Kseniia Romanova, Keaton Mangone, Aidan Syrgak Uulu

CS/DS 4433 - Big Data Management & Analytics
Project 3 - Report

Install and Execution Instructions

The project should be fully functional after reloading Maven. The queries were tested locally using small datasets and the screenshots are presented for them. The scala files are located in *src/main/scala*.

For creation of datasets a class named **dataGenerator.java** was created. It's located in *src/main/java/dataGenerator.java*. The datasets are created by running main inside this class and by changing variables inside it:

- **sizePeople** => # tuples in PEOPLE-large.csv
- **sizeInfected** => # tuples in INFECTED-small.csv
- **maxValueXY** => upper bound for x and y coordinate values
- **sizeCustomers** => # tuples in customers.csv
- **sizePurchases** => # tuples in purchases.csv

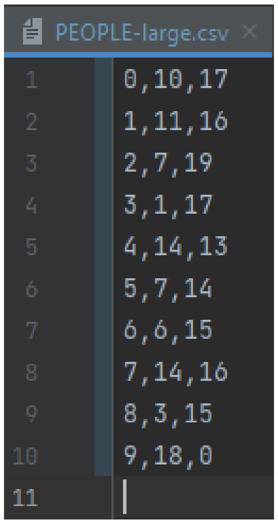
All csv files will be created at once. Files for the first part of the project will be stored in *src/main/data/people* and for the second part in *src/main/data/transactions*. Everything will be run locally using main functions.

Queries and tasks were also tested on big datasets provided in the data folder. Screenshots with the first couple of resulted tuples are provided. Please adjust the size of datasets if needed, since the solutions were run on a powerful laptop and still took a while.

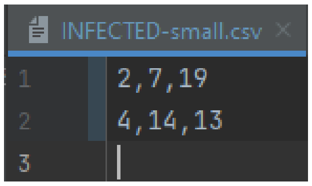
Analytics Queries Logistics

1. **Query 1**

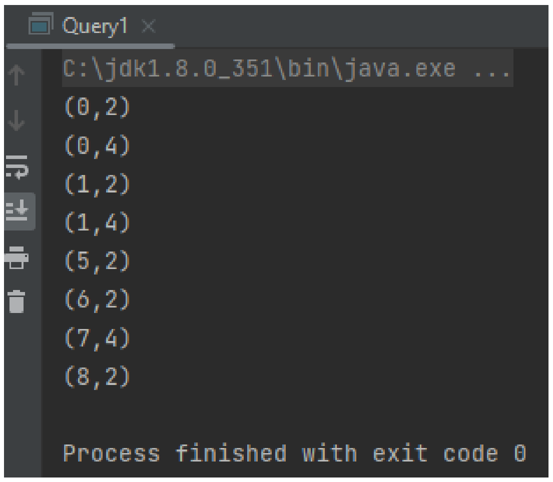
   Two files were loaded into the system first: PEOPLE-large.csv and INFECTED-small.csv. Then a broadcast variable was created for caching on each machine rather than shipping a copy of it with tasks. This can reduce communication cost. After that, we applied filtering to compute close contacts by iterating through each person in the people file and calculating the distance. If the distance is <= 6 feet to the infected person, then we want to store both healthy and infected people's ids. We then filter out pairs of ids where both ids are the same and output the result in the form (healthy person, infected person).



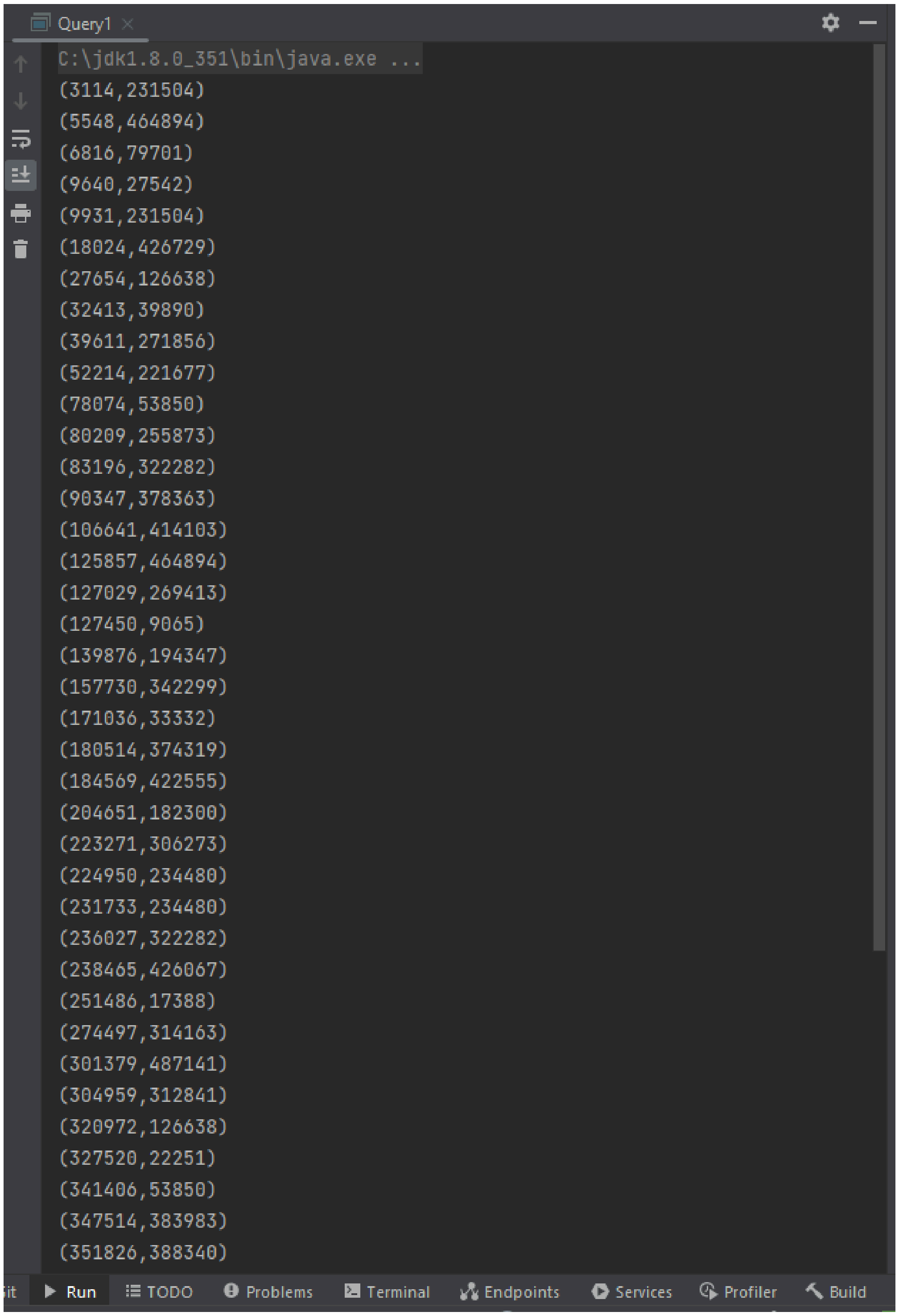
```
0,10,17
1,11,16
2,7,19
3,1,17
4,14,13
5,7,14
6,6,15
7,14,16
8,3,15
9,18,0
```



```
2,7,19
4,14,13
```



