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

spark = SparkSession\
    .builder\
    .appName("chapter-05-basic-operation")\
    .getOrCreate()

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

In [2]:
```
file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/json/2015-summary
df = spark.read.format("json").load(file_path)
```

In [3]:
```
df.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 [6]:
```
# COMMAND ----------

df.schema
```

Out[6]:
```
StructType(List(StructField(DEST_COUNTRY_NAME,StringType,true),StructF
ield(ORIGIN_COUNTRY_NAME,StringType,true),StructField(count,LongType,t
rue)))
```

In [7]:
```
df.printSchema()
```

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

In [8]:
```
# COMMAND ----------

from pyspark.sql.types import StructField, StructType, StringType, Long

myManualSchema = StructType([
  StructField("DEST_COUNTRY_NAME", StringType(), True),
  StructField("ORIGIN_COUNTRY_NAME", StringType(), True),
  StructField("count", LongType(), False, metadata={"hello":"world"})
])
```

In [14]:
```
df2 = spark.read.format("json").schema(myManualSchema).load(file_path)
```

In [15]:
```
df2.printSchema()
```

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

In [16]:
```
df2.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]:
```
# COMMAND ----------

from pyspark.sql.functions import col, column
col("someColumnName")
# column("someColumnName")
```

Out[17]:
```
Column<b'someColumnName'>
```

In [18]:
```
# COMMAND ----------

from pyspark.sql.functions import expr
expr("(((someCol + 5) * 200) - 6) < otherCol")
```

Out[18]:
```
Column<b'((((someCol + 5) * 200) - 6) < otherCol)'>
```

In [19]:
```
# COMMAND ----------

from pyspark.sql import Row
myRow = Row("Hello", None, 1, False)
```

In [20]:
```
# COMMAND ----------

myRow[0], myRow[2]
```

Out[20]:
```
('Hello', 1)
```

In [23]:
```
# COMMAND ----------

from pyspark.sql import Row
from pyspark.sql.types import StructField, StructType, StringType, LongT
myManualSchema = StructType([
  StructField("some", StringType(), True),
  StructField("col", StringType(), True),
  StructField("names", LongType(), False)
])
myRow = Row("Hello", None, 1)
myDf = spark.createDataFrame([myRow], myManualSchema)
myDf.show()
```

```
+-----+----+-----+
| some| col|names|
+-----+----+-----+
|Hello|null|    1|
+-----+----+-----+
```

In [24]:
```
# COMMAND ----------

## df = spark.read.format("json").load("../data/flight-data/json/2015-su
df.createOrReplaceTempView("dfTable")
```

In [25]:
```
# COMMAND ----------

df.select("DEST_COUNTRY_NAME").show(2)
```

```
+-----------------+
|DEST_COUNTRY_NAME|
+-----------------+
|    United States|
|    United States|
+-----------------+
only showing top 2 rows
```

In [26]:
```
# COMMAND ----------

df.select("DEST_COUNTRY_NAME", "ORIGIN_COUNTRY_NAME").show(2)
```

```
+-----------------+-------------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|
+-----------------+-------------------+
|    United States|            Romania|
|    United States|            Croatia|
+-----------------+-------------------+
only showing top 2 rows
```

In [28]:
```
# COMMAND ----------

from pyspark.sql.functions import expr, col, column
df.select(
    "ORIGIN_COUNTRY_NAME",
    expr("DEST_COUNTRY_NAME as dest"),
    col("DEST_COUNTRY_NAME").alias("dest_country"),
    column("DEST_COUNTRY_NAME"))\
  .show(5)
```

```
+-------------------+-------------+-------------+-----------------+
|ORIGIN_COUNTRY_NAME|         dest| dest_country|DEST_COUNTRY_NAME|
+-------------------+-------------+-------------+-----------------+
|            Romania|United States|United States|    United States|
|            Croatia|United States|United States|    United States|
|            Ireland|United States|United States|    United States|
|      United States|        Egypt|        Egypt|            Egypt|
|              India|United States|United States|    United States|
+-------------------+-------------+-------------+-----------------+
only showing top 5 rows
```

In [29]:
```
# COMMAND ----------

df.select(expr("DEST_COUNTRY_NAME AS destination")).show(2)
```

```
+-------------+
|  destination|
+-------------+
|United States|
|United States|
+-------------+
only showing top 2 rows
```

In [30]:
```
# COMMAND ----------

df.select(expr("DEST_COUNTRY_NAME as destination"),
    .alias("DEST_COUNTRY_NAME"))\
    .show(2)
```

```
+-----------------+
|DEST_COUNTRY_NAME|
+-----------------+
|    United States|
|    United States|
+-----------------+
only showing top 2 rows
```

In [31]:
```
# COMMAND ----------

df.selectExpr("DEST_COUNTRY_NAME as newColumnName", "DEST_COUNTRY_NAME"
```

```
+-------------+-----------------+
|newColumnName|DEST_COUNTRY_NAME|
+-------------+-----------------+
|United States|    United States|
|United States|    United States|
+-------------+-----------------+
only showing top 2 rows
```

