



Chapter 4: Spark SQL & Dataframe

Ex2: Flights

Cho tập tin `flights_small.csv` và `airports.csv`

Yêu cầu:

1. Đọc tập tin `flights_small.csv` vào data.
2. In schema của data. Hiển thị 3 dòng đầu tiên của dữ liệu. Cho biết dữ liệu có bao nhiêu dòng?
3. Từ data hãy tạo một view có tên là `flights_small`
4. Từ view `flights_small` hãy tạo một dataframe có tên là `flights`. Hiển thị 3 dòng đầu tiên của dữ liệu
5. Trong `flights`, hãy tạo thêm cột `duration_hrs` ($= \text{air_time}/60$)
6. Lọc dữ liệu các chuyến bay có `distance > 2000` => `long_flights1`. Cho biết có bao nhiêu dòng dữ liệu thỏa điều kiện này? (bằng cả SQL và Dataframe)
7. Lọc dữ liệu các chuyến bay có `300 <= air_time <= 600` => `time_flights`. Cho biết có bao nhiêu dòng dữ liệu thỏa điều kiện này? (bằng cả SQL và Dataframe)
8. Tạo dữ liệu `selected1` từ `flights` với các cột "origin", "dest", "carrier". Tạo biến điều kiện lọc thứ nhất `filterA` là các chuyến bay có origin là "SEA", tạo biến điều kiện lọc thứ hai `filterB` là các chuyến bay bay có dest là "PDX". Từ đó hãy tạo Dataframe kết quả `selected2` từ `selected1` thỏa cả `filterA` và `filterB`. Cho biết có bao nhiêu dòng dữ liệu thỏa điều kiện trên.
9. Tạo riêng một biến `avg_speed` (alias "avg_speed") là tốc độ trung bình của chuyến bay (tính theo giờ) $\text{distance}/(\text{air_time}/60)$. Tạo Dataframe `speed1` từ `flights` với các cột "origin", "dest", "tailnum" và cột "avg_speed" vừa tạo
10. Tạo Dataframe `speed2` từ `flights` với các cột "origin", "dest", "tailnum" và cột `avg_speed = distance/(\text{air_time}/60)` trong cùng một lệnh (dùng `selectExpr`)
11. Sử dụng aggregation method để: tìm `air_time` nhỏ nhất của các chuyến bay có origin là "PDX"; tìm `distance` lớn nhất của các chuyến bay có origin là "SEA"; tìm tổng `duration` các chuyến bay (theo giờ).
12. Nhóm dữ liệu và đếm số chuyến bay theo từng "tailnum"; nhóm dữ liệu và đếm số chuyến bay theo từng "origin"; nhóm dữ liệu và tính trung bình của `air_time` theo từng "origin"
13. Nhóm các chuyến bay theo "month", "dest" => `by_month_dest`. Tính trung bình "dep_delay" theo `by_month_dest`. Tính std "dep_delay" theo `by_month_dest`.
14. Đọc tập tin `airports.csv` vào `airports`. In schema của `airports`. Hiển thị 3 dòng đầu tiên của dữ liệu. Cho biết dữ liệu có bao nhiêu dòng?
15. Đổi tên cột "faa" trong `airports` thành "dest".
16. Tạo một Dataframe mới bằng cách kết hợp `flights` và `airports` theo "dest"

```
In [1]: import findspark
findspark.init()
```



In [2]: `import pyspark`

In [3]: `from pyspark import SparkContext
from pyspark.conf import SparkConf
from pyspark.sql import SparkSession`

In [4]: `sc = SparkContext()`

In [5]: `spark = SparkSession(sc)`

In [6]: `#1.
data = spark.read.csv("flights_small.csv", header=True,
inferSchema=True)`

In [7]: `#2.
data.printSchema()`

```
root  
|-- year: integer (nullable = true)  
|-- month: integer (nullable = true)  
|-- day: integer (nullable = true)  
|-- dep_time: string (nullable = true)  
|-- dep_delay: string (nullable = true)  
|-- arr_time: string (nullable = true)  
|-- arr_delay: string (nullable = true)  
|-- carrier: string (nullable = true)  
|-- tailnum: string (nullable = true)  
|-- flight: integer (nullable = true)  
|-- origin: string (nullable = true)  
|-- dest: string (nullable = true)  
|-- air_time: string (nullable = true)  
|-- distance: integer (nullable = true)  
|-- hour: string (nullable = true)  
|-- minute: string (nullable = true)
```

In [8]: `print("Number of rows:", data.count())`

Number of rows: 10000

In [9]: `#3.
data.createOrReplaceTempView("flights_small")`

In [10]: `#4. Create the DataFrame flights
flights = spark.table("flights_small")`



