SQL_Assignment_Reference

June 29, 2020

1 SQL Assignment

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

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

[3]:
```
conn = sqlite3.connect('drive/My Drive/FFRDB/Db-IMDB-Assignment.db')
```

1.1 Sample Code

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

query = """
        SELECT TRIM(Movie.title) AS 'Movie_Name'
        FROM Movie
        WHERE Movie.rating < 3

        """

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

```
(85, 1)
             Movie_Name
0            Mastizaade
1  Dragonball Evolution
2             Loveyatri
3                Race 3
4                Gunday
CPU times: user 11.1 ms, sys: 125 µs, total: 11.2 ms
Wall time: 17.6 ms
```

1.2 Db updates

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE M_Producer
                SET
                ID= Trim(ID),
                PID=Trim(PID),
                MID=Trim(MID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE M_Director
                SET
                ID= Trim(ID),
                PID=Trim(PID),
                MID=Trim(MID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE M_Cast
                SET
                ID= Trim(ID),
                PID=Trim(PID),
                MID=Trim(MID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE M_Genre
                SET
                ID= Trim(ID),
                GID=Trim(GID),
                MID=Trim(MID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE M_Language
                SET
                ID= Trim(ID),
                LAID=Trim(LAID),
                MID=Trim(MID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE M_Country
                SET
                ID= Trim(ID),
                CID=Trim(CID),
                MID=Trim(MID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE M_Location
                SET
                ID= Trim(ID),
                LID=Trim(LID),
                MID=Trim(MID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE Movie
                SET
                year= substr(year,-4),
                MID=Trim(MID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE Person
                SET
                PID=Trim(PID),
                Name =Trim(Name)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE Genre
                SET
                GID=Trim(GID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE Language
                SET
                LAID=Trim(LAID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE Country
                SET
                CID=Trim(CID)
                ;''')
conn.commit()
```

[ ]:
```
cursor = conn.cursor()
cursor.execute('''UPDATE Location
                SET
                LID=Trim(LID)
                ;''')
conn.commit()
```

1.3 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.

[105]:
```
%%time



query='''select distinct p.name as director, m.title as Movie, m.year as Year
         from Movie m
         join M_Director md on md.MID = m.MID
         join Person p on p.PID = md.PID
         join M_Genre mg on mg.MID=m.MID
         join Genre g on g.GID=mg.GID
         where g.name like \'%comedy%\'
         and ((m.year%4 = 0 and m.year%100 <> 0) or m.year%400 = 0)
         ;'''
q= pd.read_sql_query(query, conn)
print(q)
```

```
                director                             Movie  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
..                   ...                               ...   ...
```

```
227  Siddharth Anand Kumar                        Let's Enjoy  2004
228         Amma Rajasekhar                            Sathyam  2008
229           Oliver Paulus                      Tandoori Love  2008
230             Raja Chanda                        Le Halua Le  2012
231        K.S. Prakash Rao                 Raja Aur Rangeeli  1996

[232 rows x 3 columns]
CPU times: user 75.3 ms, sys: 0 ns, total: 75.3 ms
Wall time: 91.1 ms
```

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

[106]:
```
%%time
# Write your sql query below

query = """ select name [Actors in Anand]
            from person p
            where pid in( select distinct mc.pid
                          from Movie m
                          join M_Cast mc on mc.MID=m.MID
                          where m.title="Anand")
            order by p.name

        """

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

```
     Actors in Anand
0    Amitabh Bachchan
1      Asit Kumar Sen
2        Atam Prakash
3      Brahm Bhardwaj
4          Dara Singh
5          Dev Kishan
6         Durga Khote
7        Gurnam Singh
8       Johnny Walker
9       Lalita Kumari
10       Lalita Pawar
11          Moolchand
12      Rajesh Khanna
13         Ramesh Deo
14             Savita
15          Seema Deo
16      Sumita Sanyal
CPU times: user 113 ms, sys: 2.04 ms, total: 115 ms
```

```
Wall time: 117 ms
```