In [32]:
```
# COMMAND ----------

df.selectExpr(
  "*", # all original columns
  "(DEST_COUNTRY_NAME = ORIGIN_COUNTRY_NAME) as withinCountry")\
  .show(2)
```

```
+-----------------+-------------------+-----+-------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|withinCountry|
+-----------------+-------------------+-----+-------------+
|    United States|            Romania|   15|        false|
|    United States|            Croatia|    1|        false|
+-----------------+-------------------+-----+-------------+
only showing top 2 rows
```

In [33]:
```
# COMMAND ----------

df.selectExpr("avg(count)", "count(distinct(DEST_COUNTRY_NAME))").show(
```

```
+-----------+---------------------------------+
| avg(count)|count(DISTINCT DEST_COUNTRY_NAME)|
+-----------+---------------------------------+
|1770.765625|                              132|
+-----------+---------------------------------+
```

In [34]:
```
# COMMAND ----------

from pyspark.sql.functions import lit
df.select(expr("*"), lit(1).alias("One")).show(2)
```

```
+-----------------+-------------------+-----+---+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|One|
+-----------------+-------------------+-----+---+
|    United States|            Romania|   15|  1|
|    United States|            Croatia|    1|  1|
+-----------------+-------------------+-----+---+
only showing top 2 rows
```

In [35]:
```
# COMMAND ----------

df.withColumn("numberOne", lit(1)).show(2)
```

```
+-----------------+-------------------+-----+---------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|numberOne|
+-----------------+-------------------+-----+---------+
|    United States|            Romania|   15|        1|
|    United States|            Croatia|    1|        1|
+-----------------+-------------------+-----+---------+
only showing top 2 rows
```

In [36]:
```
# COMMAND ----------

df.withColumn("withinCountry", expr("ORIGIN_COUNTRY_NAME == DEST_COUNTR
  .show(2)
```

```
+-----------------+-------------------+-----+-------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|withinCountry|
+-----------------+-------------------+-----+-------------+
|    United States|            Romania|   15|        false|
|    United States|            Croatia|    1|        false|
+-----------------+-------------------+-----+-------------+
only showing top 2 rows
```

In [37]:
```
# COMMAND ----------

df.withColumnRenamed("DEST_COUNTRY_NAME", "dest").columns


# COMMAND ----------
```

Out[37]:
```
['dest', 'ORIGIN_COUNTRY_NAME', 'count']
```

In [38]:
```
dfWithLongColName = df.withColumn(
    "This Long Column-Name",
    expr("ORIGIN_COUNTRY_NAME"))


# COMMAND ----------
```

In [39]:
```
dfWithLongColName.selectExpr(
    "`This Long Column-Name`",
    "`This Long Column-Name` as `new col`")\
  .show(2)
```

```
+--------------------+-------+
|This Long Column-Name|new col|
+--------------------+-------+
|              Romania|Romania|
|              Croatia|Croatia|
+--------------------+-------+
only showing top 2 rows
```

In [40]:
```
# COMMAND ----------

dfWithLongColName.select(expr("`This Long Column-Name`")).columns
```

Out[40]:
```
['This Long Column-Name']
```

In [41]:
```
# COMMAND ----------

df.where(col("count") < 2).where(col("ORIGIN_COUNTRY_NAME") != "Croatia
  .show(2)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|          Singapore|    1|
|          Moldova|      United States|    1|
+-----------------+-------------------+-----+
only showing top 2 rows
```

In [42]:
```
# COMMAND ----------

df.select("ORIGIN_COUNTRY_NAME", "DEST_COUNTRY_NAME").distinct().count(
```

Out[42]:
```
256
```

In [45]:
```
# COMMAND ----------

df.select("ORIGIN_COUNTRY_NAME").distinct().count()
```

Out[45]:
```
125
```

In [46]:
```
# COMMAND ----------

seed = 5
withReplacement = False
fraction = 0.5
df.sample(withReplacement, fraction, seed).count()
```

Out[46]: 126

In [47]:
```
# COMMAND ----------

dataFrames = df.randomSplit([0.25, 0.75], seed)
dataFrames[0].count() > dataFrames[1].count() # False
```

Out[47]: False

In [49]:
```
# COMMAND ----------

from pyspark.sql import Row
schema = df.schema
newRows = [
  Row("New Country", "Other Country", 5),
  Row("New Country 2", "Other Country 3", 1)
]
parallelizedRows = spark.sparkContext.parallelize(newRows)
newDF = spark.createDataFrame(parallelizedRows, schema)
```

In [50]:
```
# COMMAND ----------

df.union(newDF)\
  .where("count = 1")\
  .where(col("ORIGIN_COUNTRY_NAME") != "United States")\
  .show()
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Croatia|    1|
|    United States|          Singapore|    1|
|    United States|          Gibraltar|    1|
|    United States|             Cyprus|    1|
|    United States|            Estonia|    1|
|    United States|          Lithuania|    1|
|    United States|           Bulgaria|    1|
|    United States|            Georgia|    1|
|    United States|            Bahrain|    1|
|    United States|   Papua New Guinea|    1|
|    United States|         Montenegro|    1|
|    United States|            Namibia|    1|
|    New Country 2|    Other Country 3|    1|
+-----------------+-------------------+-----+
```