In [11]: *# Show the head*
`flights.show(3)`

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|year|month|day|dep_time|dep_delay|arr_time|arr_delay|carrier|tailnum|flight|or
igin|dest|air_time|distance|hour|minute|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|2014|  12|  8|    658|    -7|    935|    -5|    VX| N846VA|  1780|
SEA| LAX|    132|    954|    6|    58|
|2014|   1| 22|   1040|     5|   1505|     5|    AS| N559AS|   851|
SEA| HNL|    360|   2677|   10|   40|
|2014|   3|  9|   1443|    -2|   1652|     2|    VX| N847VA|   755|
SEA| SFO|    111|    679|   14|   43|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
only showing top 3 rows

```

In [12]: *#5. Add duration_hrs*
`flights = flights.withColumn("duration_hrs", flights.air_time/60)`

In [13]: *# flights.show(3)*

In [14]: *# Show the first 3 observations*
`for row in flights.head(3):`
 `print(row)`
 `print('\n')`

Row(year=2014, month=12, day=8, dep_time='658', dep_delay='-7', arr_time='935',
arr_delay='-5', carrier='VX', tailnum='N846VA', flight=1780, origin='SEA', dest
='LAX', air_time='132', distance=954, hour='6', minute='58', duration_hrs=2.2)

Row(year=2014, month=1, day=22, dep_time='1040', dep_delay='5', arr_time='150
5', arr_delay='5', carrier='AS', tailnum='N559AS', flight=851, origin='SEA', de
st='HNL', air_time='360', distance=2677, hour='10', minute='40', duration_hrs=
6.0)

Row(year=2014, month=3, day=9, dep_time='1443', dep_delay='-2', arr_time='165
2', arr_delay='2', carrier='VX', tailnum='N847VA', flight=755, origin='SEA', de
st='SFO', air_time='111', distance=679, hour='14', minute='43', duration_hrs=1.
85)

In [15]: *#6. Filter flights by passing a string*
`long_flights2 = flights.filter("distance > 2000")`

In [16]: `print("Number of rows (distance >2000):", long_flights2.count())`

Number of rows (distance >2000): 1481



In [17]: *# Print the data to check*

```
long_flights2.show(2)
```

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|year|month|day|dep_time|dep_delay|arr_time|arr_delay|carrier|tailnum|flight|or
igin|dest|air_time|distance|hour|minute|duration_hrs|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|2014|  1| 22|  1040|      5|  1505|      5|   AS| N559AS|  851|
SEA| HNL|   360|  2677|    10|   40|    6.0|
|2014|  1| 13|  2156|     -9|   607|    -15|   AS| N597AS|   24|
SEA| BOS|   290|  2496|    21|   56|4.833333333333333|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
only showing top 2 rows
```

In [18]: *# Construct the "query"*

```
query = '''SELECT * FROM flights_small WHERE distance > 2000'''
```

Apply the SQL "query"

```
long_flights2_sql = spark.sql(query)
```

In [19]: long_flights2_sql.count()

Out[19]: 1481

In [20]: *#7.*

```
time_flights = flights.filter("air_time >= 300 and air_time <= 600")
```

In [21]: time_flights.show(2)

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|year|month|day|dep_time|dep_delay|arr_time|arr_delay|carrier|tailnum|flight|or
igin|dest|air_time|distance|hour|minute|duration_hrs|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|2014|  1| 22|  1040|      5|  1505|      5|   AS| N559AS|  851|
SEA| HNL|   360|  2677|    10|   40|    6.0|
|2014| 12|  4|   954|     -6|  1348|    -17|   HA| N395HA|   29|
SEA| OGG|   333|  2640|     9|   54|    5.55|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
only showing top 2 rows
```

In [22]: *print("Number of rows (air_time > 300 and air_time < 600):",*
time_flights.count())

Number of rows (air_time > 300 and air_time < 600): 440



```
In [23]: # Construct the "query"
query = '''SELECT * FROM flights_small WHERE air_time between 300 and 600'''
# Apply the SQL "query"
time_flights_sql = spark.sql(query)
```

```
In [24]: time_flights_sql.count()
```

```
Out[24]: 440
```

```
In [25]: #8.
# Select set of columns
selected1 = flights.select(flights.origin,
                           flights.dest, flights.carrier)

# Define first filter
filterA = flights.origin == "SEA"

# Define second filter
filterB = flights.dest == "PDX"

# Filter the data, first by filterA then by filterB
selected2 = selected1.filter(filterA).filter(filterB)
```

```
In [26]: selected2.show(5)
```

```
+-----+-----+-----+
|origin|dest|carrier|
+-----+-----+-----+
|  SEA| PDX|      00|
|  SEA| PDX|      00|
|  SEA| PDX|      00|
|  SEA| PDX|      00|
|  SEA| PDX|      00|
+-----+-----+-----+
only showing top 5 rows
```