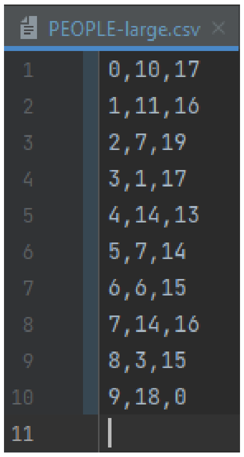
```
C:\jdk1.8.0_351\bin\java.exe ...
(0,2)
(0,4)
(1,2)
(1,4)
(5,2)
(6,2)
(7,4)
(8,2)

Process finished with exit code 0
```



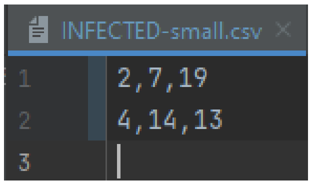
```
C:\jdk1.8.0_351\bin\java.exe ...
(3114,231504)
(5548,464894)
(6816,79701)
(9640,27542)
(9931,231504)
(18024,426729)
(27654,126638)
(32413,39890)
(39611,271856)
(52214,221677)
(78074,53850)
(80209,255873)
(83196,322282)
(90347,378363)
(106641,414103)
(125857,464894)
(127029,269413)
(127450,9065)
(139876,194347)
(157730,342299)
(171036,33332)
(180514,374319)
(184569,422555)
(204651,182300)
(223271,306273)
(224950,234480)
(231733,234480)
(236027,322282)
(238465,426067)
(251486,17388)
(274497,314163)
(301379,487141)
(304959,312841)
(320972,126638)
(327520,22251)
(341406,53850)
(347514,383983)
(351826,388340)
```

2. Query 2

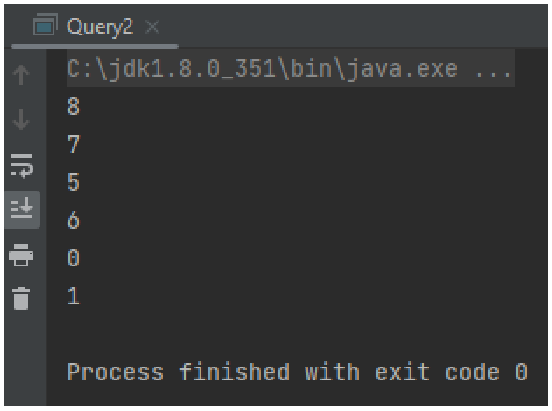
Similarly to Query 1, two files were loaded into the system: PEOPLE-large.csv and INFECTED-small.csv. Then a broadcast variable was created for caching on each machine to reduce communication cost. After that, close contacts were calculated by going through each person in the people file and filtering based on distance and whether the ids are the same. Then a distinct function was applied to remove duplicates.



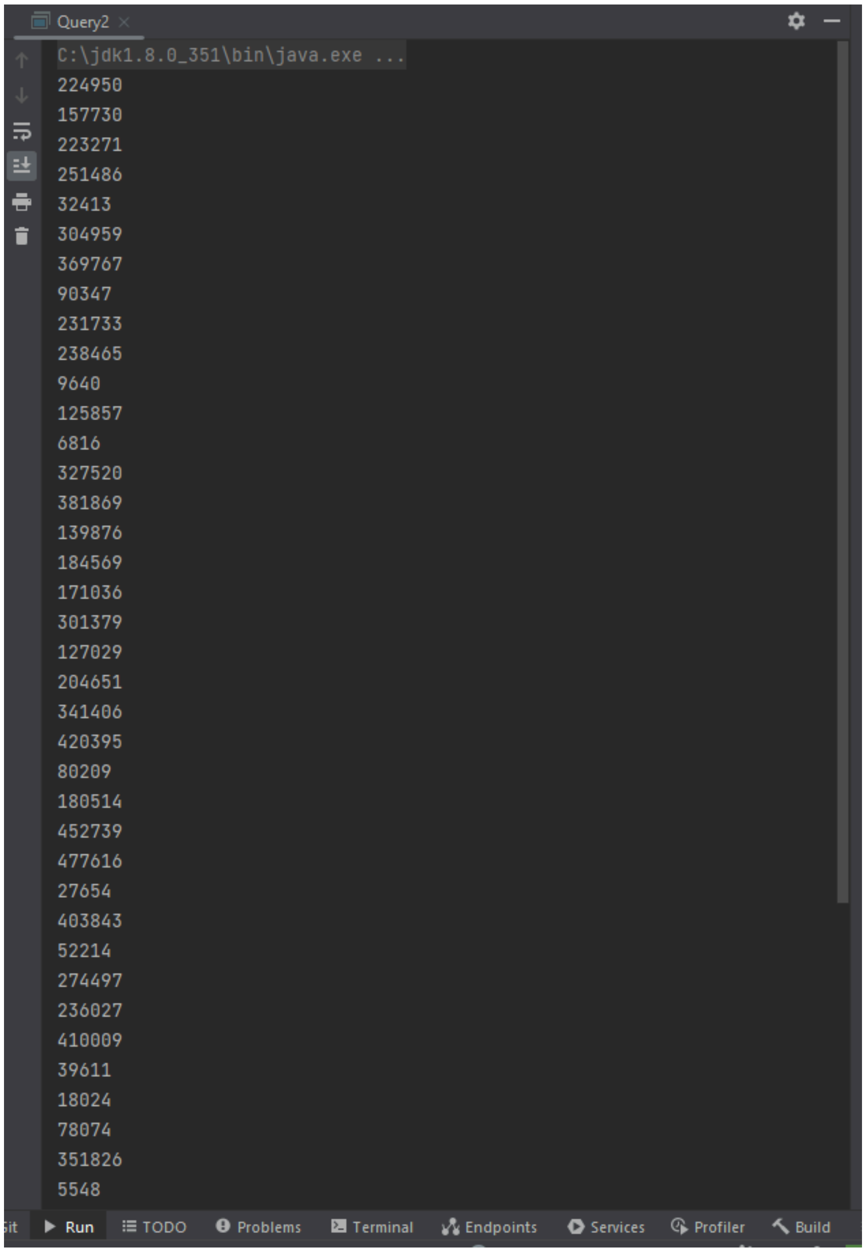
```
0,10,17
1,11,16
2,7,19
3,1,17
4,14,13
5,7,14
6,6,15
7,14,16
8,3,15
9,18,0
```



```
2,7,19
4,14,13
```



```
C:\jdk1.8.0_351\bin\java.exe ...
8
7
5
6
0
1

Process finished with exit code 0
```

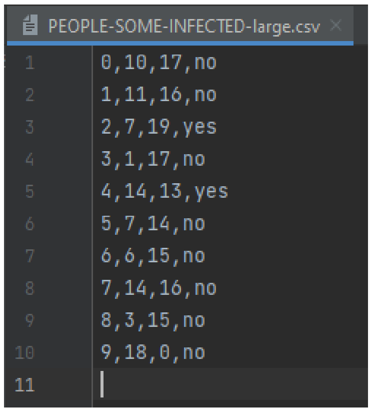


```
C:\jdk1.8.0_351\bin\java.exe ...
224950
157730
223271
251486
32413
304959
369767
90347
231733
238465
9640
125857
6816
327520
381869
139876
184569
171036
301379
127029
204651
341406
420395
80209
180514
452739
477616
27654
403843
52214
274497
236027
410009
39611
18024
78074
351826
5548
```

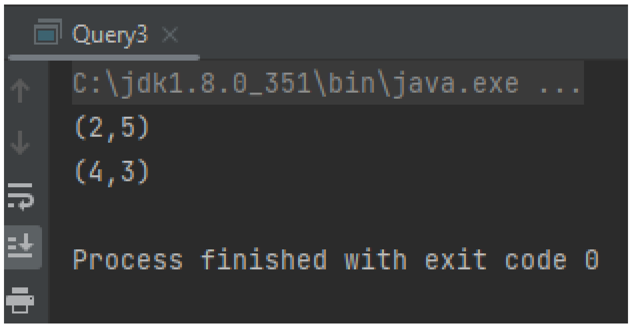
3. **Query 3**

One file is loaded into the system: PEOPLE-SOME-INFECTED-large.csv. Infected people are determined by using a filter function, then, a broadcast variable is created for the large dataset. After that, a map is applied to determine for each infected person how many people from all people were seated within 6 feet of the given infected person by applying filters and counting affected people. Return the (infected person's id, count).

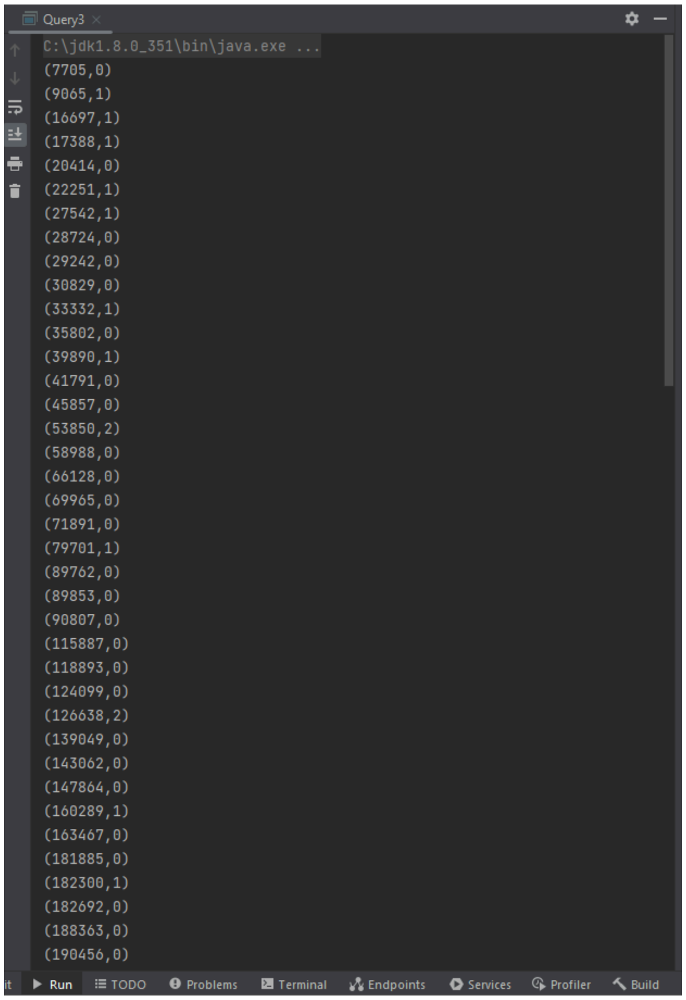
Query3 and Query3_2 are just two versions of the same query, both are correct.



```
0,10,17,no
1,11,16,no
2,7,19,yes
3,1,17,no
4,14,13,yes
5,7,14,no
6,6,15,no
7,14,16,no
8,3,15,no
9,18,0,no
```



```
C:\jdk1.8.0_351\bin\java.exe ...
(2,5)
(4,3)

Process finished with exit code 0
```

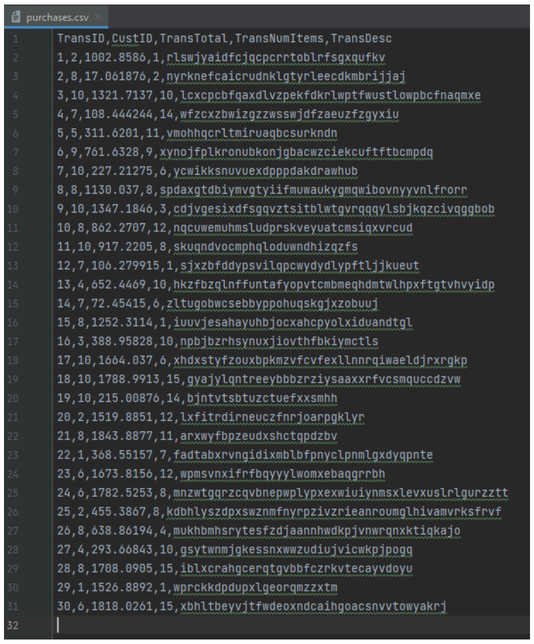


```
C:\jdk1.8.0_351\bin\java.exe ...
(7705,0)
(9065,1)
(16697,1)
(17388,1)
(20414,0)
(22251,1)
(27542,1)
(28724,0)
(29242,0)
(30829,0)
(33332,1)
(35802,0)
(39890,1)
(41791,0)
(45857,0)
(53850,2)
(58988,0)
(66128,0)
(69965,0)
(71891,0)
(79701,1)
(89762,0)
(89853,0)
(90807,0)
(115887,0)
(118893,0)
(124099,0)
(126638,2)
(139049,0)
(143062,0)
(147864,0)
(160289,1)
(163467,0)
(181885,0)
(182300,1)
(182692,0)
(188363,0)
(190456,0)
```