In [51]:
```
# COMMAND ----------

df.sort("count").show(5)
df.orderBy("count", "DEST_COUNTRY_NAME").show(5)
df.orderBy(col("count"), col("DEST_COUNTRY_NAME")).show(5)
```

```
+--------------------+-------------------+-----+
|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+--------------------+-------------------+-----+
|               Malta|      United States|    1|
|Saint Vincent and...|      United States|    1|
|       United States|            Croatia|    1|
|       United States|          Gibraltar|    1|
|       United States|          Singapore|    1|
+--------------------+-------------------+-----+
only showing top 5 rows

+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|     Burkina Faso|      United States|    1|
|    Cote d'Ivoire|      United States|    1|
|           Cyprus|      United States|    1|
|         Djibouti|      United States|    1|
|        Indonesia|      United States|    1|
+-----------------+-------------------+-----+
only showing top 5 rows

+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|     Burkina Faso|      United States|    1|
|    Cote d'Ivoire|      United States|    1|
|           Cyprus|      United States|    1|
|         Djibouti|      United States|    1|
|        Indonesia|      United States|    1|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [52]:
```
# COMMAND ----------

from pyspark.sql.functions import desc, asc
df.orderBy(expr("count desc")).show(2)
df.orderBy(col("count").desc(), col("DEST_COUNTRY_NAME").asc()).show(2)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|          Moldova|      United States|    1|
|    United States|            Croatia|    1|
+-----------------+-------------------+-----+
only showing top 2 rows

+-----------------+-------------------+------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME| count|
+-----------------+-------------------+------+
|    United States|      United States|370002|
|    United States|             Canada|  8483|
+-----------------+-------------------+------+
only showing top 2 rows
```

In [53]:
```
# COMMAND ----------


df.sortWithinPartitions("count")
```

Out[53]:
```
DataFrame[DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string, coun
t: bigint]
```

In [54]:
```
# COMMAND ----------

df.limit(5).show()


# COMMAND ----------

df.orderBy(expr("count desc")).limit(6).show()
```

```
+-----------------+-------------------+-----+
|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|
+-----------------+-------------------+-----+

+--------------------+-------------------+-----+
|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+--------------------+-------------------+-----+
|               Malta|      United States|    1|
|Saint Vincent and...|      United States|    1|
|       United States|            Croatia|    1|
|       United States|          Gibraltar|    1|
|       United States|          Singapore|    1|
|             Moldova|      United States|    1|
+--------------------+-------------------+-----+
```

In [55]:
```
# COMMAND ----------

df.rdd.getNumPartitions() # 1


# COMMAND ----------

df.repartition(5)
```

Out[55]:
```
DataFrame[DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string, coun
t: bigint]
```

In [56]:
```
# COMMAND ----------

df.repartition(col("DEST_COUNTRY_NAME"))


# COMMAND ----------
```

Out[56]:
```
DataFrame[DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string, coun
t: bigint]
```

In [57]:
```
df.repartition(5, col("DEST_COUNTRY_NAME"))
```

Out[57]:
```
DataFrame[DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string, coun
t: bigint]
```

In [58]:
```
# COMMAND ----------

df.repartition(5, col("DEST_COUNTRY_NAME")).coalesce(2)
```

Out[58]:
```
DataFrame[DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string, coun
t: bigint]
```

In [59]:
```
# COMMAND ----------

collectDF = df.limit(10)
collectDF.take(5) # take works with an Integer count
collectDF.show() # this prints it out nicely
collectDF.show(5, False)
collectDF.collect()


# COMMAND ----------
```

```
+-----------------+-------------------+-----+
|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|
|    United States|          Singapore|    1|
|    United States|            Grenada|   62|
|       Costa Rica|      United States|  588|
|          Senegal|      United States|   40|
|          Moldova|      United States|    1|
+-----------------+-------------------+-----+

+-----------------+-------------------+-----+
|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
```

Out[59]:
```
[Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Romani
a', count=15),
 Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Croati
a', count=1),
 Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Irelan
d', count=344),
 Row(DEST_COUNTRY_NAME='Egypt', ORIGIN_COUNTRY_NAME='United States',
count=15),
 Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='India',
count=62),
 Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Singapor
e', count=1),
 Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Grenad
a', count=62),
 Row(DEST_COUNTRY_NAME='Costa Rica', ORIGIN_COUNTRY_NAME='United Stat
es', count=588),
 Row(DEST_COUNTRY_NAME='Senegal', ORIGIN_COUNTRY_NAME='United State
s', count=40),
```

```
 Row(DEST_COUNTRY_NAME='Moldova', ORIGIN_COUNTRY_NAME='United State
s', count=1)]
```

In [ ]: