cognitiveclass.ai logo

Introduction to DataFrames

Estimated time needed: **15** minutes

Objectives

A DataFrame is two-dimensional. Columns can be of different data types. DataFrames accept many data inputs including series and other DataFrames. You can pass indexes (row labels) and columns (column labels). Indexes can be numbers, dates, or strings/tuples.

After completing this lab you will be able to:

- Load a data file into a DataFrame
- View the data schema of a DataFrame
- Perform basic data manipulation
- Aggregate data in a DataFrame

Setup

For this lab, we are going to be using Python and Spark (PySpark). These libraries should be installed in your lab environment or in SN Labs.

Pandas is a popular data science package for Python. In this lab, we use Pandas to load a CSV file from disc to a pandas dataframe in memory. PySpark is the Spark API for Python. In this lab, we use PySpark to initialize the spark context.

In [1]:

```
# Installing required packages
!pip install pyspark
!pip install findspark
!pip install pandas
```

```
Requirement already satisfied: pyspark in /home/jupyterlab/conda/envs/pyth
on/lib/python3.7/site-packages (3.2.1)
Requirement already satisfied: py4j==0.10.9.3 in /home/jupyterlab/conda/en
vs/python/lib/python3.7/site-packages (from pyspark) (0.10.9.3)
Requirement already satisfied: findspark in /home/jupyterlab/conda/envs/py
thon/lib/python3.7/site-packages (2.0.0)
Requirement already satisfied: pandas in /home/jupyterlab/conda/envs/pytho
n/lib/python3.7/site-packages (1.3.4)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/jupyterlab/
conda/envs/python/lib/python3.7/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /home/jupyterlab/conda/env
s/python/lib/python3.7/site-packages (from pandas) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /home/jupyterlab/conda/env
s/python/lib/python3.7/site-packages (from pandas) (1.21.4)
Requirement already satisfied: six>=1.5 in /home/jupyterlab/conda/envs/pyt
hon/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas) (1.1
6.0)
```

In [19]:

```
import findspark
findspark.init()
```

In [20]:

```
import pandas as pd
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
```

Exercise 1 - Spark session

In this exercise, you will create and initialize the Spark session needed to load the dataframes and operate on it

Task 1: Creating the spark session and context

In [22]:

```
# Creating a spark context class
sc = SparkContext()

# Creating a spark session
spark = SparkSession \
    .builder \
    .appName("Python Spark DataFrames basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
/tmp/ipykernel_566/653474904.py in <module>
      1 # Creating a spark context class
----> 2 sc = SparkContext()
      3 
      4 # Creating a spark session
      5 spark = SparkSession \

~/conda/envs/python/lib/python3.7/site-packages/pyspark/context.py in __in
it__(self, master, appName, sparkHome, pyFiles, environment, batchSize, se
rializer, conf, gateway, jsc, profiler_cls)
    142                 " is not allowed as it is a security risk.")
    143 
--> 144         SparkContext._ensure_initialized(self, gateway=gateway, co
nf=conf)
    145         try:
    146             self._do_init(master, appName, sparkHome, pyFiles, env
ironment, batchSize, serializer,

~/conda/envs/python/lib/python3.7/site-packages/pyspark/context.py in _ens
ure_initialized(cls, instance, gateway, conf)
    353                         " created by %s at %s:%s "
    354                         % (currentAppName, currentMaster,
--> 355                             callsite.function, callsite.file, call
site.linenum))
    356                 else:
    357                     SparkContext._active_spark_context = instance

ValueError: Cannot run multiple SparkContexts at once; existing SparkConte
xt(app=pyspark-shell, master=local[*]) created by __init__ at /tmp/ipykern
el_566/653474904.py:2
```

Task 2: Initialize Spark session

To work with dataframes we just need to verify that the spark session instance has been created.

In [23]:

```
spark
```

Out[23]:

SparkSession - in-memory
SparkContext

Spark UI (http://jupyterlab-joaocosta1:4040)
Version
```
v3.2.1
```
Master
```
local[*]
```
AppName
```
pyspark-shell
```

Exercise 2 - Load the data and Spark dataframe

In this section, you will first read the CSV file into a Pandas DataFrame and then read it into a Spark DataFrame. Pandas is a library used for data manipulation and analysis. Pandas offers data structures and operations for creating and manipulating Data Series and DataFrame objects. Data can be imported from various data sources, e.g., Numpy arrays, Python dictionaries, and CSV files. Pandas allows you to manipulate, organize and display the data. To create a Spark DataFrame we load an external DataFrame, called mtcars. This DataFrame includes 32 observations on 11 variables:

colIndex	colName	units/description
[, 1]	mpg	Miles per gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (lb/1000)
[, 7]	qsec	1/4 mile time
[, 8]	vs	V/S
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears
[,11]	carb	Number of carburetors

Task 1: Loading data into a Pandas DataFrame

In [25]:

```
# Read the file using `read_csv` function in pandas
mtcars = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clou
d/IBM-BD0225EN-SkillsNetwork/labs/data/mtcars.csv')
```

In [26]:

```
# Preview a few records
mtcars.head()
```

Out[26]:

	Unnamed: 0	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

Task 2: Loading data into a Spark DataFrame

In [27]:

```
# We use the `createDataFrame` function to load the data into a spark dataframe
sdf = spark.createDataFrame(mtcars)
```

In [28]:

```
# Let us look at the schema of the loaded spark dataframe
sdf.printSchema()
```

```
root
 |-- Unnamed: 0: string (nullable = true)
 |-- mpg: double (nullable = true)
 |-- cyl: long (nullable = true)
 |-- disp: double (nullable = true)
 |-- hp: long (nullable = true)
 |-- drat: double (nullable = true)
 |-- wt: double (nullable = true)
 |-- qsec: double (nullable = true)
 |-- vs: long (nullable = true)
 |-- am: long (nullable = true)
 |-- gear: long (nullable = true)
 |-- carb: long (nullable = true)
```

Exercise 3: Basic data analysis and manipulation

In this section, we perform basic data analysis and manipulation. We start with previewing the data and then applying some filtering and columwise operations.

Task 1: Displays the content of the DataFrame

We use the `show()` method for this. Here we preview the first 5 records. Compare it to a similar `head()` function in Pandas.

In [29]:

```
sdf.show(5)
```

```
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+
|       Unnamed: 0| mpg|cyl| disp| hp|drat|   wt| qsec| vs| am|gear|carb|
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+
|        Mazda RX4|21.0|  6|160.0|110| 3.9| 2.62|16.46|  0|  1|   4|   4|
|    Mazda RX4 Wag|21.0|  6|160.0|110| 3.9|2.875|17.02|  0|  1|   4|   4|
|       Datsun 710|22.8|  4|108.0| 93|3.85| 2.32|18.61|  1|  1|   4|   1|
|   Hornet 4 Drive|21.4|  6|258.0|110|3.08|3.215|19.44|  1|  0|   3|   1|
|Hornet Sportabout|18.7|  8|360.0|175|3.15| 3.44|17.02|  0|  0|   3|   2|
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+
only showing top 5 rows
```

We use the `select()` function to select a particular column of data. Here we show the `mpg` column.

In [30]:

```
sdf.select('mpg').show(5)
```

```
+----+
| mpg|
+----+
|21.0|
|21.0|
|22.8|
|21.4|
|18.7|
+----+
only showing top 5 rows
```

Task 2: Filtering and Columnar operations

Filtering and Column operations are important to select relevant data and apply useful transformations.

We first filter to only retain rows with mpg > 18. We use the `filter()` function for this.

In [31]:

```
sdf.filter(sdf['mpg'] < 18).show(5)
```

```
+-----------+----+---+-----+---+----+----+-----+---+---+----+----+
| Unnamed: 0| mpg|cyl| disp| hp|drat|  wt| qsec| vs| am|gear|carb|
+-----------+----+---+-----+---+----+----+-----+---+---+----+----+
| Duster 360|14.3|  8|360.0|245|3.21|3.57|15.84|  0|  0|   3|   4|
|  Merc 280C|17.8|  6|167.6|123|3.92|3.44| 18.9|  1|  0|   4|   4|
| Merc 450SE|16.4|  8|275.8|180|3.07|4.07| 17.4|  0|  0|   3|   3|
| Merc 450SL|17.3|  8|275.8|180|3.07|3.73| 17.6|  0|  0|   3|   3|
|Merc 450SLC|15.2|  8|275.8|180|3.07|3.78| 18.0|  0|  0|   3|   3|
+-----------+----+---+-----+---+----+----+-----+---+---+----+----+
only showing top 5 rows
```

Operating on Columns

Spark also provides a number of functions that can be directly applied to columns for data processing and aggregation. The example below shows the use of basic arithmetic functions to convert the weight values from `lb` to `metric ton` . We create a new column called `wtTon` that has the weight from the `wt` column converted to metric tons.

In [13]:

```
sdf.withColumn('wtTon', sdf['wt'] * 0.45).show(5)
```

```
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+-
------+
|       Unnamed: 0| mpg|cyl| disp| hp|drat|   wt| qsec| vs| am|gear|carb|
wtTon|
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+-
------+
|        Mazda RX4|21.0|  6|160.0|110| 3.9| 2.62|16.46|  0|  1|   4|   4|
1.179|
|    Mazda RX4 Wag|21.0|  6|160.0|110| 3.9|2.875|17.02|  0|  1|   4|   4|
1.29375|
|       Datsun 710|22.8|  4|108.0| 93|3.85| 2.32|18.61|  1|  1|   4|   1|
1.044|
|   Hornet 4 Drive|21.4|  6|258.0|110|3.08|3.215|19.44|  1|  0|   3|   1|
1.44675|
|Hornet Sportabout|18.7|  8|360.0|175|3.15| 3.44|17.02|  0|  0|   3|   2|
1.548|
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+-
------+
only showing top 5 rows
```

Exercise 4: Grouping and Aggregation

Spark DataFrames support a number of commonly used functions to aggregate data after grouping. In this example we compute the average weight of cars by their cylinders as shown below.

In [32]:

```
sdf.groupby(['cyl'])\
.agg({"wt": "AVG"})\
.show(5)
```

```
[Stage 22:==========>                                          (3 + 13)
/ 16]

+---+------------------+
|cyl|           avg(wt)|
+---+------------------+
|  6| 3.117142857142857|
|  4|2.2857272727272724|
|  8| 3.999214285714286|
+---+------------------+
```

We can also sort the output from the aggregation to get the most common cars.

In [33]:

```
car_counts = sdf.groupby(['cyl'])\
.agg({"wt": "count"})\
.sort("count(wt)", ascending=False)\
.show(5)
```

```
[Stage 25:========================>                              (7 + 9)
/ 16]

+---+---------+
|cyl|count(wt)|
+---+---------+
|  8|       14|
|  4|       11|
|  6|        7|
+---+---------+
```

Practice Questions

Question 1 - DataFrame basics

Display the first 5 rows of all cars that have atleast 5 cylinders.

In [34]:

```
# Code block for learners to answer
sdf.filter(sdf['cyl'] > 4 ).show(5)
```

```
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+
|       Unnamed: 0| mpg|cyl| disp| hp|drat|   wt| qsec| vs| am|gear|carb|
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+
|        Mazda RX4|21.0|  6|160.0|110| 3.9| 2.62|16.46|  0|  1|   4|   4|
|    Mazda RX4 Wag|21.0|  6|160.0|110| 3.9|2.875|17.02|  0|  1|   4|   4|
|   Hornet 4 Drive|21.4|  6|258.0|110|3.08|3.215|19.44|  1|  0|   3|   1|
|Hornet Sportabout|18.7|  8|360.0|175|3.15| 3.44|17.02|  0|  0|   3|   2|
|          Valiant|18.1|  6|225.0|105|2.76| 3.46|20.22|  1|  0|   3|   1|
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+
only showing top 5 rows
```

Question 2 - DataFrame aggregation

Using the functions and tables shown above, print out the mean weight of a car in our database in metric tons.

In [43]:

```
sdf.withColumn('wtTon', sdf['wt'] * 0.45).filter(sdf['cyl'] > 4).agg({"wt": "AVG"}).sho
w(5)
```

```
+-----------------+
|          avg(wt)|
+-----------------+
|3.705190476190476|
+-----------------+
```

Question 3 - DataFrame columnar operations

In the earlier sections of this notebook, we have created a new column called `wtTon` to indicate the weight in metric tons using a standard conversion formula. In this case we have applied this directly to the dataframe column `wt` as it is a linear operation (multiply by 0.45). Similarly, as part of this exercise, create a new column for mileage in `kmpl` (kilometer-per-liter) instead of `mpg` (miles-per-gallon) by using a conversion factor of 0.425.

Additionally sort the output in decreasing order of mileage in kmpl.

In [44]:

```
# Code block for learners to answer
sdf.withColumn('kmpl', sdf['mpg'] * 0.425).sort('mpg', ascending=False).show()
```

```
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+-
-----------------+
|       Unnamed: 0| mpg|cyl| disp| hp|drat|   wt| qsec| vs| am|gear|carb|
kmpl|
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+-
-----------------+
|   Toyota Corolla|33.9|  4| 71.1| 65|4.22|1.835| 19.9|  1|  1|   4|   1|1
4.407499999999999|
|         Fiat 128|32.4|  4| 78.7| 66|4.08|  2.2|19.47|  1|  1|   4|   1|
13.77|
|      Honda Civic|30.4|  4| 75.7| 52|4.93|1.615|18.52|  1|  1|   4|   2|
12.92|
|     Lotus Europa|30.4|  4| 95.1|113|3.77|1.513| 16.9|  1|  1|   5|   2|
12.92|
|        Fiat X1-9|27.3|  4| 79.0| 66|4.08|1.935| 18.9|  1|  1|   4|   1|
11.6025|
|    Porsche 914-2|26.0|  4|120.3| 91|4.43| 2.14| 16.7|  0|  1|   5|   2|1
1.049999999999999|
|        Merc 240D|24.4|  4|146.7| 62|3.69| 3.19| 20.0|  1|  0|   4|   2|
10.37|
|       Datsun 710|22.8|  4|108.0| 93|3.85| 2.32|18.61|  1|  1|   4|   1|
9.69|
|         Merc 230|22.8|  4|140.8| 95|3.92| 3.15| 22.9|  1|  0|   4|   2|
9.69|
|    Toyota Corona|21.5|  4|120.1| 97| 3.7|2.465|20.01|  1|  0|   3|   1|
9.1375|
|   Hornet 4 Drive|21.4|  6|258.0|110|3.08|3.215|19.44|  1|  0|   3|   1|
9.094999999999999|
|       Volvo 142E|21.4|  4|121.0|109|4.11| 2.78| 18.6|  1|  1|   4|   2|
9.094999999999999|
|    Mazda RX4 Wag|21.0|  6|160.0|110| 3.9|2.875|17.02|  0|  1|   4|   4|
8.924999999999999|
|        Mazda RX4|21.0|  6|160.0|110| 3.9| 2.62|16.46|  0|  1|   4|   4|
8.924999999999999|
|     Ferrari Dino|19.7|  6|145.0|175|3.62| 2.77| 15.5|  0|  1|   5|   6|
8.372499999999999|
| Pontiac Firebird|19.2|  8|400.0|175|3.08|3.845|17.05|  0|  0|   3|   2|
8.16|
|         Merc 280|19.2|  6|167.6|123|3.92| 3.44| 18.3|  1|  0|   4|   4|
8.16|
|Hornet Sportabout|18.7|  8|360.0|175|3.15| 3.44|17.02|  0|  0|   3|   2|
7.9475|
|          Valiant|18.1|  6|225.0|105|2.76| 3.46|20.22|  1|  0|   3|   1|
7.692500000000001|
|        Merc 280C|17.8|  6|167.6|123|3.92| 3.44| 18.9|  1|  0|   4|   4|
7.565|
+-----------------+----+---+-----+---+----+-----+-----+---+---+----+----+-
-----------------+
only showing top 20 rows
```

Double-click **here** for a hint.

Double-click **here** for the solution.

Authors

Karthik Muthuraman (https://www.linkedin.com/in/karthik-muthuraman/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id
SkillsNetwork-Channel-SkillsNetworkCoursesIBMBD0225ENSkillsNetwork25716109-2021-01-01)

Other Contributors

Jerome Nilmeier (https://github.com/nilmeier)

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-07-02	0.2	Karthik	Beta launch
2021-06-30	0.1	Karthik	First Draft