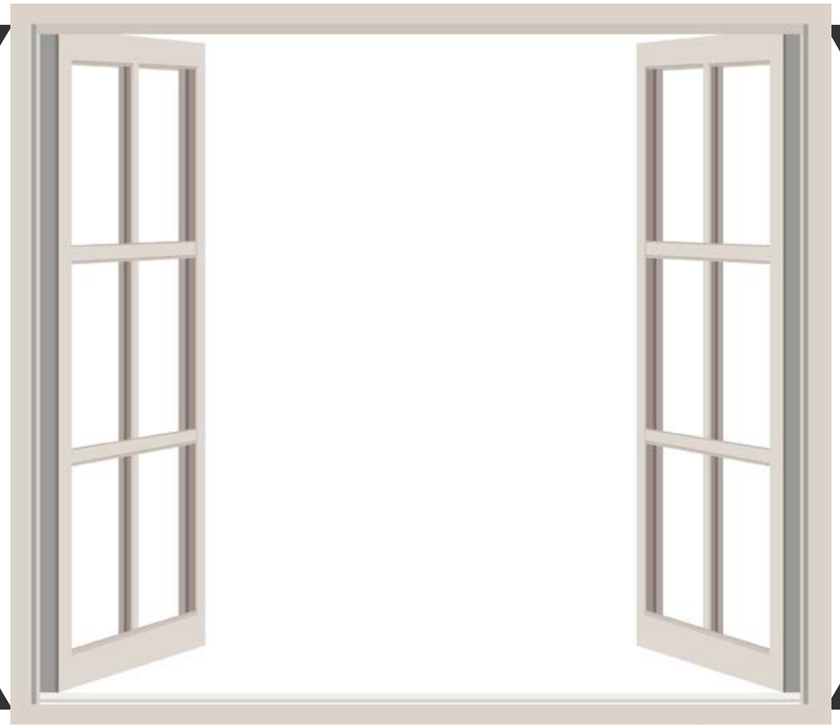
Window Functions

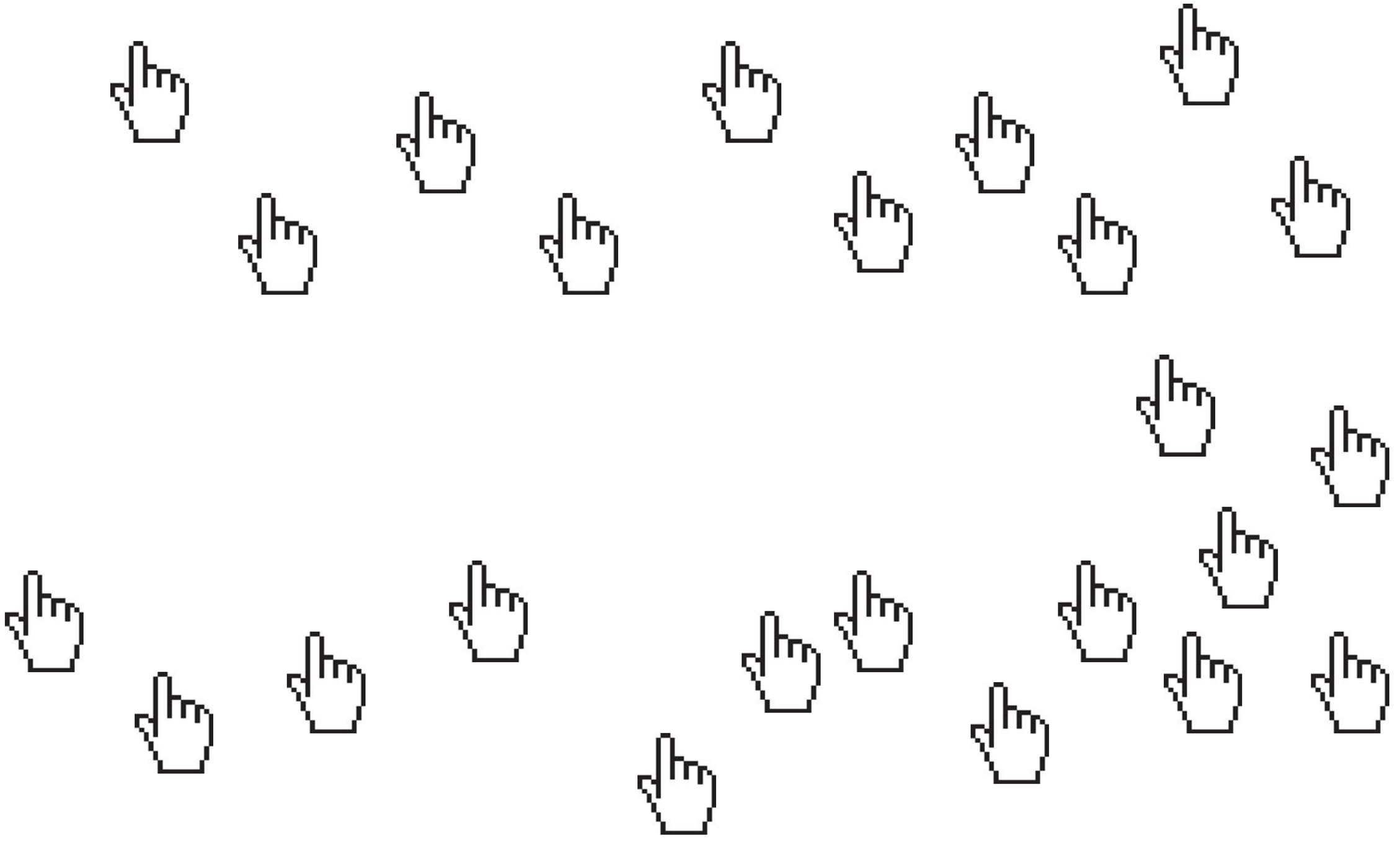
f()