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

[107]:
```
%%time
# Write your sql query below


query = """ select name [Actors]
            from person p
            where p.pid in (select distinct mc.pid
                            from Movie m
                            join M_Cast mc on mc.MID=m.MID
                            where m.year<=1970

                            Intersect

                            select distinct mc.pid
                            from Movie m
                            join M_Cast mc on mc.MID=m.MID
                            where m.year>=1990)

        """

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

```
              Actors
0        Rishi Kapoor
1    Amitabh Bachchan
2              Asrani
3       Gurdeep Singh
4        Zohra Sehgal
..                ...
340            Poonam
341     Jamila Massey
342       K.R. Vijaya
343             Sethi
344      Suryakantham

[345 rows x 1 columns]
CPU times: user 264 ms, sys: 11.8 ms, total: 276 ms
Wall time: 280 ms
```

1.6 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.

[108]:
```
%%time
# Write your sql query below

query = """ Select distinct p.Name as Directors, count(m.title) [Number of␣
 ↪films directed]
            from Movie m
            join M_Director md on md.MID=m.MID
            join Person p on p.PID=md.PID
            group by p.Name
            having count(m.title)>=10
            order by count(m.title) desc

        """

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

	Directors	Number of films directed
0	David Dhawan	39
1	Mahesh Bhatt	36
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
10	Abbas Alibhai Burmawalla	17
11	Rama Rao Tatineni	17
12	Shyam Benegal	17
13	Gulzar	16
14	Manmohan Desai	16
15	Raj N. Sippy	16
16	Mahesh Manjrekar	15
17	Raj Kanwar	15
18	Indra Kumar	14
19	Rahul Rawail	14
20	Raj Khosla	14
21	Rajkumar Santoshi	14
22	Ananth Narayan Mahadevan	13
23	Anurag Kashyap	13
24	Dev Anand	13
25	Harry Baweja	13

```
26        K. Raghavendra Rao                        13
27             Rakesh Roshan                        13
28               Vijay Anand                        13
29              Anees Bazmee                        12
30               Anil Sharma                        12
31              Guddu Dhanoa                        12
32         Madhur Bhandarkar                        12
33           Nagesh Kukunoor                        12
34               Prakash Jha                        12
35             Prakash Mehra                        12
36              Rohit Shetty                        12
37             Satish Kaushik                       12
38               Umesh Mehra                        12
39           Govind Nihalani                        11
40               Ketan Mehta                        11
41                Mohit Suri                        11
42             Nasir Hussain                        11
43        Pramod Chakravorty                        11
44              Sanjay Gupta                        11
45                 Bimal Roy                        10
46              Hansal Mehta                        10
47             J. Om Prakash                        10
48                J.P. Dutta                        10
49                K. Bapaiah                        10
50       K. Muralimohana Rao                        10
51                Mehul Kumar                       10
52                 N. Chandra                       10
53            Pankaj Parashar                       10
54                 Raj Kapoor                       10
55              Sudhir Mishra                       10
56           Tigmanshu Dhulia                       10
57            Vishal Bhardwaj                       10
CPU times: user 65.5 ms, sys: 1.85 ms, total: 67.4 ms
Wall time: 68.5 ms
```

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

[109]:
```
%%time
# Write your sql query below

query = """ select Year, count(distinct Movie) [Number of movies with actresses␣
 ↪only]
            from  ( select
                    m.year [Year],m.title [Movie],
                    count(case when p.gender='Male' then 1 end) as male_cnt,
                    count(case when p.gender='Female' then 1 end) as female_cnt
```

```
                    from movie m
                    join m_cast mc on mc.mid = m.mid
                    join person p on p.pid = mc.pid
                    group by m.title )
            where male_cnt=0
            group by Year
        """




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

