SQL Assignment

In [29]:
```
import pandas as pd
import sqlite3

from IPython.display import display, HTML
```

In [30]:
```
conn = sqlite3.connect("C:/Users/91888/Desktop/DataSet/Assignment/SQL Assignment/Db-IMDB-Assignment.db")
```

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. ¶

In [33]:
```
%%time
def grader_1(q1):
    q1_results  = pd.read_sql_query(q1,conn)
    print(q1_results.head(10))
    assert (q1_results.shape == (232,3))

query1 = """
    SELECT m.Title,p.Name,m.year year
FROM Movie m JOIN
     M_director d
     ON m.MID = d.MID JOIN
     Person p
     ON d.PID = p.PID JOIN
     M_Genre mg
     ON m.MID = mg.MID JOIN
     Genre g
     ON g.GID = mg.GID
WHERE g.Name LIKE '%Comedy%'
AND
(CAST(SUBSTR(m.year,-4) AS UNSIGNED)%4 = 0 AND CAST(SUBSTR(m.year,-4) AS UNSIGNED)%100 != 0  OR CAST(SUBSTR(m.year,-4)
AS UNSIGNED)%400= 0)
"""
grader_1(query1)
```

```
                             title              Name  year
0                        Mastizaade       Milap Zaveri  2016
1  Harold & Kumar Go to White Castle       Danny Leiner  2004
2                 Gangs of Wasseypur     Anurag Kashyap  2012
3        Around the World in 80 Days       Frank Coraci  2004
4             The Accidental Husband     Griffin Dunne  2008
5                             Barfi!        Anurag Basu  2012
6                  Bride & Prejudice    Gurinder Chadha  2004
7    Beavis and Butt-Head Do America         Mike Judge  1996
8                            Dostana   Tarun Mansukhani  2008
9                     Kapoor & Sons       Shakun Batra  2016
Wall time: 157 ms
```

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

In [37]:
```
%%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 p.Name Actor_Names
            FROM Person p
            JOIN M_Cast m
            ON TRIM(p.PID) = TRIM(m.PID)
            WHERE MID IN
            (SELECT MID FROM Movie WHERE title= 'Anand' AND  CAST(SUBSTR(year,-4) AS Integer)=1971)
        """
grader_2(query2)
```

```
        Actor_Names
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
Wall time: 194 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 [31]:
```
%%time

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

query_less_1970 ="""
Select p.PID from Person p
inner join
(
    select trim(mc.PID) PD, mc.MID from M_cast mc
where mc.MID
in
(
    select mv.MID from Movie mv where CAST(SUBSTR(mv.year,-4) AS Integer)<1970
)
) r1
on r1.PD=p.PID
"""
query_more_1990 ="""
Select p.PID from Person p
inner join
(
    select trim(mc.PID) PD, mc.MID from M_cast mc
where mc.MID
in
(
    select mv.MID from Movie mv where CAST(SUBSTR(mv.year,-4) AS Integer)>1990
)
) r1
on r1.PD=p.PID """
print(grader_3a(query_less_1970, query_more_1990))
```

```
(4942, 1)
(62570, 1)
True
Wall time: 461 ms
```

In [39]:
```
%%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 =""" WITH
            A1 AS
            (
              SELECT DISTINCT TRIM(c.PID) PID
              FROM Movie m
              JOIN M_Cast c
              ON m.MID=c.MID
              WHERE
              CAST(SUBSTR(m.year,-4) AS UNSIGNED) > 1990
              ),
             A2 AS
            (
              SELECT DISTINCT TRIM(c.PID) PID
              FROM Movie m
              JOIN M_Cast c
              ON m.MID=c.MID
              WHERE
              CAST(SUBSTR(m.year,-4) AS UNSIGNED) < 1970
              )
    SELECT DISTINCT TRIM(Name) Actor_Name
    FROM Person
    WHERE TRIM(PID) IN A1 AND TRIM(PID) IN A2


 """
grader_3(query3)
```

```
         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
Wall time: 583 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 [40]:
```
%%time

def grader_4a(query_4a):
    query_4a = pd.read_sql_query(query_4a,conn)
    print(query_4a.head(10))
    return (query_4a.shape == (1462,2))

query_4a ="""SELECT d.PID Director_ID,COUNT(*) Movies_Count
             FROM M_Director d
             GROUP BY TRIM(d.PID)"""
print(grader_4a(query_4a))
```

```
  Director_ID  Movies_Count
0   nm0000180             1
1   nm0000187             1
2   nm0000229             1
3   nm0000269             1
4   nm0000386             1
5   nm0000487             2
6   nm0000965             1
7   nm0001060             1
8   nm0001162             1
9   nm0001241             1
True
Wall time: 16 ms
```

In [41]:
```
%%time
def grader_4(q4):
    q4_results  = pd.read_sql_query(q4,conn)
    print(q4_results.head(10))
    assert (q4_results.shape == (58,2))

query4 = """SELECT  TRIM(p.name) Director_Name, COUNT(*) movies_count
            from Person p
            JOIN M_Director d
            ON p.PID=TRIM(d.PID)
            GROUP BY TRIM(d.PID)
            having count(*)>=10
            ORDER BY movies_count DESC"""
grader_4(query4)
```

```
          Director_Name  movies_count
0          David Dhawan            39
1          Mahesh Bhatt            35
2          Priyadarshan            30
3       Ram Gopal Varma            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: 130 ms
```

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

In [42]:
```
%%time

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,p.Gender Gend,COUNT(*) count
              FROM Person p
              JOIN M_Cast m
              ON p.PID = TRIM(m.PID)
              GROUP BY m.MID,p.Gender"""
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,p.Gender Gend,COUNT(*) count
              FROM Person p
              JOIN M_Cast m
              ON p.PID = TRIM(m.PID)\
              AND p.Gender='Male'\
              GROUP BY m.MID,p.Gender"""

print(grader_5ab(query_5ab))
```

```
         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: 596 ms
```

In [43]:
```
%%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 CAST(SUBSTR(m.year,-4) AS Integer) Year,
            count(*) Female_Cast_Only_Movies
            FROM Movie m
            WHERE m.MID
            NOT IN
           (SELECT c.MID FROM Person p
            JOIN M_Cast c
            ON
            p.PID = TRIM(c.PID)
            AND p.Gender='Male')
            GROUP BY m.year"""
grader_5a(query5a)
```

```
   Year  Female_Cast_Only_Movies
0  1939                        1
1  1999                        1
2  2000                        1
3  2018                        1
Wall time: 189 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 [44]:
```
%%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 F.Year Year,
             CAST(F.Female_cast as real)/T.Total Percentage_Female_Only_Movie,
             T.Total Total_Movies
             FROM
(
    (SELECT CAST(SUBSTR(m.year,-4) AS Integer) Year,count(*) Female_cast
         FROM Movie m
          WHERE m.MID
          NOT IN
             (SELECT c.MID FROM Person p
             JOIN M_Cast c
             ON
             p.PID = TRIM(c.PID)
             AND p.Gender='Male')

     GROUP BY m.year

    ) F
    JOIN
        (SELECT CAST(SUBSTR(m.year,-4) AS Integer) Year,COUNT(*) Total
         FROM Movie m
         GROUP BY CAST(SUBSTR(m.year,-4) AS Integer)
         ) T

    ON F.Year=T.Year
)
         """

grader_5b(query5b)
```

	Year	Percentage_Female_Only_Movie	Total_Movies
0	1939	0.500000	2
1	1999	0.015152	66
2	2000	0.015625	64
3	2018	0.009615	104

```
Wall time: 289 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 [46]:
```
%%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,COUNT(DISTINCT(TRIM(c.PID))) count
          FROM Movie m
          JOIN M_Cast c
          ON m.MID=c.MID
          GROUP BY m.MID
          ORDER BY count DESC"""
grader_6(query6)
```

```
                        title  count
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: 362 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 [47]:
```
%%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 CAST(SUBSTR(m.year,-4) AS Integer) Movie_year,
             COUNT(*) Total_Movies
             FROM Movie m
             GROUP BY  Movie_year
             ORDER BY  Movie_year"""
grader_7a(query7a)
```

```
   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: 8 ms
```

In [48]:
```
%%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 CAST(SUBSTR(m.year,-4) AS Integer) Movie_year,
             COUNT(*) Total_Movies
             FROM Movie m
             GROUP BY  Movie_year
             ORDER BY  Movie_year
    )A

    JOIN

    (SELECT CAST(SUBSTR(m.year,-4) AS Integer) Movie_year,
             COUNT(*) Total_Movies
             FROM Movie m
             GROUP BY  Movie_year
             ORDER BY  Movie_year
    )B

  ON  B.Movie_year <=A.Movie_year+9 AND A.Movie_year<=B.Movie_year
  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: 16 ms
```

In [50]:
```
%%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 COUNT(*)  Decade_Movie_Count,
              Y.year Decade
             FROM
             (SELECT DISTINCT CAST(SUBSTR(m.year,-4) AS Integer) year
              FROM
              Movie m)Y
              JOIN
              Movie m
              ON
              CAST(SUBSTR(m.year,-4) AS Integer) >= Y.year
              AND
              CAST(SUBSTR(m.year,-4) AS Integer) < Y.year + 10
              GROUP BY
              Y.year
              ORDER BY
              COUNT(*)
              desc LIMIT 1"""
grader_7(query7)
```

