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

Posted on September 15, 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
```

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

You can verify the image with the “docker images” command.

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.

How to aggregate values into collection after groupBy?

collect_list(expr) – Collects and returns a list of non-unique elements.

collect_set(expr) – Collects and returns a set of unique elements.

concat(str1, str2, …, strN) – Returns the concatenation of str1, str2, …, strN.

concat_ws(sep, [str | array(str)]+) – Returns the concatenation of the strings separated by sep.

```
%spark


import org.apache.spark.sql.functions.collect_list

case class Employee (id: Integer, name: String, ex

val employees = Seq(
    Employee(1, "John", "Java" ),
    Employee(2, "Peter","Scala"),
    Employee(2, "Peter", "Python"),
    Employee(2, "Peter","Spark"),
```

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

```
    Employee(1, "John", "JEE"),
    Employee(6, "Elliot", "Unix")
)

val employeeDF  = spark.createDataFrame(
  spark.sparkContext.parallelize(employees)
)

val expertise = collect_list($"expertise").alias("

val resultDF = employeeDF.groupBy($"id")
                         .agg(expertise)

resultDF.show(false)

```

Output:

```
import org.apache.spark.sql.functions.collect_list
defined class Employee
employees: Seq[Employee] = List(Employee(1,John,Ja
employeeDF: org.apache.spark.sql.DataFrame = [id:
expertise: org.apache.spark.sql.Column = collect_
resultDF: org.apache.spark.sql.DataFrame = [id: in
+---+--------------------+
|id |expertise           |
+---+--------------------+
|1  |[Java, JEE]         |
|6  |[Unix]              |
|2  |[Scala, Python, Spark]|
+---+--------------------+

```

What if expertise is an array? explode first

```
%spark


import org.apache.spark.sql.functions.{collect_lis

case class Employee (id: Integer, name: String, ex

val employees = Seq(
    Employee(1, "John", Array("Java", "Scala") ),
    Employee(2, "Peter",Array("Scala")),
    Employee(2, "Peter", Array("Python", "Git")),
    Employee(2, "Peter", Array("Spark")),

```

```
    Employee(1, "John", Array("JEE", "Spring")),
    Employee(6, "Elliot", Array("Unix", "DevOps"))
)

val employeeDF  = spark.createDataFrame(
  spark.sparkContext.parallelize(employees)
)

employeeDF.show()

val resultDF = employeeDF.select($"id", explode($"
                         .groupBy($"id")
                         .agg(collect_list($"exper

resultDF.show(false)

```

Output:

```
import org.apache.spark.sql.functions.{collect_lis
defined class Employee
employees: Seq[Employee] = List(Employee(1,John,[J
employeeDF: org.apache.spark.sql.DataFrame = [id:
+---+------+-------------+
| id|  name|     expertise|
+---+------+-------------+
|  1|  John| [Java, Scala]|
|  2| Peter|       [Scala]|
|  2| Peter| [Python, Git]|
|  2| Peter|       [Spark]|
|  1|  John| [JEE, Spring]|
|  6|Elliot|[Unix, DevOps]|
+---+------+-------------+
resultDF: org.apache.spark.sql.DataFrame = [id: in
+---+--------------------------+
|id |collect_list(expertise)   |
+---+--------------------------+
|1  |[Java, Scala, JEE, Spring]|
|6  |[Unix, DevOps]            |
|2  |[Scala, Python, Git, Spark]|
+---+--------------------------+

```

Cumulative expertise the window function

```
%spark

```

```
3
4  import org.apache.spark.sql.functions.{collect_se
5  import org.apache.spark.sql.expressions.Window
6
7  case class Employee (id: Integer, name: String, e
8
9  val employees = Seq(
10     Employee(1, "John", Array("Java", "Scala") ),
11     Employee(2, "Peter",Array("Scala")),
12     Employee(2, "Peter", Array("Python", "Git")),
13     Employee(2, "Peter", Array("Spark")),
14     Employee(1, "John", Array("JEE", "Spring")),
15     Employee(6, "Elliot", Array("Unix", "DevOps")
16 )
17
18 val employeeDF  = spark.createDataFrame(
19   spark.sparkContext.parallelize(employees)
20 )
21
22 employeeDF.show()
23
24 val resultDF = employeeDF.select($"id", explode($"
25                          .withColumn("expertise_c
26
27 resultDF.show(false)
28
```

Output:

```
1  import org.apache.spark.sql.functions.{collect_se
2  import org.apache.spark.sql.expressions.Window
3  defined class Employee
4  employees: Seq[Employee] = List(Employee(1,John,[
5  employeeDF: org.apache.spark.sql.DataFrame = [id:
6  +---+------+-------------+
7  | id|  name|     expertise|
8  +---+------+-------------+
9  |  1|  John| [Java, Scala]|
10 |  2| Peter|       [Scala]|
11 |  2| Peter| [Python, Git]|
12 |  2| Peter|       [Spark]|
13 |  1|  John| [JEE, Spring]|
14 |  6|Elliot|[Unix, DevOps]|
15 +---+------+-------------+
16 resultDF: org.apache.spark.sql.DataFrame = [id: i
17 +---+---------+-------------------------+
18 |id |expertise|expertise_aggregated     |
19 +---+---------+-------------------------+
20 |1  |JEE      |[JEE]                    |
21 |1  |Java     |[Java, JEE]              |
22 |1  |Scala    |[Scala, Java, JEE]       |
23 |1  |Spring   |[Scala, Java, Spring, JEE] |
```

```
|6  |DevOps    |[DevOps]                    |
|6  |Unix      |[Unix, DevOps]              |
|2  |Git       |[Git]                       |
|2  |Python    |[Python, Git]               |
|2  |Scala     |[Scala, Python, Git]        |
|2  |Spark     |[Scala, Spark, Python, Git]|
+---+---------+---------------------------+
```

‹ 10: Spark on Zeppelin – union, udf and explode

12: Spark on Zeppelin – Dataframe pivot ›

Disclaimer