4. **Task A.1**

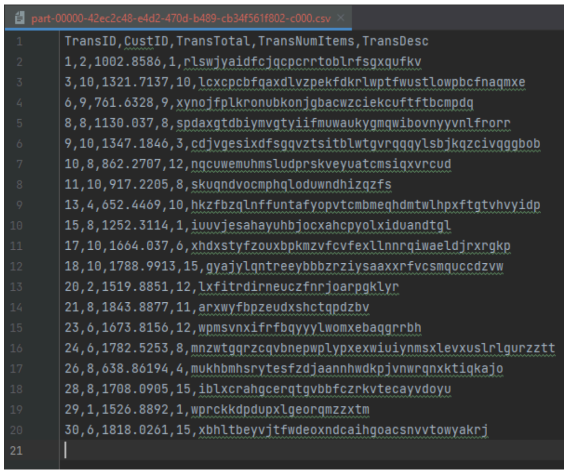
All tasks from option A were done both in Scala and Python. You could choose whichever version is easier for you to run 🙂. The Python version was run in PyCharm.

TaskA1 in Scala starts with getting or creating a spark session. Then, purchases.csv file is read into the system and a temporary view is created. After that, we use the sql() function to select purchases which have TransTotal > 600 and write it to output/T1.csv.



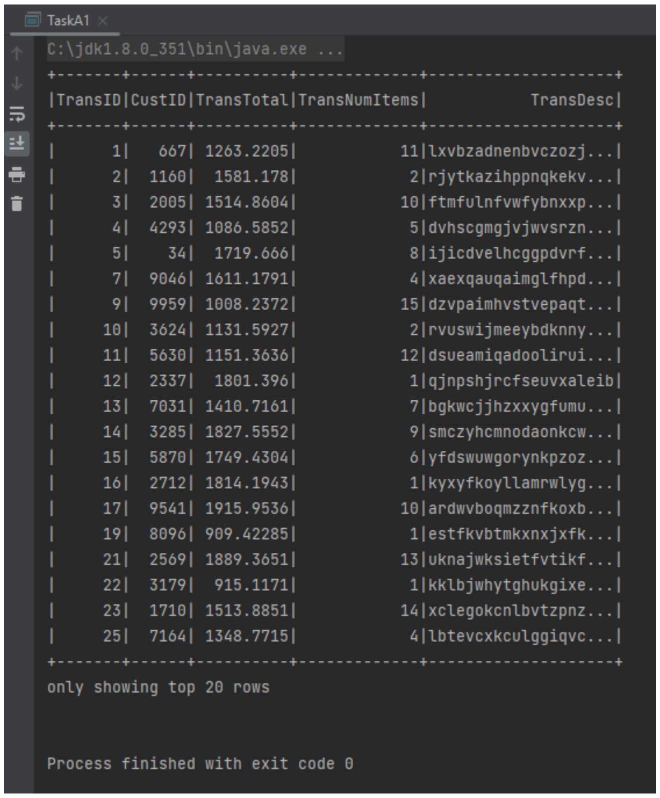
purchases.csv

```
TransID,CustID,TransTotal,TransNumItems,TransDesc
1,2,1002.8586,1,rlswjyaidfcjqcpcrrtoblrfsgxqufkv
2,8,17.061876,2,nyrknefcaicrudnklgtyrleecdkmbrijjaj
3,10,1321.7137,10,lcxcpcbfqaxdlvzpekfdkrlwptfwustlowpbcfnaqmxe
4,7,108.444244,14,wfzcxzbwizgzzwsswjdfzaeuzfzgyxiu
5,5,311.6201,11,vmohhqcrltmiruaqbcsurkndn
6,9,761.6328,9,xynojfplkronubkonjgbacwzciekcuftftbcmpdq
7,10,227.21275,6,ycwikksnuvuexdpppdakdrawhub
8,8,1130.037,8,spdaxgtdbiymvgtyiifmuwaukygmqwibovnyyvnlfrorr
9,10,1347.1846,3,cdjvgesixdfsgqvztsitblwtgvrqqqylsbjkqzcivqggbob
10,8,862.2707,12,nqcuwemuhmsludprskveyuatcmsiqxvrcud
11,10,917.2205,8,skuqndvocmphqloduwndhizqzfs
12,7,106.279915,1,sjxzbfddypsvilqpcwydydlypftljjkueut
13,4,652.4469,10,hkzfbzqlnffuntafyopvtcmbmeqhdmtwlhpxftgtvhvyidp
14,7,72.45415,6,zltugobwcsebbyppohuqskgjxzobuuj
15,8,1252.3114,1,iuuvjesahayuhbjocxahcpyolxiduandtgl
16,3,388.95828,10,npbjbzrhsynuxjiovthfbkiymctls
17,10,1664.037,6,xhdxstyfzouxbpkmzvfcvfexllnnrqiwaeldjrxrgkp
18,10,1788.9913,15,gyajylqntreeybbbzrziysaaxxrfvcsmquccdzvw
19,10,215.00876,14,bjntvtsbtuzctuefxxsmhh
20,2,1519.8851,12,lxfitrdirneuczfnrjoarpgklyr
21,8,1843.8877,11,arxwyfbpzeudxshctqpdzbv
22,1,368.55157,7,fadtabxrvngidixmblbfpnyclpnmlgxdyqpnte
23,6,1673.8156,12,wpmsvnxifrfbqyyylwomxebaqgrrbh
24,6,1782.5253,8,mnzwtgqrzcqvbnepwplypxexwiuiynmsxlevxuslrlgurzztt
25,2,455.3867,8,kdbhlyszdpxswznmfnyrpzivzrieanroumglhivamvrksfrvf
26,8,638.86194,4,mukhbmhsrytesfzdjaannhwdkpjvnwrqnxktiqkajo
27,4,293.66843,10,gsytwnmjgkessnxwwzudiujvicwkpjpogq
28,8,1708.0905,15,iblxcrahgcerqtgvbbfczrkvtecayvdoyu
29,1,1526.8892,1,wprckkdpdupxlgeorqmzzxtm
30,6,1818.0261,15,xbhltbeyvjtfwdeoxndcaihgoacsnvvtowyakrj
```



part-00000-42ec2c48-e4d2-470d-b489-cb34f561f802-c000.csv

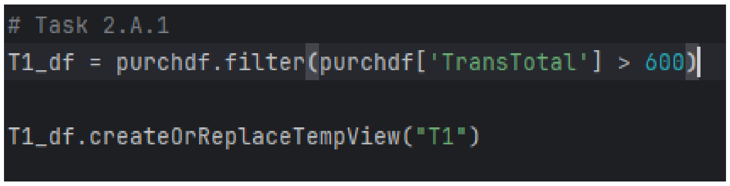
```
TransID,CustID,TransTotal,TransNumItems,TransDesc
1,2,1002.8586,1,rlswjyaidfcjqcpcrrtoblrfsgxqufkv
3,10,1321.7137,10,lcxcpcbfqaxdlvzpekfdkrlwptfwustlowpbcfnaqmxe
6,9,761.6328,9,xynojfplkronubkonjgbacwzciekcuftftbcmpdq
8,8,1130.037,8,spdaxgtdbiymvgtyiifmuwaukygmqwibovnyyvnlfrorr
9,10,1347.1846,3,cdjvgesixdfsgqvztsitblwtgvrqqqylsbjkqzcivqggbob
10,8,862.2707,12,nqcuwemuhmsludprskveyuatcmsiqxvrcud
11,10,917.2205,8,skuqndvocmphqloduwndhizqzfs
13,4,652.4469,10,hkzfbzqlnffuntafyopvtcmbmeqhdmtwlhpxftgtvhvyidp
15,8,1252.3114,1,iuuvjesahayuhbjocxahcpyolxiduandtgl
17,10,1664.037,6,xhdxstyfzouxbpkmzvfcvfexllnnrqiwaeldjrxrgkp
18,10,1788.9913,15,gyajylqntreeybbbzrziysaaxxrfvcsmquccdzvw
20,2,1519.8851,12,lxfitrdirneuczfnrjoarpgklyr
21,8,1843.8877,11,arxwyfbpzeudxshctqpdzbv
23,6,1673.8156,12,wpmsvnxifrfbqyyylwomxebaqgrrbh
24,6,1782.5253,8,mnzwtgqrzcqvbnepwplypxexwiuiynmsxlevxuslrlgurzztt
26,8,638.86194,4,mukhbmhsrytesfzdjaannhwdkpjvnwrqnxktiqkajo
28,8,1708.0905,15,iblxcrahgcerqtgvbbfczrkvtecayvdoyu
29,1,1526.8892,1,wprckkdpdupxlgeorqmzzxtm
30,6,1818.0261,15,xbhltbeyvjtfwdeoxndcaihgoacsnvvtowyakrj
```



TaskA1

```
C:\jdk1.8.0_351\bin\java.exe ...
+-------+------+----------+-------------+--------------------+
|TransID|CustID|TransTotal|TransNumItems|           TransDesc|
+-------+------+----------+-------------+--------------------+
|      1|   667| 1263.2205|           11|lxvbzadnenbvczozj...|
|      2|  1160|  1581.178|            2|rjytkazihppnqkekv...|
|      3|  2005| 1514.8604|           10|ftmfulnfvwfybnxxp...|
|      4|  4293| 1086.5852|            5|dvhscgmgjvjwvsrzn...|
|      5|    34|  1719.666|            8|ijicdvelhcggpdvrf...|
|      7|  9046| 1611.1791|            4|xaexqauqaimglfhpd...|
|      9|  9959| 1008.2372|           15|dzvpaimhvstvepaqt...|
|     10|  3624| 1131.5927|            2|rvuswijmeeybdknny...|
|     11|  5630| 1151.3636|           12|dsueamiqadoolirui...|
|     12|  2337|  1801.396|            1|qjnpshjrcfseuvxaleib|
|     13|  7031| 1410.7161|            7|bgkwcjjhzxxygfumu...|
|     14|  3285| 1827.5552|            9|smczyhcmnodaonkcw...|
|     15|  5870| 1749.4304|            6|yfdswuwgorynkpzoz...|
|     16|  2712| 1814.1943|            1|kyxyfkoyllamrwlyg...|
|     17|  9541| 1915.9536|           10|ardwvboqmzznfkoxb...|
|     19|  8096| 909.42285|            1|estfkvbtmkxnxjxfk...|
|     21|  2569| 1889.3651|           13|uknajwksietfvtikf...|
|     22|  3179|  915.1171|            1|kklbjwhytghukgixe...|
|     23|  1710| 1513.8851|           14|xclegokcnlbvtzpnz...|
|     25|  7164| 1348.7715|            4|lbtevcxkculggiqvc...|
+-------+------+----------+-------------+--------------------+
only showing top 20 rows


Process finished with exit code 0
```

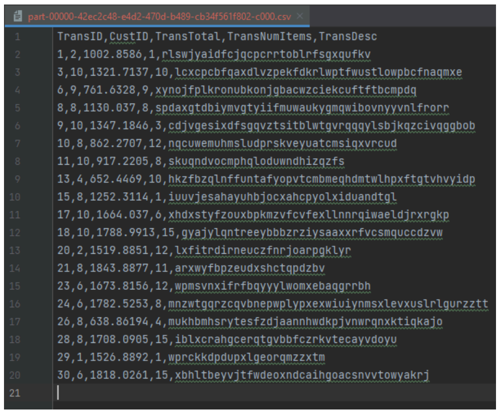
For the Python version, we read in our generated dataset “purchases.csv” and store it in datafram purchdf. Then, using the filter() function, we filtered this dataframe by Transaction Totals > 600, and stored this in new datafram “T1_df”. The result is stored as view T1 to be used in the next steps.



```
# Task 2.A.1
T1_df = purchdf.filter(purchdf['TransTotal'] > 600)

T1_df.createOrReplaceTempView("T1")
```

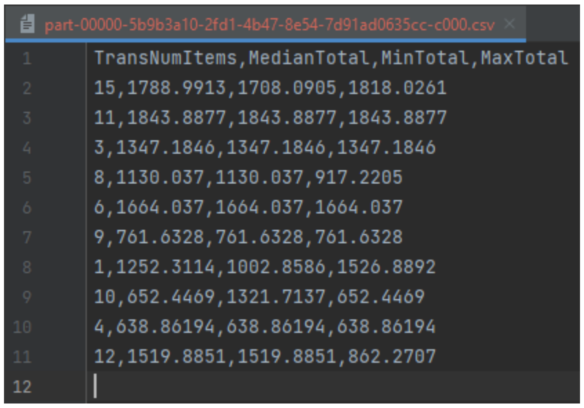
5. **Task A.2**

TaskA2 in Scala uses approximate median computation, which, in case there is an even amount of numbers, doesn’t compute the average of two middle numbers and returns only one of them instead. First, the spark session is set up and a T1.csv is read into the system. Then, the tuples are grouped by TransNumItems value, and median, min, and max are computed using percentile_approx(), as well as min() and max() functions. The resulting tuples are returned and stored as T2.csv.



