```
scala> import org.apache.log4j.{Level, Logger}
import org.apache.log4j.{Level, Logger}
scala> import org.apache.spark.sql.{Column, SparkSession}
import org.apache.spark.sql.{Column, SparkSession}
scala> import
org.apache.spark.sql.functions.{regexp_extract,sum,col,to_date,udf,to_tim
estamp,desc,dayofyear,year}
import org.apache.spark.sql.functions.{regexp_extract, sum, col, to_date,
udf, to_timestamp, desc, dayofyear, year}
scala> val spark =
SparkSession.builder().appName("WebLog").master("local[*]").getOrCreate()
23/05/09 11:16:34 WARN SparkSession: Using an existing Spark session;
only runtime SQL configurations will take effect.
spark: org.apache.spark.sql.SparkSession =
org.apache.spark.sql.SparkSession@2fe172fe
scala> val base_df =
spark.read.text("/home/student/Downloads/weblog.csv")
base_df: org.apache.spark.sql.DataFrame = [value: string]
scala> import spark.implicits._
import spark.implicits._
scala> val base_df =
spark.read.text("/home/student/Downloads/weblog.csv")
base_df: org.apache.spark.sql.DataFrame = [value: string]
scala> base_df.printSchema()
root
 |-- value: string (nullable = true)
scala> base_df.show(3,false)
+-----------------------------------------------------------------+
|value                                                            |
+-----------------------------------------------------------------+
|IP,Time,URL,Staus                                                |
|10.128.2.1,[29/Nov/2017:06:58:55,GET /login.php HTTP/1.1,200     |
|10.128.2.1,[29/Nov/2017:06:59:02,POST /process.php HTTP/1.1,302|
+-----------------------------------------------------------------+
only showing top 3 rows
scala> val parsed_df =
base_df.select(regexp_extract($"value","""^([^(\s|,)]+)""",1).alias("host
"),
     |
regexp_extract($"value","""^.*\[(\d\d/\w{3}/\d{4}:\d{2}:\d{2}:\d{2})""",1
).as("timestamp"),
     |
regexp_extract($"value","""^.*\w+\s+([^\s]+)\s+HTTP.*""",1).as("path"),
     |
regexp_extract($"value","""^.*,([^\s]+)$""",1).cast("int").alias("status"
))
parsed_df: org.apache.spark.sql.DataFrame = [host: string, timestamp:
string ... 2 more fields]
scala> parsed_df.show(5,false)
+----------+--------------------+------------------------+------+
|host      |timestamp           |path                    |status|
+----------+--------------------+------------------------+------+
|IP        |                    |                        |null  |
|10.128.2.1|29/Nov/2017:06:58:55|/login.php              |200   |
```

```
|10.128.2.1|29/Nov/2017:06:59:02|/process.php            |302   |
|10.128.2.1|29/Nov/2017:06:59:03|/home.php               |200   |
|10.131.2.1|29/Nov/2017:06:59:04|/js/vendor/moment.min.js|200   |
+----------+--------------------+------------------------+------+
only showing top 5 rows
scala> parsed_df.printSchema()
root
 |-- host: string (nullable = true)
 |-- timestamp: string (nullable = true)
 |-- path: string (nullable = true)
 |-- status: integer (nullable = true)
scala>  println("Number of bad row in the initial dataset : " +
base_df.filter($"value".isNull).count())
Number of bad row in the initial dataset : 0
scala>  val bad_rows_df = parsed_df.filter($"host".isNull ||
$"timestamp".isNull || $"path".isNull || $"status".isNull)
bad_rows_df: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] =
[host: string, timestamp: string ... 2 more fields]
scala>   println("Number of bad rows : " + bad_rows_df.count())
Number of bad rows : 219
scala>   val t = parsed_df.columns.map(col_name =>
count_null(col(col_name)))
t: Array[org.apache.spark.sql.Column] = Array(sum(CAST((host IS NULL) AS
INT)) AS host, sum(CAST((timestamp IS NULL) AS INT)) AS timestamp,
sum(CAST((path IS NULL) AS INT)) AS path, sum(CAST((status IS NULL) AS
INT)) AS status)
scala>   parsed_df.select(t: _*).show()
+----+---------+----+------+
|host|timestamp|path|status|
+----+---------+----+------+
|   0|        0|   0|   219|
+----+---------+----+------+
scala>  val bad_status_df =
base_df.select(regexp_extract($"value","""([^\d]+)$""",1).as("bad_status"
)).filter($"bad_status".notEqual(""))