facebook
Регистрация


facebook
Регистрация

Session

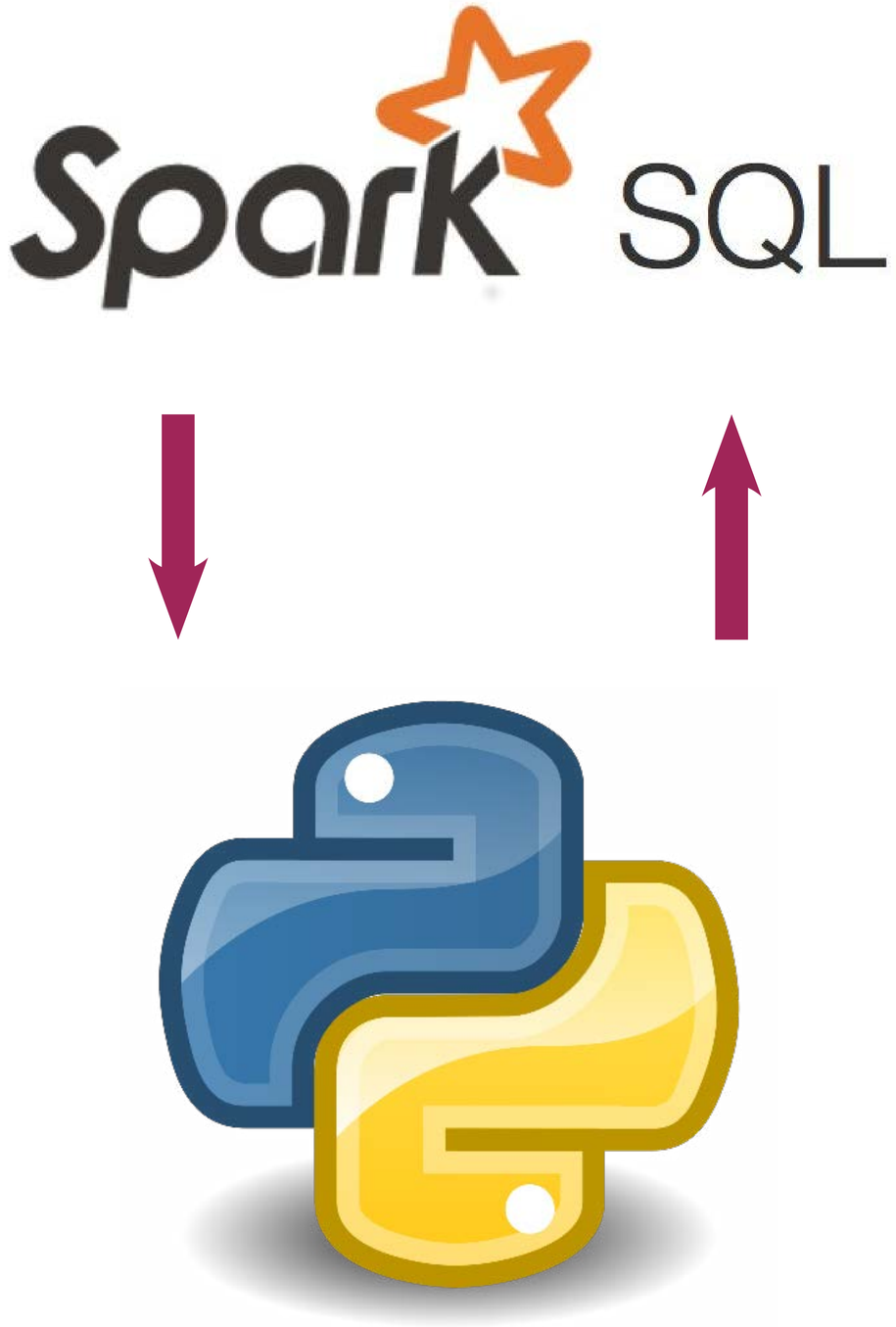




	time
0	10/Dec/2015:00:29:05 +0400
1	10/Dec/2015:00:29:05 +0400
2	10/Dec/2015:00:29:10 +0400
3	11/Dec/2015:00:36:04 +0400
4	11/Dec/2015:00:36:04 +0400
5	11/Dec/2015:00:36:06 +0400
6	11/Dec/2015:00:36:15 +0400
7	11/Dec/2015:00:36:18 +0400
8	11/Dec/2015:00:36:35 +0400
9	11/Dec/2015:00:36:43 +0400

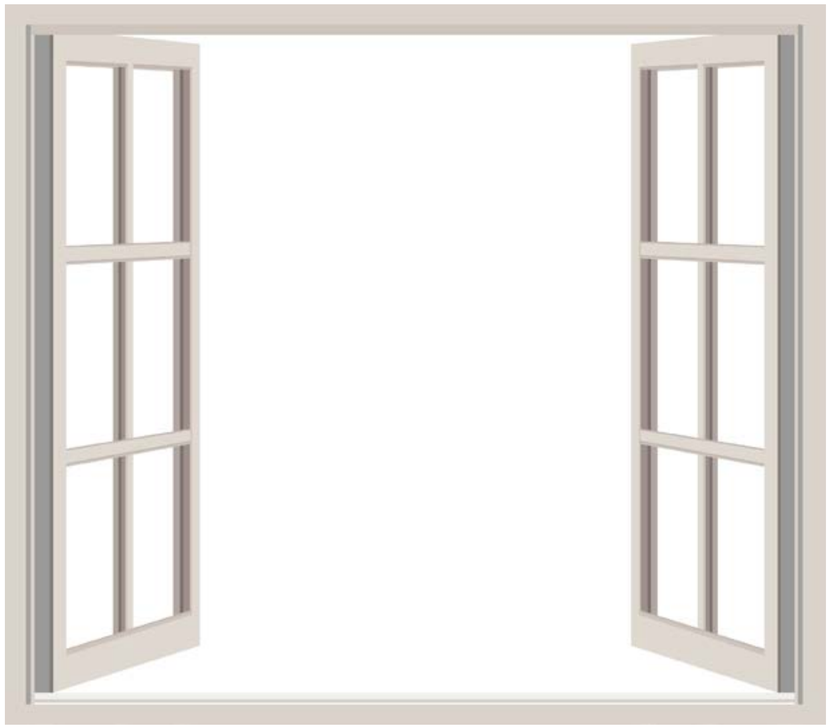

Spark SQL


Spark
SQL


Spark SQL

In this video you will learn

- what window functions are
- what type of windows exists
- how to count number of sessions



≈


EL SOL

```
access_log_ts.select(
    "ip",
    "time",
    f.count("*").over(Window.partitionBy("ip")).alias("cnt"))\
  .limit(5).toPandas()
```

	ip	time	cnt
0	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
1	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
2	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
3	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
4	109.60.192.67	10/Dec/2015:01:41:24 +0400	13

```
access_log_ts.select(
    "ip",
    "time",
    f.count("*").over(Window.partitionBy("ip")).alias("cnt"))\
  .limit(5).toPandas()
```

	ip	time	cnt
0	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
1	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
2	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
3	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
4	109.60.192.67	10/Dec/2015:01:41:24 +0400	13

```
access_log_ts.select(
    "ip",
    "time",
    f.count("*").over(Window.partitionBy("ip")).alias("cnt"))\
  .limit(5).toPandas()
```

	ip	time	cnt
0	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
1	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
2	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
3	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
4	109.60.192.67	10/Dec/2015:01:41:24 +0400	13

