```
'''
Created on 07-Jun-2019

@author: SiddharthMohanty

TOPICS COVERED:
a) Reading a file into Spark Dataframe: spark.read
b) Creating a dataframe on the fly: spark.createDataframe
c) Projecting columns from a dataframe: select, selectExpr
d) Subseting a dataframe: filter, where
e) Vertical Merging of dataframes: union
f) Ordering of rows in a dataframe: sort, orderBy
g) Selecting distinct records from a dataframe: df.distinct()
h) Selecting a random sample from a dataframe: df.sample()
i) Selecting a fixed number of rows from a dataframe: df.limit(n)
j) Collecting rows to the driver: df.collect()

'''


''' Import Statements '''
from pyspark.sql import SparkSession

'''CREATE SPARK SESSION '''

spark = SparkSession.builder.master('local').appName("basic-operations").getOrCreate()



''' CREATE SCHEMA FOR READ '''

from pyspark.sql.types import StructField, StructType, StringType, LongType, IntegerType

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


''' Defining the column "FLT_Count" not there in json data '''

airline_schema_1 = StructType([StructField("DEST_COUNTRY_NAME", StringType(), True),
                               StructField("ORIGIN_COUNTRY_NAME", StringType(), True),
                               StructField("FLT_Count", IntegerType(), False,
                                           metadata={"hello":"world"})])


''' READ THE CONTENTS OF A JSON FILE TO A SPARK DATAFRAME'''

airline_df = spark.read.format("json").schema(airline_schema).\
                    load(r"C:\Users\SiddharthMohanty\Downloads\
                    Spark-The-Definitive-Guide-master\Spark-The-Definitive-Guide-master\
                    data\flight-data\json\2015-summary.json")

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

```
# |-- count: integer (nullable = true)


''' Without mentioning the schema, spark infers the schema automatically'''

airline_df = spark.read.json(r"C:\Users\SiddharthMohanty\Downloads\
                Spark-The-Definitive-Guide-master\Spark-The-Definitive-Guide-master\
                data\flight-data\json\2015-summary.json")

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

''' Mentioning the schema again'''

airline_df = spark.read.schema(airline_schema).json(r"C:\Users\SiddharthMohanty\
                    Downloads\Spark-The-Definitive-Guide-master\
                    Spark-The-Definitive-Guide-master\data\flight-data\json\
                    2015-summary.json")

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

''' Reading the json file with the schema airline_schema_1 '''

airline_df = spark.read.format("json").schema(airline_schema).\
load(r"C:\Users\SiddharthMohanty\Downloads\Spark-The-Definitive-Guide-master\
                    Spark-The-Definitive-Guide-master\data\flight-data\json\
                    2015-summary.json")

#root
# |-- DEST_COUNTRY_NAME: string (nullable = true)
# |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
# |-- FLT_Count: integer (nullable = true)

#airline_df.show()
#+-------------------+-------------------+---------+
#|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|FLT_Count|
#+-------------------+-------------------+---------+
#|       United States|            Romania|     null|
#|       United States|            Croatia|     null|
#|       United States|            Ireland|     null|

''' Reading a csv file with schema which does not have headers'''

airline_df_csv = spark.read.format("csv").schema(airline_schema_1).\
load(r"C:\Users\SiddharthMohanty\Downloads\Spark-The-Definitive-Guide-master\
                Spark-The-Definitive-Guide-master\data\flight-data\csv\2015-summary.csv")

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

```
#+-------------------+-------------------+---------+
#|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|FLT_Count|
#+-------------------+-------------------+---------+
#|       United States|            Romania|       15|
#|       United States|            Croatia|        1|
#|       United States|            Ireland|      344|
#|              Egypt|      United States|       15|
```

```
''' CREATE A DATAFRAME ON THE FLY USING  airline_schema'''

from pyspark.sql import Row

''' From python list '''

myRow1 = Row("United States", "India", 3)
myRow2 = Row("United States", "Sri Lanka", 2)
rowList = [myRow1, myRow2]
ontheFlyDf = spark.createDataFrame(rowList, airline_schema)

''' From RDD '''

myRdd = spark.sparkContext.parallelize(rowList)
ontheFlyDfFromRdd = spark.createDataFrame(myRdd, airline_schema)

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

#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
#+-----------------+-------------------+-----+
#|    United States|              India|    3|
#|    United States|          Sri Lanka|    2|
#+-----------------+-------------------+-----+


''' SELECT COLUMNS FROM A DATAFRAME '''

from pyspark.sql.functions import col, expr, concat

''' Selecting a single column from a dataframe '''

one_col_df = airline_df.select(col("DEST_COUNTRY_NAME"))
one_col_expr_df = airline_df.select(expr("DEST_COUNTRY_NAME"))
one_col_df_1 = airline_df.select("DEST_COUNTRY_NAME")
one_col_py = airline_df.select(airline_df.DEST_COUNTRY_NAME)
one_col_py_1 = airline_df.select(airline_df["DEST_COUNTRY_NAME"])

#root
# |-- DEST_COUNTRY_NAME: string (nullable = true)

#+-----------------+
#|DEST_COUNTRY_NAME|
#+-----------------+
#|    United States|
#+-----------------+
```

```
''' Selecting multiple columns from a df '''

mul_col_df = airline_df.select(col("DEST_COUNTRY_NAME"), col("count"))
mul_col_expr_df = airline_df.select(expr("DEST_COUNTRY_NAME"), expr("count"))
mul_col_df_1 = airline_df.select("DEST_COUNTRY_NAME", "count")
mul_col_py_1 = airline_df.select(airline_df["DEST_COUNTRY_NAME"], airline_df["count"])

# |-- DEST_COUNTRY_NAME: string (nullable = true)
# |-- count: integer (nullable = true)

#+-----------------+-----+
#|DEST_COUNTRY_NAME|count|
#+-----------------+-----+
#|    United States|   15|
#|    United States|    1|
#|    United States|  344|
#+-----------------+-----+

mul_col_py = airline_df.select(airline_df.DEST_COUNTRY_NAME, airline_df.count)

# TypeError: Invalid argument, not a string or column: <bound method DataFrame.count of
#        DataFrame[DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string, count: int]>
#        of type <class 'method'>. For column literals, use 'lit', 'array', 'struct' or
#        'create_map' function.

# Doesn't work because "count" is a reserved word in python and airline_df.count returns
#        something else instead of column

''' To fix the error, in airline_schema_1 defined the colname for third col
    as FLT_Count '''

mul_col_py = airline_df_csv.select(airline_df_csv.DEST_COUNTRY_NAME,
                                   airline_df_csv.FLT_Count)

#root
# |-- DEST_COUNTRY_NAME: string (nullable = true)
# |-- FLT_Count: integer (nullable = true)

#+-----------------+---------+
#|DEST_COUNTRY_NAME|FLT_Count|
#+-----------------+---------+
#|    United States|       15|
#|    United States|        1|
#|    United States|      344|
#+-----------------+---------+

mul_col_py = airline_df_csv.select(airline_df_csv.DEST_COUNTRY_NAME,
                                   "FLT_Count") # WORKS

mul_col_py = airline_df_csv.select(col("DEST_COUNTRY_NAME"),
                                   "FLT_Count") # WORKS

mul_col_py = airline_df_csv.select(col("DEST_COUNTRY_NAME"),
                                   expr("FLT_Count")) # WORKS
```

```
mul_col_py = airline_df_csv.select(col("DEST_COUNTRY_NAME"),
                                   airline_df_csv.FLT_Count) # WORKS

mul_col_py = airline_df_csv.select(expr("DEST_COUNTRY_NAME"),
                                   airline_df_csv["FLT_Count"]) # WORKS


''' RENAMING AN EXPRESSION IN SELECT'''

mul_col_py = airline_df_csv.select(col("DEST_COUNTRY_NAME"),
                                   col("FLT_Count").alias("COUNT_FLT"))


mul_col_py = airline_df_csv.select(col("DEST_COUNTRY_NAME"),
                                   expr("FLT_Count as COUNT_FLT"))

mul_col_py = airline_df_csv.select(col("DEST_COUNTRY_NAME"),
                                   expr("FLT_Count").alias("COUNT_FLT"))

#+--------------------+---------+
#|   DEST_COUNTRY_NAME|COUNT_FLT|
#+--------------------+---------+
#|       United States|       15|


''' selectExpr DEMO '''

mul_col_py.selectExpr("DEST_COUNTRY_NAME AS DCN", "COUNT_FLT AS FC").show(2)

#+-------------+---+
#|          DCN| FC|
#+-------------+---+
#|United States| 15|
#|United States|  1|


''' COLUMN LEVEL ARITHMATIC OPERATIONS AND AGGREGATIONS '''

''' Arithmatic operation on each element of a column '''

airline_df.select((col("count")*10).alias("Count_10")).show(3)
airline_df.select(expr("count*10 as Count_10")).show(3)

#+--------+
#|Count_10|
#+--------+
#|     150|
#|      10|
#|    3440|
#+--------+

''' Substring operation on each element of a column '''

airline_df.select((col("DEST_COUNTRY_NAME").substr(0,5)).alias("SUB_DEST")).show(3)
airline_df.select(expr("SUBSTRING(DEST_COUNTRY_NAME,0,5) as SUB_DEST")).show(3)

#+--------+
```

```
#|SUB_DEST|
#+--------+
#|   Unite|
#|   Unite|
#|   Unite|
#+--------+
```

```
''' distinct of entire column '''

airline_df.select((col("DEST_COUNTRY_NAME")).alias("DCN")).distinct().show(2)
airline_df.select(expr("DEST_COUNTRY_NAME as DCN")).distinct().show(2)
```

```
#+--------+
#|     DCN|
#+--------+
#|Anguilla|
#|  Russia|
#+--------+
```

```
''' Logical and arithmatic operations in select '''

airline_df.select(expr("*"), expr("count > 100").alias("CntGr100")).show()
airline_df.select(expr("*"), (col("count") > 100).alias("CntGr100")).show(3)
airline_df.selectExpr("*", "(count > 100) as CntGr100")
```

```
#+--------------------+-------------------+-----+--------+
#|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|CntGr100|
#+--------------------+-------------------+-----+--------+
#|       United States|            Romania|   15|   false|
#|       United States|            Croatia|    1|   false|
#|       United States|            Ireland|  344|    true|
```

```
airline_df.select(expr("*"), expr("(count * 100)/1000").alias("CntGr100")).show(3)
airline_df.select(expr("*"), ((col("count") * 100)/1000).alias("CntGr100")).show(3)
```

```
#+-----------------+-------------------+-----+--------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|CntGr100|
#+-----------------+-------------------+-----+--------+
#|    United States|            Romania|   15|     1.5|
#|    United States|            Croatia|    1|     0.1|
#|    United States|            Ireland|  344|    34.4|
#+-----------------+-------------------+-----+--------+
```

```
''' Performing aggregation on entire columns using selectExpr'''

airline_df.selectExpr("avg(count)").show()
airline_df.selectExpr("min(count)").show()
airline_df.selectExpr("max(count)").show()
airline_df.selectExpr("sum(count)").show()

from pyspark.sql.functions import avg, min, max, sum

airline_df.select(avg(col("count"))).show()
airline_df.select(min(col("count"))).show()
airline_df.select(max(col("count"))).show()
```

```
airline_df.select(sum(col("count"))).show()


''' ADD A NEW COLUMN TO THE DATAFRAME '''

''' by using lit() function : CREATE A COLUMN OF LITERAL VALUE '''

from pyspark.sql.functions import lit

airline_df.select("*", lit(1).alias("One")).show(2)
airline_df.select("*", lit("HI").alias("HI_STR")).show(2)

''' withColumn function defined in DataFrame '''

airline_df.withColumn("One", lit(1)).show(2)
airline_df.withColumn("HI_STR", lit("HI")).show(2)

#+-----------------+-------------------+-----+---+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|One|
#+-----------------+-------------------+-----+---+
#|    United States|            Romania|   15|  1|
#|    United States|            Croatia|    1|  1|
#+-----------------+-------------------+-----+---+

#+-----------------+-------------------+-----+------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|HI_STR|
#+-----------------+-------------------+-----+------+
#|    United States|            Romania|   15|    HI|
#|    United States|            Croatia|    1|    HI|
#+-----------------+-------------------+-----+------+

airline_df.withColumn("IS_CNT_GR3", expr("count > 3")).show(2)
airline_df.withColumn("IS_CNT_GR3", col("count") > 3).show(2)

#+-----------------+-------------------+-----+----------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|IS_CNT_GR3|
#+-----------------+-------------------+-----+----------+
#|    United States|            Romania|   15|      true|
#|    United States|            Croatia|    1|     false|
#+-----------------+-------------------+-----+----------+

airline_df.withColumn("DCN_OCN",
concat(concat(col("DEST_COUNTRY_NAME"),lit(":")),col("ORIGIN_COUNTRY_NAME"))).show(2)
airline_df.withColumn("DCN_OCN", expr("DEST_COUNTRY_NAME || ':' || ORIGIN_COUNTRY_NAME")).show(2)
airline_df.withColumn("DCN_OCN", expr("CONCAT(CONCAT(DEST_COUNTRY_NAME, ':'),
ORIGIN_COUNTRY_NAME)")).show(2)

#+-----------------+-------------------+-----+--------------------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|             DCN_OCN|
#+-----------------+-------------------+-----+--------------------+
#|    United States|            Romania|   15|United States:Rom...|
#|    United States|            Croatia|    1|United States:Cro...|
#+-----------------+-------------------+-----+--------------------+


''' RENAMING EXISTING COLUMN IN A DATAFRAME withColumnRenamed '''
```

```
airline_df.withColumn("my_cnt", expr("count>100")).\
            withColumnRenamed("my_cnt","IS_CNT_GR").show()

#+-------------------+-------------------+-----+---------+
#|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|IS_CNT_GR|
#+-------------------+-------------------+-----+---------+
#|       United States|            Romania|   15|    false|
#|       United States|            Croatia|    1|    false|


''' REMOVE A COLUMN USING drop '''

airline_df_newcol = airline_df.withColumn("my_cnt", expr("count>100")).\
                                withColumnRenamed("my_cnt","IS_CNT_GR")

#root
# |-- DEST_COUNTRY_NAME: string (nullable = true)
# |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
# |-- count: integer (nullable = true)
# |-- IS_CNT_GR: boolean (nullable = true)

airline_df_drop = airline_df_newcol.drop("IS_CNT_GR")

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


''' CHANGING THE TYPE OF A COLUMN USING withColumn: change IS_CNT_GR to string '''

airline_df_newcol.withColumn("IS_CNT_GR_Str", col("IS_CNT_GR").cast("string")).printSchema()

#root
# |-- DEST_COUNTRY_NAME: string (nullable = true)
# |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
# |-- count: long (nullable = true)
# |-- IS_CNT_GR: boolean (nullable = true)
# |-- IS_CNT_GR_Str: string (nullable = true)



''' FILTER ROWS FROM A DF USING filter or where '''

airline_df_newcol.where("IS_CNT_GR = true").show(3)
airline_df_newcol.where(col("IS_CNT_GR") == "true").show(3)

#+-----------------+-------------------+-----+---------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|IS_CNT_GR|
#+-----------------+-------------------+-----+---------+
#|    United States|            Ireland|  344|     true|
#|       Costa Rica|      United States|  588|     true|
#|    United States|       Sint Maarten|  325|     true|

airline_df_newcol.where("count < 2").show(2)
airline_df_newcol.where(col("count") < 2).show(2)
```

```
#+-----------------+-------------------+-----+---------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|IS_CNT_GR|
#+-----------------+-------------------+-----+---------+
#|    United States|            Croatia|    1|    false|
#|    United States|          Singapore|    1|    false|
#+-----------------+-------------------+-----+---------+
```

```
airline_df_newcol.where("ORIGIN_COUNTRY_NAME = 'Croatia'").show(2)
airline_df_newcol.where(col("ORIGIN_COUNTRY_NAME") == 'Croatia').show(2)
```

```
#+-----------------+-------------------+-----+---------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|IS_CNT_GR|
#+-----------------+-------------------+-----+---------+
#|    United States|            Croatia|    1|    false|
#+-----------------+-------------------+-----+---------+
```

```
''' Multiple logical conditions '''

airline_df_newcol.where("ORIGIN_COUNTRY_NAME = 'Croatia' AND DEST_COUNTRY_NAME = 'United
States'").show(2)
airline_df_newcol.where((col("ORIGIN_COUNTRY_NAME") == 'Croatia') &
                        (col("DEST_COUNTRY_NAME") == 'United States')).show(2)
```

```
#+-----------------+-------------------+-----+---------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|IS_CNT_GR|
#+-----------------+-------------------+-----+---------+
#|    United States|            Croatia|    1|    false|
#+-----------------+-------------------+-----+---------+
```

```
airline_df_newcol.where("ORIGIN_COUNTRY_NAME = 'Croatia' OR DEST_COUNTRY_NAME = 'United
States'").show(2)
airline_df_newcol.where((col("ORIGIN_COUNTRY_NAME") == 'Croatia') |
