Spark: The Definitive Guide

Notebooks here are created from book's Code (https://github.com/databricks/Spark-The-Definitive-Guide/tree/master/code) and Data (https://github.com/databricks/Spark-The-Definitive-Guide/tree/master/data)

After cloning the Git repo (https://github.com/databricks/Spark-The-Definitive-Guide) locally, set os env var `SPARK_BOOK_DATA_PATH` to that folder.

How to Get Started

Databrick Cloud Sandbox

Use Spark Cluster free at :

https://community.cloud.databricks.com/ (https://community.cloud.databricks.com/)

Local Installation

To install pyspark

```
$ pip install pyspark
```

To start jupyter notebook

```
$ PYSPARK_DRIVER_PYTHON="jupyter" PYSPARK_DRIVER_PYTHON_OPTS="n
otebook" pyspark
```

To use other pyspark packages, add `--packages <pkg-name>` , e.g.

```
$ PYSPARK_DRIVER_PYTHON="jupyter" PYSPARK_DRIVER_PYTHON_OPTS="n
otebook" pyspark --packages graphframes:graphframes:0.7.0-spark
2.4-s_2.11
```

In [1]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-02-intro")\
    .getOrCreate()

import os
SPARK_BOOK_DATA_PATH = os.environ['SPARK_BOOK_DATA_PATH']
```

In [2]:
```
sc = spark.sparkContext
```

In [2]:
```
spark
```

Out[2]:

SparkSession - hive

SparkContext

Spark UI (http://192.168.1.2:4041)

Version

v2.4.3

Master

local[*]

AppName

PySparkShell

In [3]:
```
# spark.range(1000) returns a RDD, toDF() converts it to DataFrame
myRange = spark.range(10).toDF("number")
myRange.show()
```

```
+------+
|number|
+------+
|     0|
|     1|
|     2|
|     3|
|     4|
|     5|
|     6|
|     7|
|     8|
|     9|
+------+
```

In [4]:
```
divisBy2 = myRange.where("number % 2 = 0")
divisBy2.collect()
```

Out[4]:
```
[Row(number=0), Row(number=2), Row(number=4), Row(number=6), Row(numbe
r=8)]
```

In [5]:
```
# convert collection to RDD
rdd = sc.parallelize(range(10))

rdd.collect()
```

Out[5]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [6]:
```
from pyspark.sql import Row

test_rdd = spark.sparkContext.parallelize([Row(1), Row(2), Row(3)])
```

In [7]:
```
type(test_rdd)
```

Out[7]:
```
pyspark.rdd.RDD
```

In [8]:
```
type(spark.range(10))
```

Out[8]:
```
pyspark.sql.dataframe.DataFrame
```

In [9]:
```
test_df = test_rdd.toDF()
```

In [10]:
```
type(test_df)
```

Out[10]:
```
pyspark.sql.dataframe.DataFrame
```

In [11]:
```
test_df.toDF("id").show()
```

```
+---+
| id|
+---+
|  1|
|  2|
|  3|
+---+
```

In [12]:
```
# read from file
file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/csv/2015-summary.
flightData2015 = spark\
  .read\
  .option("inferSchema", "true")\
  .option("header", "true")\
  .csv(file_path)
```

short form:

```
flightData2015 = spark.read.csv(file_path, header=True,
inferSchema=True)
```

In [18]:
```
# write to parquet file
file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/parquet/2015-summa
flightData2015.write\
    .format("parquet")\
    .mode("overwrite")\
    .save(file_path)
```

In [22]:
```
flightData2015.show(5)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|    United States|            Croatia|    1|
|    United States|            Ireland|  344|
|            Egypt|      United States|   15|
|    United States|              India|   62|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [20]:
```
# read it back
flightData2015_2 = spark\
    .read\
    .format("parquet")\
    .load(file_path)
```

In [21]:
```
flightData2015_2.show(5)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|    United States|            Croatia|    1|
|    United States|            Ireland|  344|
|            Egypt|      United States|   15|
|    United States|              India|   62|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [13]:
```
flightData2015.printSchema()
```

```
root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: integer (nullable = true)
```

In [14]:
```
flightData2015.schema
```

Out[14]:
```
StructType(List(StructField(DEST_COUNTRY_NAME,StringType,true),StructF
ield(ORIGIN_COUNTRY_NAME,StringType,true),StructField(count,IntegerTyp
e,true)))
```

In [15]:
```
flightData2015.columns
```

Out[15]:
```
['DEST_COUNTRY_NAME', 'ORIGIN_COUNTRY_NAME', 'count']
```

In [16]:
```
flightData2015.count()
```

Out[16]:
```
256
```

In [17]:
```
flightData2015.show(5)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|    United States|            Croatia|    1|
|    United States|            Ireland|  344|
|            Egypt|      United States|   15|
|    United States|              India|   62|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [17]:
```
# convert DataFrame to temp Table
flightData2015.createOrReplaceTempView("flight_data_2015")
```

In [26]:
```
# run SQL directly against temp table
sqlWay = spark.sql("""
SELECT DEST_COUNTRY_NAME, count(1)
FROM flight_data_2015
GROUP BY DEST_COUNTRY_NAME
--having count(1) > 4
""")
```