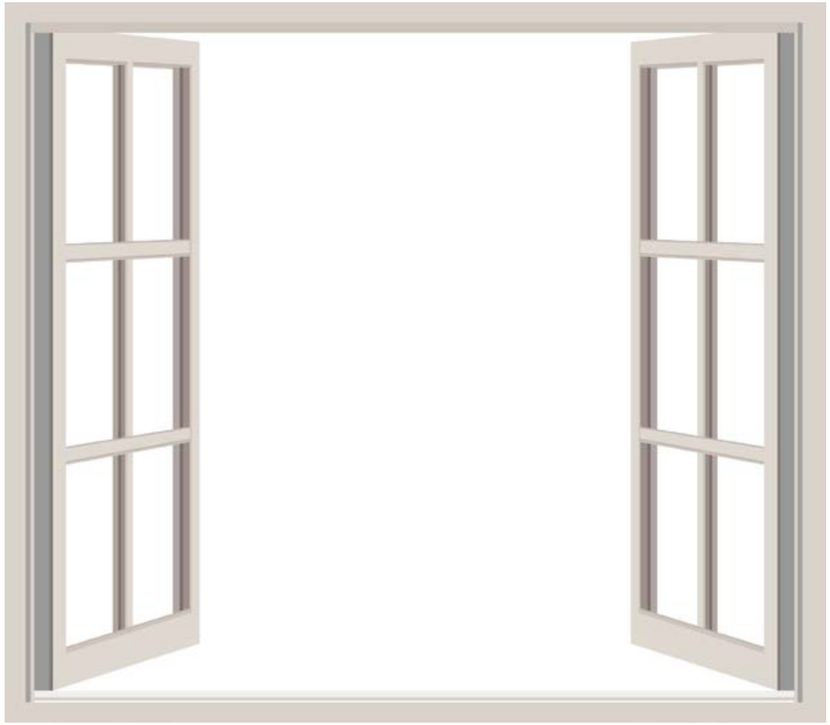
```
access_log_ts.select(
    "ip",
    "time",
    f.count("*").over(Window.partitionBy("ip")).alias("cnt"))\
  .limit(5).toPandas()
```

	ip	time	cnt
0	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
1	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
2	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
3	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
4	109.60.192.67	10/Dec/2015:01:41:24 +0400	13

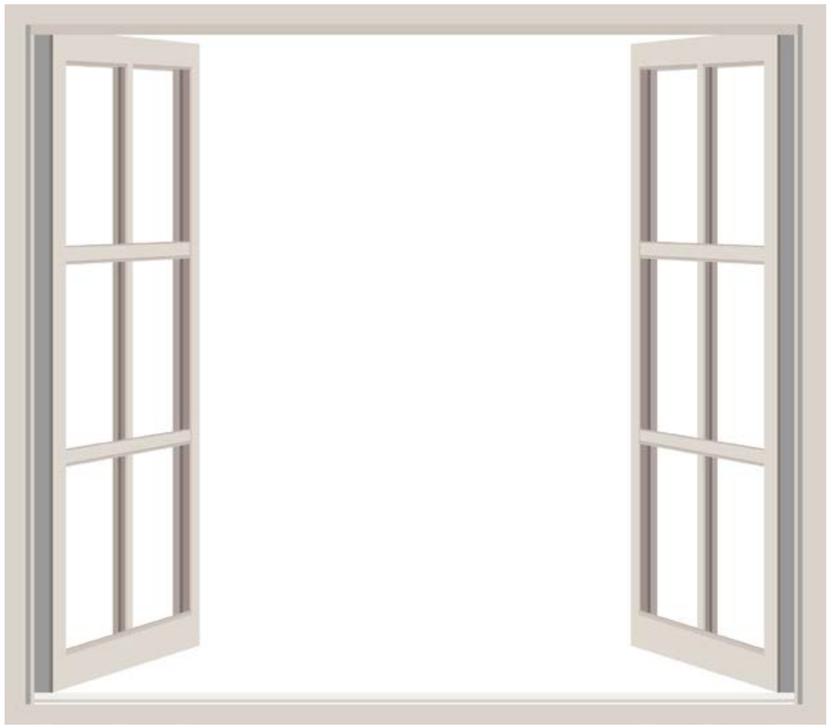
```
access_log_ts.select(
    "ip",
    "time",
    f.count("*").over(Window.partitionBy("ip")).alias("cnt"))\
  .limit(5).toPandas()
```

	ip	time	cnt
0	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
1	109.120.126.97	10/Dec/2015:00:13:16 +0400	2
2	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
3	109.60.192.67	10/Dec/2015:01:41:17 +0400	13
4	109.60.192.67	10/Dec/2015:01:41:24 +0400	13

	aggregation	window function
applied to	whole table	column
number of rows	reduces	remains unchanged
grouping condition	goes first **df.groupby(…).agg(…)**	goes last **func(“column”).over(…)**
values in a group…	unordered	ordered

