SQL_Assignment

February 13, 2021

1 SQL Assignment

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

     from IPython.display import display, HTML,Image
```

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

Overview of all tables

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

```
[4]: 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

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

2 Preprocessing

```
[5]: cursor=conn.execute("Update Movie SET␣
      ↪MID=TRIM(MID),title=TRIM(title),year=CAST(SUBSTR(TRIM(year),-4) AS INTEGER)")
     conn.commit()
```

```
[6]: cursor1=conn.execute("Update Genre SET Name=TRIM(Name)")
     conn.commit()
```

```
[7]: cursor2=conn.execute("Update Language SET Name=TRIM(Name)")
     conn.commit()
```

```
[8]: cursor3=conn.execute("Update Country SET Name=TRIM(Name)")
     conn.commit()
```

```
[9]: cursor4=conn.execute("Update Location SET Name=TRIM(Name)")
     conn.commit()
```

```
[10]: cursor5=conn.execute("Update M_Location SET MID=TRIM(MID)")
      conn.commit()
```

```
[11]: cursor6=conn.execute("Update M_Country SET MID=TRIM(MID)")
      conn.commit()
```

```
[12]: cursor7=conn.execute("Update M_Language SET MID=TRIM(MID)")
      conn.commit()
```

```
[13]: cursor8=conn.execute("Update M_Genre SET MID=TRIM(MID)")
      conn.commit()
```

```
[14]: cursor9=conn.execute("Update Person SET␣
       ↪PID=TRIM(PID),Name=TRIM(Name),Gender=TRIM(Gender)")
      conn.commit()
```

```
[15]: cursor10=conn.execute("Update M_Producer SET MID=TRIM(MID),PID=TRIM(PID)")
      conn.commit()
```

```
[16]: cursor11=conn.execute("Update M_Director SET MID=TRIM(MID),PID=TRIM(PID)")
      conn.commit()
```

```
[17]: cursor12=conn.execute("Update M_Cast SET MID=TRIM(MID),PID=TRIM(PID)")
      conn.commit()
```

2.1 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.

[18]:
```
%%time
def grader_1(q1):
    q1_results  = pd.read_sql_query(q1,conn)
    print(q1_results)
    assert (q1_results.shape == (232,3))

query1 = """select p.Name,tb.year,tb.title
from Person p
JOIN M_Director md on p.PID=md.PID
JOIN (select MID,title,year from Movie where year%4=0 and year%100!=0 or␣
 ↪year%400=0) tb on md.MID=tb.MID
JOIN (select mg.MID MID,g.GID GID from Genre g JOIN M_Genre mg on g.GID=mg.GID␣
 ↪where g.Name like "%Comedy%") tc on tc.MID=tb.MID"""
grader_1(query1)
```

```
                      Name  year                             title
0             Milap Zaveri  2016                        Mastizaade
1             Danny Leiner  2004  Harold & Kumar Go to White Castle
2           Anurag Kashyap  2012                 Gangs of Wasseypur
3             Frank Coraci  2004         Around the World in 80 Days
4            Griffin Dunne  2008              The Accidental Husband
..                     ...   ...                               ...
227  Siddharth Anand Kumar  2004                       Let's Enjoy
228        Amma Rajasekhar  2008                           Sathyam
229          Oliver Paulus  2008                     Tandoori Love
230            Raja Chanda  2012                        Le Halua Le
231        K.S. Prakash Rao  1996                 Raja Aur Rangeeli

[232 rows x 3 columns]
Wall time: 89.8 ms
```

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

[19]:
```
%%time
def grader_2(q2):
    q2_results  = pd.read_sql_query(q2,conn)
    print(q2_results)
    assert (q2_results.shape == (17,1))


query2 = """select a.name
from Person a
```

```
JOIN M_Cast b on a.PID=b.PID
JOIN Movie c on b.MID=c.MID
where c.title='Anand'"""
grader_2(query2)
```

```
                Name
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
10     Sumita Sanyal
11    Asit Kumar Sen
12        Dara Singh
13     Johnny Walker
14         Moolchand
15      Gurnam Singh
16            Savita
Wall time: 202 ms
```

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

[20]:
```
%%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
Wall time: 299 ms
```

[21]:
```
%%time
def grader_3(q3):
    q3_results  = pd.read_sql_query(q3,conn)
    print(q3_results)
    assert (q3_results.shape == (300,1))

query3 = """select a.Name from Person a
where a.PID in
(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)
and a.PID in
(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)"""
grader_3(query3)
```

```
                 Name
0        Rishi Kapoor
1    Amitabh Bachchan
2              Asrani
3        Zohra Sehgal
4     Parikshat Sahni
..                ...
295            Poonam
296     Jamila Massey
297       K.R. Vijaya
298             Sethi
299      Suryakantham

[300 rows x 1 columns]
Wall time: 297 ms
```

2.4 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.

[22]:
```
%%time

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

query_4a ="""select PID,count(MID)
from M_Director
group by PID"""
print(grader_4a(query_4a))

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

```
            PID  count(MID)
0     nm0000180           1
1     nm0000187           1
2     nm0000229           1
```

```
3     nm0000269          1
4     nm0000386          1
…          …          …
1457  nm9742183          1
1458  nm9751348          1
1459  nm9793365          1
1460  nm9795684          1
1461  nm9872536          1

[1462 rows x 2 columns]
True
Wall time: 11 ms
```

[23]:
```
%%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 a.Name,b.count
from Person a
join (select PID,count(MID) count
from M_Director
group by PID) b on a.PID=b.PID
group by a.PID having b.count>=10
order by b.count DESC"""
grader_4(query4)
```

```
                   Name  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: 41.9 ms
```

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

[24]:
```
%%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 m.year,count(mid)
from Movie m
where m.MID not in
(select mc.MID from M_Cast mc join Person p on mc.PID=p.PID where p.
 ↪Gender='Male')
group by m.year"""
grader_5a(query5a)
```

```
   year  count(mid)
0  1939           1
1  1999           1
2  2000           1
3  2018           1
Wall time: 415 ms
```

2.6 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.

[25]:
```
%%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 ta.year,ta.count*100.0/tb.totalmoviescount␣
 ↪percentofmovies_female,tb.totalmoviescount from
(select m.year year,count(mid) count from Movie m where m.MID not in
(select mc.MID from M_Cast mc join Person p on mc.PID=p.PID where p.
 ↪Gender='Male') group by m.year) ta
join (select year,count(mid) totalmoviescount from Movie group by year) tb on␣
 ↪ta.year=tb.year"""
grader_5b(query5b)
```

```
   year  percentofmovies_female  totalmoviescount
0  1939               50.000000                 2
1  1999                1.515152                66
2  2000                1.562500                64
3  2018                0.961538               104
Wall time: 216 ms
```

2.7 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.

[26]:
```
%%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(mc.PID)) castsize
from Movie m
join M_Cast mc on m.MID=mc.MID
group by m.MID
order by castsize DESC"""
grader_6(query6)
```

```
                        title  castsize
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: 229 ms
```

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

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

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

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

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

[27]:
```
%%time
def grader_7a(q7a):
    q7a_results  = pd.read_sql_query(q7a,conn)
    print(q7a_results)
    assert (q7a_results.shape == (78, 2))
```

```
query7a = """select year,count(MID)
from Movie group by year"""
grader_7a(query7a)
#*** Write a query that computes number of movies in each year ***
# using the above query, you can write the answer to the given question
```

```
    year  count(MID)
0   1931           1
1   1936           3
2   1939           2
3   1941           1
4   1943           1
..   …           …
73  2014         126
74  2015         119
75  2016         129
76  2017         126
77  2018         104

[78 rows x 2 columns]
Wall time: 9.97 ms
```

[28]:
```
%%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 f.year,f.count,s.year,s.count
from (select year,count(MID) count from Movie group by year) f
join (select year,count(MID) count from Movie group by year) s on s.year<=f.
 ↪year+9
and s.year>=f.year"""
grader_7b(query7b)
# ***
#   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
#    ***
# 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  count  year  count
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: 15 ms
```

[29]:
```
%%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 max(moviecount),startyear
from (select sum(count2) moviecount,year1 startyear
from (select f.year year1,f.count count1,s.year year2,s.count count2
from (select year,count(MID) count from Movie group by year) f
join (select year,count(MID) count from Movie group by year) s on s.year<=f.
 ↪year+9
and s.year>=f.year)
group by year1)"""
grader_7(query7)
#*** Write a query that will return the decade that has maximum number of␣
 ↪movies ***
# if you check the output we are printinng all the year in that decade, its␣
 ↪fine you can print 2008 or 2008-2017
```

```
   max(moviecount) startyear
0             1203      2008
Wall time: 11 ms
```

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

[30]:
```
%%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 mc.PID actorid,md.PID directorid,count(md.MID) moviecount
from M_Cast mc
join M_Director md on mc.MID=md.MID
```

```
group by mc.PID,md.PID"""
grader_8a(query8a)
# *** Write a query that will results in number of movies actor-director worked␣
↪together ***
# using the above query, you can write the answer to the given question
```

```
     actorid directorid  moviecount
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: 304 ms
```

[31]:
```
%%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 p.Name,ta.moviecount Count
from (select actorid,moviecount
from (select mc.PID actorid,md.PID directorid,count(md.MID) moviecount
from M_Cast mc
join M_Director md on mc.MID=md.MID
group by mc.PID,md.PID)
where (actorid,moviecount) in
(select actorid,max(moviecount)
from (select mc.PID actorid,md.PID directorid,count(md.MID) moviecount
from M_Cast mc
join M_Director md on mc.MID=md.MID
group by mc.PID,md.PID)
group by actorid)
and directorid=(select PID from Person where Name='Yash Chopra')) ta
join Person p on ta.actorid=p.PID
order by Count DESC"""
grader_8(query8)
```

```
                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: 610 ms
```

2.9 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.

[32]:
```
%%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 distinct PID S1_PID
from M_Cast
where MID in (select MID from M_Cast where PID=(select PID from Person where␣
 ↪Name like '%Shah Rukh Khan%'))
and PID!=(select PID from Person where 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
```

```
      S1_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: 47.9 ms
```

[33]:
```
%%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 distinct PID S2_PID from M_Cast
where MID in (select MID from M_Cast where PID in
(select distinct PID S1_PID from M_Cast
where MID in (select MID from M_Cast where PID=(select PID from Person where␣
 ↪Name like '%Shah Rukh Khan%'))
and PID!=(select PID from Person where Name like '%Shah Rukh Khan%'))))
and PID not in (select distinct PID S1_PID from M_Cast
where MID in (select MID from M_Cast where PID=(select PID from Person where␣
 ↪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)
Wall time: 305 ms
```