bad_status_df: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] =
[bad_status: string]
scala>   println("Number of bad rows : " + bad_status_df.count())
Number of bad rows : 219
scala>   bad_status_df.show(5)
+--------------------+
|          bad_status|
+--------------------+
|   IP,Time,URL,Staus|
|chmod:,cannot,'a....|
|chmod:,cannot,'er...|
|rm:,cannot,'*.o':,No|
|rm:,cannot,'a.out...|
+--------------------+
only showing top 5 rows
scala>   println("The count of null value : " +
cleaned_df.filter($"host".isNull || $"timestamp".isNull ||
$"path".isNull|| $"status".isNull).count())
The count of null value : 0
```

```
scala>   println("Before : " + parsed_df.count() + " | After : " +
cleaned_df.count())
Before : 16008 | After : 15789
scala>
     |   cleaned_df.select(to_date($"timestamp")).show(2)
+------------------+
|to_date(timestamp)|
+------------------+
|              null|
|              null|
+------------------+
only showing top 2 rows
scala>   val month_map = Map("Jan" -> 1, "Feb" -> 2, "Mar" -> 3, "Apr" ->
4, "May" -> 5, "Jun" -> 6, "Jul" -> 7, "Aug" -> 8
     |     , "Sep" -> 9, "Oct" -> 10, "Nov" -> 11, "Dec" -> 12)
month_map: scala.collection.immutable.Map[String,Int] = Map(Nov -> 11,
Jul -> 7, Mar -> 3, Jan -> 1, Oct -> 10, Dec -> 12, Feb -> 2, May -> 5,
Apr -> 4, Aug -> 8, Sep -> 9, Jun -> 6)
scala>   def parse_clf_time(s: String) ={
     |     "%3$s-%2$s-%1$s
%4$s:%5$s:%6$s".format(s.substring(0,2),month_map(s.substring(3,6)),s.sub
string(7,11)
     |       ,s.substring(12,14),s.substring(15,17),s.substring(18))
     |   }
parse_clf_time: (s: String)String
scala>   val toTimestamp = udf[String, String](parse_clf_time(_))
toTimestamp: org.apache.spark.sql.expressions.UserDefinedFunction =
SparkUserDefinedFunction($Lambda$4429/0x0000000841833840@2f414e82,StringT
ype,List(Some(class[value[0]: string])),Some(class[value[0]:
string]),None,true,true)
scala>   val logs_df =
cleaned_df.select($"*",to_timestamp(toTimestamp($"timestamp")).alias("tim
e")).drop("timestamp")
logs_df: org.apache.spark.sql.DataFrame = [host: string, path: string ...
2 more fields]
scala>   logs_df.printSchema()
root
 |-- host: string (nullable = true)
 |-- path: string (nullable = true)
 |-- status: integer (nullable = true)
 |-- time: timestamp (nullable = true)
scala>   logs_df.show(2)
+----------+------------+------+-------------------+
|      host|        path|status|               time|
+----------+------------+------+-------------------+
|10.128.2.1|  /login.php|   200|2017-11-29 06:58:55|
|10.128.2.1|/process.php|   302|2017-11-29 06:59:02|
+----------+------------+------+-------------------+
only showing top 2 rows
scala>   logs_df.cache()
res20: logs_df.type = [host: string, path: string ... 2 more fields]
scala> logs_df.describe("status").show()
+-------+------------------+
|summary|            status|
```

```
+-------+------------------+
|  count|             15789|
|   mean|230.19469250744189|
| stddev| 50.05853522906924|
|    min|               200|
|    max|               404|
+-------+------------------+
scala> logs_df.groupBy("status").count().sort("status").show()
+------+-----+
|status|count|
+------+-----+
|   200|11330|
|   206|   52|
|   302| 3498|
|   304|  658|
|   404|  251|
+------+-----+
scala> logs_df.groupBy("host").count().filter($"count" > 10).show()
+----------+-----+
|      host|count|
+----------+-----+
|10.131.2.1| 1626|
|10.128.2.1| 4257|
|10.130.2.1| 4056|
|10.131.0.1| 4198|
|10.129.2.1| 1652|
+----------+-----+
scala> val unique_host_count = logs_df.select("host").distinct().count()
unique_host_count: Long = 5
scala>   println("Unique hosts : %d".format(unique_host_count))
Unique hosts : 5
scala>   val avg_daily_request_per_host_df =
total_req_per_day_df.join(daily_hosts_df,total_req_per_day_df("day") ===
daily_hosts_df("day")&& total_req_per_day_df("year") ===
daily_hosts_df("year")).select(daily_hosts_df("day"),daily_hosts_df("year
"),(total_req_per_day_df("count")
/daily_hosts_df("count")).alias("avg_req_per_host_per_day")).cache()
<console>:30: error: not found: value daily_hosts_df
         val avg_daily_request_per_host_df =
total_req_per_day_df.join(daily_hosts_df,total_req_per_day_df("day") ===
daily_hosts_df("day")&& total_req_per_day_df("year") ===
daily_hosts_df("year")).select(daily_hosts_df("day"),daily_hosts_df("year
"),(total_req_per_day_df("count")
/daily_hosts_df("count")).alias("avg_req_per_host_per_day")).cache()
scala> val daily_hosts_df =
logs_df.withColumn("day",dayofyear($"time")).withColumn("year",year($"tim
e")).select("host","day","year").distinct().groupBy("day","year").count()
.sort("year","day").cache()
daily_hosts_df: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] =
[day: int, year: int ... 1 more field]
scala>   daily_hosts_df.show(5)
+---+----+-----+
|day|year|count|
+---+----+-----+
```

```
|311|2017|    1|
|312|2017|    5|
|313|2017|    5|
|314|2017|    5|
|315|2017|    5|
+---+----+-----+
only showing top 5 rows
scala>
     |   val total_req_per_day_df = logs_df.withColumn("day",
dayofyear($"time")).withColumn("year", year($"time")).groupBy("day",
"year").count()
total_req_per_day_df: org.apache.spark.sql.DataFrame = [day: int, year:
int ... 1 more field]
scala>   val avg_daily_request_per_host_df =
total_req_per_day_df.join(daily_hosts_df,total_req_per_day_df("day") ===
daily_hosts_df("day")&& total_req_per_day_df("year") ===
daily_hosts_df("year")).select(daily_hosts_df("day"),daily_hosts_df("year
"),(total_req_per_day_df("count")
/daily_hosts_df("count")).alias("avg_req_per_host_per_day")).cache()
avg_daily_request_per_host_df:
org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [day: int, year:
int ... 1 more field]
scala>   avg_daily_request_per_host_df.show(5)
+---+----+-----------------------+
|day|year|avg_req_per_host_per_day|
+---+----+-----------------------+
|335|2017|                   93.6|
|327|2017|                   76.0|
| 60|2018|     10.333333333333334|
|350|2017|     51.666666666666664|
| 46|2018|      6.666666666666667|
+---+----+-----------------------+
only showing top 5 rows
scala>   println("found %d 404 Urls".format(not_found_df.count()))
found 251 404 Urls
scala> not_found_df.select("path").distinct().show(40,false)
+--------------------------------------+
|path                                  |
+--------------------------------------+
|/css/bootstrap.min.css.map            |
|/robots.txt                           |
|/djs/vendor/bootstrap-datetimepicker.js|
|/favicon.ico                          |
+--------------------------------------+
scala>
not_found_df.groupBy("path").count().sort("count").show(20,false)
+--------------------------------------+-----+
|path                                  |count|
+--------------------------------------+-----+
|/css/bootstrap.min.css.map            |1    |
|/djs/vendor/bootstrap-datetimepicker.js|7    |
|/favicon.ico                          |19   |
|/robots.txt                           |224  |
+--------------------------------------+-----+
```

```
scala>   not_found_df.groupBy("path").agg("host" ->
"collect_list","status" -> "count").sort("count(status)").show(20)
+--------------------+--------------------+-------------+
|                path|  collect_list(host)|count(status)|
+--------------------+--------------------+-------------+
|/css/bootstrap.mi...|        [10.130.2.1]|            1|
|/djs/vendor/boots...|[10.131.0.1, 10.1...|            7|
|        /favicon.ico|[10.128.2.1, 10.1...|           19|
|         /robots.txt|[10.131.0.1, 10.1...|          224|
+--------------------+--------------------+-------------+
scala>   not_found_df.groupBy("path").agg("host" ->
"collect_set","status" -> "count").sort("count(status)").show(20)
+--------------------+--------------------+-------------+
|                path|   collect_set(host)|count(status)|
+--------------------+--------------------+-------------+
|/css/bootstrap.mi...|        [10.130.2.1]|            1|
|/djs/vendor/boots...|[10.130.2.1, 10.1...|            7|
|        /favicon.ico|[10.130.2.1, 10.1...|           19|
|         /robots.txt|[10.130.2.1, 10.1...|          224|
+--------------------+--------------------+-------------+
```