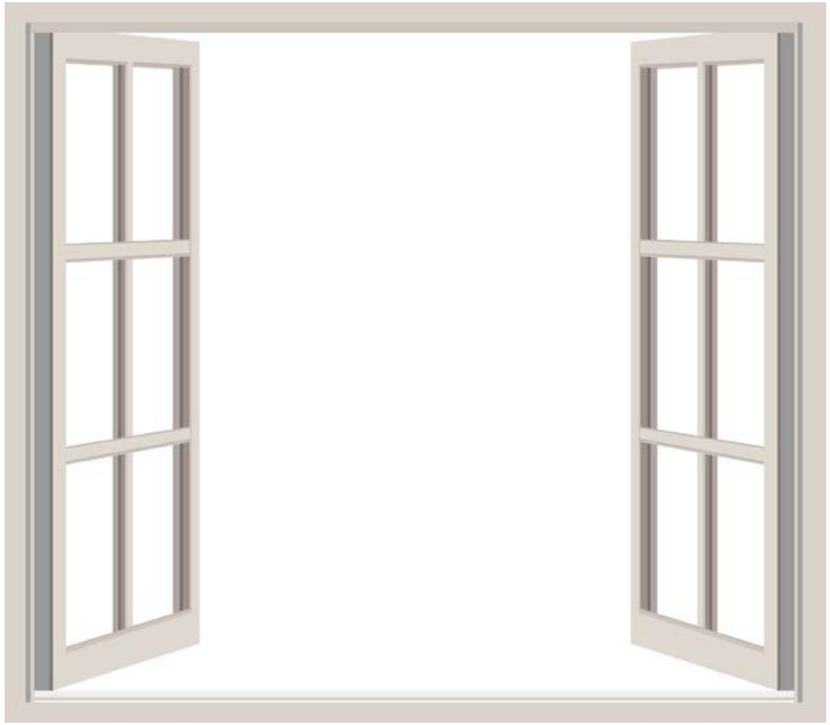


```
Window.partitionBy("ip")
```



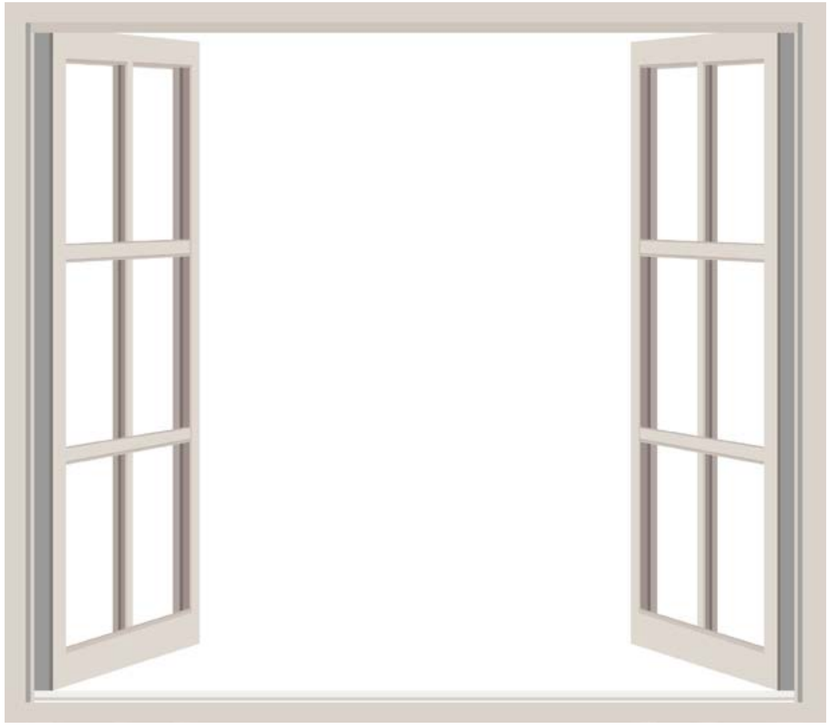
```
Window.partitionBy("ip")
```

[12:10, 7:30, 10:20, 7:31, 10:21, 10:19, 7:32, 12:14, 12:11]



```
Window.partitionBy("ip")\
      .orderBy("unixtime")
```

[7:30, 7:31, 7:32, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]



```
Window.partitionBy("ip")\
      .orderBy("unixtime")
```

[7:30, 7:31, **7:32**, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]

[7:30, 7:31, **7:32**, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]

[7:30, 7:31, **7:32**, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]

first () 7:30

[7:30, 7:31, **7:32**, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]

first ()	7:30
last ()	12:14

[7:30, 7:31, **7:32**, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]

first ()	7:30
last ()	12:14
lag ()	7:31

[7:30, 7:31, **7:32**, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]

first ()	7:30
last ()	12:14
lag ()	7:31
lead ()	10:19

[7:30, 7:31, **7:32**, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]

first ()	7:30
last ()	12:14
lag ()	7:31
lead ()	10:19
row_number ()	3

[7:30, 7:31, **7:32**, 10:19, 10:20, 10:21, 12:10, 12:11 , 12:14]

first () 7:30

last () 12:14

lag () 7:31

lead () 10:19

row_number () 3

min() max() sum() ...

```
user_window = Window.orderBy("unixtime").partitionBy("ip")
```

```
access_log_ts.select("ip", "unixtime",
                     f.row_number().over(user_window).alias("count"),
                     f.lag("unixtime").over(user_window).alias("lag"),
                     f.lead("unixtime").over(user_window).alias("lead"),
                    )\
             .limit(5).toPandas()
```

