SQL Assignment

```
import pandas as pd
import sqlite3
import re

from IPython.display import display, HTML

# 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/10-1-L1DdNxEK6O6nG2jS31MbrMh-OnXM/
view?usp=sharing

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

```
<sqlite3.Connection at 0x7f4cb53ed810>
```

Overview of all tables

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

tables
```

```
['Movie',
 'Genre',
 'Language',
 'Country',
 'Location',
 'M_Location',
 'M_Country',
 'M_Language',
 'M_Genre',
 'Person',
 'M_Producer',
 'M_Director',
 'M_Cast']
```

```
#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")
```

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.

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

```
%%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 p.name,m.title,m.year from Movie m join M_Director
md \
             on m.MID = md.MID join Person p on md.PID = trim(P.PID)
join M_Genre mg\
             on m.MID = mg.Mid join Genre g on mg.GID = g.GID\
             and g.Name like '%Comedy%' and CAST(SUBSTR(TRIM(m.year),-
4)as INTEGER)%4=0 '''
grader_1(query1)

                Name                              title  year
0      Griffin Dunne             The Accidental Husband  2008
1   Mahesh Manjrekar  Jis Desh Mein Ganga Rehta Hain  2000
2            Madonna                  Filth and Wisdom  2008
3    Gurinder Chadha                 Bride & Prejudice  2004
4       Frank Coraci        Around the World in 80 Days  2004
5   Tarun Mansukhani                            Dostana  2008
6        Lekh Tandon                   Jhuk Gaya Aasman  1968
7     S.S. Rajamouli                               Eega  2012
8      Jugal Hansraj                     Roadside Romeo  2008
9         Mike Judge  Beavis and Butt-Head Do America  1996
CPU times: user 44.4 ms, sys: 4.2 ms, total: 48.6 ms
Wall time: 49.8 ms
```

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

```
%%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 from Person p join M_Cast mc on
trim(p.PID)=trim(mc.PID)
             join Movie m on mc.MID=m.MID\
             and m.title='Anand' and m.year=1971 """
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
CPU times: user 313 ms, sys: 8.24 ms, total: 321 ms
Wall time: 318 ms
```

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

```
%%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))

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

```
(4942, 1)
(62570, 1)
True
CPU times: user 269 ms, sys: 9.34 ms, total: 279 ms
Wall time: 285 ms
```

```
%%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 Name as Actor from Person where PID in\
(select TRIM(PID) from M_Cast mc where mc.MID in\
(select MID from Movie m1 where cast(substr(m1.year,-4)as
INTEGER)>1990)and \
PID in(select PID from M_Cast where MID in\
(select MID from Movie m2 where cast(substr(m2.year,-4)as
INTEGER)<1970))) """
grader_3(query3)
```

```
               Actor
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
CPU times: user 122 ms, sys: 4.69 ms, total: 127 ms
Wall time: 129 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.

```
%%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,count(*) as count from M_Director md
           join Person p on md.PID = p.PID
           group by md.PID'''
print(grader_4a(query_4a))

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

```
         PID  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
CPU times: user 66.8 ms, sys: 567 µs, total: 67.4 ms
Wall time: 71.7 ms
```

```
%%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 p.Name,count(*) as no_of_movies from person p
            join M_Director md on p.PID=md.PID\
            group by p.Name
            having count(*)>=10
            order by count(*) desc'''
grader_4(query4)
```

```
                    Name  no_of_movies
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
CPU times: user 41.1 ms, sys: 2.67 ms, total: 43.8 ms
Wall time: 49.7 ms
```

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

```
%%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 mc.MID,p.Gender,count(*) as no_of_movies from
person p
               join M_cast mc on p.PID=trim(mc.PID)\
               group by mc.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 mc.MID,p.Gender,count(*) as no_of_movies from
person p
              join M_cast mc on p.PID = trim(mc.PID)
              group by mc.MID,p.Gender
              having p.Gender='Male' and count(*)>=1"""

print(grader_5ab(query_5ab))


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

         MID  Gender  no_of_movies
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  no_of_movies
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
CPU times: user 308 ms, sys: 14.6 ms, total: 323 ms
Wall time: 325 ms
```