```
   Decade_Movie_Count  Decade
0                1203    2008
Wall time: 108 ms
```

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

In [49]:
```
%%time
def grader_8a(q8a):
    q8a_results  = pd.read_sql_query(q8a,conn)
    print(q8a_results.head(10))
    assert (q8a_results.shape == (73408, 3))

query8a = """SELECT TRIM(c.PID) actor,TRIM(d.PID) director,COUNT(*) Movies
             FROM M_Director d
             JOIN M_Cast c
             ON d.MID=c.MID
             GROUP BY actor,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
Wall time: 671 ms
```

In [51]:
```
%%time

def grader_8(q8):
    q8_results  = pd.read_sql_query(q8,conn)
    print(q8_results.head(10))
    print(q8_results.shape)
    assert (q8_results.shape == (245, 2))

query8 ="""
With yashmoviesmore AS
(SELECT actor,director,Movies FROM
(
SELECT TRIM(c.PID) actor,TRIM(d.PID) director,COUNT(*) Movies
             FROM M_Director d
             JOIN M_Cast c
             ON d.MID=c.MID
             GROUP BY actor,director
   )

WHERE (actor,Movies) IN

(SELECT actor,max(Movies) FROM

   (
SELECT TRIM(c.PID) actor,TRIM(d.PID) director,COUNT(*) Movies
             FROM M_Director d
             JOIN M_Cast c
             ON d.MID=c.MID
             GROUP BY actor,director
   )
   GROUP BY actor
)
AND
director='nm0007181'


)

SELECT TRIM(p.Name) Actor_Name,yc.Movies count
FROM
Person p
```

```
JOIN yashmoviesmore yc
ON
yc.actor=p.PID
ORDER BY count DESC


"""
grader_8(query8)
```

```
         Actor_Name  count
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: 893 ms
```

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 [52]:
```
%%time
def grader_9a(q9a):
    q9a_results  = pd.read_sql_query(q9a,conn)
    print(q9a_results.head(10))
    print(q9a_results.shape)
    assert (q9a_results.shape == (2382, 1))

query9a = """
WITH
Shahrukh_Movies AS
 (
    SELECT
        DISTINCT
        TRIM(c.MID) MID,p.PID PID

    FROM
        M_Cast c JOIN
        Person p ON
        TRIM(c.PID)=p.PID
        WHERE p.PID IN (SELECT
        TRIM(p.PID) PID
    FROM
        Person p
    WHERE
        Trim(p.Name) like '%Shah%rukh%Khan%')


)

 SELECT
        DISTINCT
        TRIM(c.PID) PID
    FROM
        M_Cast c JOIN
        Shahrukh_Movies S1M
    ON
        TRIM(c.MID) = S1M.MID AND
        TRIM(c.PID) <> S1M.PID

"""
grader_9a(query9a)
```

```
        PID
0  nm0004418
1  nm1995953
2  nm2778261
3  nm0631373
4  nm0241935
5  nm0792116
6  nm1300111
7  nm0196375
8  nm1464837
9  nm2868019
(2382, 1)
Wall time: 255 ms
```

In [53]:
```
%%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 = """
WITH
Shahrukh_Movies AS
 (
    SELECT
        DISTINCT
        TRIM(c.MID) MID, TRIM(c.PID) PID

    FROM
        M_Cast c JOIN
        Person p ON
        TRIM(c.PID)=p.PID
        WHERE TRIM(c.PID) IN (SELECT
        TRIM(p.PID) PID
    FROM
        Person p
    WHERE
        Trim(p.Name) like '%Shah%rukh%Khan%')


),

Shahrukh_1actorMovies AS
(  SELECT
        DISTINCT
        TRIM(c.PID) PID,TRIM(c.MID) MID
    FROM
        M_Cast c JOIN
        Shahrukh_Movies S1M
    ON
        TRIM(c.MID) = S1M.MID

),
```

```
Shahrukh_2Movies AS
( SELECT
        DISTINCT
        TRIM(c.MID) MID,
        S1A.PID PID
    FROM
        M_Cast c JOIN
        Shahrukh_1actorMovies S1A
        ON
        TRIM(c.PID) = S1A.PID
        AND
        TRIM(c.MID) <> S1A.MID
 ),

 Shahrukh_2actor AS
 (
SELECT
        DISTINCT
        TRIM(c.PID) PID
    FROM
        M_Cast c JOIN
        Shahrukh_2Movies S2M
    ON
        TRIM(c.MID) = S2M.MID


),

Only_S2Actor AS
(
SELECT S2A.PID PID
FROM
   Shahrukh_2actor S2A
   LEFT JOIN
   Shahrukh_1actorMovies S1A
   ON
   S2A.PID = S1A.PID
   WHERE S1A.PID IS NULL
 )

SELECT TRIM(p.Name) Actor_Name
FROM Person p
```

```
WHERE TRIM(p.PID) IN
(SELECT a2.PID FROM Only_S2Actor a2 )


"""
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: 1.46 s
```