	ip	unixtime	count	lag	lead
0	109.120.126.97	1449691996	1	NaN	1.449692e+09
1	109.120.126.97	1449691996	2	1.449692e+09	NaN
2	109.60.192.67	1449697277	1	NaN	1.449697e+09
3	109.60.192.67	1449697277	2	1.449697e+09	1.449697e+09
4	109.60.192.67	1449697284	3	1.449697e+09	1.449697e+09

```
access_log_ts.select("ip", "unixtime",
                     f.row_number().over(user_window).alias("count"),
                     f.lag("unixtime").over(user_window).alias("lag"),
                     f.lead("unixtime").over(user_window).alias("lead"),
                    )\
             .limit(5).toPandas()
```

	ip	unixtime	count	lag	lead
0	109.120.126.97	1449691996	1	NaN	1.449692e+09
1	109.120.126.97	1449691996	2	1.449692e+09	NaN
2	109.60.192.67	1449697277	1	NaN	1.449697e+09
3	109.60.192.67	1449697277	2	1.449697e+09	1.449697e+09
4	109.60.192.67	1449697284	3	1.449697e+09	1.449697e+09

```
access_log_ts.select("ip", "unixtime",
                     f.row_number().over(user_window).alias("count"),
                     f.lag("unixtime").over(user_window).alias("lag"),
                     f.lead("unixtime").over(user_window).alias("lead"),
                    )\
             .limit(5).toPandas()
```

	ip	unixtime	count	lag	lead
0	109.120.126.97	1449691996	1	NaN	1.449692e+09
1	109.120.126.97	1449691996	2	1.449692e+09	NaN
2	109.60.192.67	1449697277	1	NaN	1.449697e+09
3	109.60.192.67	1449697277	2	1.449697e+09	1.449697e+09
4	109.60.192.67	1449697284	3	1.449697e+09	1.449697e+09

```
access_log_ts.select("ip",
                     f.col("unixtime"),
                     f.lead("unixtime").over(user_window).alias("lead"))\
             .limit(5).toPandas()
```

	ip	unixtime	lead
0	109.120.126.97	1449691996	1.449692e+09
1	109.120.126.97	1449691996	NaN
2	109.60.192.67	1449697277	1.449697e+09
3	109.60.192.67	1449697277	1.449697e+09
4	109.60.192.67	1449697284	1.449697e+09

```
access_log_ts.select("ip",
                     f.col("unixtime"),
                     f.lead("unixtime").over(user_window).alias("lead"))\
             .select("ip", (f.col("lead")-f.col("unixtime")).alias("diff"))\
             .limit(5).toPandas()
```

	ip	diff
0	109.120.126.97	0.0
1	109.120.126.97	NaN
2	109.60.192.67	0.0
3	109.60.192.67	7.0
4	109.60.192.67	9.0

```
access_log_ts.select("ip",
                     f.col("unixtime"),
                     f.lead("unixtime").over(user_window).alias("lead"))\
             .select("ip","unixtime",
                     (f.col("lead")-f.col("unixtime")).alias("diff"))\
             .where("diff >= 1800 or diff is NULL")\
             .limit(5).toPandas()
```

	ip	unixtime	diff
5	149.126.79.68	1449778014	175090.0
6	149.126.79.68	1449953116	NaN
7	154.37.229.66	1449780872	90757.0
8	154.37.229.66	1449871722	NaN
9	158.69.69.225	1449782033	86732.0

```
access_log_ts.select("ip",
                     f.col("unixtime"),
                     f.lead("unixtime").over(user_window).alias("lead"))\
             .select("ip","unixtime",
                     (f.col("lead")-f.col("unixtime")).alias("diff"))\
             .where("diff >= 1800 or diff is NULL")\
             .groupBy("ip").count()\
             .limit(5).toPandas()
```

	ip	count
0	109.120.126.97	1
1	109.60.192.67	1
2	145.225.216.216	1
3	147.215.145.110	1
4	148.188.9.218	1

```
access_log_ts.select("ip",
                     f.col("unixtime"),
                     f.lead("unixtime").over(user_window).alias("lead"))\
             .select("ip","unixtime",
                     (f.col("lead")-f.col("unixtime")).alias("diff"))\
             .where("diff >= 1800 or diff is NULL")\
             .groupBy("ip").count()\
             .limit(5).toPandas()
```

	ip	count
0	109.120.126.97	1
1	109.60.192.67	1
2	145.225.216.216	1
3	147.215.145.110	1
4	148.188.9.218	1

```
access_log_ts.select("ip",
                     f.col("unixtime"),
                     f.lead("unixtime").over(user_window).alias("lead"))\
             .select("ip","unixtime",
                     (f.col("lead")-f.col("unixtime")).alias("diff"))\
             .where("diff >= 1800 or diff is NULL")\
             .groupBy("ip").count()\
             .orderBy(f.col("count").desc())\
             .limit(5).toPandas()
```

	ip	count
0	194.196.26.65	6
1	31.192.111.243	6
2	193.187.255.229	6
3	193.32.68.52	5
4	41.224.169.6	5