```
In [27]: print("Number of rows (origin = 'SEA' & dest = 'PDX'):",selected2.count())
```

```
Number of rows (origin = 'SEA' & dest = 'PDX'): 157
```

```
In [28]: #9.
# Define avg_speed
avg_speed = (flights.distance/(flights.air_time/60)).alias("avg_speed")

# Select the correct columns
speed1 = flights.select("origin", "dest", "tailnum", avg_speed)
```



In [29]: `speed1.show(2)`

```
+-----+-----+-----+-----+
|origin|dest|tailnum|      avg_speed|
+-----+-----+-----+-----+
|   SEA| LAX| N846VA|433.6363636363636|
|   SEA| HNL| N559AS|446.1666666666667|
+-----+-----+-----+-----+
only showing top 2 rows
```

In [30]: *#10. Create the same table using a SQL expression*
`speed2 = flights.selectExpr("origin", "dest", "tailnum",
 "distance/(air_time/60) as avg_speed")`

In [31]: `speed2.show(2)`

```
+-----+-----+-----+-----+
|origin|dest|tailnum|      avg_speed|
+-----+-----+-----+-----+
|   SEA| LAX| N846VA|433.6363636363636|
|   SEA| HNL| N559AS|446.1666666666667|
+-----+-----+-----+-----+
only showing top 2 rows
```

In [32]: `from pyspark.sql.types import IntegerType`

In [33]: *#11.*
`flights = flights.withColumn("air_time",
 flights["air_time"].cast(IntegerType()))
Find the shortest time from PDX in terms of air_time
flights.filter(flights.origin == "PDX").groupBy().min("air_time").show()`

```
+-----+
|min(air_time)|
+-----+
|           24|
+-----+
```

In [34]: *# Find the Longest distance from SEA in terms of distance*
`flights.filter(flights.origin == "SEA").groupBy().max("distance").show()`

```
+-----+
|max(distance)|
+-----+
|          2724|
+-----+
```



```
In [35]: # Total hours in the air
flights.withColumn("duration_hrs",
                  flights.air_time/60).groupBy().sum("duration_hrs").show()
```

```
+-----+
| sum(duration_hrs)|
+-----+
|25289.600000000126|
+-----+
```

```
In [36]: from pyspark.sql.functions import avg
```

```
In [37]: #12.
# Group by tailnum
by_plane = flights.groupBy("tailnum")
```

```
In [38]: # Number of flights each plane made
by_plane.count().show()
```

```
+-----+-----+
|tailnum|count|
+-----+-----+
| N442AS|    38|
| N102UW|     2|
| N36472|     4|
| N38451|     4|
| N73283|     4|
| N513UA|     2|
| N954WN|     5|
| N388DA|     3|
| N567AA|     1|
| N516UA|     2|
| N927DN|     1|
| N8322X|     1|
| N466SW|     1|
|  N6700|     1|
| N607AS|    45|
| N622SW|     4|
| N584AS|    31|
| N914WN|     4|
| N654AW|     2|
| N336NW|     1|
+-----+-----+
```

only showing top 20 rows

```
In [39]: # Group by origin
by_origin = flights.groupBy("origin").count()
```



In [40]: `by_origin.show()`

```
+-----+-----+
|origin|count|
+-----+-----+
|   SEA| 6754|
|   PDX| 3246|
+-----+-----+
```

In [41]: `# Average air_time`
`flights.groupBy("origin").avg("air_time").show()`

```
+-----+-----+
|origin|    avg(air_time)|
+-----+-----+
|   SEA| 160.4361496051259|
|   PDX|137.11543248288737|
+-----+-----+
```

In [42]: `from pyspark.sql.types import IntegerType`
`import pyspark.sql.functions as F`



```
In [43]: # 13
flights = flights.withColumn("dep_delay",
                             flights["dep_delay"].cast(IntegerType()))

# Group by month and dest
by_month_dest = flights.groupBy("month", "dest")

# Average departure delay by month and destination
by_month_dest.avg("dep_delay").show()
```

```
+-----+-----+-----+
|month|dest|    avg(dep_delay)|
+-----+-----+-----+
|    4| PHX| 1.6833333333333333|
|    1| RDM|          -1.625|
|    5| ONT| 3.5555555555555554|
|    7| OMA|          -6.5|
|    8| MDW|          7.45|
|    6| DEN| 5.418181818181818|
|    5| IAD|          -4.0|
|   12| COS|          -1.0|
|   11| ANC| 7.529411764705882|
|    5| AUS|          -0.75|
|    5| COS| 11.666666666666666|
|    2| PSP|           0.6|
|    4| ORD| 0.14285714285714285|
|   10| DFW| 18.176470588235293|
|   10| DCA|          -1.5|
|    8| JNU|         18.125|
|   11| KOA|          -1.0|
|   10| OMA| -0.6666666666666666|
|    6| ONT|          9.625|
|    3| MSP|           3.2|
+-----+-----+-----+
```

only showing top 20 rows



