Assignment18

May 9, 2022

1 Assignment 18

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

        from IPython.display import display, HTML
```

```
In [42]: # 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 [2]: conn = sqlite3.connect("Db-IMDB-Assignment.db")
```

Overview of all tables

```
In [44]: tables = pd.read_sql_query("SELECT name AS 'Table_Name' FROM sqlite_master WHERE \
                                     type='table' ",conn)
         tables = tables["Table_Name"].values.tolist()
```

```
In [45]: 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

1.1 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(*)

1.2 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:

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

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

```
In [46]: %%time
         def grader_1(q1):
             q1_results  = pd.read_sql_query(q1,conn)
             print(q1_results.head(10))
             return (q1_results.shape == (232,3))

         query1 = """
                     SELECT
                            p.name AS 'Director_Name',
                            m.title AS 'Movie_Name',
                            m.year AS 'Year'

                     FROM movie m,
                          m_genre mg,
                          genre g,
```

```
                    person p,
                    m_director md
                WHERE
                    mg.mid=m.mid
                AND mg.gid=g.gid
                AND md.mid=m.mid
                AND md.pid=p.pid
                AND g.name LIKE '%Comedy%'
                AND (CASE WHEN CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)%4==0
                     THEN
                        CASE WHEN CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)%100==0
                        THEN
                         CASE WHEN CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)%400==0
                          THEN 1
                          END
                         ELSE
                           1
                         END
                       END )=1

                """
        print(grader_1(query1))
```

```
       Director_Name                         Movie_Name  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   Tarun Mansukhani                           Dostana  2008
9       Shakun Batra                      Kapoor & Sons  2016
True
Wall time: 183 ms
```

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

```
In [47]: %%time
         def grader_2(q2):
             q2_results  = pd.read_sql_query(q2,conn)
             print(q2_results.head(10))
             return (q2_results.shape == (17,1))


         query2 = """SELECT
```

```
                    p.name AS 'Actors'
                FROM
                    movie m,
                    person p,
                    m_cast mc
                WHERE
                    mc.mid=m.mid
                AND TRIM(mc.pid)=p.pid
                AND m.title='Anand'
                """
        print(grader_2(query2))
```

```
              Actors
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
True
Wall time: 455 ms
```

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

In [48]:
```
%%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 MID from Movie where CAST(SUBSTR(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 MID from Movie where CAST(SUBSTR(year,-4) AS Integer)>1990
        )
        ) r1
        on r1.PD=p.PID """
        print(grader_3a(query_less_1970, query_more_1990))

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

```
(4942, 1)
(62570, 1)
True
Wall time: 742 ms
```

```
In [64]: %%time
        def grader_3(q3):
            q3_results  = pd.read_sql_query(q3,conn)
            print(q3_results.shape)
            assert (q3_results.shape == (300,1))

        query3 = """
        SELECT DISTINCT a.name
        FROM
                (SELECT DISTINCT p.pid,p.name FROM person p
                INNER JOIN
                    (
                        SELECT TRIM(mc.pid) pd, mc.mid FROM m_cast mc
                        WHERE mc.mid
                        IN
                        (
                        SELECT mid FROM movie WHERE CAST(SUBSTR(year,-4) AS INTEGER)<1970
                        )
                    ) r1
                    ON r1.pd=p.pid
            )a
            WHERE a.pid IN
```

```
                (SELECT DISTINCT p.pid FROM person p
                INNER JOIN
                    (
                        SELECT TRIM(mc.pid) pd, mc.mid FROM m_cast mc
                        WHERE mc.mid
                        IN
                        (
                        SELECT mid FROM movie WHERE CAST(SUBSTR(year,-4) AS INTEGER)>1990
                        )
                    ) r1
                   ON r1.pd=p.pid
            )
        """


        grader_3(query3)
```

```
(300, 1)
Wall time: 250 ms
```

1.5 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 [50]: %%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))

        #*** Write a query, which will return all the directors(id's) along
        #with the number of movies they directed ***
        query_4a ="""SELECT
                        md.pid AS Director_id,
                        count(md.mid) AS Movie_count
                     FROM
                         m_director md
                     GROUP BY md.pid
                      """
        print(grader_4a(query_4a))

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

```
  Director_id  Movie_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: 21.2 ms
```

```
In [51]: %%time
         def grader_4(q4):
             q4_results  = pd.read_sql_query(q4,conn)
             print(q4_results.head(10))
             assert (q4_results.shape == (58,2))

         #*** Write your query for the question 4 ***
         query4 = """
                     SELECT DISTINCT p.name AS Director,
                                     md.movie_count AS No_of_movies
                     FROM
                         person p
                     INNER JOIN
                         (
                             SELECT
                                 md.pid,
                                 COUNT(md.mid) AS movie_count
                              FROM
                                  m_director md
                              GROUP BY
                                  md.pid
                              HAVING
                                  movie_count >=10
                          ) md
                      ON
                          md.pid=p.pid
                      ORDER BY
                          movie_count desc
                     """
         grader_4(query4)
```

```
              Director  No_of_movies
0          David Dhawan            39
1          Mahesh Bhatt            35
2       Ram Gopal Varma            30
3          Priyadarshan            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
Wall time: 54 ms
```

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

```
In [52]: %%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))

         # Write your query that will get movie id, and number of people for each gender
         query_5aa =""" SELECT
                              m.mid,
                              p.gender,
                              count(p.pid) Gender_count
                        FROM
                            movie m,
                            m_cast mc,
                            person p
                         WHERE
                            TRIM(mc.pid)=p.pid
                         AND mc.mid=m.mid
                         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))

         #Write your query that will have at least one male actor
         #try to use query that you have written above
```

```
        query_5ab =""" SELECT
                            m.mid,
                            p.gender,
                            count(p.pid) as Male_count
                        FROM
                           movie m,
                           m_cast mc,
                           person p
                         WHERE
                           TRIM(mc.pid)=p.pid
                         AND p.gender='Male'
                         AND mc.mid=m.mid
                         GROUP BY
                            m.mid,
                            p.gender
                         HAVING Male_count>=1


        """

        print(grader_5ab(query_5ab))
```

```
        MID  Gender  Gender_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 Gender  Male_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: 1.9 s
```

```
In [53]: %%time
         def grader_5a(q5a):
             q5a_results  = pd.read_sql_query(q5a,conn)
             print(q5a_results.head(10))
             assert (q5a_results.shape == (4,2))

         #*** Write your query for the question 5a ***
         query5a = """ SELECT SUBSTR(TRIM(year),-4) AS Year,
                              COUNT(mid) Movie_count
                       FROM
                           movie
                       WHERE mid NOT IN
                           (SELECT
                               mid
                            FROM
                                (SELECT
                                   m.mid,
                                   p.gender,
                                   COUNT(p.pid) AS gender_count
                                 FROM
                                   movie m,
                                   m_cast mc,
                                   person p
                                 WHERE
                                   TRIM(mc.pid)=p.pid
                                 AND p.gender='Male'
                                 AND mc.mid=m.mid
                                GROUP BY
                                   m.mid,
                                   p.gender
                               HAVING
                                   gender_count>=1)
                           )GROUP BY SUBSTR(TRIM(year),-4)
                    """
         grader_5a(query5a)

   Year  Movie_count
0  1939            1
1  1999            1
2  2000            1
3  2018            1
Wall time: 727 ms
```

1.7 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 [54]: %%time
         def grader_5b(q5b):
             q5b_results  = pd.read_sql_query(q5b,conn)
             print(q5b_results.head(10))
             assert (q5b_results.shape == (4,3))

         #*** Write your query for the question 5b ***
         query5b = """
             SELECT
             r1.Year,
             CAST(Movie_count AS REAL) * 100/COUNT(m.mid) AS percentage_with_female_actors,
             COUNT(m.mid) Movie_count
             FROM
                      (
                         SELECT SUBSTR(TRIM(year),-4) AS year,
                             COUNT(mid) Movie_count
                         FROM
                             movie
                         WHERE mid NOT IN
                             (SELECT
                                 mid
                              FROM
                                  (SELECT
                                     m.mid,
                                     p.gender,
                                     COUNT(p.pid) AS gender_count
                                   FROM
                                     movie m,
                                     m_cast mc,
                                     person p
                                   WHERE
                                     TRIM(mc.pid)=p.pid
                                   AND p.gender='Male'
                                   AND mc.mid=m.mid
                                  GROUP BY
                                     m.mid,
                                     p.gender
                                 HAVING
                                     gender_count>=1)
                             )GROUP BY SUBSTR(TRIM(year),-4)
                     ) r1 INNER JOIN movie m ON r1.year=m.year
                         GROUP BY r1.year,Movie_count
```

```
                    """
        grader_5b(query5b)
```

```
   year  percentage_with_female_actors  Movie_count
0  1939                      50.000000            2
1  1999                       1.515152           66
2  2000                       1.562500           64
3  2018                       1.075269           93
Wall time: 774 ms
```

