Grouping Functions

In Datameer grouping functions are split into two distinct types of functions, group series functions and aggregate functions.

Group series functions in Datameer are similar to aggregation functions - they group values that have the same key, but they don't aggregate values. The biggest difference between group series functions and aggregation functions is that group series functions usually have more than one result per key where as aggregation functions have only one result per key.
In order to use these functions you must first define groups within your workbook. This can be done using the GROUPBY (), GROUPBYBIN (), or GROUPBYGAP () functions.

Group Series Functions

Group series functions operate row-wise within a group. The function is applied to every row and therefore returns a value for every argument in the group.

INFO

Functions IF, AND, and OR (as well as operators && and ||) arguments are evaluated lazily. This evaluation strategy is used to prevent failures caused by evaluating expressions that may result in an exception.

Problems can occur around lazy evaluation when you are working with the functions that maintain state. This is true for all of the group series functions.

When using a group series function within an IF()'s 2nd or 3rd argument or the functions AND() / OR() in any argument position, Datameer can't guarantee that all of their arguments will be evaluated.

The following functions in Datameer are group series functions:

- GROUPACCUMULATE()
- GROUPBOTTOMN()
- GROUPBYBIN()
- GROUPBYCUSTOMBIN()
- GROUPBYGAP()
- GROUPCOMBIN ()
- GROUPPREDICTIVEWINDOWS ()
- GROUPROWNUMBER ()
- GROUPSESSIONS ()
- GROUPTOPN ()
- GROUPUNIQUES ()
- GROUP_DIFF ()
- GROUP_PAIR ()
- GROUP_PATH ()
- GROUP_PATH_CHANGES ()
- GROUP_PREVIOUS ()
- GROUP_SORT_ASC ()
- GROUP_SORT_DESC ()

Aggregate Functions

Aggregate functions combine and then operate on all the values in a group. The function returns one value for each group.

INFO

Due to the different nature of group series functions and aggregate functions, they can't be used in the same workbook sheet.

INFO

Aggregations functions applied to empty groups (all records have been filtered) will result in no entry, e.g. GROUPCOUNT() will not return 0 but no record.

The following functions in Datameer are aggregate functions:

- GROUPAND ()
- GROUPANOVA()
- GROUPANY ()
- GROUPAVERAGE ()
- GROUPCONCAT ()
- GROUPCONCATDISTINCT ()
- GROUPCOUNT ()
- GROUPCOUNTSDISTINCT ()
- GROUPFIRST ()
- GROUPJSONOBJECTMERGE()
- GROUPLAST ()
- GROUPMAP ()
- GROUPMAX ()
- GROUPMEDIAN ()
- GROUPMIN ()
- GROUPOR()
- GROUPPERCENTILE ()
- GROUPSELECT()
- GROUPSTDEVP ()
- GROUPSTDEVS ()
- GROUPSUM ()
- GROUPTTEST()
- GROUP_HOLT_WINTERS()
- GROUP_JACCARD_DIST ()
- GROUP_MOVING_AVERAGE()

Real World Examples of Group Series Functions

Here we show a few examples of the power of using group series functions and how they can be used in your analyses:

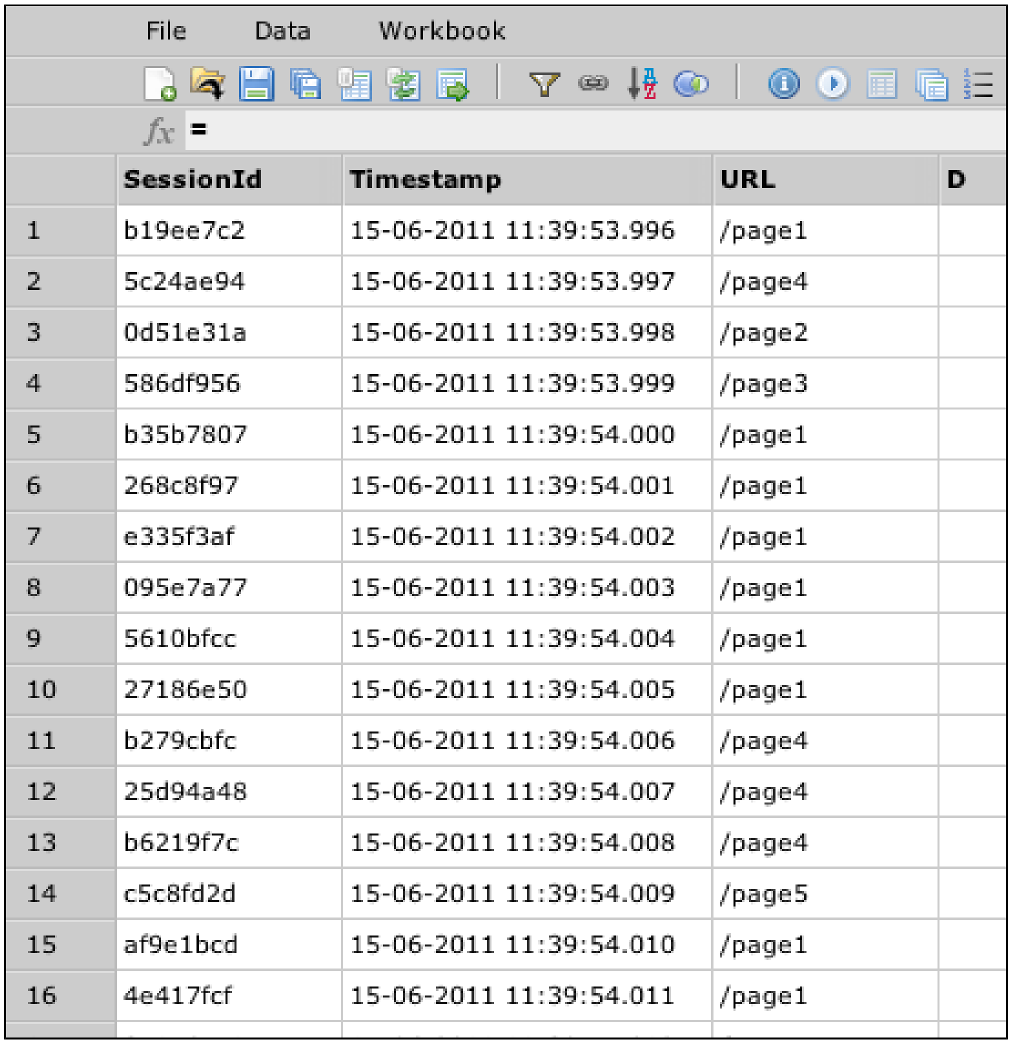
- Click stream analysis with session ID
- Click stream analysis without Session ID
- Market basket analysis

Click stream analysis with session ID

Group series functions can be used for click stream analysis:

1. Group according to the session key.
2. Sort all values within each session by the timestamp.
3. Generate click paths using the URLs.

For example, suppose you have the following input data:

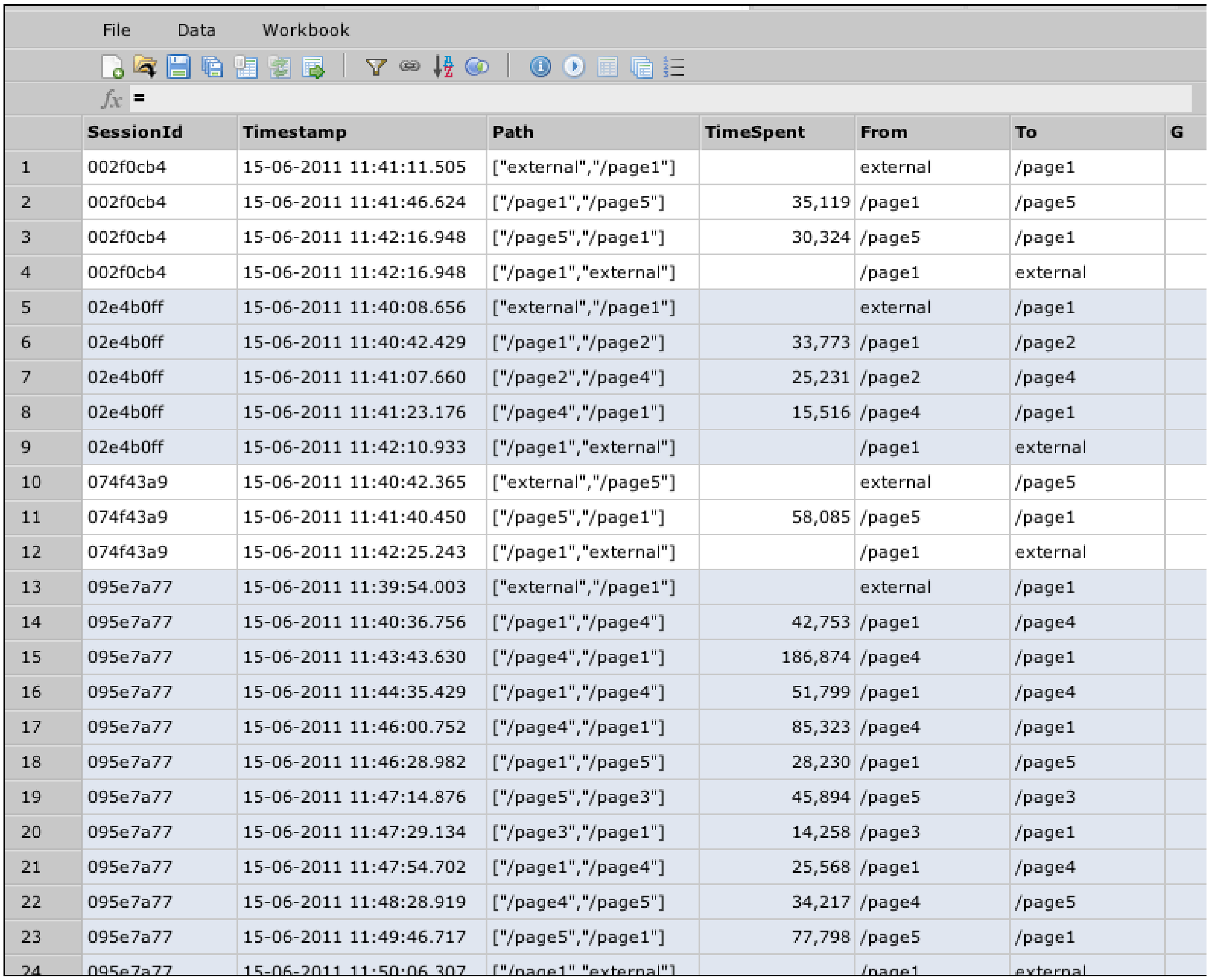


	SessionId	Timestamp	URL	D
1	b19ee7c2	15-06-2011 11:39:53.996	/page1	
2	5c24ae94	15-06-2011 11:39:53.997	/page4	
3	0d51e31a	15-06-2011 11:39:53.998	/page2	
4	586df956	15-06-2011 11:39:53.999	/page3	
5	b35b7807	15-06-2011 11:39:54.000	/page1	
6	268c8f97	15-06-2011 11:39:54.001	/page1	
7	e335f3af	15-06-2011 11:39:54.002	/page1	
8	095e7a77	15-06-2011 11:39:54.003	/page1	
9	5610bfcc	15-06-2011 11:39:54.004	/page1	
10	27186e50	15-06-2011 11:39:54.005	/page1	
11	b279cbfc	15-06-2011 11:39:54.006	/page4	
12	25d94a48	15-06-2011 11:39:54.007	/page4	
13	b6219f7c	15-06-2011 11:39:54.008	/page4	
14	c5c8fd2d	15-06-2011 11:39:54.009	/page5	
15	af9e1bcd	15-06-2011 11:39:54.010	/page1	
16	4e417fcf	15-06-2011 11:39:54.011	/page1	

Generating click stream information is done within one formula sheet:

```
=GROUPBY(#Input!SessionId)
=GROUP_SORT_ASC(#Input!Timestamp)
=GROUP_PATH_CHANGES(#Input!URL)
=GROUP_DIFF(#Timestamp)
=JSON_ELEMENT(#Path;0)
=JSON_ELEMENT(#Path;1)
```

The result of this sheet looks like this:



	SessionId	Timestamp	Path	TimeSpent	From	To	G
1	002f0cb4	15-06-2011 11:41:11.505	["external","/page1"]		external	/page1	
2	002f0cb4	15-06-2011 11:41:46.624	["/page1","/page5"]	35,119	/page1	/page5	
3	002f0cb4	15-06-2011 11:42:16.948	["/page5","/page1"]	30,324	/page5	/page1	
4	002f0cb4	15-06-2011 11:42:16.948	["/page1","external"]		/page1	external	
5	02e4b0ff	15-06-2011 11:40:08.656	["external","/page1"]		external	/page1	
