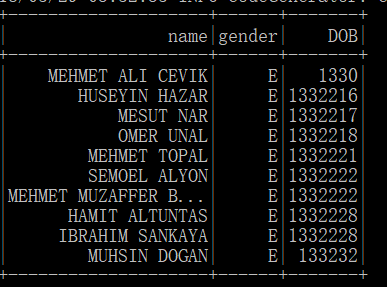
E1：

In most of our questions, I found that it’s easier for me to create a dateFrame structure and use a SQL query to solve the question.

We use str(yyyy)+str(mm)+str(dd) as the key of sort, then use a SQL query to solve this problem.



Here is the 10 eldest man in the dataset. We can know that Turkey men are very long-lived. :)

E2.

In this and the next question, use simple RDD calculation is more convenient than SQL query.

We use an algorithm which seems like “wordcount” in this part. We use flatMap to trans origin RDD into an RDD of characters, and then we map character into (character,1), and use reduceByKey(add) to sum them.

Here we use a trick: list(some string) can make a string into list of chars. For example:



Then flat Map function can transform this into RDD of chars.

**[('A', 57623513), ('E', 39619607), ('I', 34493003), ('N', 26440665), ('R', 24978398), ('', 24072703), ('L', 22802085), ('M', 20796307), ('S', 20375160), ('K', 19204812), ('T', 17139058), ('Y', 14881732), ('C', 11752704), ('H', 11162106), ('O', 11013168), ('D', 10683664), ('G', 9980618), ('Z', 9896629), ('B', 7507056), ('F', 5180773), ('V', 3432261), ('P', 2585236), (' ', 1870865), ('J', 94123), ('.', 8586), ('W', 973), ('-', 507), ('X', 286), ('Q', 145), ('(', 118), (')', 117), ("'", 112), ('3', 71), ('2', 70), ('<', 49), ('>', 49), ('u', 25), ('0', 17), ('o', 13), ('/', 11), ('1', 9), ('`', 7), ('5', 6), ('4', 5), (',', 4), ('\\', 4), ('?', 3), ('c', 3), ('s', 3), ('g', 2), ('7', 1), ('\_', 1)]**

We can find that character “A” “E” “I” “N” “R” appears most in Turkey people’s name.

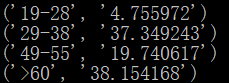
E3.

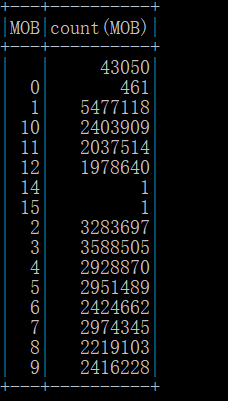
We use a self-defined function to map age into string “19-28” or something like this, and then we map each key into (key,1) and use reduceByKey to sum up. We get the following answer:

**[(None, 10947617), ('19-28', 1130969), ('29-38', 8881641), ('49-55', 4694314), ('>60', 9073052)]**

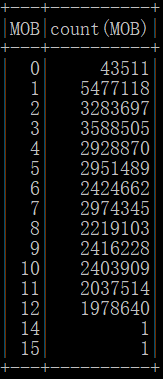
Here, ‘None’ means this line of data is lack of month of birth information, or month of birth does not exist in these buckets.

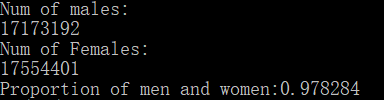
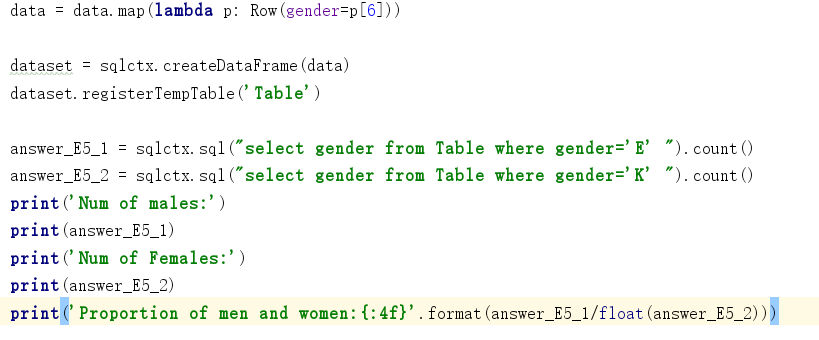
Then we can calculate the proportion of each part, the percentage is shown below.



E4. We first scan the whole table with a SQL query, just use string type month-of-birth as key, and found some exception in the data. There are 43050 lines of data has no MOB information and have two lines of data which have a month over 12, maybe for some religious reason. Also have 461 lines of data which have month 0. 

Then we parse month information into int type and parse no-month information data into month 0, then we get the result shown in second picture.

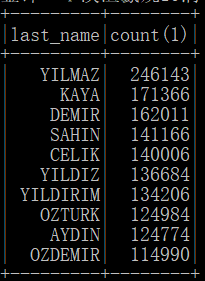


E5. 

N1:use a simple SQL query we can solve this problem:

select last\_name, count(1) from Table where gender='E' group by last\_name order by count(1) desc

We just need to change gender=’E’ into gender=’K’ then we can solve the rest half question.

For male, we get: 

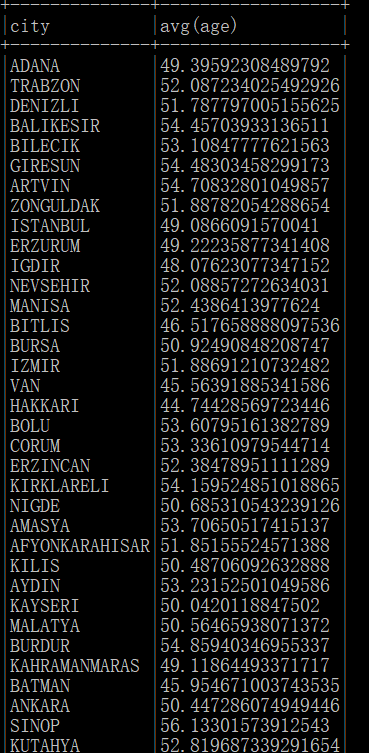
And for female: 

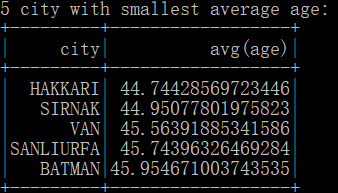
N2+N3:

We write these two questions together in one python file, because they are very similar.

For question N2, I just show part of the results, since the results have so much lines.

For question N3, we can pick up answer from N2 and sort them, then show the top5 use dataframe.show(5).

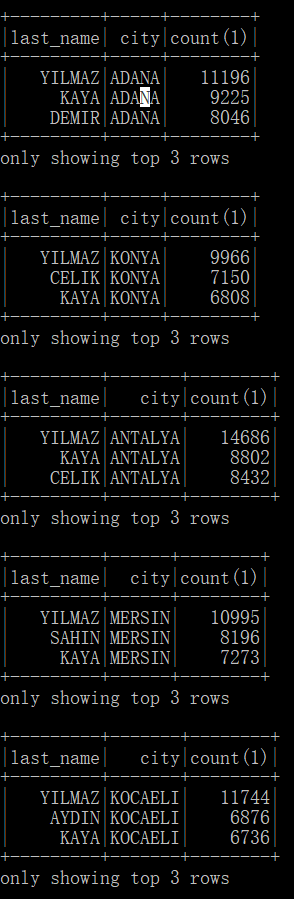
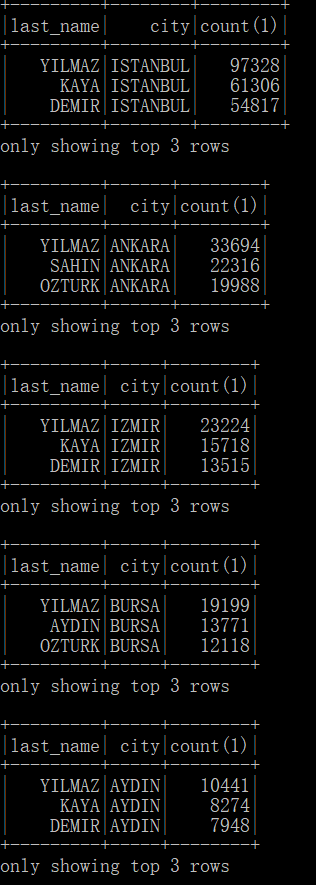




N4:

In this question, we first select 10 cities with most population, and save them in python. Then use a ‘for’ loop in python, doing a SQL query in each loop, and select most popular last name for each city.

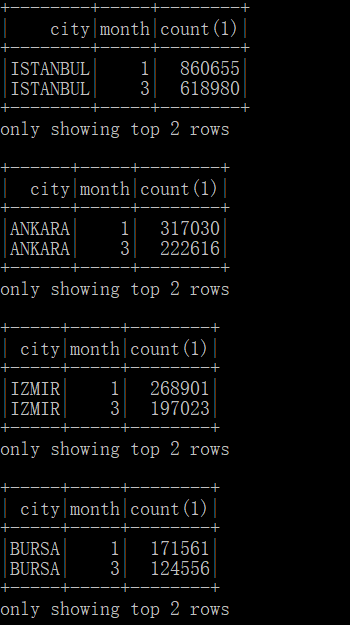
Here is the answer. We can find that the answer for each city is quite same.



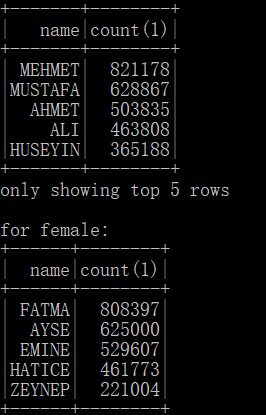
N5:

In N5, we use a same strategy as used in N4. We just need to change a little in the inner loop SQL query.

Here is part of the results. We can find that for each city, result is quite the same.



N6: This problem is quite similar as N1, and we don’t need to talk about it for more.



N7:

This problem is quite same as N4. However, we try to solve this problem with a pure SQL method and without using python ‘for’ loop, which will made it more concise. Instead, we use multi-layer SQL query to solve this question. Our SQL query and answer is shown below.