                        (col("DEST_COUNTRY_NAME") == 'United States')).show(2)
```

```
#+-----------------+-------------------+-----+---------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|IS_CNT_GR|
#+-----------------+-------------------+-----+---------+
#|    United States|            Romania|   15|    false|
#|    United States|            Croatia|    1|    false|
#+-----------------+-------------------+-----+---------+
```

```
# In all the above queries just replace where with filer and all the queries will
#     give the same result

''' use of isin() synonymous to IN clause in SQL '''

airline_df_newcol.where(col("ORIGIN_COUNTRY_NAME").isin(['Croatia','India', 'China'])).show(3)
```

```
#+-----------------+-------------------+-----+---------+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|IS_CNT_GR|
#+-----------------+-------------------+-----+---------+
#|    United States|            Croatia|    1|    false|
#|    United States|              India|   62|    false|
#|    United States|              China|  920|     true|
#+-----------------+-------------------+-----+---------+
```

```
''' GENERATING A RANDOM SAMPLE USING sample()'''

airline_10 = airline_df_newcol.selectExpr("*").limit(10)
airline_10.count()
#10

seed = 5
withRep = False
fraction = 0.5

airline_10.sample(withRep, fraction, seed).count()
#5

''' COMBINING TWO DATAFRAMES USING union  '''

ontheFlyDf.show()

#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
#+-----------------+-------------------+-----+
#|    United States|              India|    3|
#|    United States|          Sri Lanka|    2|
#+-----------------+-------------------+-----+

airline_10.drop("IS_CNT_GR").union(ontheFlyDf).count()

#12


''' SORTING DATA IN A DATAFRAME USING sort and orderBy '''

from pyspark.sql.functions import desc, asc

airline_df.sort("count").show()

#+--------------------+-------------------+-----+
#|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
#+--------------------+-------------------+-----+
#|       United States|            Estonia|    1|
#|              Kosovo|      United States|    1|
#|              Zambia|      United States|    1|

airline_df.sort("DEST_COUNTRY_NAME").show()

#+--------------------+-------------------+-----+
#|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
#+--------------------+-------------------+-----+
#|             Algeria|      United States|    4|
#|              Angola|      United States|   15|
#|            Anguilla|      United States|   41|
#| Antigua and Barbuda|      United States|  126|

airline_df.sort("DEST_COUNTRY_NAME", desc("count")).show()

#+--------------------+-------------------+-----+
#|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
#+--------------------+-------------------+-----+
#|             Algeria|      United States|    4|
```

```
#|             Angola|      United States|   15|
#|           Anguilla|      United States|   41|
#| Antigua and Barbuda|      United States|  126|
```

```
airline_df.sort(col("DEST_COUNTRY_NAME").desc(),asc("count")).show()
```

```
#+-----------------+-------------------+-----+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
#+-----------------+-------------------+-----+
#|           Zambia|      United States|    1|
#|        Venezuela|      United States|  290|
#|          Uruguay|      United States|   43|
#|    United States|          Gibraltar|    1|
```

```
airline_df.orderBy("count").show()
airline_df.orderBy("DEST_COUNTRY_NAME").show()
airline_df.orderBy("DEST_COUNTRY_NAME","count").show()
airline_df.orderBy(desc("DEST_COUNTRY_NAME"),asc("count")).show()
airline_df.orderBy(col("DEST_COUNTRY_NAME").desc(),asc("count")).show()


''' SELECTING A FIXED NUMBER OF ROWS '''

airline_df.limit(3).show()
```

```
#+-----------------+-------------------+-----+
#|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
#+-----------------+-------------------+-----+
#|    United States|            Romania|   15|
#|    United States|            Croatia|    1|
#|    United States|            Ireland|  344|
#+-----------------+-------------------+-----+
```

```
''' COLLECT ROWS TO DRIVER '''

myList = airline_df.limit(3).collect()

#[Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Romania', count=15),
#    Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Croatia', count=1),
#    Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Ireland', count=344)]
```