*) total_number_of_films \
           from (select distinct(year) from movie ) y, movie m where m.year>= y.year and \
           m.year < y.year+10 group by  y.year order by count(*) desc limit 1 '''

grader_7(query7)
```

```
  start_decade  end_decade  total_number_of_films
0         2008        2017                   1126
Wall time: 852 ms
```

Q8 --- Find all the actors that made more movies with Yash Chopra than any other director.

In [5]:
```python
%%time
def grader_8a(q8a):
    q8a_results  = pd.read_sql_query(q8a,conn)
    print(q8a_results.head(10))
    print(q8a_results.shape)
    assert (q8a_results.shape == (73408, 3))

query8a = '''select a.actor,b.director, count(*) movies from (select p.pid actor, m.mid mids \
             from person p, movie m, m_cast m_c where trim(m_c.pid) = p.pid and m_c.mid = m.mid) a, \
             (select p.pid director, m.mid mids from person p, movie m, m_director m_d where \
             m_d.pid = p.pid and m_d.mid = m.mid) b where a.mids = b.mids group by a.actor,\
             b.director'''


grader_8a(query8a)
```

```
       actor   director  movies
0  nm0000002  nm0496746       1
1  nm0000027  nm0000180       1
2  nm0000039  nm0896533       1
3  nm0000042  nm0896533       1
4  nm0000047  nm0004292       1
5  nm0000073  nm0485943       1
6  nm0000076  nm0000229       1
7  nm0000092  nm0178997       1
8  nm0000093  nm0000269       1
9  nm0000096  nm0113819       1
(73408, 3)
Wall time: 914 ms
```

In [8]:
```python
%%time
def grader_8a(q8a):
    q8a_results  = pd.read_sql_query(q8a,conn)
    print(q8a_results.head(10))
    print(q8a_results.shape)
    assert (q8a_results.shape == (245, 2))

query8a = '''select p.name actors, a.movies no_of_movies from  (select act_pid ,dir_pid,directors, \
             movies from (select act_pid ,dir_pid,directors, \
             movies, dense_rank() over(partition by act_pid order by movies desc) r from \
             (select a.actor act_pid ,b.director dir_pid ,directors, count(*) movies from \
             (select p.pid actor, m.mid mids from person p, movie m, m_cast m_c where \
             trim(m_c.pid) = p.pid and m_c.mid = m.mid) a, (select p.pid director, p.name \
             directors, m.mid mids from person p, movie m, m_director m_d where \
             m_d.pid = p.pid and m_d.mid = m.mid) b where a.mids = b.mids group by a.actor,\
             b.director)) where r = 1 and trim(directors) = 'Yash Chopra') a , person p where\
             a.act_pid = p.pid order by a.movies desc '''


grader_8a(query8a)
```

```
              actors  no_of_movies
0        Jagdish Raj            11
1   Manmohan Krishna            10
2           Iftekhar             9
3      Shashi Kapoor             7
4     Waheeda Rehman             5
5      Rakhee Gulzar             5
6     Achala Sachdev             4
7        Neetu Singh             4
8           Ravikant             4
9    Parikshat Sahni             3
(245, 2)
Wall time: 1.03 s
```

Q9 --- 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 [21]:
```python
%%time
def grader_9(q9):
    q9_results  = pd.read_sql_query(q9,conn)
    print(q9_results.head(10))
    print(q9_results.shape)
    assert (q9_results.shape == (25698, 1))

query9 = '''select name Actor_Name from (select * from person where rowid in(select min(rowid) from \
            person group by pid)) where pid in(select trim(pid) pi from (select  pid from m_cast \
            where mid in(select mid from (select mid from m_cast where mid in(select mid from \
            m_cast where trim(pid) in(select pid from (select pid from person where pid \
            in(select pi from (select p.pid pi  from person p , (select mid, pid from (select \
            mid, pid from m_cast where mid in (select mid from m_cast m_c , person p where \
            trim(m_c.pid)=p.pid and trim(p.name)='Shah Rukh Khan'))) a where \
            trim(p.name)!='Shah Rukh Khan' and p.pid=trim(a.pid)))))) and mid not in(select mid \
            from m_cast m_c , person p where trim(m_c.pid)=p.pid and \
            trim(p.name)='Shah Rukh Khan'))))) and trim(name)!='Shah Rukh Khan' and pid not \
            in(select pi from (select p.pid pi , p.name from person p , (select mid, pid from \
            (select mid, pid from m_cast where mid in (select mid from m_cast m_c , person p \
            where trim(m_c.pid)=p.pid and trim(p.name)='Shah Rukh Khan'))) a where \
            trim(p.name)!='Shah Rukh Khan' and p.pid=trim(a.pid)))'''

grader_9(query9)
```

```
              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
(25698, 1)
Wall time: 1.65 s
```