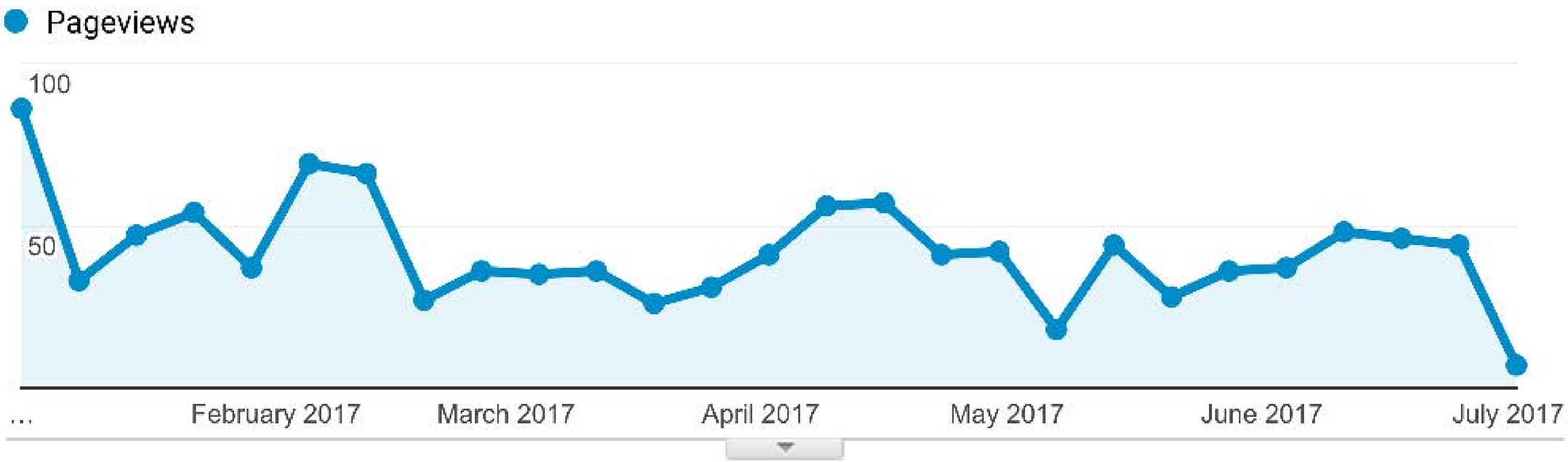
How to analyze data

http_code		ip	response_length	time	url	user_agent
0	200	109.106.133.8	21546	12/Dec /2015:01:31:46 +0400	/id53821	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)...
1	200	46.31.82.254	8777	12/Dec /2015:01:31:47 +0400	/id33929	Mozilla/5.0 (Windows NT 5.1; U; de; rv:1.9.1.6...
2	200	193.124.254.46	8731	12/Dec /2015:01:31:48 +0400	/id35754	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT...

	Page	Pageviews	% Pageviews
1.	/2012/10/cloudera-certified-hadoop-developer.html	363	32.18%
2.	/2011/12/hadoop.html	153	13.56%
3.	/2012/06/hive-web.html	110	9.75%
4.	/	96	8.51%



Pageviews

1,128

Unique Pageviews

983

Avg. Time on Page

00:03:08

	Country	Sessions	% Sessions
1.	Russia	373	43.17%
2.	Germany	211	24.42%
3.	Ukraine	160	18.52%
4.	United States	26	3.01%
5.	(not set)	23	2.66%
6.	Belarus	17	1.97%

Today we will study

Today we will study

- Aggregation in SQL

Today we will study

- Aggregation in SQL
- Aggregation of DataFrames

Today we will study

- Aggregation in SQL
- Aggregation of DataFrames
- Aggregate functions

```
access_log = spark_session.read.table("web.access_log")
```

```
access_log = spark_session.read.table("web.access_log")
```

```
access_log.limit(3).toPandas()
```

http_code		ip	response_length	time	url	user_agent
0	200	109.106.133.8	21546	12/Dec /2015:01:31:46 +0400	/id53821	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)...
1	200	46.31.82.254	8777	12/Dec /2015:01:31:47 +0400	/id33929	Mozilla/5.0 (Windows NT 5.1; U; de; rv:1.9.1.6...
2	200	193.124.254.46	8731	12/Dec /2015:01:31:48 +0400	/id35754	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT...

```
spark_session.sql("""
    select url,
           count(ip)
    from web.access_log
    group by url
""").limit(4).toPandas()
```

