12: Spark on Zeppelin – Dataframe pivot

Posted on September 17, 2018

Pre-requisite: Docker is installed on your machine for Mac OS X (E.g. $ brew cask install docker) or Windows 10. Docker interview Q&As. This extends setting up Apache Zeppelin Notebook.

Step 1: Pull this from the docker hub, and build the image with the following command.

```
$ docker pull apache/zeppelin:0.7.3

```

You can verify the image with the "docker images" command.

300+ Java Interview FAQs

300+ Java FAQs
16+ Java Key Areas Q&As
150+ Java Architect FAQs
80+ Java Code Quality Q&As
150+ Java Coding Q&As

300+ Big Data Interview FAQs

300+ Big Data FAQs
Tutorials - Big Data
TUT - Starting Big Data
TUT - Starting Spark & Scala

Step 2: Run the container with the above image.

```
$ docker run --rm -it -p 8080:8080 apache/zeppelin

```

Step 3: Open Zeppelin notebook via a web browser "http:localhost:8080". Create a note book with "spark" as a default interpreter.

Calculate the average temperature for each station for each month

The pivot function to the rescue. A pivot is an aggregation where one or more of the grouping columns with distinct values (E.g. Statuses like pending, completed, etc or Types like Basic, Advanced, etc) transposed into individual columns. Pivot tables are an essential part of data analysis and reporting. In the following example let's transpose the distinct "year-month" values foreach weather station.

```
%spark


import java.sql.Date
import org.apache.spark.sql.functions._

case class Weather (stationId: Integer, date: jav

val weather = Seq(
    Weather(1, Date.valueOf("2018-06-01"),34.0 ),
    Weather(2, Date.valueOf("2017-12-01"), 32.0),
    Weather(2, Date.valueOf("2018-06-01"), 28.0),
    Weather(2, Date.valueOf("2017-01-01"), 26.0),
    Weather(1, Date.valueOf("2017-01-01"), 24.0),
    Weather(1, Date.valueOf("2017-12-01"), 30.0),
    Weather(2, Date.valueOf("2017-01-01"), 26.0),
    Weather(1, Date.valueOf("2017-12-01"), 24.0),
    Weather(1, Date.valueOf("2018-06-01"), 30.0)
)

```

TUT - Starting with Python
TUT - Kafka
TUT - Pig
TUT - Apache Storm
TUT - Spark Scala on Zeppelin
TUT - Cloudera
TUT - Cloudera on Docker
TUT - File Formats
TUT - Spark on Docker
TUT - Flume
TUT - Hadoop (HDFS)
TUT - HBase (NoSQL)
TUT - Hive (SQL)
TUT - Hadoop & Spark
TUT - MapReduce
TUT - Spark and Scala
TUT - Spark & Java
TUT - PySpark on Databricks
TUT - Zookeeper

800+ Java Interview Q&As

300+ Core Java Q&As
300+ Enterprise Java Q&As
150+ Java Frameworks Q&As
120+ Companion Tech Q&As
Tutorials - Enterprise Java

```
val weathereDF  = spark.createDataFrame(
  spark.sparkContext.parallelize(weather)
)

weathereDF.show()


val monthlyWeatherDF = weathereDF.withColumn("year
                                   year($"date"),
                                 .groupBy("statio
                                 .pivot("year_mon

val monthlyAvgTempDF = monthlyWeatherDF.agg(avg($

monthlyAvgTempDF.show()
```

Output:

```
import java.sql.Date
import org.apache.spark.sql.functions._
defined class Weather
weather: Seq[Weather] = List(Weather(1,2018-06-01
weathereDF: org.apache.spark.sql.DataFrame = [sta
+---------+----------+-----------+
|stationId|      date|temperature|
+---------+----------+-----------+
|        1|2018-06-01|       34.0|
|        2|2017-12-01|       32.0|
|        2|2018-06-01|       28.0|
|        2|2017-01-01|       26.0|
|        1|2017-01-01|       24.0|
|        1|2017-12-01|       30.0|
|        2|2017-01-01|       26.0|
|        1|2017-12-01|       24.0|
|        1|2018-06-01|       30.0|
+---------+----------+-----------+
monthlyWeatherDF: org.apache.spark.sql.Relational
monthlyAvgTempDF: org.apache.spark.sql.DataFrame =
+---------+-------+-------+-------+
|stationId|2017-01|2017-12|2018-06|
+---------+-------+-------+-------+
|        1|   24.0|   27.0|   32.0|
|        2|   26.0|   32.0|   28.0|
+---------+-------+-------+-------+

```

‹ 11: Spark on Zeppelin – Dataframe groupBy, collect_list, explode & window

17+ DevOps CI/CD Jenkins interview Q&As ›