In [27]:
```
sqlWay.show(5)
```

```
+-----------------+--------+
|DEST_COUNTRY_NAME|count(1)|
+-----------------+--------+
|         Anguilla|       1|
|           Russia|       1|
|         Paraguay|       1|
|          Senegal|       1|
|           Sweden|       1|
+-----------------+--------+
only showing top 5 rows
```

In [22]:
```
dataFrameWay = flightData2015\
  .groupBy("DEST_COUNTRY_NAME")\
  .count()
```

In [25]:
```
dataFrameWay.show(5)
```

```
+-----------------+-----+
|DEST_COUNTRY_NAME|count|
+-----------------+-----+
|         Anguilla|    1|
|           Russia|    1|
|         Paraguay|    1|
|          Senegal|    1|
|           Sweden|    1|
+-----------------+-----+
only showing top 5 rows
```

Spark Catalyst turns logic plans to optimized physical plan

In [30]:
```
sqlWay.explain()
```

```
== Physical Plan ==
*(2) HashAggregate(keys=[DEST_COUNTRY_NAME#26], functions=[count(1)])
+- Exchange hashpartitioning(DEST_COUNTRY_NAME#26, 200)
   +- *(1) HashAggregate(keys=[DEST_COUNTRY_NAME#26], functions=[parti
al_count(1)])
      +- *(1) FileScan csv [DEST_COUNTRY_NAME#26] Batched: false, Form
at: CSV, Location: InMemoryFileIndex[file:/home/gong/spark/books/Spark
-The-Definitive-Guide/data/flight-data/csv/201..., PartitionFilters:
[], PushedFilters: [], ReadSchema: struct<DEST_COUNTRY_NAME:string>
```

In [31]:
```
dataFrameWay.explain()
```

```
== Physical Plan ==
*(2) HashAggregate(keys=[DEST_COUNTRY_NAME#26], functions=[count(1)])
+- Exchange hashpartitioning(DEST_COUNTRY_NAME#26, 200)
   +- *(1) HashAggregate(keys=[DEST_COUNTRY_NAME#26], functions=[parti
al_count(1)])
      +- *(1) FileScan csv [DEST_COUNTRY_NAME#26] Batched: false, Form
at: CSV, Location: InMemoryFileIndex[file:/home/gong/spark/books/Spark
-The-Definitive-Guide/data/flight-data/csv/201..., PartitionFilters:
[], PushedFilters: [], ReadSchema: struct<DEST_COUNTRY_NAME:string>
```

In [32]:
```
# Spark SQL Functions
from pyspark.sql.functions import max

flightData2015.select(max("count")).take(1)
```

Out[32]:
```
[Row(max(count)=370002)]
```

In [41]:
```
max_count = spark.sql("""
SELECT max(count) as max_count
FROM flight_data_2015
""")
```

In [42]:
```
type(max_count)
```

Out[42]:
```
pyspark.sql.dataframe.DataFrame
```

In [43]:
```
max_count.collect()[0]
```

Out[43]:
```
Row(max_count=370002)
```

In [44]:
```
max_count.collect()[0].max_count
```

Out[44]:
```
370002
```

In [45]:
```
maxSql = spark.sql("""
SELECT DEST_COUNTRY_NAME, sum(count) as destination_total
FROM flight_data_2015
GROUP BY DEST_COUNTRY_NAME
ORDER BY sum(count) DESC
LIMIT 5
""")

maxSql.show()
```

```
+-----------------+-----------------+
|DEST_COUNTRY_NAME|destination_total|
+-----------------+-----------------+
|    United States|           411352|
|           Canada|             8399|
|           Mexico|             7140|
|   United Kingdom|             2025|
|            Japan|             1548|
+-----------------+-----------------+
```

In [46]:
```
from pyspark.sql.functions import desc

top5_destDF = flightData2015\
  .groupBy("DEST_COUNTRY_NAME")\
  .sum("count")\
  .withColumnRenamed("sum(count)", "destination_total")\
  .sort(desc("destination_total"))\
  .limit(5)

top5_destDF.show()
```

```
+-----------------+-----------------+
|DEST_COUNTRY_NAME|destination_total|
+-----------------+-----------------+
|    United States|           411352|
|           Canada|             8399|
|           Mexico|             7140|
|   United Kingdom|             2025|
|            Japan|             1548|
+-----------------+-----------------+
```

In [47]:
```
top5_destDF.explain()
```

```
== Physical Plan ==
TakeOrderedAndProject(limit=5, orderBy=[destination_total#171L DESC NU
LLS LAST], output=[DEST_COUNTRY_NAME#26,destination_total#171L])
+- *(2) HashAggregate(keys=[DEST_COUNTRY_NAME#26], functions=[sum(cast
(count#28 as bigint))])
   +- Exchange hashpartitioning(DEST_COUNTRY_NAME#26, 200)
      +- *(1) HashAggregate(keys=[DEST_COUNTRY_NAME#26], functions=[pa
rtial_sum(cast(count#28 as bigint))])
         +- *(1) FileScan csv [DEST_COUNTRY_NAME#26,count#28] Batched:
false, Format: CSV, Location: InMemoryFileIndex[file:/home/gong/spark/
books/Spark-The-Definitive-Guide/data/flight-data/csv/201..., Partitio
nFilters: [], PushedFilters: [], ReadSchema: struct<DEST_COUNTRY_NAME:
string,count:int>
```

In [ ]:
```

```