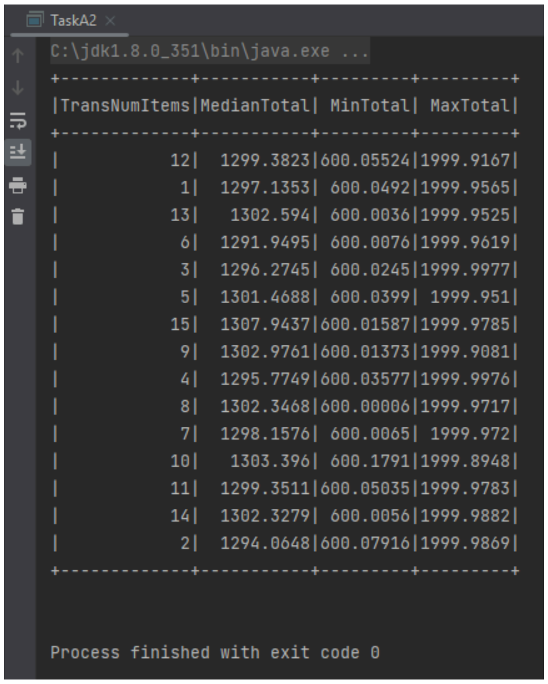
part-00000-42ec2c48-e4d2-470d-b489-cb34f561f802-c000.csv

```
TransID,CustID,TransTotal,TransNumItems,TransDesc
1,2,1002.8586,1,rlswjyaidfcjqcpcrrtoblrfsgxqufkv
3,10,1321.7137,10,lcxcpcbfqaxdlvzpekfdkrlwptfwustlowpbcfnaqmxe
6,9,761.6328,9,xynojfplkronubkonjgbacwzciekcuftftbcmpdq
8,8,1130.037,8,spdaxgtdbiymvgtyiifmuwaukygmqwibovnyyvnlfrorr
9,10,1347.1846,3,cdjvgesixdfsgqvztsitblwtgvrqqqylsbjkqzcivqggbob
10,8,862.2707,12,nqcuwemuhmsludprskveyuatcmsiqxvrcud
11,10,917.2205,8,skuqndvocmphqloduwndhizqzfs
13,4,652.4469,10,hkzfbzqlnffuntafyopvtcmbmeqhdmtwlhpxftgtvhvyidp
15,8,1252.3114,1,iuuvjesahayuhbjocxahcpyolxiduandtgl
17,10,1664.037,6,xhdxstyfzouxbpkmzvfcvfexllnnrqiwaeldjrxrgkp
18,10,1788.9913,15,gyajylqntreeybbbzrziysaaxxrfvcsmquccdzvw
20,2,1519.8851,12,lxfitrdirneuczfnrjoarpgklyr
21,8,1843.8877,11,arxwyfbpzeudxshctqpdzbv
23,6,1673.8156,12,wpmsvnxifrfbqyyylwomxebaqgrrbh
24,6,1782.5253,8,mnzwtgqrzcqvbnepwplypxexwiuiynmsxlevxuslrlgurzztt
26,8,638.86194,4,mukhbmhsrytesfzdjaannhwdkpjvnwrqnxktiqkajo
28,8,1708.0905,15,iblxcrahgcerqtgvbbfczrkvtecayvdoyu
29,1,1526.8892,1,wprckkdpdupxlgeorqmzzxtm
30,6,1818.0261,15,xbhltbeyvjtfwdeoxndcaihgoacsnvvtowyakrj
```



part-00000-5b9b3a10-2fd1-4b47-8e54-7d91ad0635cc-c000.csv

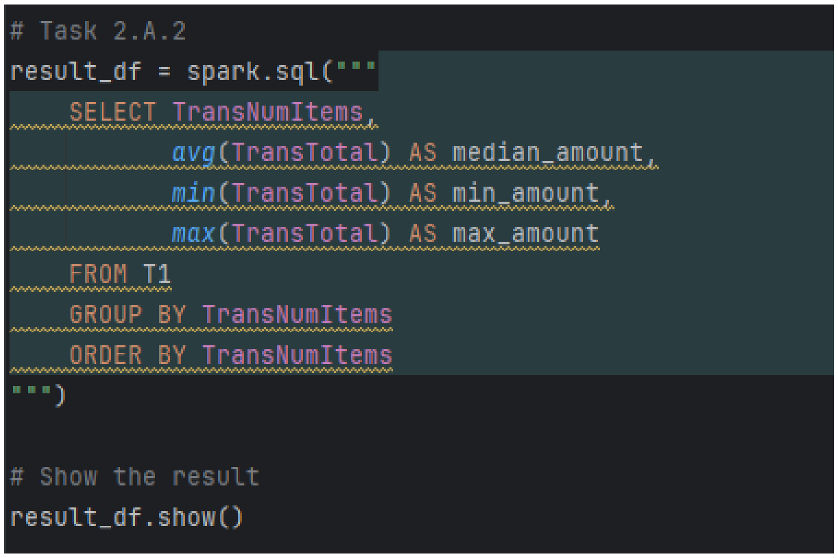
```
TransNumItems,MedianTotal,MinTotal,MaxTotal
15,1788.9913,1708.0905,1818.0261
11,1843.8877,1843.8877,1843.8877
3,1347.1846,1347.1846,1347.1846
8,1130.037,1130.037,917.2205
6,1664.037,1664.037,1664.037
9,761.6328,761.6328,761.6328
1,1252.3114,1002.8586,1526.8892
10,652.4469,1321.7137,652.4469
4,638.86194,638.86194,638.86194
12,1519.8851,1519.8851,862.2707
```



TaskA2

```
C:\jdk1.8.0_351\bin\java.exe ...
+-------------+-----------+---------+---------+
|TransNumItems|MedianTotal| MinTotal| MaxTotal|
+-------------+-----------+---------+---------+
|           12|  1299.3823|600.05524|1999.9167|
|            1|  1297.1353| 600.0492|1999.9565|
|           13|   1302.594| 600.0036|1999.9525|
|            6|  1291.9495| 600.0076|1999.9619|
|            3|  1296.2745| 600.0245|1999.9977|
|            5|  1301.4688| 600.0399| 1999.951|
|           15|  1307.9437|600.01587|1999.9785|
|            9|  1302.9761|600.01373|1999.9081|
|            4|  1295.7749|600.03577|1999.9976|
|            8|  1302.3468|600.00006|1999.9717|
|            7|  1298.1576| 600.0065| 1999.972|
|           10|   1303.396| 600.1791|1999.8948|
|           11|  1299.3511|600.05035|1999.9783|
|           14|  1302.3279| 600.0056|1999.9882|
|            2|  1294.0648|600.07916|1999.9869|
+-------------+-----------+---------+---------+


Process finished with exit code 0
```

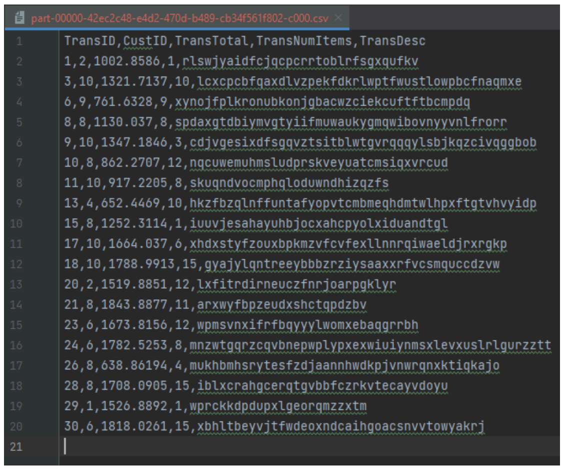
In the Python version, with T1 view created previously, we used spark.sql to create a new dataframe that gets the median, max, and min transaction totals based on groups by number of items purchased in T1. Using sparksql in this case made the most sense to us, as having this query in sql form, rather than say databricks form would be easier to read and write. We then print this dataframe for the user to see.



```python
# Task 2.A.2
result_df = spark.sql("""
    SELECT TransNumItems,
           avg(TransTotal) AS median_amount,
           min(TransTotal) AS min_amount,
           max(TransTotal) AS max_amount
    FROM T1
    GROUP BY TransNumItems
    ORDER BY TransNumItems
""")

# Show the result
result_df.show()
```

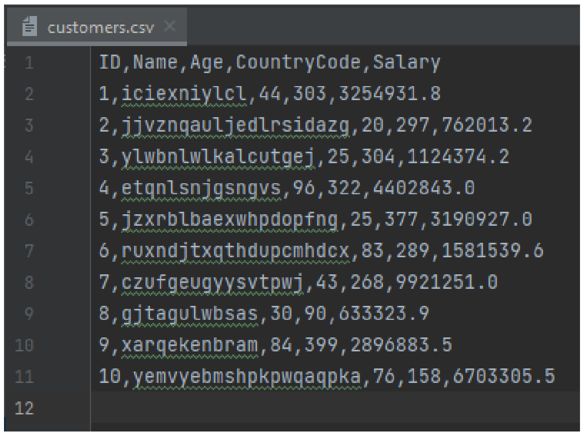
6. **Task A.3**

TaskA3 in Scala sets up a spark session and reads two files into the system first: T1.csv and customers.csv between 18 and 25 years of age. Then, those files are joined on customer's id and a group by function is performed on customer's id and age. After that, a total number of transactions and transactions' totals is calculated. The result is written to the T3.csv file as well as outputted to the console window.



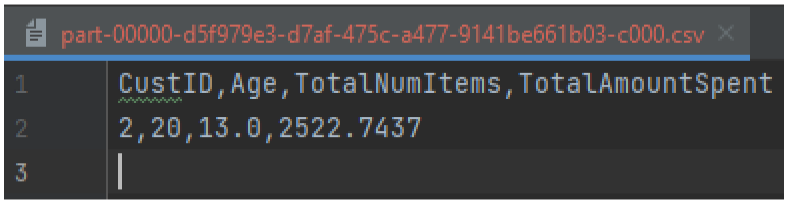
part-00000-42ec2c48-e4d2-470d-b489-cb34f561f802-c000.csv

```
TransID,CustID,TransTotal,TransNumItems,TransDesc
1,2,1002.8586,1,rlswjyaidfcjqcpcrrtoblrfsgxqufkv
3,10,1321.7137,10,lcxcpcbfqaxdlvzpekfdkrlwptfwustlowpbcfnaqmxe
6,9,761.6328,9,xynojfplkronubkonjgbacwzciekcuftftbcmpdq
8,8,1130.037,8,spdaxgtdbiymvgtyiifmuwaukygmqwibovnyyvnlfrorr
9,10,1347.1846,3,cdjvgesixdfsgqvztsitblwtgvrqqqylsbjkqzcivqggbob
10,8,862.2707,12,nqcuwemuhmsludprskveyuatcmsiqxvrcud
11,10,917.2205,8,skuqndvocmphqloduwndhizqzfs
13,4,652.4469,10,hkzfbzqlnffuntafyopvtcmbmeqhdmtwlhpxftgtvhvyidp
15,8,1252.3114,1,iuuvjesahayuhbjocxahcpyolxiduandtgl
17,10,1664.037,6,xhdxstyfzouxbpkmzvfcvfexllnnrqiwaeldjrxrgkp
18,10,1788.9913,15,gyajylqntreeybbbzrziysaaxxrfvcsmquccdzvw
20,2,1519.8851,12,lxfitrdirneuczfnrjoarpgklyr
21,8,1843.8877,11,arxwyfbpzeudxshctqpdzbv
23,6,1673.8156,12,wpmsvnxifrfbqyyylwomxebaqgrrbh
24,6,1782.5253,8,mnzwtgqrzcqvbnepwplypxexwiuiynmsxlevxuslrlgurzztt
26,8,638.86194,4,mukhbmhsrytesfzdjaannhwdkpjvnwrqnxktiqkajo
28,8,1708.0905,15,iblxcrahgcerqtgvbbfczrkvtecayvdoyu
29,1,1526.8892,1,wprckkdpdupxlgeorqmzzxtm
30,6,1818.0261,15,xbhltbeyvjtfwdeoxndcaihgoacsnvvtowyakrj
```



