For Whitehouse:

Below are the plans printed during runtime using the explain(true) function on the SQL Query on the data frames.

1) Dataframe for most frequent visitors

```
== Parsed Logical Plan ==
'Project [unresolvedalias('NAMELAST),unresolvedalias('NAMEFIRST),unresolvedalias('NAMEMID)]
 'UnresolvedRelation [whitehouse_table], None

== Analyzed Logical Plan ==
NAMELAST: string, NAMEFIRST: string, NAMEMID: string
Project [NAMELAST#0,NAMEFIRST#1,NAMEMID#2]
 Subquery whitehouse_table
  Relation[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27] CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)

== Optimized Logical Plan ==
Project [NAMELAST#0,NAMEFIRST#1,NAMEMID#2]
 Relation[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27] CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)

== Physical Plan ==
TungstenProject [NAMELAST#0,NAMEFIRST#1,NAMEMID#2]
 Scan CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27]
```

2) Dataframe for most frequent visitees

```
== Parsed Logical Plan ==
'Project [unresolvedalias('visitee_namefirst),unresolvedalias('visitee_namelast)]
 'UnresolvedRelation [whitehouse_table], None

== Analyzed Logical Plan ==
visitee_namefirst: string, visitee_namelast: string
Project [visitee_namefirst#20,visitee_namelast#19]
 Subquery whitehouse_table
  Relation[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27] CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)

== Optimized Logical Plan ==
Project [visitee_namefirst#20,visitee_namelast#19]
 Relation[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27] CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)

== Physical Plan ==
TungstenProject [visitee_namefirst#20,visitee_namelast#19]
 Scan CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27]
```

3) Dataframe for most frequent combinations

```
== Parsed Logical Plan ==
'Project [unresolvedalias('NAMELAST),unresolvedalias('NAMEFIRST),unresolvedalias('NAMEMID),unresolvedalias('visitee_namefirst),unresolvedalias('visitee_namelast)]
 'UnresolvedRelation [whitehouse_table], None

== Analyzed Logical Plan ==
NAMELAST: string, NAMEFIRST: string, NAMEMID: string, visitee_namefirst: string, visitee_namelast: string
Project [NAMELAST#0,NAMEFIRST#1,NAMEMID#2,visitee_namefirst#20,visitee_namelast#19]
 Subquery whitehouse_table
  Relation[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27] CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)

== Optimized Logical Plan ==
Project [NAMELAST#0,NAMEFIRST#1,NAMEMID#2,visitee_namefirst#20,visitee_namelast#19]
 Relation[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27] CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)

== Physical Plan ==
TungstenProject [NAMELAST#0,NAMEFIRST#1,NAMEMID#2,visitee_namefirst#20,visitee_namelast#19]
 Scan CsvRelation(s3n://AKIAIPHL5F57GM7JDVHA:c070LWcEK0w46kCIqDtHY4SWfekoVw4m3UqTpVJr@spark-farrukh/whitehouse_waves-2015_09.csv,true,,,",null,PERMISSIVE,COMMONS,false,false,null)[NAMELAST#0,NAMEFIRST#1,NAMEMID#2,UIN#3,BDGNBR#4,ACCESS_TYPE#5,TOA#6,POA#7,TOD#8,POD#9,APPT_MADE_DATE#10,APPT_START_DATE#11,APPT_END_DATE#12,APPT_CANCEL_DATE#13,Total_People#14,LAST_UPDATEDBY#15,POST#16,LASTENTRYDATE#17,TERMINAL_SUFFIX#18,visitee_namelast#19,visitee_namefirst#20,MEETING_LOC#21,MEETING_ROOM#22,CALLER_NAME_LAST#23,CALLER_NAME_FIRST#24,CALLER_ROOM#25,Description#26,Release_Date#27]
```

In the above physical plans, the operators used are:

- Scan (on the entire table, all relations from the input csv file)
- Project (the `NAMELAST, NAMEFIRST, NAMEMID, visitee_namefirst, visitee_namelast` relations)

Alternative Logical plan:

Instead of scanning all the relations and processing on the subquery, only the ones relevant i.e. `NAMELAST, NAMEFIRST, NAMEMID, visitee_namefirst, visitee_namelast,` the relevant relations can be joined and then processed (filtering, project). This would be more efficient on the mapper side where all the records are being scanned currently.

Alternative Physical plan:

a clustered index could be used to identify the exact relations needed for the query instead of scanning the entire table of the dataset. This would reduce the time to scan all the irrelevant entries not needed for our programs at all.