```
%%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 = """ with
              male_MIDS as(select mc.MID from M_cast mc, person p
where trim(mc.PID)=p.PID and p.Gender='Male'),
              female_MIDS as(select m.MID from Movie m,M_cast mc where
trim(mc.MID)=m.MID and trim(m.MID) not in(select MID from male_MIDS))

              select substr(year,-4) year,count(*) count from Movie
              where trim(MID) in (select MID from female_MIDS)
              group by year
              order by year"""
grader_5a(query5a)
```

```
   year  count
0  1939      1
1  1999      1
2  2000      1
3  2018      1
CPU times: user 212 ms, sys: 3.45 ms, total: 216 ms
Wall time: 221 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.

```
%%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 = """ with
              male_MIDS as(select mc.MID from M_cast mc, person p
where trim(mc.PID)=p.PID and p.Gender='Male'),
              female_MIDS as(select m.MID from Movie m,M_cast mc where
trim(mc.MID)=m.MID and trim(m.MID) not in(select MID from male_MIDS)),
              ALL_years as (select year, count(*) total_movies from
Movie group by substr(year,-4))

              select substr(m.year,-4)
year,ay.total_movies,count(m.year)*100.0 / total_movies as present
from movie m
              join female_MIDS fm on fm.MID = m.MID
              join ALL_years ay on substr(ay.year,-4) =
substr(m.year,-4)
              group by m.year
              order by m.year"""
grader_5b(query5b)

   year  total_movies    present
0  1939             2  50.000000
1  1999            66  16.666667
2  2000            64  15.625000
3  2018           104   1.923077
CPU times: user 6.33 s, sys: 14.5 ms, total: 6.35 s
Wall time: 6.36 s
```

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.

```
%%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(mc.PID)total_cast from Movie m
            join M_cast mc on m.MID=mc.MID
            group by m.MID
            order by total_cast desc """
grader_6(query6)

                        title  total_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
CPU times: user 169 ms, sys: 9.04 ms, total: 178 ms
Wall time: 186 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

```
%%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,count(*) as no_of_movies from Movie
              group by substr(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
CPU times: user 11.2 ms, sys: 978 µs, total: 12.2 ms
Wall time: 14.8 ms

%%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 cast(substr(m.year,-4) as INTERGER)
first,cast(substr(n.year,-4) as INTEGER) second,\
              cast(substr(m.year,-4) as INTEGER)+9 last,count(*) count
from Movie m, Movie n\
              where second<=last and second>=first\
              group by last,second
              order by count(*)
          """
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

   first  second  last  count
0   1931    1931  1940      1
1   1941    1941  1950      1
2   1941    1943  1950      1
3   1943    1943  1952      1
4   1931    1939  1940      2
5   1939    1941  1948      2
6   1939    1943  1948      2
7   1941    1946  1950      2
8   1941    1947  1950      2
9   1941    1950  1950      2
CPU times: user 6.73 s, sys: 129 ms, total: 6.86 s
Wall time: 6.86 s

%%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 = """ with\
             ALL_count as (select cast(substr(m.year,-4) as
INTEGER)first,cast(substr(n.year,-4) as INTEGERP) second,\
             cast(substr(m.year,-4) as INTEGER)+9 last,count(*) count
from Movie m, Movie n\
             where second <= last and second >= first\
             group by last,second
             order by count(*))

             select first,max(count) from ALL_count"""
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

   first  max(count)
0   2013       18496
CPU times: user 6.68 s, sys: 107 ms, total: 6.78 s
Wall time: 6.77 s
```

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

```
%%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 pd.PID director,pa.PID actor, count(*) from
M_Director md\
             join Person pd on pd.PID = trim(md.PID)
             join M_Cast mc on mc.MID = trim(md.MID)
             join Person pa on pa.PID = trim(mc.PID)
             group by director,actor"""
grader_8a(query8a)

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

    director      actor  count(*)
0  nm0000180  nm0000027         1
1  nm0000180  nm0001114         1
2  nm0000180  nm0001919         1
3  nm0000180  nm0006762         1
4  nm0000180  nm0030062         1
5  nm0000180  nm0038970         1
6  nm0000180  nm0051856         1
7  nm0000180  nm0085966         1
8  nm0000180  nm0097889         1
9  nm0000180  nm0125497         1
CPU times: user 456 ms, sys: 24 ms, total: 480 ms
Wall time: 487 ms
```

```
%%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 = """select actor,count from
            (select pd.PID director, pa.PID actor, count(*)count from
M_Director md\
            join person pd on pd.PID = trim(md.PID)
            join M_Cast mc on mc.MID = trim(md.MID)
            join person pa on pa.PID = trim(mc.PID)
            group by director,actor)
            where (actor,count) in\
            (select actor, max(max_count) from\
            (select pd.PID director, pa.PID actor, count(*) max_count
from M_Director md\
            join Person pd on pd.PID = trim(md.PID)
            join M_Cast mc on mc.MID = trim(md.MID)
            join Person pa on pa.PID = trim(mc.PID)
            group by director,actor)
            group by actor) and director = 'nm0007181' """
grader_8(query8)

       actor  count
0  nm0004434      7
1  nm0007181      2
2  nm0015296      1
3  nm0019463      1
4  nm0046230      1
5  nm0052570      1
6  nm0080266      1
7  nm0080385      1
8  nm0081070      1
9  nm0085944      1
(245, 2)
CPU times: user 572 ms, sys: 19.3 ms, total: 591 ms
Wall time: 592 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.

```
%%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 = """select trim(mc3.PID) from M_cast mc3 where trim(mc3.MID)
                       IN
                         (
                           select trim(mc2.MID) from M_cast mc2 where
trim(mc2.PID)
                           IN
                             (
                               select trim(mc1.PID) from M_cast mc1,
Person p where p.PID = trim(mc1.PID)
                               and trim(p.name) like '%Shah Rukh Khan
%'
                             )
                         )
                       except
                       select trim(mc1.PID) from M_cast mc1, Person p
where p.PID=trim(mc1.PID)
                       and trim(p.name) like '%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

  trim(mc3.PID)
0     nm0000818
1     nm0000821
2     nm0001934
3     nm0002043
4     nm0004109
```

```
5     nm0004334
6     nm0004335
7     nm0004363
8     nm0004418
9     nm0004429
(2382, 1)
CPU times: user 315 ms, sys: 7.76 ms, total: 322 ms
Wall time: 327 ms

%%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 from Person where PID
                IN
                (
                  select trim(mc5.PID) from M_cast mc5 where
trim(mc5.MID)
                  IN
                  (
                    select trim(mc4.MID) from M_cast mc4 where
trim(mc4.PID)
                    IN
                    (
                      select trim(mc3.PID) from M_cast mc3 where
trim(mc3.MID)
                      IN
                      (
                        select trim(mc2.MID) from M_cast mc2 where
trim(mc2.PID)
                        IN
                        (
                          select trim(mc1.PID) from M_cast mc1,Person
p where trim(mc1.PID)=p.PID
                          and p.name like '%Shah Rukh Khan%'

                        )
                      )
                    except
                    select trim(mc1.PID) from M_cast mc1,Person p
where trim(mc1.PID) = p.PID
                    and p.name like '%Shah Rukh Khan%'
                    )
                  )
                  except
                  select trim(mc3.PID) from M_cast mc3 where
trim(mc3.MID)
```

```
                     IN
                     (
                       select trim(mc2.MID) from M_cast mc2 where
trim(mc2.PID)
                       IN
                       (
                         select trim(mc1.PID) from M_cast mc1,Person p
where trim(mc1.PID) = p.PID
                         and p.name like '%Shah Rukh Khan%'
                       )
                     )
                   except
                   select trim(mc1.PID) from M_cast mc1,Person p where
trim(mc1.PID) = p.PID
                   and p.name like '%Shah Rukh Khan%'

                      )"""
grader_9(query9)

                     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)
CPU times: user 787 ms, sys: 17.9 ms, total: 804 ms
Wall time: 806 ms
```