1.8 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 [55]: %%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 p.pid) AS Cast_size

                      FROM
                              movie m,
                              m_cast mc,
                              person p
                       WHERE
                              TRIM(mc.pid)=p.pid
                       AND mc.mid=m.mid
                       GROUP BY m.mid
                       ORDER BY Cast_size DESC

                 """
         grader_6(query6)
```

```
                        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
5                     Geostorm        170
6                      Striker        165
7                         2012        154
```

```
8                     Pixels        144
9       Yamla Pagla Deewana 2       140
Wall time: 900 ms
```

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

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

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

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

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

```
In [56]: %%time
         def grader_7a(q7a):
             q7a_results  = pd.read_sql_query(q7a,conn)
             print(q7a_results.head(10))
             assert (q7a_results.shape == (78, 2))

         #*** Write a query that computes number of movies in each year ***

         query7a = """SELECT
                         SUBSTR(TRIM(year),-4) AS year,
                         COUNT(mid) AS no_of_movies
                      FROM
                          movie
                      GROUP BY
                          SUBSTR(TRIM(year),-4)
                     """
         grader_7a(query7a)

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

```
   year  no_of_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: 15.5 ms
```

```
In [57]: %%time
         def grader_7b(q7b):
```

```
    q7b_results  = pd.read_sql_query(q7b,conn)
    print(q7b_results.head(10))
    assert (q7b_results.shape == (713, 4))


#Write a query that will do joining of the above table(7a) with itself
#such that you will join with only rows if the second tables year is <= current_year+9
#and more than or equal current_year

query7b = """
            SELECT * FROM
                (SELECT
                    SUBSTR(TRIM(year),-4) AS year,
                    COUNT(mid) AS no_of_movies
                 FROM
                     movie
                 GROUP BY
                     SUBSTR(TRIM(year),-4)) query1,
                  (SELECT
                    SUBSTR(TRIM(year),-4) AS year,
                    COUNT(mid) AS no_of_movies
                 FROM
                     movie
                 GROUP BY
                     SUBSTR(TRIM(year),-4)) query2
               WHERE CAST(query2.year AS INTEGER)<=CAST(query1.year AS INTEGER)+9
               AND CAST(query2.year AS INTEGER)>=CAST(query1.year AS INTEGER)
              """
grader_7b(query7b)
# if you see the below results the first movie year is less than 2nd movie year and
# 2nd movie year is less or equal to the first movie year+9

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

	year	no_of_movies	year	no_of_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: 29.3 ms
```

```
In [58]: %%time
```

```
        def grader_7(q7):
            q7_results  = pd.read_sql_query(q7,conn)
            print(q7_results.head(10))
            assert (q7_results.shape == (1, 2))

        #Write a query that will return the decade that has maximum number of movies
        query7 = """
                    SELECT query1.year ||'-'|| MAX(query2.year) AS Decade,
                            SUM(query2.no_of_movies) AS No_of_movies FROM
                        (SELECT
                            SUBSTR(TRIM(year),-4) AS year,
                            COUNT(mid) AS no_of_movies
                         FROM
                             movie
                         GROUP BY
                             SUBSTR(TRIM(year),-4)) query1,
                          (SELECT
                            SUBSTR(TRIM(year),-4) AS year,
                            COUNT(mid) AS no_of_movies
                         FROM
                             movie
                         GROUP BY
                             SUBSTR(TRIM(year),-4)) query2
                       WHERE CAST(query2.year AS INTEGER)<=CAST(query1.year AS INTEGER)+9
                       AND CAST(query2.year AS INTEGER)>=CAST(query1.year AS INTEGER)

                       GROUP BY query1.year
                        ORDER BY SUM(query2.no_of_movies) DESC
                        LIMIT 1

                 """
        grader_7(query7)
        # if you check the output we are printinng all the year in that decade,
        #its fine you can print 2008 or 2008-2017
```

```
      Decade  No_of_movies
0  2008-2017          1203
Wall time: 22.4 ms
```

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

```
In [59]: %%time
        def grader_8a(q8a):
            q8a_results  = pd.read_sql_query(q8a,conn)
            print(q8a_results.head(10))
            assert (q8a_results.shape == (73408, 3))
```

```
#Write a query that will results in number of movies actor-director worked together
query8a = """
            SELECT
                query1.name AS Director,
                query2.name AS Actor,
                COUNT(query1.mid) AS No_of_movies
            FROM
                (SELECT
                    m.mid,
                    p.pid,
                    p.name
                 FROM
                    movie m,
                    m_director md,
                    person p
                 WHERE
                    m.mid=md.mid
                 AND p.pid=TRIM(md.pid)
                 )query1,
                (SELECT
                    m.mid,
                    p.pid,
                    p.name
                 FROM
                    movie m,
                    m_cast mc,
                    person p
                 WHERE
                    m.mid=mc.mid
                 AND p.pid=TRIM(mc.pid)
                 )query2

                WHERE query1.mid=query2.mid
                GROUP BY query1.pid,query2.pid
                """
grader_8a(query8a)

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

	Director	Actor	No_of_movies
0	David Lean	Alec Guinness	1
1	David Lean	Judy Davis	1
2	David Lean	Peggy Ashcroft	1
3	David Lean	Saeed Jaffrey	1
4	David Lean	Paul Anil	1
5	David Lean	Mohammed Ashiq	1
6	David Lean	Victor Banerjee	1

```
7  David Lean    Adam Blackwood             1
8  David Lean      Phyllis Bose             1
9  David Lean         Ishaq Bux             1
Wall time: 1.53 s
```

In [60]: %%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))

        # *** Write a query that answers the 8th question ***
        query8 = """SELECT actor,
                           movie_count
                    FROM
                        (SELECT
                               actorid,
                               directorid,
                               actor,
                               director,
                               count(*) AS movie_count
                         FROM
                             (SELECT DISTINCT
                                        m.mid,
                                        p.pid AS directorid,
                                        p.name AS director
                             FROM
                                        movie m,
                                        m_director md,
                                        person p
                             WHERE
                                    m.mid=TRIM(md.mid)
                             AND    p.pid=TRIM(md.pid)
                             )query1,
                            (SELECT distinct
                                        m.mid,
                                        p.pid AS actorid,
                                        p.name AS actor
                             FROM
                                        movie m,
                                        m_cast mc,
                                        person p
                             WHERE
                                    m.mid=TRIM(mc.mid)
                             AND    p.pid=TRIM(mc.pid)
```

```
                        )query2
                    WHERE
                    query1.mid=query2.mid
                    GROUP BY actorid,directorid)
                WHERE (actorid,movie_count) IN
                 (SELECT actorid,max_count
                 FROM
                    (SELECT actorid, MAX(movie_count) AS max_count
                     FROM
                         (SELECT actorid,directorid,
                         actor,director,count(*) AS movie_count
                         FROM (SELECT DISTINCT
                                          m.mid,
                                          p.pid AS directorid,
                                          p.name AS director
                               FROM
                                     movie m,
                                     m_director md,
                                     person p
                               WHERE
                                     m.mid=TRIM(md.mid)
                               AND   p.pid=TRIM(md.pid)
                              )query1,
                             (SELECT DISTINCT
                                           m.mid,
                                           p.pid AS actorid,
                                           p.name AS actor
                              FROM
                                     movie m,
                                     m_cast mc,
                                     person p
                              WHERE
                                     m.mid=TRIM(mc.mid)
                              AND    p.pid=TRIM(mc.pid)
                              )query2
                         WHERE query1.mid=query2.mid GROUP BY actorid,directorid)
                         GROUP BY actorid

                     )
                )
                AND director like '%Yash%Chopra%'"""
```

grader_8(query8)

	actor	movie_count
0	Shashi Kapoor	7
1	Yash Chopra	2
2	Akhtar-Ul-Iman	1

```
3         Murad Ali            1
4      Badri Prasad            1
5        Saira Banu            1
6        Raj Bharti            1
7     Ashwini Bhave            1
8   Andrew Bicknell            1
9    Paul Blackwell            1
(245, 2)
Wall time: 4.12 s
```

