```
import org.apache.log4j.{Level, Logger}
import org.apache.spark.sql.{Column, SparkSession}
import org.apache.spark.sql.functions.
{regexp_extract,sum,col,to_date,udf,to_timestamp,desc,dayofyear,year
}

val spark =
SparkSession.builder().appName("WebLog").master("local[*]").getOrCre
ate()
val base_df = spark.read.text("/home/deptii/Web_Log/weblog.csv")
base_df.printSchema()

import spark.implicits._

val base_df = spark.read.text("/home/deptii/Web_Log/weblog.csv")
base_df.printSchema()

base_df.show(3,false)


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.show(5,false)
parsed_df.printSchema()


  println("Number of bad row in the initial dataset : " +
base_df.filter($"value".isNull).count())

  val bad_rows_df = parsed_df.filter($"host".isNull ||
$"timestamp".isNull || $"path".isNull || $"status".isNull)
  println("Number of bad rows : " + bad_rows_df.count())

  //val bad_rows_df =
parsed_df.filter($"host".isNull.or($"timestamp".isNull).or($"path".i
sNull)
  // .or($"status".isNull)).count


  def count_null(col_name: Column): Column =
sum(col_name.isNull.cast("int")).alias(col_name.toString())
  val t = parsed_df.columns.map(col_name =>
count_null(col(col_name)))
  parsed_df.select(t: _*).show()

  val bad_status_df =
base_df.select(regexp_extract($"value","""([^\d]+)
```

```
$"""",1).as("bad_status")).filter($"bad_status".notEqual(""))
  println("Number of bad rows : " + bad_status_df.count())
  bad_status_df.show(5)


  val cleaned_df = parsed_df.na.drop()

  println("The count of null value : " +
cleaned_df.filter($"host".isNull || $"timestamp".isNull ||
$"path".isNull|| $"status".isNull).count())

  println("Before : " + parsed_df.count() + " | After : " +
cleaned_df.count())

  cleaned_df.select(to_date($"timestamp")).show(2)

  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)
  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.substrin
g(7,11)
      ,s.substring(12,14),s.substring(15,17),s.substring(18))
  }
  val toTimestamp = udf[String, String](parse_clf_time(_))
  val logs_df =
cleaned_df.select($"*",to_timestamp(toTimestamp($"timestamp")).alias
("time")).drop("timestamp")
  logs_df.printSchema()
  logs_df.show(2)
  logs_df.cache()



  logs_df.describe("status").show()

  logs_df.groupBy("status").count().sort("status").show()


  logs_df.groupBy("host").count().filter($"count" > 10).show()

  logs_df.groupBy("path").count().sort(desc("count")).show()

  logs_df.groupBy("path").count().sort(desc("count")).show(10)


  logs_df.filter($"status" =!=
200).groupBy("path").count().sort(desc("count")).show(10)

  val unique_host_count = logs_df.select("host").distinct().count()
  println("Unique hosts : %d".format(unique_host_count))

  val daily_hosts_df =
```

```
logs_df.withColumn("day",dayofyear($"time")).withColumn("year",year(
$"time")).select("host","day","year").distinct().groupBy("day","year
").count().sort("year","day").cache()
  daily_hosts_df.show(5)

  val total_req_per_day_df = logs_df.withColumn("day",
dayofyear($"time")).withColumn("year", year($"time")).groupBy("day",
"year").count()
  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.show(5)


  val not_found_df = logs_df.where($"status" === 404).cache()
  println("found %d 404 Urls".format(not_found_df.count()))

  not_found_df.select("path").distinct().show(40,false)


  not_found_df.groupBy("path").count().sort("count").show(20,false)
  not_found_df.groupBy("path").agg("host" -> "collect_list","status"
-> "count").sort("count(status)").show(20)
  not_found_df.groupBy("path").agg("host" -> "collect_set","status"
-> "count").sort("count(status)").show(20)



not_found_df.groupBy("host").count().sort(desc("count")).show(trunca
te = false)


  val errors_by_date_pair_df = not_found_df.withColumn("day",
dayofyear($"time")).withColumn("year",
year($"time")).groupBy("day","year").count()
  not_found_df.withColumn("day",
dayofyear($"time")).withColumn("year",
year($"time")).groupBy("day","year").count().sort($"year",
$"day").show(10)
```