25
28
31
34
37
40
43
46
49

```
access_log_ts.select("ip",
                     f.col("unixtime"),
                     f.lead("unixtime").over(user_window).alias("lead"))\
             .select("ip","unixtime",
                     (f.col("lead")-f.col("unixtime")).alias("diff"))\
             .where("diff >= 1800 or diff is NULL")\
             .groupBy("ip").count()\
             .where("count = 3")\
             .limit(5).toPandas()
```

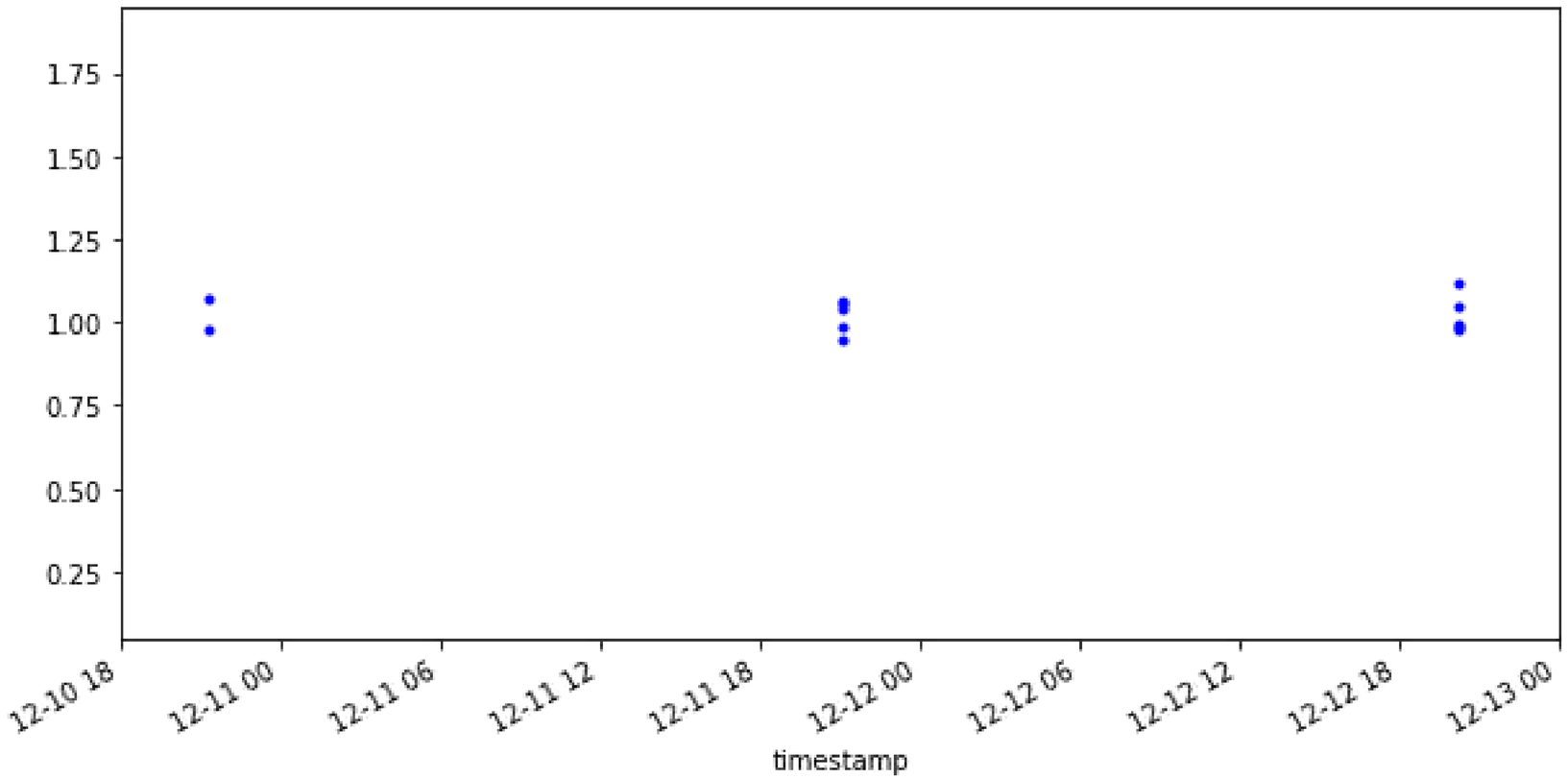
	ip	count
0	178.208.51.84	3
1	185.20.133.106	3
2	194.50.60.51	3
3	185.8.139.237	3
4	78.159.120.94	3

```
event = access_log_ts[access_log_ts.ip=="185.8.139.237"].toPandas()
```