```
In [44]: # Standard deviation of departure delay
by_month_dest.agg(F.stddev("dep_delay")).show()
```

```
+-----+-----+-----+
|month|dest|stddev_samp(dep_delay)|
+-----+-----+-----+
|    4| PHX|    15.003380033491737|
|    1| RDM|     8.830749846821778|
|    5| ONT|    18.895178691342874|
|    7| OMA|     2.1213203435596424|
|    8| MDW|    14.467659032985843|
|    6| DEN|    13.536905534420026|
|    5| IAD|     3.8078865529319543|
|   12| COS|     1.4142135623730951|
|   11| ANC|    18.604716401245316|
|    5| AUS|     4.031128874149275|
|    5| COS|    33.38163167571851|
|    2| PSP|     4.878524367060187|
|    4| ORD|    11.593882803741764|
|   10| DFW|    45.53019017606675|
|   10| DCA|     0.7071067811865476|
|    8| JNU|    40.79368823727514|
|   11| KOA|     1.8708286933869707|
|   10| OMA|     5.8594652770823155|
|    6| ONT|    25.98316762829351|
|    3| MSP|    21.556779370817555|
+-----+-----+-----+
only showing top 20 rows
```

```
In [45]: #14.
airports = spark.read.csv("airports.csv", header=True,
                           inferSchema=True)
```

```
In [46]: airports.printSchema()
```

```
root
|-- faa: string (nullable = true)
|-- name: string (nullable = true)
|-- lat: double (nullable = true)
|-- lon: double (nullable = true)
|-- alt: integer (nullable = true)
|-- tz: integer (nullable = true)
|-- dst: string (nullable = true)
```

```
In [47]: print("Number of lines:", airports.count())
```

```
Number of lines: 1397
```



In [48]: *# Examine the data*
`airports.show(3)`

```
+---+-----+-----+-----+-----+-----+
|faa|          name|      lat|      lon| alt| tz|dst|
+---+-----+-----+-----+-----+-----+
|04G|  Lansdowne Airport|41.1304722|-80.6195833|1044| -5| A|
|06A|Moton Field Munic...|32.4605722|-85.6800278| 264| -5| A|
|06C| Schaumburg Regional|41.9893408|-88.1012428| 801| -6| A|
+---+-----+-----+-----+-----+-----+
only showing top 3 rows
```

In [49]: *#15. Rename the faa column*
`airports = airports.withColumnRenamed("faa", "dest")`

In [50]: `airports.show(3)`

```
+---+-----+-----+-----+-----+-----+
|dest|          name|      lat|      lon| alt| tz|dst|
+---+-----+-----+-----+-----+-----+
| 04G|  Lansdowne Airport|41.1304722|-80.6195833|1044| -5| A|
| 06A|Moton Field Munic...|32.4605722|-85.6800278| 264| -5| A|
| 06C| Schaumburg Regional|41.9893408|-88.1012428| 801| -6| A|
+---+-----+-----+-----+-----+-----+
only showing top 3 rows
```

In [51]: *#16. Join the DataFrames*
`flights_with_airports = flights.join(airports,
 on="dest", how="leftouter")`



In [52]: *# Examine the new DataFrame*
 flights_with_airports.show(3)

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+
|dest|year|month|day|dep_time|dep_delay|arr_time|arr_delay|carrier|tailnum|fligh
ht|origin|air_time|distance|hour|minute|duration_hrs|          name|      1
at|      lon|alt|  tz|dst|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+
| LAX|2014|  12|  8|    658|    -7|    935|    -5|    VX| N846VA|  17
80|  SEA|   132|    954|    6|    58|    2.2| Los Angeles Intl|33.9425
36|-118.408075|126| -8|  A|
| HNL|2014|   1| 22|   1040|     5|   1505|     5|    AS| N559AS|   8
51|  SEA|   360|   2677|   10|   40|    6.0| Honolulu Intl|21.3186
81|-157.922428| 13|-10|  N|
| SFO|2014|   3|  9|   1443|    -2|   1652|     2|    VX| N847VA|   7
55|  SEA|   111|    679|   14|   43|    1.85|San Francisco Intl|37.6189
72|-122.374889| 13| -8|  A|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+
only showing top 3 rows

```