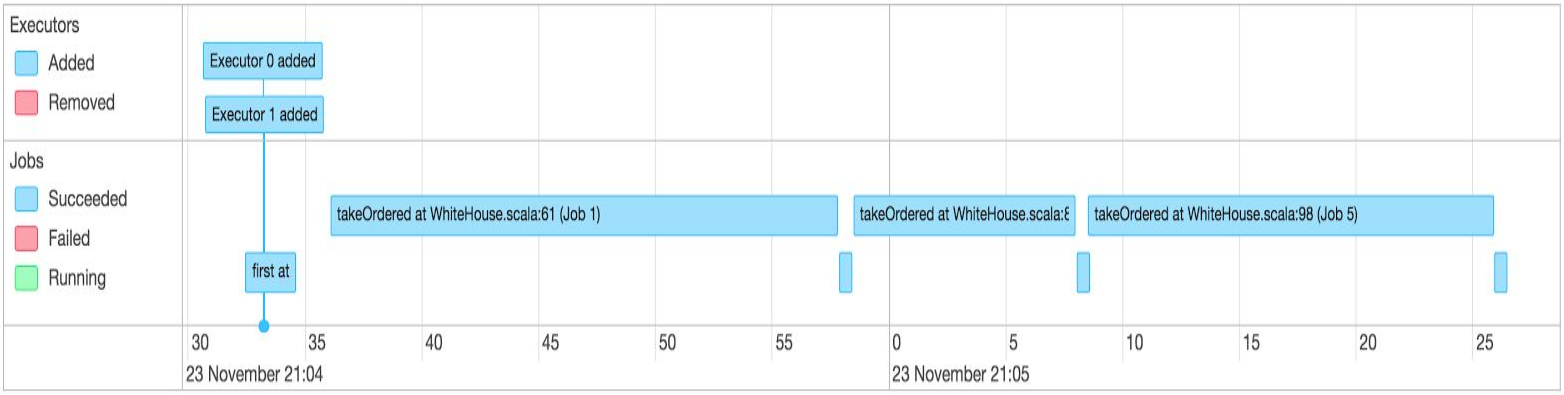
Once the count for one of the names is determined, a TNLJ could be used to store records in memory and compare with the records that are being issued. This would save the time and cost for reading them again.

Also, once the first two jobs (most frequent visitors and visitees) are processed, these contain all the needed for the processing of the third job. In this case, a join can be used to form a new relation, and stored in memory, and that can be queried on, instead of scanning all the relations for another time to determine the most frequent combination.

Spark Jobs (?)

Scheduling Mode: FIFO
Completed Jobs: 7

▾ Event Timeline
☐ Enable zooming



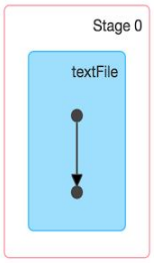
Completed Jobs (7)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
6	saveAsTextFile at WhiteHouse.scala:104	2015/11/24 02:05:25	0.2 s	1/1	8/8
5	takeOrdered at WhiteHouse.scala:98	2015/11/24 02:05:08	17 s	2/2	4/4
4	saveAsTextFile at WhiteHouse.scala:83	2015/11/24 02:05:08	0.2 s	1/1	8/8
3	takeOrdered at WhiteHouse.scala:80	2015/11/24 02:04:58	10 s	2/2	4/4
2	saveAsTextFile at WhiteHouse.scala:65	2015/11/24 02:04:57	0.3 s	1/1	8/8
1	takeOrdered at WhiteHouse.scala:61	2015/11/24 02:04:36	22 s	2/2	4/4
0	first at CsvRelation.scala:129	2015/11/24 02:04:32	2 s	1/1	1/1

Details for Job 0

Status: SUCCEEDED
Completed Stages: 1

▸ Event Timeline
▾ DAG Visualization



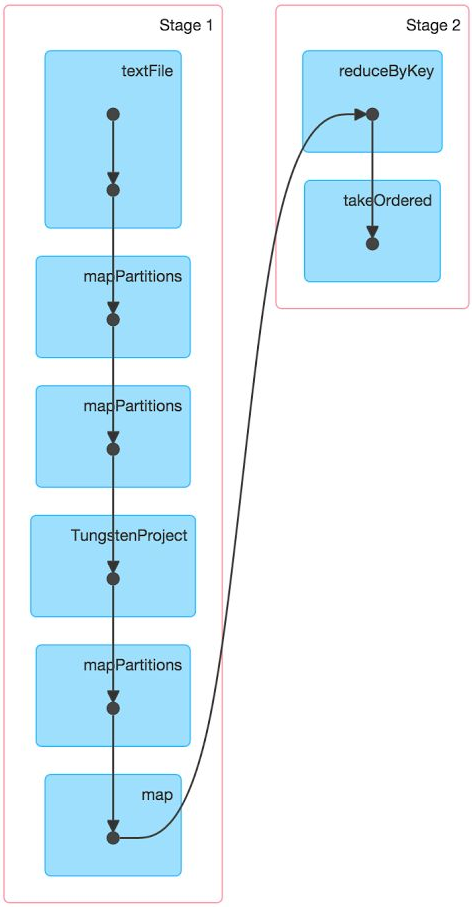
Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	first at CsvRelation.scala:129 +details	2015/11/24 02:04:32	2 s	1/1	57.1 MB			

Details for Job 1

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▾ DAG Visualization



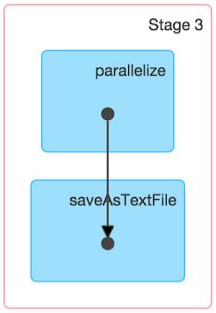
Completed Stages (2)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	takeOrdered at WhiteHouse.scala:61	+details	2015/11/24 02:04:51	6 s	2/2			11.6 MB	
1	map at WhiteHouse.scala:55	+details	2015/11/24 02:04:36	16 s	2/2	114.1 MB			11.6 MB

Details for Job 2

Status: SUCCEEDED
Completed Stages: 1

▸ Event Timeline
▾ DAG Visualization



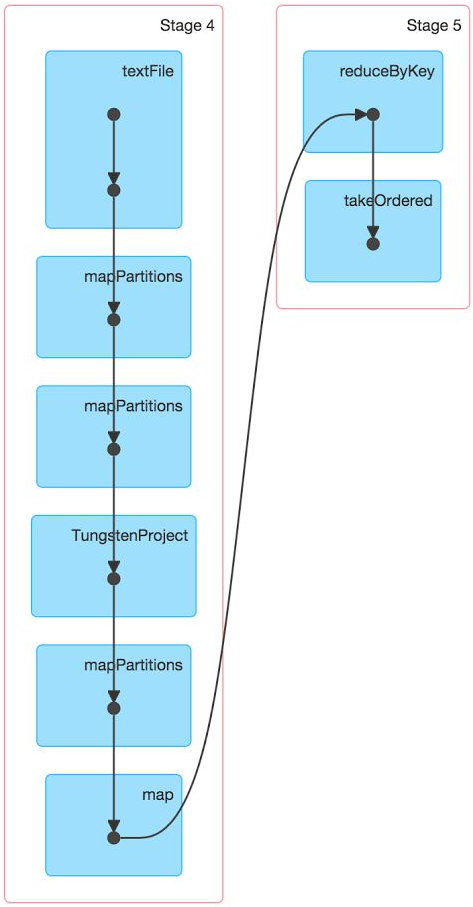
Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
3	saveAsTextFile at WhiteHouse.scala:65 +details	2015/11/24 02:04:57	0.3 s	8/8				

Details for Job 3

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▾ DAG Visualization



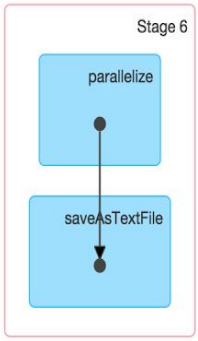
Completed Stages (2)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
5	takeOrdered at WhiteHouse.scala:80	+details	2015/11/24 02:05:06	1 s	2/2			221.8 KB	
4	map at WhiteHouse.scala:74	+details	2015/11/24 02:04:58	8 s	2/2	114.1 MB			221.8 KB

Details for Job 4

Status: SUCCEEDED
Completed Stages: 1

▸ Event Timeline
▾ DAG Visualization



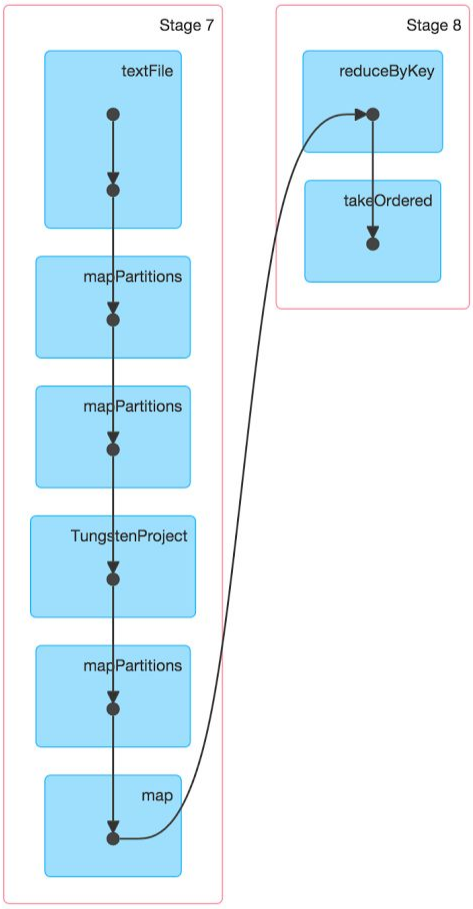
Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
6	saveAsTextFile at WhiteHouse.scala:83 +details	2015/11/24 02:05:08	0.1 s	8/8				

Details for Job 5

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▾ DAG Visualization



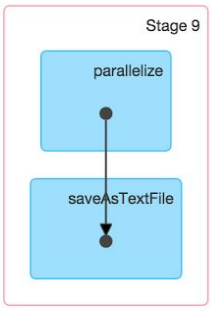
Completed Stages (2)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
8	takeOrdered at WhiteHouse.scala:98	+details	2015/11/24 02:05:20	6 s	2/2			16.7 MB	
7	map at WhiteHouse.scala:92	+details	2015/11/24 02:05:08	12 s	2/2	114.1 MB			16.7 MB

Details for Job 6

Status: SUCCEEDED
Completed Stages: 1

▸ Event Timeline
▾ DAG Visualization



Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
9	saveAsTextFile at WhiteHouse.scala:104 +details	2015/11/24 02:05:25	0.2 s	8/8				