	url	count(ip)
0	/id53821	3
1	/id33929	2
2	/id35754	1
3	/id11744	1

```
spark_session.sql("""
    select url,
           count(ip)
    from web.access_log
    group by url
""").limit(4).toPandas()
```

	url	count(ip)
0	/id53821	3
1	/id33929	2
2	/id35754	1
3	/id11744	1

```
spark_session.sql("""
    select url,
           count(ip)
    from web.access_log
    group by url
""").limit(4).toPandas()
```

	url	count(ip)
0	/id53821	3
1	/id33929	2
2	/id35754	1
3	/id11744	1

```
spark_session.sql("""
    select url,
           count(ip)
           time
    from web.access_log
    group by url
""").limit(4).toPandas()
```

```
spark_session.sql("""
    select url,
           count(ip)
           time
    from web.access_log
    group by url
""").limit(4).toPandas()
```

```
---------------------------------------------------------------------------
AnalysisException                         Traceback (most recent call last)
<ipython-input-13-dbfe5d8d8a2e> in <module>()
      5     from web.access_log
      6     group by url
--> 7 """).limit(4).toPandas()

AnalysisException: u"expression 'access_log.`time`' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first_value) if you don't care which value you get.;;\nAggregate [url#78], [url#78, count(distinct ip#75) AS count(DISTINCT ip)#99L, time#77]\n+- SubqueryAlias access_log\n   +- Relation[http_code#74L,ip#75,response_length#76L,-time#77,url#78,user_agent#79] parquet\n"
```

```
dataframe
  .groupBy(...)
  .agg(...)
```

```python
import pyspark.sql.functions as f
```

```python
import pyspark.sql.functions as f
```

```python
access_log.groupBy("url")\
```

```python
import pyspark.sql.functions as f
```

```python
access_log.groupBy("url")\
           .agg(f.count("ip"))\
```

```python
import pyspark.sql.functions as f
```

```python
access_log.groupBy("url")\
          .agg(f.count("ip"))\
          .limit(3).toPandas()
```

```python
import pyspark.sql.functions as f
```

```python
access_log.groupBy("url")\
          .agg(f.count("ip"))\
          .limit(3).toPandas()
```

	url	count(DISTINCT ip)
0	/id37020	4
1	/id47695	2
2	/id77559	1

```
access_log.groupBy("url")\
```

```
access_log.groupBy("url")\
           .agg({"ip":"count"})\
           .limit(3).toPandas()
```

```
access_log.groupBy("url")\
          .agg({"ip":"count"})\
          .limit(3).toPandas()
```

	url	count(DISTINCT ip)
0	/id37020	4
1	/id47695	2
2	/id77559	1


OMG
404
not found

```
spark_session.sql("""
    select url,
           http_code,
           count(distinct ip)
    from web.access_log
    group by url, http_code
""").limit(4).toPandas()
```

	url	http_code	count(DISTINCT ip)
0	/id72542	200	2
1	/id10319	200	1
2	/id98806	200	1
3	/id29102	200	1

```
access_log.groupBy("url","http_code")\
           .agg(f.count("ip"))\
           .limit(4).toPandas()
```

	url	http_code	count(DISTINCT ip)
0	/id12307	200	1
1	/id84392	200	1
2	/id42947	200	1
3	/id75595	200	1

```
access_log.groupBy(f.length("url"))\
          .agg(f.count("*"))\
          .limit(4).toPandas()
```

```
access_log.groupBy(f.length("url"))\
          .agg(f.count("*"))\
          .limit(4).toPandas()
```

	length(url)	count(1)
0	31	5
1	34	4
2	28	4
3	27	7

```
access_log.groupBy()\
          .agg(f.count("*"))\
          .limit(4).toPandas()
```

```
access_log.groupBy()\
          .agg(f.count("*"))\
          .limit(4).toPandas()
```

	count(ip)
0	89206

```
access_log\
          .agg(f.count("*"))\
          .limit(4).toPandas()
```

	count(ip)
0	89206

[1, 1, 2, 2, 2, 3, 3, 3, 3]

[1, 1, 2, 2, 2, 3, 3, 3, 3]

collect_list = [1, 1, 2, 2, 2, 3, 3, 3, 3]

[1, 1, 2, 2, 2, 3, 3, 3, 3]

collect_list = [1, 1, 2, 2, 2, 3, 3, 3, 3]

count = 9

[1, 1, 2, 2, 2, 3, 3, 3, 3]

collect_list = [1, 1, 2, 2, 2, 3, 3, 3, 3]

count = 9

sum = 20

[1, 1, 2, 2, 2, 3, 3, 3, 3]

distinct

[1, 1, 2, 2, 2, 3, 3, 3, 3]

distinct

collect_set = [1, 2, 3]

[1, 1, 2, 2, 2, 3, 3, 3, 3]

distinct

collect_set = [1, 2, 3]

countDistinct = 3

[1, 1, 2, 2, 2, 3, 3, 3, 3]

distinct

collect_set = [1, 2, 3]

countDistinct = 3

sumDistinct = 6

[1, 1, 2, 2, 2, 3, 3, 3, 3]

[1, 1, 2, 2, 2, 3, 3, 3, 3]

math

[1, 1, 2, 2, 2, 3, 3, 3, 3]

math

min max avg var ...

```
access_log.groupBy("url")\
          .agg(f.count("response_length"),
               f.min("response_length"),
               f.max("response_length"),
               f.avg("response_length"),
               )\
          .limit(4).toPandas()
```


leave
moment
time
whisper
spring
those
behind
your
picture
see
stare
through
may
worship
size
claim
would
delirious
blue
pole
sing
hine
eat
like
an

```
access_log.select(f.split("user_agent", " ").alias("words"))\
          .limit(10).toPandas()
```

	words
0	[Mozilla/5.0, (Macintosh; Intel, Mac, OS, X,...
1	[Mozilla/5.0, (Windows, NT 5.1; U; de; rv:...
2	[Mozilla/4.0, (compatible; MSIE 7.0; Window...
3	[Mozilla/5.0, (Linux; Android, 4.4.4;, nb-no;...
4	[Mozilla/5.0, (Linux; Android, 4.4.4;, nb-no;...
5	[Mozilla/5.0, (Macintosh; Intel, Mac, OS, X,...
6	[Mozilla/5.0, (Windows; NT, 10.0;,WOW64), App...
7	[Mozilla/5.0, (Macintosh; Intel, Mac, OS, X,...
8	[Mozilla/5.0, (Windows; U;, Windows, NT, 6.0;...
9	[Mozilla/5.0, (Macintosh; U;, PPC, Mac, OS, X...

```
access_log.select(f.split("user_agent", " ").alias("words"))\
          .select(f.explode("words").alias("word"))\
          .limit(10).toPandas()
```

	words
0	Mozilla/5.0
1	(Macintosh;
2	Intel
3	Mac
4	OS
5	X
6	10_9_4)
7	AppleWebKit/537.36
8	(KHTML,
9	like

```
access_log.select(f.split("user_agent", " ").alias("words"))\
         .select(f.explode("words").alias("word"))\
         .groupBy("word")\
         .agg(f.count("*").alias("count"))\
         .limit(10).toPandas()
```

	words	count
0	Firefox/3.6.8	90
1	3.5.21022;	801
2	rv:1.9.2.19)	6
3	Firefox/33.1.1	5
4	Release/02.10.2014	9
5	OPR/25.0.1614.71	47
6	Chrome/38.0.2125.104	42
7	AppleWebKit/537.21	34
8	Gecko/20110420	63
9	MASN)	4

```
access_log.select(f.split("user_agent", " ").alias("words"))\
         .select(f.explode("words").alias("word"))\
         .groupBy("word")\
         .agg(f.count("*").alias("count"))\
         .orderBy(f.col("count").desc())\
         .limit(10).toPandas()
```

	words	count
0	Mozilla/5.0	75565
1	like	63791
2	Gecko)	58926
3	(KHTML,	58551
4	NT	50439
5	AppleWebKit/537.21	48648
6	Safari/537.36	48648
7	(Windows	37942
8	CLR	32140
9	.NET	31648

```
access_log.select(f.split("user_agent", " ").alias("words"))\
         .select(f.explode("words").alias("word"))\
         .groupBy("word")\
         .agg(f.count("*").alias("count"))\
         .orderBy(f.col("count").desc())\
         .limit(10).toPandas()
```

	words	count
0	Mozilla/5.0	75565
1	like	63791
2	Gecko)	58926
3	(KHTML,	58551
4	NT	50439
5	AppleWebKit/537.21	48648
6	Safari/537.36	48648
7	(Windows	37942
8	CLR	32140
9	.NET	31648

Today you have:

- recapped how to do aggregation in SQL
- found the ways how to aggregate data in DataFrames
- had a lot of fun and practice with aggregation