```
   Year  Number of movies with actresses only
0  1939                                     1
1  1999                                     1
2  2000                                     1
3  2018                                     2
CPU times: user 280 ms, sys: 5.65 ms, total: 285 ms
Wall time: 290 ms
```

1.8 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.

[110]:
```
%%time
# Write your sql query below
# case when male_cnt!=0 or female_cnt!=0 then 1 end
query = """ select Year,
            (cast(count(case when male_cnt=0 then 1 end) as float)/
 ↪cast(count(distinct movie)as float)*100)␣
 ↪[percentage_of_movies_with_actresses_only],
            count(distinct movie) [Total Number of movies]
            from  ( select
                    m.year [Year],m.title [Movie],
                    count(case when p.gender='Male' then 1 end) as male_cnt,
                    count(case when p.gender='Female' then 1 end) as female_cnt
                    from movie m
                    join m_cast mc on mc.mid = m.mid
                    join person p on p.pid = mc.pid
                    group by m.title )
            group by Year
            having percentage_of_movies_with_actresses_only!=0
        """
```

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

```
   Year  percentage_of_movies_with_actresses_only  Total Number of movies
0  1939                                 50.000000                       2
1  1999                                  1.538462                      65
2  2000                                  1.639344                      61
3  2018                                  1.980198                     101
CPU times: user 280 ms, sys: 11.6 ms, total: 292 ms
Wall time: 297 ms
```

1.9 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.

[111]:
```
%%time
# Write your sql query below

query = """ select Movie , max(size) [cast size]
            from (
              select m.title [Movie], count(distinct pid) [size]
              from movie m
              join m_cast mc on m.mid=mc.mid
              group by m.title

            )


        """

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

```
          Movie  cast size
0  Ocean's Eight        238
CPU times: user 171 ms, sys: 11 ms, total: 182 ms
Wall time: 191 ms
```

1.10 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.

[112]:
```
%%time
# Write your sql query below
#understood the approach from https://stackoverflow.com/questions/51609285/
 ↪query-for-find-the-decade-with-the-largest-number-of-records

query = """        select y.year  [start decade], y.year + 9 [end decade],␣
 ↪count(*) as num_movies
                   from (select distinct year from movie) y
                   join movie m on m.year >= y.year and m.year < y.year + 9
                   group by y.year
                   order by count(*) desc
                   limit 1;
        """

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

```
  start decade  end decade  num_movies
0         2009        2018        1098
CPU times: user 73.4 ms, sys: 0 ns, total: 73.4 ms
Wall time: 75.5 ms
```

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

[113]:
```
#creating a table containing total number of movies a actor-director duo has␣
 ↪done together, lets say t1
%%time
query = """
                              select actor.name [Actor], director.name␣
 ↪[Director], count(m.title) [Movie_cnt]
                              from movie m
                              join m_cast mc on mc.mid = m.mid
                              join person actor on actor.pid = mc.pid
                              join m_director md on md.mid=m.mid
                              join person director on director.pid=md.pid
                              group by director.pid,actor.pid


              """
```

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

```
                  Actor      Director  Movie_cnt
0         Alec Guinness    David Lean          1
1            Judy Davis    David Lean          1
2        Peggy Ashcroft    David Lean          1
3         Saeed Jaffrey    David Lean          1
4             Paul Anil    David Lean          1
...                  ...           ...        ...
73403         Mukul Dev  Vinod Tiwari          1
73404  Krishna Abhishek  Vinod Tiwari          1
73405  Rajniesh Duggall  Vinod Tiwari          1
73406     Nazia Hussain  Vinod Tiwari          1
73407      Nancy Marwah  Vinod Tiwari          1

[73408 rows x 3 columns]
CPU times: user 602 ms, sys: 35.9 ms, total: 638 ms
Wall time: 642 ms
```

[114]:
```
#counting total number of acotrs
%%time
query = """
                        select count(distinct pid)  [Total number of actors]␣
 ↪from m_cast


              """


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

```
   Total number of actors
0                   32127
CPU times: user 68.7 ms, sys: 2.95 ms, total: 71.7 ms
Wall time: 73.9 ms
```

[115]:
```
# creating table which stores the movie count for actors with the director they␣
 ↪have worked the most with lets say t2,
#which has the same row count as number of actors


query = """
                        select distinct aid,Actor, Director, max(Movie_cnt)
                        from(
```

```
                              select actor.name [Actor], actor.pid as aid,␣
 ↪director.name [Director], director.pid as did, count(m.title) [Movie_cnt]
                              from movie m
                              join m_cast mc on mc.mid = m.mid
                              join person actor on actor.pid = mc.pid
                              join m_director md on md.mid=m.mid
                              join person director on director.pid=md.pid
                              group by director.pid,actor.pid
                            )
                          group by aid
                          order by aid




              """


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

```
            aid            Actor          Director  max(Movie_cnt)
0      nm0000002    Lauren Bacall   J. Lee Thompson               1
1      nm0000027    Alec Guinness        David Lean               1
2      nm0000039     Deborah Kerr     Charles Vidor               1
3      nm0000042        Alan Ladd     Charles Vidor               1
4      nm0000047     Sophia Loren        Raj Kapoor               1
...          ...              ...              ...             ...
32122  nm9988016    Ashiqa Salvan     Sudha Kongara               1
32123  nm9988018   Ravindra Vijay     Sudha Kongara               1
32124  nm9990703     Godaan Kumar  Shashank Khaitan               1
32125  nm9990704  Shridhar Watsar  Shashank Khaitan               1
32126  nm9990705     Janhavi Dave  Shashank Khaitan               1

[32127 rows x 4 columns]
```

[117]:
```
# finding actors whoes movie count in t1(number of movies with yash chopra) is␣
 ↪greater than movies count in t2

%%time


query = """             select t1.actor, t1.Movie_cnt [Movies with Raj]
                        from (select distinct actor.pid as aid,actor.name␣
 ↪[Actor], director.name [Director],  director.pid,count(m.title) [Movie_cnt]
                              from movie m
                              join m_cast mc on mc.mid = m.mid
```

```
                              join person actor on actor.pid = mc.pid
                              join m_director md on md.mid=m.mid
                              join person director on director.pid=md.pid
                              where Director.name =  'Yash Chopra'
                              group by director.pid,actor.pid) t1
                         join(select distinct aid,Actor, Director,␣
 ↪max(Movie_cnt) [Movie_cnt]
                              from(
                              select actor.name [Actor], actor.pid as aid,␣
 ↪director.name [Director], director.pid as did, count(m.title) [Movie_cnt]
                              from movie m
                              join m_cast mc on mc.mid = m.mid
                              join person actor on actor.pid = mc.pid
                              join m_director md on md.mid=m.mid
                              join person director on director.pid=md.pid
                              group by director.pid,actor.pid
                                )
                              group by aid
                              order by aid) t2
                        on t1.aid = t2.aid
                        and t1.movie_cnt>=t2.movie_cnt



              """



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

```
              Actor  Movies with Raj
0     Shashi Kapoor                7
1       Yash Chopra                2
2    Akhtar-Ul-Iman                1
3         Murad Ali                1
4      Badri Prasad                1
..              ...              ...
240     Saul George                3
241   Vinita Sharma                1
242       Sean Moon                1
243    Varun Thakur                1
244   Surendra Rahi                3

[245 rows x 2 columns]
CPU times: user 768 ms, sys: 23.5 ms, total: 792 ms
```

```
Wall time: 795 ms
```

1.12 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.

[118]:
```
%%time
# mid of movies in which shah rukh khan acted

query = """ select distinct mc.mid
            from m_cast mc
            join person actor on actor.pid = mc.pid
            where actor.name = 'Shah Rukh Khan'

        """

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

```
          MID
0   tt0101732
1   tt0106333
2   tt0107321
3   tt0109134
4   tt0109555
..        ...
85  tt4535650
86  tt4559006
87  tt5882970
88  tt5946128
89  tt5997666

[90 rows x 1 columns]
CPU times: user 118 ms, sys: 10 ms, total: 128 ms
Wall time: 132 ms
```

[120]:
```
%%time
# pid of co actors of shah rukh khan /actors with  shah rukh number 1 excluding␣
 ↪shah rukh khan

query = """ select distinct mc.pid
            from m_cast mc
            join person actor on actor.pid = mc.pid
            where mc.mid in (
```

```
                  select mc.mid
                  from m_cast mc
                  join person actor on actor.pid = mc.pid
                  where actor.name = 'Shah Rukh Khan')

            and actor.pid not in       (select actor.pid
                                        from person actor
                                        where actor.name = 'Shah Rukh Khan'
                                            )

        """

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

```
           PID
0     nm0004418
1     nm1995953
2     nm2778261
3     nm0631373
4     nm0241935
...         ...
2377  nm4173451
2378  nm7620177
2379  nm3093045
2380  nm0451154
2381  nm3385526

[2382 rows x 1 columns]
CPU times: user 191 ms, sys: 4.85 ms, total: 196 ms
Wall time: 197 ms
```

[121]:
```
%%time
# mid of movies of co actors of shah rukh khan /actors with  shah rukh number 1

query = """ select distinct mc.mid
            from m_cast mc
            where mc.pid in(

                  select mc.pid
                  from m_cast mc
                  join person actor on actor.pid = mc.pid
                  where mc.mid in (
                        select mc.mid
                        from m_cast mc
                        join person actor on actor.pid = mc.pid
                        where actor.name = 'Shah Rukh Khan')
```

```
                and actor.pid not in       (select actor.pid
                                            from person actor
                                            where actor.name = 'Shah Rukh␣
 ↪Khan'
                                                ))

        """

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

```
           MID
0     tt1365519
1     tt3498820
2     tt8108198
3     tt3741834
4     tt6747420
...         ...
3201  tt0165795
3202  tt0090611
3203  tt0106270
3204  tt0852989
3205  tt0375890

[3206 rows x 1 columns]
CPU times: user 242 ms, sys: 7.95 ms, total: 250 ms
Wall time: 251 ms
```

[122]:
```
%%time
# All the actors who acted in movies of actors having shah rukh number 1(let␣
 ↪say sh1) excluding sh1

query = """
                select actor.name [Actor with Shahrukh number 2]
                from person actor
                where actor.pid in (select distinct actor.pid
                                    from m_cast mc
                                    join person actor on actor.pid = mc.pid
                                    where mc.mid in ( select  distinct mc.mid
                                                      from m_cast mc
                                                      join person actor on␣
 ↪actor.pid = mc.pid
                                                      where actor.pid in␣
 ↪(select distinct mc.pid
                                                                          from␣
 ↪m_cast mc
```

```
                                                                        where␣
↪mc.mid in ( select distinct  mc.mid
                                                                             ␣
↪            from m_cast mc
                                                                             ␣
↪            join person actor on actor.pid = mc.pid
                                                                             ␣
↪            where actor.name = 'Shah Rukh Khan')
                                                                        and␣
↪mc.pid not in ( select mc.pid
                                                                             ␣
↪            from m_cast mc
                                                                             ␣
↪            join person actor on actor.pid = mc.pid
                                                                             ␣
↪            where actor.name = 'Shah Rukh Khan')))

                                    and mc.pid not in (select distinct  mc.pid
                                                       from m_cast mc
                                                       where mc.mid in (␣
↪select distinct  mc.mid
                                                                         from␣
↪m_cast mc
                                                                         join␣
↪person actor on actor.pid = mc.pid
                                                                        ␣
↪where actor.name = 'Shah Rukh Khan')
                                                        and mc.pid not in (␣
↪select   mc.pid
                                                                           ␣
↪from m_cast mc
                                                                           ␣
↪join person actor on actor.pid = mc.pid
                                                                           ␣
↪where actor.name = 'Shah Rukh Khan')))
                  group by actor.pid;



       """

  ### trying to find co actors of shahrkh khan but no other name than david␣
↪lean works

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

```
      Actor with Shahrukh number 2
0                    Alec Guinness
1                     Sophia Loren
2                        Brad Pitt
3                  Gillian Anderson
4                    Pierce Brosnan
...                             ...
25694                Fernando Cruz
25695                 Sohail Mirza
25696              Shreyas Sanghavi
25697                 Ashiqa Salvan
25698                Ravindra Vijay

[25699 rows x 1 columns]
CPU times: user 861 ms, sys: 39.7 ms, total: 901 ms
Wall time: 905 ms
```