6	02e4b0ff	15-06-2011 11:40:42.429	["/page1","/page2"]	33,773	/page1	/page2	
7	02e4b0ff	15-06-2011 11:41:07.660	["/page2","/page4"]	25,231	/page2	/page4	
8	02e4b0ff	15-06-2011 11:41:23.176	["/page4","/page1"]	15,516	/page4	/page1	
9	02e4b0ff	15-06-2011 11:42:10.933	["/page1","external"]		/page1	external	
10	074f43a9	15-06-2011 11:40:42.365	["external","/page5"]		external	/page5	
11	074f43a9	15-06-2011 11:41:40.450	["/page5","/page1"]	58,085	/page5	/page1	
12	074f43a9	15-06-2011 11:42:25.243	["/page1","external"]		/page1	external	
13	095e7a77	15-06-2011 11:39:54.003	["external","/page1"]		external	/page1	
14	095e7a77	15-06-2011 11:40:36.756	["/page1","/page4"]	42,753	/page1	/page4	
15	095e7a77	15-06-2011 11:43:43.630	["/page4","/page1"]	186,874	/page4	/page1	
16	095e7a77	15-06-2011 11:44:35.429	["/page1","/page4"]	51,799	/page1	/page4	
17	095e7a77	15-06-2011 11:46:00.752	["/page4","/page1"]	85,323	/page4	/page1	
18	095e7a77	15-06-2011 11:46:28.982	["/page1","/page5"]	28,230	/page1	/page5	
19	095e7a77	15-06-2011 11:47:14.876	["/page5","/page3"]	45,894	/page5	/page3	
20	095e7a77	15-06-2011 11:47:29.134	["/page3","/page1"]	14,258	/page3	/page1	
21	095e7a77	15-06-2011 11:47:54.702	["/page1","/page4"]	25,568	/page1	/page4	
22	095e7a77	15-06-2011 11:48:28.919	["/page4","/page5"]	34,217	/page4	/page5	
23	095e7a77	15-06-2011 11:49:46.717	["/page5","/page1"]	77,798	/page5	/page1	
24	095e7a77	15-06-2011 11:50:06.307	["/page1","external"]		/page1	external	

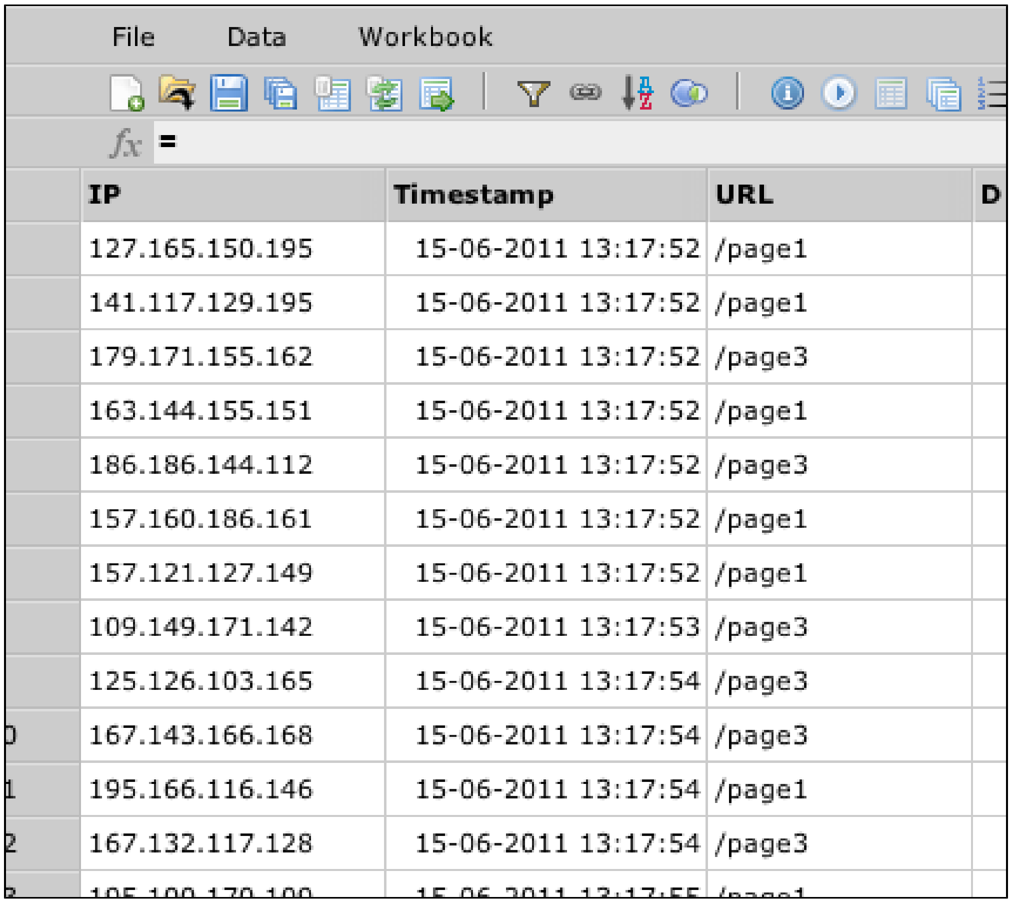
After that, you can do whatever analysis is required, for example, you could do the following:

- Find the page where users spend most time on by:
  - Doing GROUPBY(#ClickStream!From) and GROUPMEDIAN(#ClickStream!TimeSpent).
  - Sorting by TimeSpent.
- Find the page where a user most often leaves the application by:
  - Filtering all records, where #ClickStream!To == "external".
  - Doing GROUPBY(#From) and GROUPCOUNT().
  - Sorting by count in descending order.
- Find most often page transitions by:
  - Doing GROUPBY(#ClickStream!From), GROUPBY(#ClickStream!To) and GROUPCOUNT().
  - Sorting by count in descending order.

Click stream analysis without session ID

In many cases you might not have a session ID in your data so you can't group on a session as easily. But usually it's possible to extract a session from your data. For example, if you have IP addresses and timestamps in your server log, you could assume that for each IP address a new session begins when there is a gap of more than x minutes between one and the next log entry. This means that the request is coming from the same machine, but it's probably a different session. The GROUPBYGAP function helps you determine the session information.

For example, suppose you have the following input data:

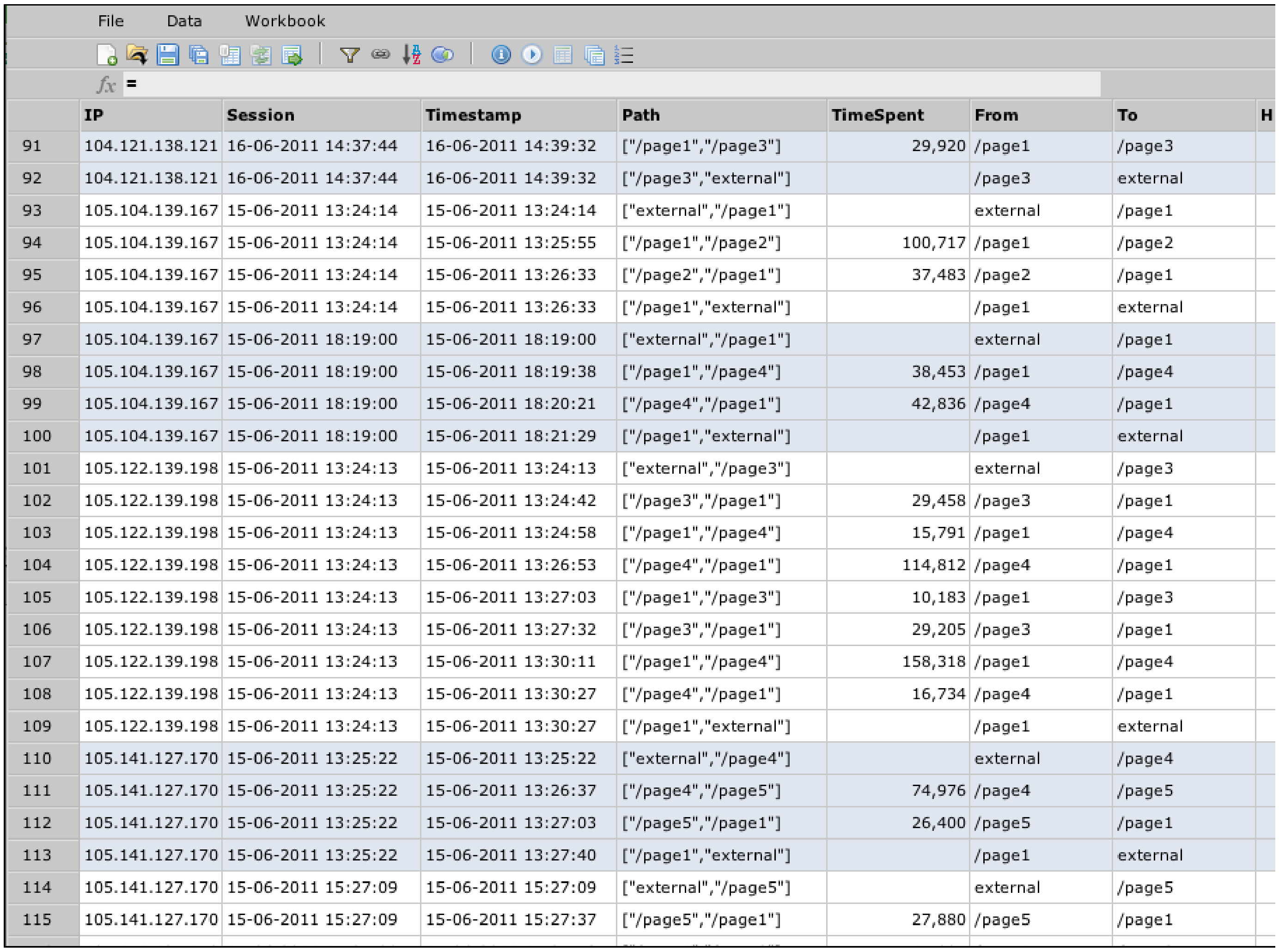


IP	Timestamp	URL	D
127.165.150.195	15-06-2011 13:17:52	/page1	
141.117.129.195	15-06-2011 13:17:52	/page1	
179.171.155.162	15-06-2011 13:17:52	/page3	
163.144.155.151	15-06-2011 13:17:52	/page1	
186.186.144.112	15-06-2011 13:17:52	/page3	
157.160.186.161	15-06-2011 13:17:52	/page1	
157.121.127.149	15-06-2011 13:17:52	/page1	
109.149.171.142	15-06-2011 13:17:53	/page3	
125.126.103.165	15-06-2011 13:17:54	/page3	
167.143.166.168	15-06-2011 13:17:54	/page3	
195.166.116.146	15-06-2011 13:17:54	/page1	
167.132.117.128	15-06-2011 13:17:54	/page3	
[illegible]	[illegible]	[illegible]	

Generating click streams can be done within one formula sheet:

```
=GROUPBY(#Input!IP)
=GROUPBYGAP(#Input!Timestamp;10m)
=#Input!Timestamp
=GROUP_PATH_CHANGES(#Input!URL)
=GROUP_DIFF(#Timestamp)
=JSON_ELEMENT(#Path;0)
=JSON_ELEMENT(#Path;1)
```

The result of this sheet looks like this:

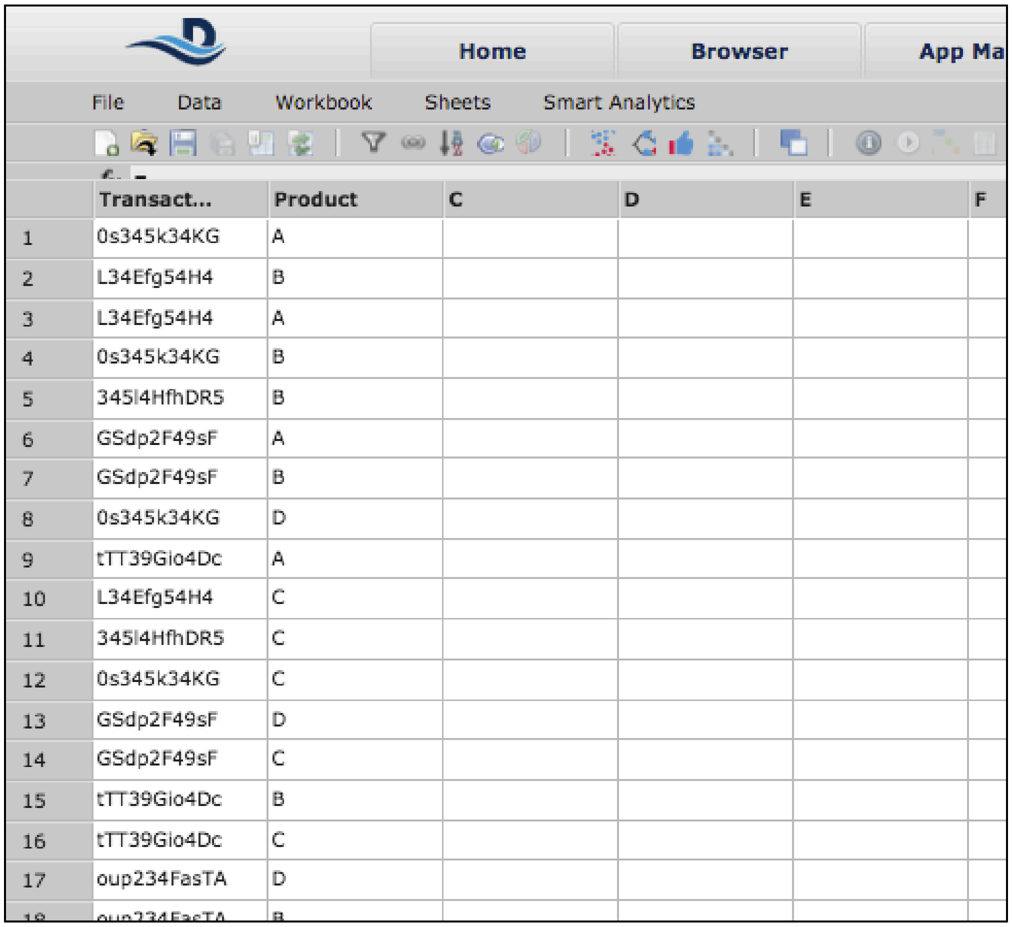

File Data Workbook

	IP	Session	Timestamp	Path	TimeSpent	From	To	H
91	104.121.138.121	16-06-2011 14:37:44	16-06-2011 14:39:32	["/page1","/page3"]	29,920	/page1	/page3	
92	104.121.138.121	16-06-2011 14:37:44	16-06-2011 14:39:32	["/page3","external"]		/page3	external	
93	105.104.139.167	15-06-2011 13:24:14	15-06-2011 13:24:14	["external","/page1"]		external	/page1	
94	105.104.139.167	15-06-2011 13:24:14	15-06-2011 13:25:55	["/page1","/page2"]	100,717	/page1	/page2	
95	105.104.139.167	15-06-2011 13:24:14	15-06-2011 13:26:33	["/page2","/page1"]	37,483	/page2	/page1	
96	105.104.139.167	15-06-2011 13:24:14	15-06-2011 13:26:33	["/page1","external"]		/page1	external	
97	105.104.139.167	15-06-2011 18:19:00	15-06-2011 18:19:00	["external","/page1"]		external	/page1	
98	105.104.139.167	15-06-2011 18:19:00	15-06-2011 18:19:38	["/page1","/page4"]	38,453	/page1	/page4	
99	105.104.139.167	15-06-2011 18:19:00	15-06-2011 18:20:21	["/page4","/page1"]	42,836	/page4	/page1	
100	105.104.139.167	15-06-2011 18:19:00	15-06-2011 18:21:29	["/page1","external"]		/page1	external	
101	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:24:13	["external","/page3"]		external	/page3	
102	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:24:42	["/page3","/page1"]	29,458	/page3	/page1	
103	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:24:58	["/page1","/page4"]	15,791	/page1	/page4	
104	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:26:53	["/page4","/page1"]	114,812	/page4	/page1	
105	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:27:03	["/page1","/page3"]	10,183	/page1	/page3	
106	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:27:32	["/page3","/page1"]	29,205	/page3	/page1	
107	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:30:11	["/page1","/page4"]	158,318	/page1	/page4	
108	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:30:27	["/page4","/page1"]	16,734	/page4	/page1	
109	105.122.139.198	15-06-2011 13:24:13	15-06-2011 13:30:27	["/page1","external"]		/page1	external	
110	105.141.127.170	15-06-2011 13:25:22	15-06-2011 13:25:22	["external","/page4"]		external	/page4	
111	105.141.127.170	15-06-2011 13:25:22	15-06-2011 13:26:37	["/page4","/page5"]	74,976	/page4	/page5	
112	105.141.127.170	15-06-2011 13:25:22	15-06-2011 13:27:03	["/page5","/page1"]	26,400	/page5	/page1	
113	105.141.127.170	15-06-2011 13:25:22	15-06-2011 13:27:40	["/page1","external"]		/page1	external	
114	105.141.127.170	15-06-2011 15:27:09	15-06-2011 15:27:09	["external","/page5"]		external	/page5	
115	105.141.127.170	15-06-2011 15:27:09	15-06-2011 15:27:37	["/page5","/page1"]	27,880	/page5	/page1	

Market basket analysis

Another use case for group series analysis is determining which products have been bought together in one transaction in order to do recommendations. To do this analysis, you can use a GROUP_PAIR function that creates all pairs of a value within one group.

Suppose you have a data structure such as this:

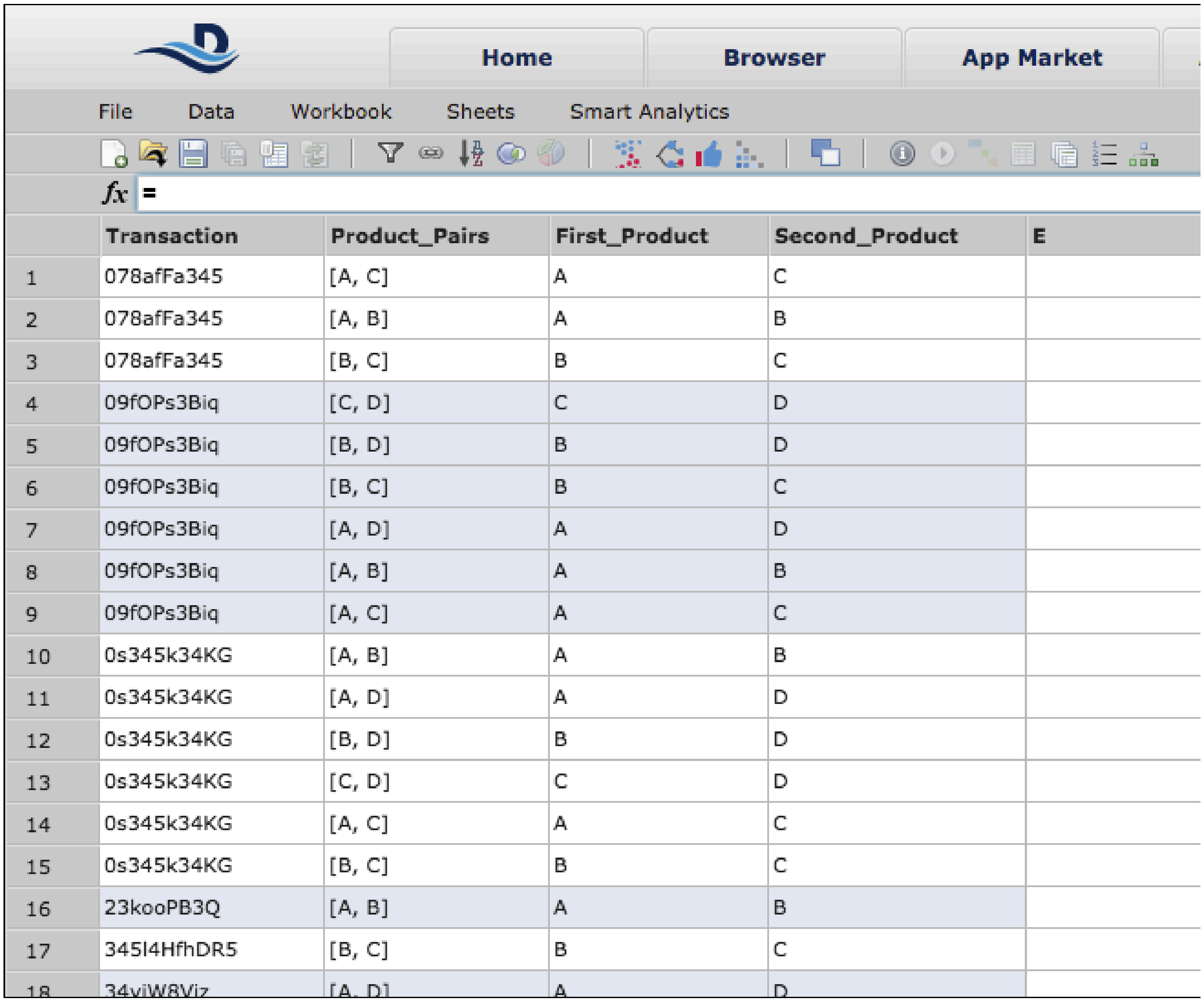


	Transact...	Product	C	D	E	F
1	0s345k34KG	A				
2	L34Efg54H4	B				
3	L34Efg54H4	A				
4	0s345k34KG	B				
5	345l4HfhDR5	B				
6	GSdp2F49sF	A				
7	GSdp2F49sF	B				
8	0s345k34KG	D				
9	tTT39Gio4Dc	A				
10	L34Efg54H4	C				
11	345l4HfhDR5	C				
12	0s345k34KG	C				
13	GSdp2F49sF	D				
14	GSdp2F49sF	C				
15	tTT39Gio4Dc	B				
16	tTT39Gio4Dc	C				
17	oup234FasTA	D				
18	oup234FasTA	B				

You can now do the following analysis:

```
=GROUPBY(#Input!Transaction)
 -The GROUPBY function groups all the transactions of the same value
together.
=GROUP_PAIR(#Input!Product)
 -The GROUP_PAIR function can be used after a GROUPBY function to return
all pairs of values of each GROUPBY value in a list.
=LISTELEMENT(#Product_Pairs;0)
=LISTELEMENT(#Product_Pairs;1)
 -The LISTELEMENT functions are used to extract the first and second
elements of the list into separate columns.
```

The result sheet looks like this:



	Transaction	Product_Pairs	First_Product	Second_Product	E
1	078afFa345	[A, C]	A	C	
2	078afFa345	[A, B]	A	B	
3	078afFa345	[B, C]	B	C	
4	09fOPs3Biq	[C, D]	C	D	
5	09fOPs3Biq	[B, D]	B	D	
6	09fOPs3Biq	[B, C]	B	C	
7	09fOPs3Biq	[A, D]	A	D	
8	09fOPs3Biq	[A, B]	A	B	
9	09fOPs3Biq	[A, C]	A	C	
10	0s345k34KG	[A, B]	A	B	
11	0s345k34KG	[A, D]	A	D	
12	0s345k34KG	[B, D]	B	D	
13	0s345k34KG	[C, D]	C	D	
14	0s345k34KG	[A, C]	A	C	
15	0s345k34KG	[B, C]	B	C	
16	23kooPB3Q	[A, B]	A	B	
17	345l4HfhDR5	[B, C]	B	C	
18	34viW8Viz	[A, D]	A	D	

Next, you can aggregate on pairs to see which pairs occur most often in one transaction.