sahiltinky94gmail.com_1

March 28, 2020

1 SQL Assignment

[2]:
```
import pandas as pd
import sqlite3
```

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

[15]:
```
# http://www.appliedaicourse.com/lecture/11/
 ↪applied-machine-learning-online-course/4142/
 ↪assignment-22-sql-assignment-on-imdb-data/1/
 ↪module-1-fundamentals-of-programming#comment91709
# Preprocessed the DB

cursor = conn.cursor()
# Movie table
cursor.execute('UPDATE Movie SET year = REPLACE(year, "I", "");')
cursor.execute('UPDATE Movie SET year = REPLACE(year, "V", "");')
cursor.execute('UPDATE Movie SET year = REPLACE(year, "X ", "");')
cursor.execute('UPDATE Movie SET title = TRIM(title);')
cursor.execute('UPDATE Movie SET year = TRIM(year);')
cursor.execute('UPDATE Movie SET rating = TRIM(rating);')
cursor.execute('UPDATE Movie SET num_votes = TRIM(num_votes);')

# M_Producer
cursor.execute('UPDATE M_Producer SET PID = TRIM(PID);')
cursor.execute('UPDATE M_Producer SET MID = TRIM(MID);')

# M_Director
cursor.execute('UPDATE M_Director SET PID = TRIM(PID);')
cursor.execute('UPDATE M_Director SET MID = TRIM(MID);')

# M_Cast
cursor.execute('UPDATE M_Cast SET PID = TRIM(PID);')
cursor.execute('UPDATE M_Cast SET MID = TRIM(MID);')

# M_Genre
cursor.execute('UPDATE M_Genre SET GID = TRIM(GID);')
```

```
cursor.execute('UPDATE M_Genre SET MID = TRIM(MID);')

# Genre
cursor.execute('UPDATE Genre SET GID = TRIM(GID);')
cursor.execute('UPDATE Genre SET Name = TRIM(Name);')

# Person
cursor.execute('UPDATE Person SET Name = TRIM(Name);')
cursor.execute('UPDATE Person SET PID = TRIM(PID);')
cursor.execute('UPDATE Person SET Gender = TRIM(Gender);')

#conn.commit() temporary ( un-comment it to make permanent)
```

[15]:
```
<sqlite3.Cursor at 0x1d0b21d8960>
```

1.1 Sample Code

[105]:
```
%%time
# Write your sql query below

query = """
        SELECT count(*)
        FROM Movie

        """

q = pd.read_sql_query(query, conn)
print(q.shape)
q.head()
```

```
(1, 1)
Wall time: 1.99 ms
```

[105]:

	count(*)
0	3475

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.

[39]:
```
%%time
# Write your sql query below

query = """
        SELECT p.Name,m.title,m.year as year
```

```
        FROM Movie AS m, Person AS p, Genre AS g, M_Director AS md, M_Genre AS␣
↪mg
        WHERE mg.GID = g.GID and mg.MID = md.MID and md.PID = p.PID and md.MID␣
↪= m.MID
        AND g.GID IN (SELECT distinct GID
        FROM Genre
        where Name like '%Comedy%')
        AND (((m.year % 4 == 0) AND (m.year % 100 != 0)) OR (m.year % 400 == 0))
        ORDER BY m.year ASC

        """

q1 = pd.read_sql_query(query, conn)
print(q1.shape)
q1.head()
```

```
(232, 3)
Wall time: 49.5 ms
```

[39]:

	Name	title	year
0	Amit Mitra	Jagte Raho	1956
1	Chetan Anand	Funtoosh	1956
2	Satyen Bose	Jagriti	1956
3	Mohan Segal	New Delhi	1956
4	S.U. Sunny	Kohinoor	1960

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

[7]:
```
%%time
# Write your sql query below

query = """
        SELECT distinct(p.Name)
        FROM Person AS p, M_Cast AS mc, Movie AS m
        WHERE mc.PID = p.PID AND mc.MID = m.MID AND m.year = 1971 AND m.
↪title='Anand'

        """

q2 = pd.read_sql_query(query, conn)
print(q2.shape)
q2.head()
```

```
(17, 1)
Wall time: 119 ms
```

[7]:

	Name
0	Amitabh Bachchan
1	Rajesh Khanna
2	Brahm Bhardwaj
3	Ramesh Deo
4	Seema Deo

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.)

[41]:
```
%%time
# Write your sql query below

query = """
        SELECT distinct(p.Name)
        FROM Movie AS m, Person AS p, M_Cast AS mc
        WHERE mc.MID = m.MID AND mc.PID = p.PID AND m.year > 1990
        INTERSECT
        SELECT distinct(p.Name)
        FROM Movie AS m, Person AS p, M_Cast AS mc
        WHERE mc.MID = m.MID AND mc.PID = p.PID AND m.year < 1970

        """

q3 = pd.read_sql_query(query, conn)
print(q3.shape)
q3.head()
```

```
(424, 1)
Wall time: 399 ms
```

[41]:

	Name
0	A.K. Hangal
1	Aachi Manorama
2	Abbas
3	Abdul
4	Abhi Bhattacharya

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.

[20]:
```
%%time
# Write your sql query below

query = """
        SELECT distinct(p.Name), count(m.MID) as Number_of_Movies
```

```
        FROM M_Director AS md, Person AS p, Movie AS m
        WHERE md.PID = p.PID AND md.MID = m.MID
        GROUP BY md.PID
        HAVING count(m.MID) >= 10
        ORDER BY count(m.MID) DESC

        """

q4 = pd.read_sql_query(query, conn)
print(q4.shape)
```

```
(58, 2)
Wall time: 54.9 ms
```

[21]:
```
q4.head()
```

[21]:

	Name	Number_of_Movies
0	David Dhawan	39
1	Mahesh Bhatt	35
2	Priyadarshan	30
3	Ram Gopal Varma	30
4	Vikram Bhatt	29

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

[6]:
```
%%time
# Write your sql query below

query = """
        select CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) AS year, count(m.MID)␣
↪from Movie as m where m.MID not in (
        SELECT distinct mc.MID from M_Cast as mc where
        (trim(mc.PID) IN ( select trim(p.PID) from Person p where ((p.
↪Gender='Male') or (p.Gender is NULL)))) or trim(mc.PID)
        is NULL) group by year
        """

q5a = pd.read_sql_query(query, conn)
print(q5a.shape)
q5a
```

```
(4, 2)
Wall time: 73 ms
```

[6]:

	year	count(m.MID)
0	1939	1

1	1999	1
2	2000	1
3	2018	1

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.

From my assumption, I calculated as

> Calculate female percentage: number of female for that year [answer generated- see prev question q5a]/ total number of movies in that year (incl Male, None, Female)

[17]:
```
%%time
# Write your sql query below

query = """
        SELECT Y.year, (IFNULL(X.Mov,0)*1.0/Y.Mov)*100 AS FemalePercentage, Y.
 ↪Mov AS TotalMovie FROM
        (select m.year as year, count(m.MID) as Mov from Movie as m, M_Cast as␣
 ↪mc, Person as p where
        mc.PID = p.PID and mc.MID = m.MID
        group by m.year
        ORDER BY m.year ASC) AS Y
        Left join
        (select CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) AS year, count(m.MID)␣
 ↪as Mov from Movie as m where m.MID not in (
        SELECT distinct mc.MID from M_Cast as mc where
        (trim(mc.PID) IN ( select trim(p.PID) from Person p where ((p.
 ↪Gender='Male') or (p.Gender is NULL)))) or trim(mc.PID)
        is NULL)
        group by m.year
        ORDER BY m.year ASC) AS X
        on Y.year = X.year
        """

q5b = pd.read_sql_query(query, conn)
print(q5b.shape)
q5b
```

```
(78, 3)
Wall time: 258 ms
```

[17]:

	year	FemalePercentage	TotalMovie
0	1931	0.000000	9

```
1    1936          0.000000          47
2    1939          2.222222          45
3    1941          0.000000          54
4    1943          0.000000          14
..    …               …         …
73   2014          0.000000        2780
74   2015          0.000000        2956
75   2016          0.000000        3315
76   2017          0.000000        3592
77   2018          0.033990        2942

[78 rows x 3 columns]
```

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.

[160]:
```
%%time
# Write your sql query below

query = """
        SELECT X.* FROM
        (Select m.title AS Movie_Name, count(*) AS count_distinct_actor
        FROM Movie AS m, Person AS p, M_Cast AS mc
        WHERE m.MID = TRIM(mc.MID) AND TRIM(mc.PID) = p.PID
        GROUP BY m.title) AS X
        ORDER BY X.count_distinct_actor DESC
        LIMIT 1
        """

q6 = pd.read_sql_query(query, conn)
print(q6.shape)
```

```
(1, 2)
Wall time: 174 ms
```

[32]:
```
q6.head()
```

[32]:
```
       Movie_Name  count_distinct_actor
0  Ocean's Eight                   238
```

1.9 Q7 — A decade is a sequence of 10 consecutive years. For example, say in your database you have movie information starting from 1965. Then the first decade is 1965, 1966, ..., 1974; the second one is 1967, 1968, ..., 1976 and so on. Find the decade D with the largest number of films and the total number of films in D.

[85]:
```
%%time
# Write your sql query below

query = """
        with initial_year(year) as
        (
            select distinct year from Movie
        )
        , dest_year(year) as
        (
            select year+9 from initial_year
        )
        , decade_ten(startyear, endyear) as
        (
            select initial_year.year,dest_year.year from initial_year,dest_year␣
 ↪where initial_year.year = dest_year.year - 9
        )
        , num_mov_yr(styr, endyr, num_mov) as
        (
            select decade_ten.startyear,decade_ten.endyear, count(*) from Movie␣
 ↪as m, decade_ten
            where m.year >= decade_ten.startyear and m.year <= decade_ten.
 ↪endyear
            group by decade_ten.startyear
        )
        select num_mov_yr.styr, num_mov_yr.endyr, num_mov_yr.num_mov from␣
 ↪num_mov_yr where
        num_mov_yr.num_mov in (select max(num_mov_yr.num_mov) from num_mov_yr)
        """

q = pd.read_sql_query(query, conn)
print(q.shape)
q
```

```
(1, 3)
Wall time: 165 ms
```

[85]:

	styr	endyr	num_mov
0	2008	2017	1205

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

[16]:
```
%%time
# Write your sql query below

query = """
         with yc(ycID) as
         (
             select distinct PID from Person as p
             where p.Name = 'Yash Chopra'
         )
         , num_mov_actor_director(acID, dirID, numMov) as
         (
             select mc.PID, md.PID, count(md.MID) from M_Cast as mc, M_director␣
↪as md
             where mc.MID = md.MID
             group by mc.PID, md.PID
         )
         , num_mov_yc(acID, ycID, numMov) as
         (
             select mc.PID, md.PID, count(md.MID) from M_Cast as mc, M_director␣
↪as md, yc
             where mc.MID = md.MID and md.PID = yc.ycID
             group by mc.PID
         )
         , num_mov_noyc(acID, noycID, numMov) as
         (
             select mc.PID, md.PID, count(md.MID) from M_Cast as mc, M_director␣
↪as md, yc
             where mc.MID = md.MID and md.PID <> yc.ycID
             group by mc.PID
         )
         , actor_mov_comp(acID, comval) as
         (
             select num_mov_yc.acID,
             case when num_mov_yc.numMov >= IFNULL(num_mov_noyc.numMov,0) then␣
↪'true' else 'false' end more_mov_yc
             from
             num_mov_yc LEFT OUTER JOIN num_mov_noyc on num_mov_yc.acID =␣
↪num_mov_noyc.acID
         )
         select distinct p.Name from Person as p, actor_mov_comp where p.PID =␣
↪actor_mov_comp.acID and
         actor_mov_comp.comval = 'true'
        """
```

```
q8 = pd.read_sql_query(query, conn)
print(q8.shape)
q8
```

```
(133, 1)
Wall time: 288 ms
```

[16]:

	Name
0	Yash Chopra
1	Akhtar-Ul-Iman
2	Andrew Bicknell
3	Steve Box
4	Pamela Chopra
..	…
128	Martin Crossingham
129	Katy Kartwheel
130	Abbie Murphy
131	Richard Broom
132	Sean Moon

[133 rows x 1 columns]

1.11 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.

[50]:
```
%%time
# Write your sql query below

query = """

        WITH srk(srkPID) as
        (
            select PID from Person as p where
            p.Name = 'Shah Rukh Khan'
        )
        , moviewithsrk(srkmovMID) as
        (
            select distinct mc.MID from M_Cast as mc, srk where
            mc.PID = srk.srkPID

        )
        , actorplayinsrk(actorID) as
        (
```

```
            select distinct mc.PID from M_Cast as mc, moviewithsrk where
            mc.MID = moviewithsrk.srkmovMID
        )
        , movienotsrk(srknotMID) as
        (
            select distinct mc.MID from M_Cast as mc, actorplayinsrk,␣
↪moviewithsrk where
            mc.PID = actorplayinsrk.actorID and mc.MID NOT IN (
                select distinct mc.MID from M_Cast as mc, srk where
                mc.PID = srk.srkPID
            )
        )
        , actornotplaysrk(acID) as
        (
            select distinct PID from M_Cast as mc, movienotsrk, actorplayinsrk␣
↪where
            mc.MID = movienotsrk.srknotMID  and mc.PID not in (
                select distinct mc.PID from M_Cast as mc, moviewithsrk where
                mc.MID = moviewithsrk.srkmovMID
            )
        )
        select Name from Person as p, actornotplaysrk where p.PID =␣
↪actornotplaysrk.acID
        """

q9 = pd.read_sql_query(query, conn)
print(q9.shape)
q9.head()
```

```
(25698, 1)
Wall time: 1min 8s
```

[50]:

	Name
0	Freida Pinto
1	Rohan Chand
2	Damian Young
3	Waris Ahluwalia
4	Caroline Christl Long

[0]: