BIG DATA IMPLEMENTATION WITH PYSPARK

PySpark and SparkSQL Basics

How to implement Spark with Python Programming

Pınar Ersoy Follow
Jan 10 · 8 min read

(Source)

Apache Spark is a cluster computing system that offers comprehensive libraries and APIs for developers and supports languages including **Java**, **Python**, **R**, and **Scala**.

SparkSQL can be represented as the module in Apache Spark for processing unstructured data with the help of **DataFrame API**.

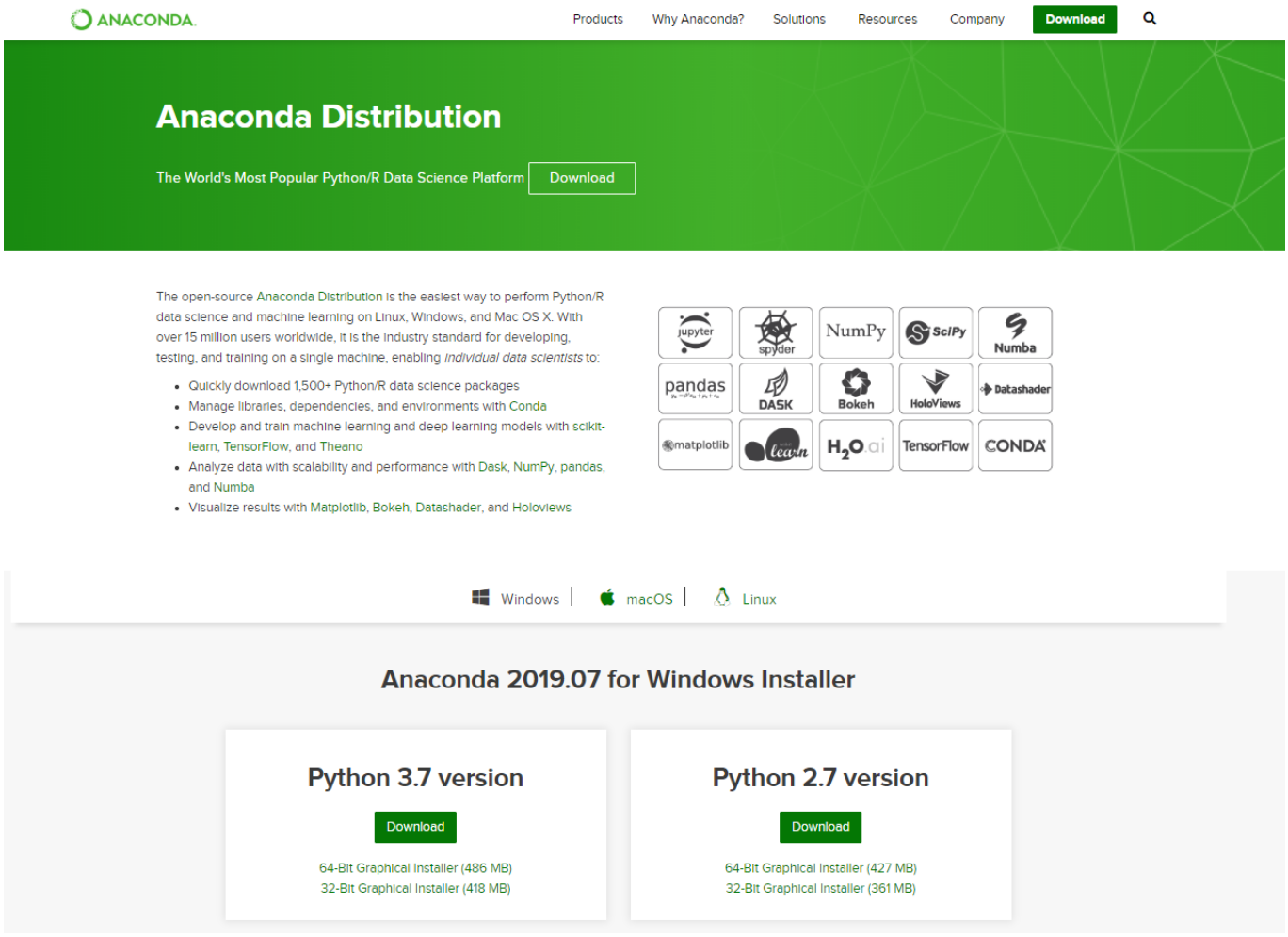
Python is revealed the Spark programming model to work with structured data by the Spark Python API which is called as **PySpark**.

This post's objective is to demonstrate how to run Spark with PySpark and execute common functions.

Python programming language requires an installed IDE. The easiest way to use Python with Anaconda since it installs sufficient IDE's and crucial packages along with itself.

1. Downloading Anaconda and Installing PySpark

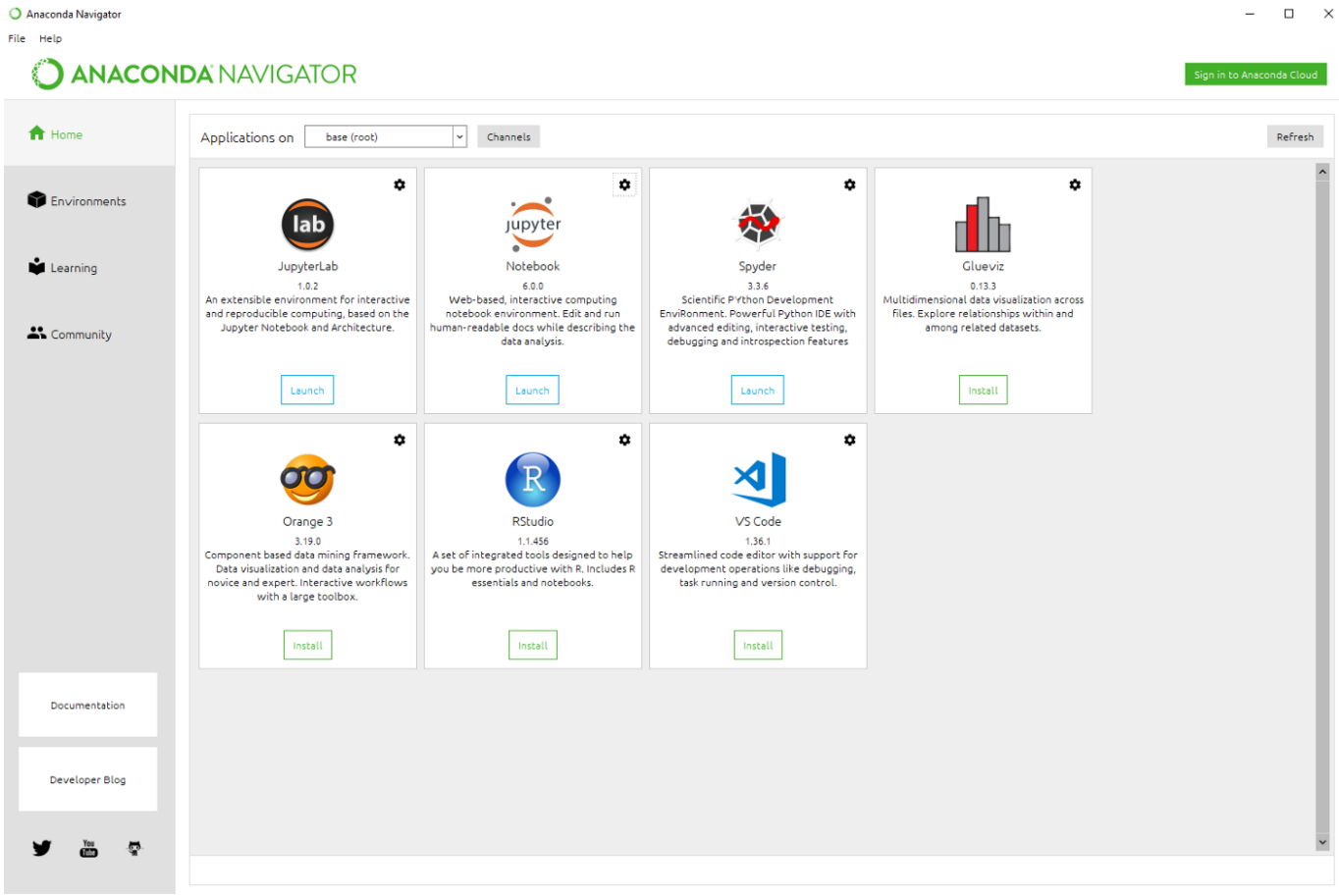
With the help of this **link** you can download Anaconda. You can choose between Windows, macOS, Linux operating systems and 64-Bit/32-Bit Graphical Installer types. The latest version of Python is recommended to be installed.



Download page for Anaconda (https://www.anaconda.com/distribution/)

After the suitable Anaconda version is downloaded, click on it to proceed with the installation procedure which is explained step by step in the **Anaconda Documentation**.

When the installation is completed, the Anaconda Navigator Homepage will be opened. In order to use Python, simply click on the "Launch" button of "Notebook" module.



Anaconda Navigator Home Page

To be able to use Spark through Anaconda, the following package installation steps shall be followed.

> ***Step 1:*** *Open "Anaconda Prompt" terminal from your computer.*
>
> ***Step 2:*** *Type " conda install pyspark" on Anaconda Prompt terminal and hit Enter to install PySpark package.*

> ***Step 3:*** *Type " conda install pyarrow" on Anaconda Prompt terminal and hit Enter to install PyArrow package.*

After PySpark and PyArrow package installations are completed, simply close the terminal and go back to Jupyter Notebook and import the required packages at the top of your code.

```
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.context import SparkContext
from pyspark.sql.functions
import *from pyspark.sql.types
import *from datetime import date, timedelta, datetime
import time
```

2. Initializing SparkSession

First of all, a Spark session needs to be initialized. With the help of SparkSession, DataFrame can be created and registered as tables. Moreover, SQL tables be executed, tables can be cached, and parquet/json/csv/avro data formatted files can be read.

```
sc = SparkSession.builder.appName("PysparkExample")\
.config ("spark.sql.shuffle.partitions", "50")\
.config("spark.driver.maxResultSize","5g")\
.config ("spark.sql.execution.arrow.enabled", "true")\
.getOrCreate()
```

For detailed explanations for each parameter of SparkSession, kindly visit pyspark.sql.SparkSession.

3. Creating Data Frames

A DataFrame can be accepted as a distributed and tabulated collection of titled columns which is similar to a table in a relational database. In this post, we will be using DataFrame operations on PySpark API while working with datasets.

You can download the Kaggle dataset from this link.

3.1. From Spark Data Sources

DataFrames can be created by reading txt, csv, json and parquet file formats. In our example, we will be using .json formatted file. You can also find and read text, csv and parquet file formats by using the related read functions as shown below.

```
#Creates a spark data frame called as raw_data.

#JSON
dataframe = sc.read.json('dataset/nyt2.json')

#TXT FILES#
dataframe_txt = sc.read.text('text_data.txt')

#CSV FILES#
dataframe_csv = sc.read.csv('csv_data.csv')

#PARQUET FILES#
dataframe_parquet = sc.read.load('parquet_data.parquet')
```

4. Duplicate Values

Duplicate values in a table can be eliminated by using dropDuplicates() function.

```
dataframe = sc.read.json('dataset/nyt2.json')
dataframe.show(10)
```

```
+--------------------+--------------------+--------------------+-----------------+--------------------+--------+--------------
--+-------------+----+--------------+--------------------+-------------+
|                 _id|  amazon_product_url|              author| bestsellers_date|         description|   price|   published_da
te|    publisher|rank|rank_last_week|               title|weeks_on_list|
+--------------------+--------------------+--------------------+-----------------+--------------------+--------+--------------
--+-------------+----+--------------+--------------------+-------------+
|[5b4aa4ead3089013...|http://www.amazon...|       Dean R Koontz|[[1211587200000]]|Odd Thomas, who c...|  [, 27]|[[121288320000
0]]|       Bantam| [1]|           [0]|           ODD HOURS|          [1]|
|[5b4aa4ead3089013...|http://www.amazon...|     Stephenie Meyer|[[1211587200000]]|Aliens have taken...|[25.99,]|[[121288320000
0]]|Little, Brown| [2]|           [1]|            THE HOST|          [3]|
|[5b4aa4ead3089013...|http://www.amazon...|        Emily Giffin|[[1211587200000]]|A woman's happy m...|[24.95,]|[[121288320000
0]]| St. Martin's| [3]|           [2]|LOVE THE ONE YOU'...|          [2]|
|[5b4aa4ead3089013...|http://www.amazon...|   Patricia Cornwell|[[1211587200000]]|A Massachusetts s...|[22.95,]|[[121288320000
0]]|       Putnam| [4]|           [0]|           THE FRONT|          [1]|
|[5b4aa4ead3089013...|http://www.amazon...|     Chuck Palahniuk|[[1211587200000]]|An aging porn que...|[24.95,]|[[121288320000
0]]|    Doubleday| [5]|           [0]|               SNUFF|          [1]|
|[5b4aa4ead3089013...|http://www.amazon...|James Patterson a...|[[1211587200000]]|A woman finds an ...|[24.99,]|[[121288320000
0]]|Little, Brown| [6]|           [3]|SUNDAYS AT TIFFANY'S|          [4]|
|[5b4aa4ead3089013...|http://www.amazon...|       John Sandford|[[1211587200000]]|The Minneapolis d...|[26.95,]|[[121288320000
0]]|       Putnam| [7]|           [4]|        PHANTOM PREY|          [3]|
|[5b4aa4ead3089013...|http://www.amazon...|       Jimmy Buffett|[[1211587200000]]|A Southern family...|[21.99,]|[[121288320000
0]]|Little, Brown| [8]|           [6]|          SWINE NOT?|          [2]|
|[5b4aa4ead3089013...|http://www.amazon...|    Elizabeth George|[[1211587200000]]|In Cornwall, tryi...|[27.95,]|[[121288320000
0]]|       Harper| [9]|           [8]|     CARELESS IN RED|          [3]|
|[5b4aa4ead3089013...|http://www.amazon...|      David Baldacci|[[1211587200000]]|An intelligence a...|[26.99,]|[[121288320000
```

```
0]]|Grand Central|[10]|          [7]|     THE WHOLE TRUTH|          [5]|
+--------------------+--------------------+--------------------+-----------------+--------------------+--------+---------------
--+-------------+----+-------------+--------------------+-------------+
only showing top 10 rows
```

After dropDuplicates() function is applied, we can observe that duplicates are removed from dataset.

```
dataframe_dropdup = dataframe.dropDuplicates()
dataframe_dropdup.show(10)
```

```
+--------------------+--------------------+--------------------+-----------------+--------------------+--------+---------------
--+-----------------+----+-------------+--------------------+-------------+
|                 _id|  amazon_product_url|              author| bestsellers_date|         description|   price|   published_da
te|       publisher|rank|rank_last_week|               title|weeks_on_list|
+--------------------+--------------------+--------------------+-----------------+--------------------+--------+---------------
--+-----------------+----+-------------+--------------------+-------------+
|[5b4aa4ead3089013...|http://www.amazon...|Clive Cussler wit...|[[1213401600000]]|Juan Cabrillo and...|[26.95,]|[[121469760000
0]]|          Putnam| [4]|          [3]|         PLAGUE SHIP|          [2]|
|[5b4aa4ead3089013...|http://www.amazon...|      Jeffery Deaver|[[1215820800000]]|Detectives Lincol...|   [, 0]|[[121711680000
0]]|Simon & Schuster|[20]|          [0]|   THE BROKEN WINDOW|          [0]|
|[5b4aa4ead3089013...|http://www.amazon...|       Daniel Silva|[[1217030400000]]|Gabriel Allon, an...|[26.95,]|[[121832640000
0]]|          Putnam| [1]|          [0]|  THE SECRET SERVANT|          [1]|
|[5b4aa4ead3089013...|http://www.amazon...|     Janet Evanovich|[[1217030400000]]|Stephanie Plum an...|[27.95,]|[[121832640000
0]]|    St. Martin’s| [9]|          [7]|   FEARLESS FOURTEEN|          [6]|
|[5b4aa4ead3089013...|http://www.amazon...|          Jane Green|[[1218240000000]]|A woman’s life ch...|   [, 0]|[[121953600000
0]]|          Viking|[18]|          [0]|     THE BEACH HOUSE|          [0]|
|[5b4aa4ead3089013...|http://www.amazon...|      Brunonia Barry|[[1220054400000]]|Secrets of a fami...|[24.95,]|[[122135040000
0]]|          Morrow|[13]|         [10]|     THE LACE READER|          [5]|
|[5b4aa4ead3089013...|http://www.amazon...|    David Wroblewski|[[1221264000000]]|A mute takes refu...|[25.95,]|[[122256000000
0]]|            Ecco| [9]|          [9]|THE STORY OF EDGA...|         [14]|
|[5b4aa4ead3089013...|http://www.amazon...|       John Sandford|[[1225497600000]]|Virgil Flowers in...|   [, 0]|[[122679360000
0]]|          Putnam|[20]|          [0]|      HEAT LIGHTNING|          [0]|
|[5b4aa4ead3089013...|http://www.amazon...|           J D Robb|[[1226102400000]]|Lt. Eve Dallas in...|[25.95,]|[[122739840000
0]]|          Putnam| [2]|          [0]|  SALVATION IN DEATH|          [1]|
|[5b4aa4ead3089013...|http://www.amazon...|       Toni Morrison|[[1226707200000]]|In 17th-century A...|[23.95,]|[[122800320000
0]]|           Knopf| [5]|          [0]|            A MERCY|          [1]|
+--------------------+--------------------+--------------------+-----------------+--------------------+--------+---------------
--+-----------------+----+-------------+--------------------+-------------+
only showing top 10 rows
```

5. Queries

Querying operations can be used for various purposes such as subsetting columns with *“select”*, adding conditions with *“when”* and filtering column contents with *“like”*. Below, some of the most commonly used operations are exemplified. For the complete list of query operations, see the **Apache Spark doc**.

5.1. “Select” Operation

It is possible to obtain columns by attribute (“author”) or by indexing (dataframe[‘author’]).

```
#Show all entries in title column
dataframe.select("author").show(10)

#Show all entries in title, author, rank, price columns
dataframe.select("author", "title", "rank", "price").show(10)
```

```
+--------------------+
|              author|
+--------------------+
|       Dean R Koontz|
|     Stephenie Meyer|
|        Emily Giffin|
|   Patricia Cornwell|
|     Chuck Palahniuk|
|James Patterson a...|
|       John Sandford|
|       Jimmy Buffett|
|    Elizabeth George|
|      David Baldacci|
+--------------------+
only showing top 10 rows

+--------------------+--------------------+----+--------+
|              author|               title|rank|   price|
+--------------------+--------------------+----+--------+
|       Dean R Koontz|           ODD HOURS| [1]|  [, 27]|
|     Stephenie Meyer|            THE HOST| [2]|[25.99,]|
|        Emily Giffin|LOVE THE ONE YOU'...| [3]|[24.95,]|
|   Patricia Cornwell|           THE FRONT| [4]|[22.95,]|
|     Chuck Palahniuk|               SNUFF| [5]|[24.95,]|
|James Patterson a...|SUNDAYS AT TIFFANY'S| [6]|[24.99,]|
|       John Sandford|        PHANTOM PREY| [7]|[26.95,]|
|       Jimmy Buffett|          SWINE NOT?| [8]|[21.99,]|
|    Elizabeth George|     CARELESS IN RED| [9]|[27.95,]|
|      David Baldacci|     THE WHOLE TRUTH|[10]|[26.99,]|
+--------------------+--------------------+----+--------+
only showing top 10 rows
```

First result table shows only "author" selection and second result table shows multiple columns

5.2. "When" Operation

In the first example, the "title" column is selected and a condition is added with a "when" condition.

```
# Show title and assign 0 or 1 depending on title
```

```
dataframe.select("title",when(dataframe.title != 'ODD HOURS',
1).otherwise(0)).show(10)
```

```
+--------------------+-------------------------------------------------------+
|               title|CASE WHEN (NOT (title = ODD HOURS)) THEN 1 ELSE 0 END|
+--------------------+-------------------------------------------------------+
|           ODD HOURS|                                                      0|
|            THE HOST|                                                      1|
|LOVE THE ONE YOU'...|                                                      1|
|           THE FRONT|                                                      1|
|               SNUFF|                                                      1|
|SUNDAYS AT TIFFANY'S|                                                      1|
|        PHANTOM PREY|                                                      1|
|          SWINE NOT?|                                                      1|
|     CARELESS IN RED|                                                      1|
|     THE WHOLE TRUTH|                                                      1|
+--------------------+-------------------------------------------------------+
only showing top 10 rows
```

Displaying 10 rows of specified conditions

In the second example, "isin" operation is applied instead of "when" which can be also used to define some conditions to rows.

```
# Show rows with specified authors if in the given options

dataframe [dataframe.author.isin("John Sandford",
"Emily Giffin")].show(5)
```

```
+--------------------+--------------------+------------+-----------------+--------------------+--------+-----------------+----
--------+----+-------------+--------------------+-------------+
|                 _id|  amazon_product_url|      author| bestsellers_date|         description|   price|   published_date|   p
ublisher|rank|rank_last_week|               title|weeks_on_list|
+--------------------+--------------------+------------+-----------------+--------------------+--------+-----------------+----
--------+----+-------------+--------------------+-------------+
|[5b4aa4ead3089013...|http://www.amazon...| Emily Giffin|[[1211587200000]]|A woman's happy m...|[24.95,]|[[1212883200000]]|St.
Martin's| [3]|           [2]|LOVE THE ONE YOU'...|          [2]|
|[5b4aa4ead3089013...|http://www.amazon...|John Sandford|[[1211587200000]]|The Minneapolis d...|[26.95,]|[[1212883200000]]|
Putnam| [7]|           [4]|        PHANTOM PREY|          [3]|
|[5b4aa4ead3089013...|http://www.amazon...| Emily Giffin|[[1212192000000]]|A woman's happy m...|[24.95,]|[[1213488000000]]|St.
Martin's| [4]|           [3]|LOVE THE ONE YOU'...|          [3]|
|[5b4aa4ead3089013...|http://www.amazon...|John Sandford|[[1212192000000]]|The Minneapolis d...|[26.95,]|[[1213488000000]]|
Putnam| [9]|           [7]|        PHANTOM PREY|          [4]|
|[5b4aa4ead3089013...|http://www.amazon...| Emily Giffin|[[1212796800000]]|A woman's happy m...|[24.95,]|[[1214092800000]]|St.
Martin's| [4]|           [4]|LOVE THE ONE YOU'...|          [4]|
+--------------------+--------------------+------------+-----------------+--------------------+--------+-----------------+----
--------+----+-------------+--------------------+-------------+
only showing top 5 rows
```

Result set displays 5 rows of specified criteria

5.3. "Like" Operation

In the brackets of “Like” function, % character is used to filter out all titles having “ THE ” word. If the condition we are looking for is the exact match, then no % character shall be used.

```
# Show author and title is TRUE if title has " THE " word in titles

dataframe.select("author", "title",
dataframe.title.like("% THE %")).show(15)
```


Result set of titles having ″ THE ″ word.

5.4. “Startswith” — “ Endswith”

StartsWith scans from the beginning of word/content with specified criteria in the brackets. In parallel, EndsWith processes the word/content starting from the end. Both of the functions are case sensitive.

```
dataframe.select("author", "title",
dataframe.title.startswith("THE")).show(5)

dataframe.select("author", "title",
dataframe.title.endswith("NT")).show(5)
```

Result sets have 5 rows of startsWith and endsWith operations.

5.5. "Substring" Operation

Substring functions to extract the text between specified indexes. In the following examples, texts are extracted from the index numbers (1, 3), (3, 6) and (1, 6).

```
dataframe.select(dataframe.author.substr(1,
3).alias("title")).show(5)

dataframe.select(dataframe.author.substr(3,
6).alias("title")).show(5)

dataframe.select(dataframe.author.substr(1,
6).alias("title")).show(5)
```

Results are displayed respectively for substring (1,3), (3,6), (1,6).

6. Add, Update & Remove Columns

Data manipulation functions are also available in the DataFrame API. Below, you can find examples to add/update/remove column operations.

6.1. Adding Columns

```
# Lit() is required while we are creating columns with exact values.

dataframe = dataframe.withColumn('new_column',
F.lit('This is a new column'))

display(dataframe)
```


New column is added at the end of the dataset

6.2. Updating Columns

For updated operations of DataFrame API, withColumnRenamed() function is used with two parameters.

```
# Update column 'amazon_product_url' with 'URL'

dataframe = dataframe.withColumnRenamed('amazon_product_url', 'URL')

dataframe.show(5)
```


'Amazon_Product_URL' column name is updated with 'URL'

6.3. Removing Columns

Removal of a column can be achieved in two ways: adding the list of column names in the drop() function or specifying columns by pointing in the drop function. Both examples are shown below.

```
dataframe_remove = dataframe.drop("publisher",
"published_date").show(5)

dataframe_remove2 = dataframe \
.drop(dataframe.publisher).drop(dataframe.published_date).show(5)
```

"publisher" and "published_date" columns are removed in two different methods.

7. Inspect Data

There exist several types of functions to inspect data. Below, you can find some of the commonly used ones. For a deeper look, visit the **Apache Spark doc**.

```
# Returns dataframe column names and data types
dataframe.dtypes

# Displays the content of dataframe
dataframe.show()

# Return first n rows
dataframe.head()

# Returns first row
dataframe.first()

# Return first n rows
dataframe.take(5)

# Computes summary statistics
dataframe.describe().show()

# Returns columns of dataframe
dataframe.columns
```

```
# Counts the number of rows in dataframe
dataframe.count()

# Counts the number of distinct rows in dataframe
dataframe.distinct().count()

# Prints plans including physical and logical
dataframe.explain(4)
```

8. "GroupBy" Operation

The grouping process is applied with GroupBy() function by adding column name in function.

```
# Group by author, count the books of the authors in the groups

dataframe.groupBy("author").count().show(10)
```


Authors are grouped by the number books published

9. "Filter" Operation

Filtering is applied by using filter() function with a condition parameter added inside of it. This function is case sensitive.

```
# Filtering entries of title
# Only keeps records having value 'THE HOST'
```

```
dataframe.filter(dataframe["title"] == 'THE HOST').show(5)
```


Title column is filtered with the content only having "THE HOST" and displaying 5 results.

10. Missing & Replacing Values

For every dataset, there is always a need for replacing, existing values, dropping unnecessary columns and filling missing values in data preprocessing stages. **pyspark.sql.DataFrameNaFunction** library helps us to manipulate data with this respect. Some examples are added below.

```
# Replacing null values
dataframe.na.fill()
dataFrame.fillna()
dataFrameNaFunctions.fill()

# Returning new dataframe restricting rows with null
valuesdataframe.na.drop()
dataFrame.dropna()
dataFrameNaFunctions.drop()

# Return new dataframe replacing one value with another
dataframe.na.replace(5, 15)
dataFrame.replace()
dataFrameNaFunctions.replace()
```

11. Repartitioning

It is possible to increase or decrease the existing level of partitioning in RDD Increasing can be actualized by using *repartition(self, numPartitions)* function which results in a

new RDD that obtains same /higher number of partitions. Decreasing can be processed with *coalesce(self, numPartitions, shuffle=False)* function that results in new RDD with a reduced number of partitions to a specified number. For more info, please visit the **Apache Spark docs**.

```
# Dataframe with 10 partitions
dataframe.repartition(10).rdd.getNumPartitions()

# Dataframe with 1 partition
dataframe.coalesce(1).rdd.getNumPartitions()
```

12. Running SQL Queries Programmatically

Raw SQL queries can also be used by enabling the "sql" operation on our SparkSession to run SQL queries programmatically and return the result sets as DataFrame structures. For more detailed information, kindly visit **Apache Spark docs**.

```
# Registering a table
dataframe.registerTempTable("df")

sc.sql("select * from df").show(3)

sc.sql("select \
CASE WHEN description LIKE '%love%' THEN 'Love_Theme' \
WHEN description LIKE '%hate%' THEN 'Hate_Theme' \                WHEN
description LIKE '%happy%' THEN 'Happiness_Theme' \
WHEN description LIKE '%anger%' THEN 'Anger_Theme' \
WHEN description LIKE '%horror%' THEN 'Horror_Theme' \
WHEN description LIKE '%death%' THEN 'Criminal_Theme' \
WHEN description LIKE '%detective%' THEN 'Mystery_Theme' \
ELSE 'Other_Themes' \                END Themes \
from df").groupBy('Themes').count().show()
```

13. Output

13.1. Data Structures

DataFrame API uses RDD as a base and it converts SQL queries into low-level RDD functions. By using .rdd operation, a dataframe can be converted into RDD. It is also

possible to convert Spark Dataframe into a string of RDD and Pandas formats.

```
# Converting dataframe into an RDD
rdd_convert = dataframe.rdd

# Converting dataframe into a RDD of string
dataframe.toJSON().first()

# Obtaining contents of df as Pandas
dataFramedataframe.toPandas()
```


Results of different data structures

13.2. Write & Save to Files

Any data source type that is loaded to our code as data frames can easily be converted and saved into other types including .parquet and .json. For more save, load, write function details, please visit **Apache Spark doc**.

```
# Write & Save File in .parquet format
dataframe.select("author", "title", "rank", "description") \
.write \
.save("Rankings_Descriptions.parquet")
```


Parquet file is created when .write .save() functions are processed.

```
# Write & Save File in .json format
dataframe.select("author", "title") \
.write \
.save("Authors_Titles.json",format="json")
```


JSON file is created when .write .save() functions are processed.

13.3. Stopping SparkSession

Spark Session can be stopped by running *stop()* function as follows.

```
# End Spark Session
sc.stop()
```

The code and Jupyter Notebook is available on my **GitHub.**

Questions and comments are highly appreciated!

. . .

References:

1. http://spark.apache.org/docs/latest/
2. https://docs.anaconda.com/anaconda/

Data Science Machine Learning Spark Python AI

About Help Legal