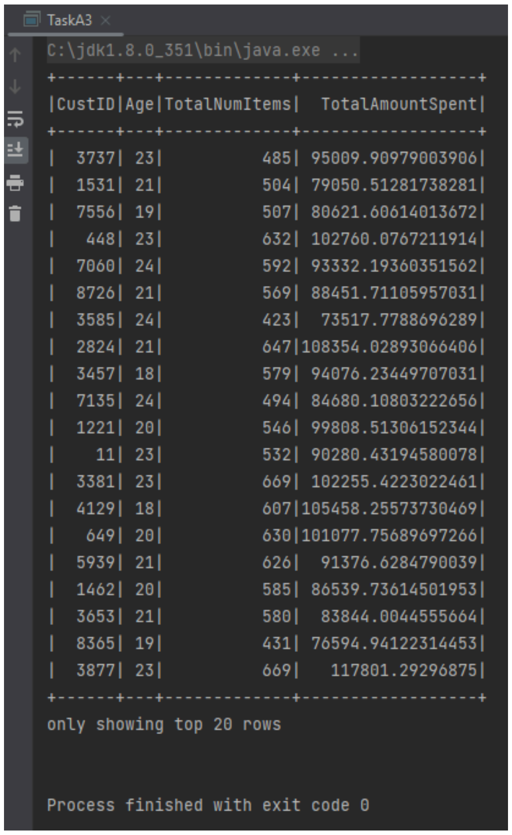
customers.csv

```
ID,Name,Age,CountryCode,Salary
1,iciexniylcl,44,303,3254931.8
2,jjvznqauljedlrsidazg,20,297,762013.2
3,ylwbnlwlkalcutgej,25,304,1124374.2
4,etqnlsnjgsngvs,96,322,4402843.0
5,jzxrblbaexwhpdopfng,25,377,3190927.0
6,ruxndjtxqthdupcmhdcx,83,289,1581539.6
7,czufgeugyysvtpwj,43,268,9921251.0
8,gjtagulwbsas,30,90,633323.9
9,xarqekenbram,84,399,2896883.5
10,yemvyebmshpkpwqaqpka,76,158,6703305.5
```



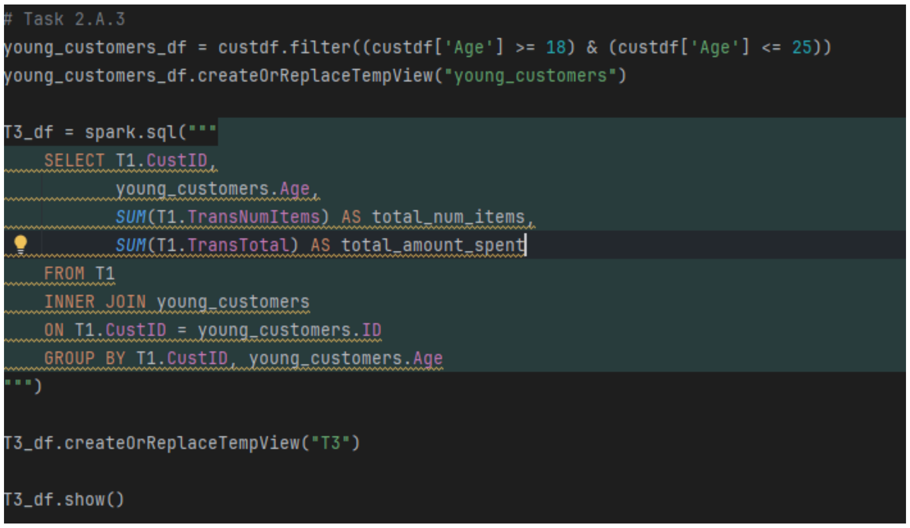
part-00000-d5f979e3-d7af-475c-a477-9141be661b03-c000.csv

```
CustID,Age,TotalNumItems,TotalAmountSpent
2,20,13.0,2522.7437
```



```
C:\jdk1.8.0_351\bin\java.exe ...
+------+---+------------+------------------+
|CustID|Age|TotalNumItems|  TotalAmountSpent|
+------+---+------------+------------------+
|  3737| 23|         485| 95009.90979003906|
|  1531| 21|         504| 79050.51281738281|
|  7556| 19|         507| 80621.60614013672|
|   448| 23|         632| 102760.0767211914|
|  7060| 24|         592| 93332.19360351562|
|  8726| 21|         569| 88451.71105957031|
|  3585| 24|         423|  73517.7788696289|
|  2824| 21|         647|108354.02893066406|
|  3457| 18|         579| 94076.23449707031|
|  7135| 24|         494| 84680.10803222656|
|  1221| 20|         546| 99808.51306152344|
|    11| 23|         532| 90280.43194580078|
|  3381| 23|         669| 102255.4223022461|
|  4129| 18|         607|105458.25573730469|
|   649| 20|         630|101077.75689697266|
|  5939| 21|         626|  91376.6284790039|
|  1462| 20|         585| 86539.73614501953|
|  3653| 21|         580|  83844.0044555664|
|  8365| 19|         431| 76594.94122314453|
|  3877| 23|         669|   117801.29296875|
+------+---+------------+------------------+
only showing top 20 rows


Process finished with exit code 0
```

In the Python version, we want to implement the customers.csv file our team generated. This was read in and saved in dataframe custdf. Then, we wanted to find customers in the dataframe whose age was between 18-25, this we accomplished using custdf.filter() and storing this in a new datafram young_customers_df. We stored this dataframe in a view "young_customers", so that it can be used in the following sql query. For the sql query, we wanted to join T1 on young_customers on customer id, and select the id, age, sum of items bought, and sum of money spent, and group this by age and ID. This is saved in data frame T3_df which is then saved as view "T3" and printed out for the user to see.



```
# Task 2.A.3
young_customers_df = custdf.filter((custdf['Age'] >= 18) & (custdf['Age'] <= 25))
young_customers_df.createOrReplaceTempView("young_customers")

T3_df = spark.sql("""
    SELECT T1.CustID,
           young_customers.Age,
           SUM(T1.TransNumItems) AS total_num_items,
           SUM(T1.TransTotal) AS total_amount_spent
    FROM T1
    INNER JOIN young_customers
    ON T1.CustID = young_customers.ID
    GROUP BY T1.CustID, young_customers.Age
""")

T3_df.createOrReplaceTempView("T3")

T3_df.show()
```

Contribution Statement

Before starting to work on this project, our team decided on how to split work between all members equally. We came up with the solution that the tasks could be split between members of the team, and then the team assembled and checked the work that has been done.

- Kseniia Romanova:
  - Created Git repo & report
  - Queries 1 & 2
  - Checked everyone's work and compared to my versions
- Keaton Mangone:
  - Task A.1 - A.3
- Aidan Syrgak Uulu:
  - Query 3

Resource Usage Statement

Our team used resources provided by the professor and TAs on canvas, such as discussion boards, helpful links, and tutorials.

- Keaton Mangone:
  - Use of the apache spark guide pages, such as https://spark.apache.org/docs/latest/quick-start.html and related pages in Tasks A.1-A.3
- Aidan Syrgak:
  - Stackoverflow for specific questions: https://stackoverflow.com/questions/24071560/using-reducebykey-in-apache-spark-scala
  - Apache spark docs: https://spark.apache.org/docs/latest/quick-start.html
  - Apache spark guide: https://spark.apache.org/docs/latest/rdd-programming-guide.html
  - Referred to Kseniia's template for the first 2 queries to create the 3rd query
- Kseniia Romanova:
  - Used sources provided in the discussion board
  - Made use of Stackoverflow for general Scala questions

Credits

We provided screenshots for each task as well as the description. Each member of the team was able to run all the tasks successfully.