Spark Project Report 1

Shun Zhang
School of Data Science
Fudan University
Shanghai, China
15300180012@fudan.edu.cn

1 E Problems

1.1 Problem E1

analysis Here our goal is to find the oldest man within the records available. This problem can be divided into two sub-problems:

- find all the MAN
- find the oldest one

and their relating solutions in pySpark requires no shuffle dependency. The *order key* is to find the one with the earliest birth date, where we find

$$old = argmin_{record}\{1000 * year + 50 * month + day\}$$

result There are a few records, of which the date of birth is censored, such as '//1932'. As for these censored record, we fill them based on the 'as latest as possible' rule. For example, '//1932' is filled as '31/12/1932', '2/3/' is filled as '2/3/2018'. By doing this, we have a greater chance to find the oldest man.

Finally, we find the target record as follows:

33475111 24422710170 MEHMET ALI CEVIK AYYUS MEHMET E OGUZELI //1330 KILIS ELBEYLI KILIS ELBEYLI DOGAN MAH. GUL SOKAK 7 <NULL>

Note that the man we found is about 700 years old if he is still alive, which is impossible as far as I know. However, here we have no idea about whether a citizen is alive or not. So, we just treat them all as alive.

1.2 Problem E2

analysis Here, our goal is to find the most popular letters in one's NAME. This problem can be divided into two sub-problems:

- split every name into letters (flatmap)
- count every letter and put them in order

and we only need one shuffle dependency: *reduceByKey*.

result Note that here we treat NAME as both first name and last name. Also, we treat 'most popular' as top-3. The results are:

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Letter	A	E	I
Frequency	57623513	39619607	34493003

1.3 Problem E3

analysis Here our goal is to find the number of people in a certain range of age. This problem is quite similar to problem E2.

result An interesting phenomenon is that there is no people under 18 in this dataset. The results are: Here we only need one shuffle dependency: *reduceByKey* and also there are other people who

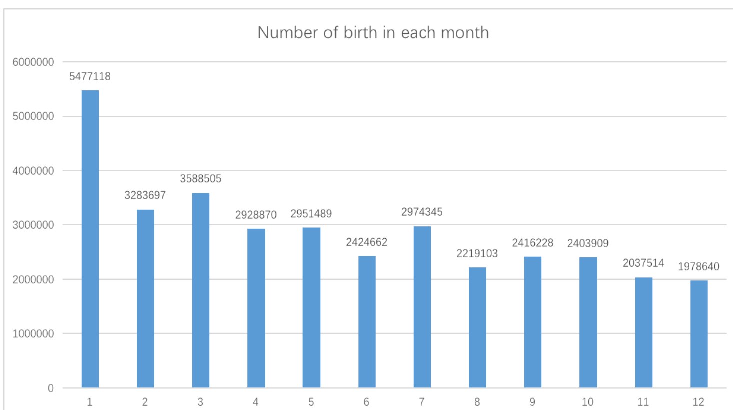
Age	0-18	19-28	29-38	49-55	>60
Frequency	0	1130969	8881641	4694314	9555000

age in range 39-48 or 56-60, of which the frequency is 10465669.

1.4 Problem E4

analysis Here our goal is to find the number of people born in a certain month. This problem is quite similar to problem E3. Also we only need one shuffle dependency: *reduceByKey*.

result There are one record with birth month '14', one record with month '15' and 461 records with month '0'. Also, there are 43050 records, whose birth month is censored. These records are all treated as invalid data. The results of the valid records are:



1.5 Problem E5

analysis Here our goal is to find the number of males and females respectively and then derive the male-female ratio. The only shuffle dependency here is: *reduceByKey*.

result The result is shown in Figure 1 and the male-female ratio is about $1 : 1.0222$.

2 N Problems

2.1 Problem N1

analysis Here our goal is to find the top-10 last name of males and females respectively. This problem is quite similar to problem E2.

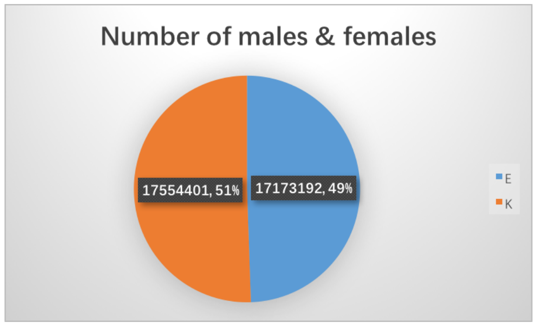
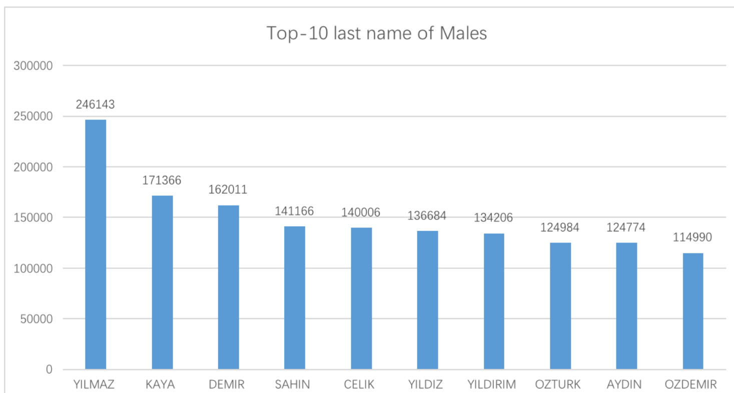


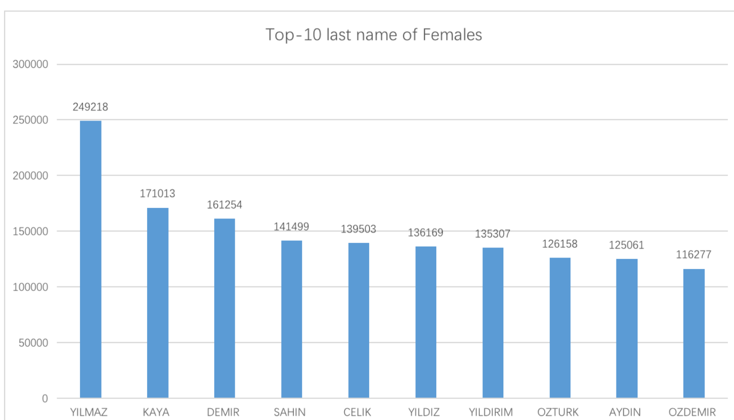
Figure 1: Total number of the males and females

result It is interesting that the results for male and female are quite similar. And the only shuffle dependency is: *reduceByKey*.

- Male



- Female



2.2 Problem N2

analysis Here, our goal is to find the average age of each city. This job can be divided into two steps:

- get the sum of ages of a certain city's citizens

- get the total number of the citizens above

The two steps can be integrated into a triad: (city, sum of age, total number), which can be derived by 'reduceByKey' in PySpark.

result The results are shown in an ascending order in Figure 2 and the only shuffle dependency is: *reduceByKey*.

City	Ave-age	City	Ave-age	City	Ave-age	City	Ave-age
HAKKARI	44.7442857	ADANA	49.3959231	SIVAS	51.7075529	YALOVA	52.7348006
SIRNAK	44.950778	HATAY	49.4626505	KIRIKKALE	51.7238296	TUNCELI	52.751176
VAN	45.5639189	AKSARAY	49.7481159	DENIZLI	51.787797	KUTAHYA	52.8196873
SANLIURFA	45.7439633	OSMANIYE	49.7561042	AFYONKAR	51.8515552	BILECIK	53.1084778
BATMAN	45.954671	KAYSERI	50.0420119	IZMIR	51.8869121	ORDU	53.1529704
MUS	46.1040027	ANTALYA	50.3358515	ZONGULDA	51.8878205	AYDIN	53.231525
AGRI	46.1168854	ELAZIG	50.3551588	SAMSUN	51.9138394	CORUM	53.3361098
DIYARBAKIR	46.1749691	MERSIN	50.4427965	ARDAHAN	51.9944242	BOLU	53.6079516
BITLIS	46.5176589	ANKARA	50.4472861	ESKISEHIR	52.044138	AMASYA	53.7065052
MARDIN	46.7418903	KILIS	50.4870609	TRABZON	52.087234	BARTIN	53.8919863
SIIRT	46.7631558	MALATYA	50.5646594	NEVSEHIR	52.0885727	KARABUK	53.9272383
BINGOL	47.4114321	KONYA	50.5664383	KIRSEHIR	52.124991	KIRKLARELI	54.1595249
GAZIANTEP	47.5827868	NIGDE	50.6853105	RIZE	52.2718554	EDIRNE	54.3379699
IGDIR	48.0762308	TEKIRDAG	50.9061788	ERZINCAN	52.3847895	BALIKESIR	54.4570393
ADIYAMAN	48.2627925	BURSA	50.9249085	MUGLA	52.3891829	GIRESUN	54.4830346
ISTANBUL	49.0866092	SAKARYA	51.1943062	MANISA	52.4386414	ARTVIN	54.708328
KAHRAMAN	49.1186449	YOZGAT	51.4480915	USAK	52.5669766	CANAKKALE	54.8514519
KARS	49.1844636	BAYBURT	51.5405421	GUMUSHAN	52.6215345	BURDUR	54.8594035
ERZURUM	49.2223588	KARAMAN	51.6071412	TOKAT	52.6215995	CANKIRI	54.9514408
KOCAELI	49.3066701	DUZCE	51.6095659	ISPARTA	52.6325023	KASTAMON	55.8195321
						SINOP	56.1330157

Figure 2: Average age of each city.

2.3 Problem N3

analysis Here our goal is to find the top-5 cities with low average age. And the only shuffle dependency is: *reduceByKey*.

result The result can be easily derived from Figure 2, which is *HAKKARI, SIRNAK, VAN, SANLIURFA and BATMAN*.

2.4 Problem N4

analysis Here our goal is to find the top-3 popular last name in the top-10 cities in population. My solution is quite simple that

1. find the top-10 cities in population
2. find the top-3 popular last name for each city

which requires two shuffle dependencies: *reduceByKey* and *groupByKey*.

result The results are shown in Table 1.

2.5 Problem N5

analysis Here our goal is to find the top-2 popular birth month in the top-10 cities in population. This problem is quite similar to problem N4, which requires two shuffle dependencies: *reduceByKey* and *groupByKey*.

result The results are shown in Table 2. An interesting thing is that the top-2 popular birth months are the same among these ten cities.

Table 1: Top-3 popular last name in the top-10 cities in population

City	Last Name 1(Num)	Last Name 2(Num)	Last Name 3(Num)
ISTANBUL	YILMAZ(97328)	KAYA(61306)	DEMIR(54817)
ANKARA	YILMAZ(33694)	SAHIN(22316)	OZTURK(19988)
IZMIR	YILMAZ(23224)	KAYA(15718)	DEMIR(13515)
BURSA	YILMAZ(19199)	AYDIN(13771)	OZTURK(12118)
AYDIN	YILMAZ(10441)	KAYA(8274)	DEMIR(7948)
ADANA	YILMAZ(11196)	KAYA(9225)	DEMIR(8046)
KONYA	YILMAZ(9966)	CELIK(7150)	KAYA(6808)
ANTALYA	YILMAZ(14686)	KAYA(8802)	CELIK(8432)
MERSIN	YILMAZ(10995)	SAHIN(8196)	KAYA(7273)
KOCAELI	YILMAZ(11744)	AYDIN(6876)	KAYA(6736)

Table 2: Top-2 popular birth month in the top-10 cities in population

City	Birth month 1(Num)	Birth month 2(Num)
ISTANBUL	Jan.(860655)	Mar.(618980)
ANKARA	Jan.(317030)	Mar.(222616)
IZMIR	Jan.(268901)	Mar.(197023)
BURSA	Jan.(171561)	Mar.(124556)
AYDIN	Jan.(135393)	Mar.(100959)
ADANA	Jan.(193277)	Mar.(94205)
KONYA	Jan.(143228)	Mar.(96847)
ANTALYA	Jan.(145205)	Mar.(94399)
MERSIN	Jan.(132807)	Mar.(77870)
KOCAELI	Jan.(96775)	Mar.(73050)