1.10 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 [61]: %%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))

         #*** Write a query that answers the 9th question ***
         query9a = """SELECT DISTINCT
                                          p.pid AS Actors_with_SRK
                              FROM
                                          movie m,
                                          m_cast mc,
                                          person p
                                   WHERE
                                          m.mid=TRIM(mc.mid)
                                   AND    p.pid=TRIM(mc.pid)
                                   AND    TRIM(p.name) <>'Shah Rukh Khan'
                                   AND m.mid IN
                                  (SELECT DISTINCT
                                      m.mid
                                   FROM
                                          movie m,
                                          m_cast mc,
                                          person p
                                   WHERE
                                          m.mid=TRIM(mc.mid)
                                   AND    p.pid=TRIM(mc.pid)
                                   AND    TRIM(p.name) ='Shah Rukh Khan')
                         """
```

```
        grader_9a(query9a)
        # using the above query, you can write the answer to the given question

        # selecting actors who acted with srk (S1)
        # selecting all movies where S1 actors acted, this forms S2 movies list
        # selecting all actors who acted in S2 movies,
        #this gives us S2 actors along with S1 actors
        # removing S1 actors from the combined list of S1 & S2 actors,
        #so that we get only S2 actors
```

```
  Actors_with_SRK
0       nm0004418
1       nm1995953
2       nm2778261
3       nm0631373
4       nm0241935
5       nm0792116
6       nm1300111
7       nm0196375
8       nm1464837
9       nm2868019
(2382, 1)
Wall time: 822 ms
```

```
In [6]: %%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))
        #*** Write a query that answers the 9th question ***
        query9 = """ SELECT name AS SRK_no_2_actors FROM (SELECT DISTINCT
                              p.pid ,p.name
                      FROM
                              movie m,
                              m_cast mc,
                              person p
                      WHERE
                              m.mid=TRIM(mc.mid)
                      AND     p.pid=TRIM(mc.pid)
                      AND     m.mid IN
                                      (SELECT DISTINCT
                                              m.mid
                                       FROM
                                           movie m,
                                           m_cast mc,
                                           person p
```

```
                    WHERE
                        m.mid=TRIM(mc.mid)
                    AND p.pid=TRIM(mc.pid)
                    AND p.pid IN

                       (SELECT DISTINCT
                               p.pid
                        FROM
                            movie m,
                            m_cast mc,
                            person p
                        WHERE
                            m.mid=TRIM(mc.mid)
                        AND p.pid=TRIM(mc.pid)
                        AND TRIM(p.name) <>'Shah Rukh Khan'
                        AND m.mid IN
                               (SELECT DISTINCT
                                       m.mid
                                FROM
                                       movie m,
                                       m_cast mc,
                                       person p
                                WHERE
                                       m.mid=TRIM(mc.mid)
                                AND    p.pid=TRIM(mc.pid)
                                AND    TRIM(p.name) ='Shah Rukh Khan')))
AND p.pid NOT IN
                    (SELECT DISTINCT
                           p.pid AS Actors_with_SRK
                    FROM
                           movie m,
                           m_cast mc,
                           person p
                    WHERE
                           m.mid=TRIM(mc.mid)
                    AND    p.pid=TRIM(mc.pid)
                    AND    TRIM(p.name) <>'Shah Rukh Khan'
                    AND m.mid IN
                           (SELECT DISTINCT
                                   m.mid
                            FROM
                                  movie m,
                                  m_cast mc,
                                  person p
                            WHERE
                                  m.mid=TRIM(mc.mid)
                            AND    p.pid=TRIM(mc.pid)
                            AND    TRIM(p.name) ='Shah Rukh Khan')
```

```
                                                        )
                      AND TRIM(p.name) <>'Shah Rukh Khan')"""
        grader_9(query9)

         SRK_no_2_actors
0        Alicia Vikander
1           Dominic West
2         Walton Goggins
3              Daniel Wu
4   Kristin Scott Thomas
5           Derek Jacobi
6     Alexandre Willaume
7           Tamer Burjaq
8         Adrian Collins
9         Keenan Arrison
(25698, 1)
Wall time: 979 ms
```