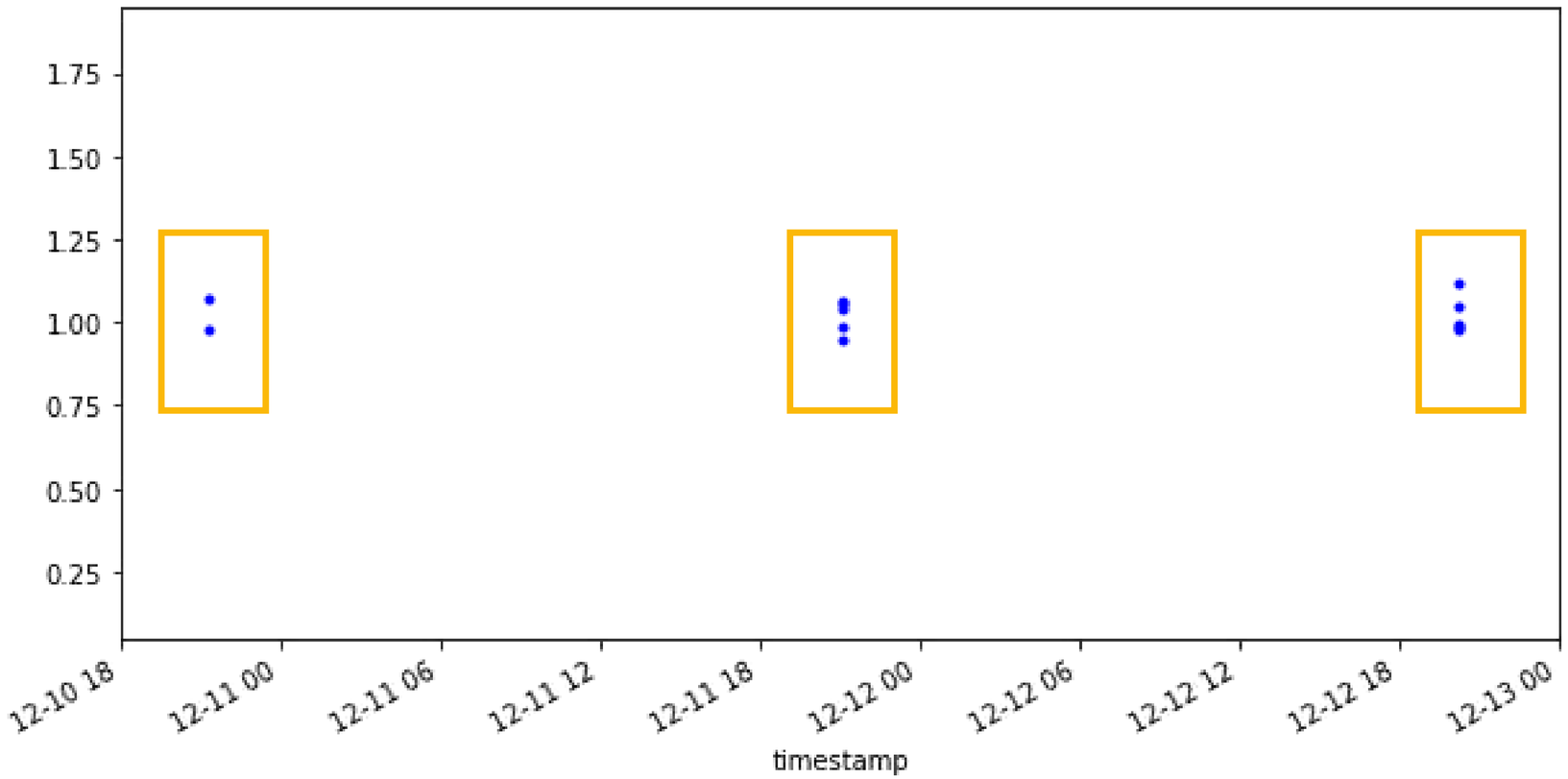
```
event = access_log_ts[access_log_ts.ip=="185.8.139.237"].toPandas()

event = event.set_index("timestamp")
```

```
event = access_log_ts[access_log_ts.ip=="185.8.139.237"].toPandas()

event = event.set_index("timestamp")

event["y"] = np.random.normal(1,0.05,size=len(event))
```

```
event = access_log_ts[access_log_ts.ip=="185.8.139.237"].toPandas()
event = event.set_index("timestamp")
event["y"] = np.random.normal(1,0.05,size=len(event))
event["y"].plot(style='b.', ylim=[0.05,1.95])
```



```
event = access_log_ts[access_log_ts.ip=="185.8.139.237"].toPandas()
event = event.set_index("timestamp")
event["y"] = np.random.normal(1,0.05,size=len(event))
event["y"].plot(style='b.', ylim=[0.05,1.95])
```



You have learned

- what window functions are
- what type of windows exists
- how to count number of sessions